Helper and Equivalent Objectives: An Efficient Approach for Constrained Optimisation

Tao Xu and Jun He and Changjing Shang

***Abstract*—Numerous multi-objective evolutionary algorithms have been designed for constrained optimisation in last two decades. This method is to transform a constrained optimisation problem into a multi-objective optimisation problem, then solve it by an evolutionary algorithm. In this paper, we propose a new multi-objective method for constrained optimisation, which is to convert a constrained optimisation problem into a problem with helper and equivalent objectives. An equivalent objective means that its optimal solution set is the same as that to the constrained problem but a helper objective not. Then this multi-objective optimisation problem is decomposed into a group of sub-problems using the weighted sum approach. Weights are dynamically adjusted so that each subproblem eventually tends to a problem with an equivalent objective. We theoretically analyse the computation time of the helper and equivalent objective method on a hard problem called "wide gap". The "wide gap" problem means that an algorithm needs exponential time to cross between two fitness levels. We prove that using helper and equivalent objectives may shorten the time of crossing the "wide gap". We conduct a case study for validating our method. An algorithm with helper and equivalent objectives is implemented. Experimental results show that its overall performance is ranked first when compared with other eight state-of-art evolutionary algorithms on the IEEE CEC 2017 benchmarks in constrained optimisation. The case study proves the efficiency of the helper and equivalent objective method for constrained optimisation.**

***Index Terms*—constrained optimisation, constraint handling techniques, evolutionary algorithms, multi-objective optimisation, algorithm analysis, objective decomposition**

I. Introduction

Optimisation problems in the real world usually are subject to some constraints. A single-objective constrained optimisation problem (COP) is formulated in a mathematical form as

$$\begin{array}{ll} \min & f(\vec{x}), \quad \vec{x}=(x_1,\cdots,x_D)\in\Omega, \\ \text{subject to} & \left\{\begin{array}{ll} g_i^I(\vec{x})\le 0, & i=1,\cdots,q, \\ g_i^E(\vec{x})=0, & i=1,\cdots,r, \end{array}\right. \end{array} \tag{1}$$

where $\Omega=\{\vec{x}\mid L_j\le x_j\le U_j,\ j=1,\cdots,D\}$ is a bounded domain in $\mathbb{R}^D$. D is the dimension. L_j and U_j denote lower and upper boundaries respectively. $g_i^I(\vec{x})\le 0$ is an inequality constraint and $g_i^E(\vec{x})=0$ is an equality constraint. A feasible solution satisfies all constraints, and an infeasible solution violating at least one. The sets of optimal feasible solution(s), infeasible solutions and feasible solutions are denoted by Ω^*, Ω_I and Ω_F respectively.

Manuscript received xx xx xxxx

Tao Xu and Changjing Shang are with the Department of Computer Science, Aberystwyth University, Aberystwyth SY23 3DB, U.K. e-mail: {tax2,cns}@aber.ac.uk.

Jun He (corresponding author) is with the School of Science and Technology, Nottingham Trent University, Nottingham NG11 8NS, U.K. Email: jun.he@ntu.ac.uk.

Evolutionary algorithms (EAs) have been applied for solving COPs using different constraint handling methods, such as the penalty function, repairing infeasible solutions and multi-objective optimisation [1]–[4]. A multi-objective method works by transforming a COP into a multi-objective optimisation problem without inequality and equality constraints and then, solving it by a multi-objective EA. A popular implementation is to minimise the original objective function f and the degree of constraint violation v simultaneously.

$$\min \vec{f}(\vec{x})=(f(\vec{x}),v(\vec{x})), \qquad \vec{x}\in\Omega. \tag{2}$$

The constraint violation degree in this paper is measured by the sum of each constraint violation degree.

$$v(\vec{x})=\textstyle\sum_{i=1}^q v_i^I(\vec{x})+\sum_{i=1}^r v_i^E(\vec{x}). \tag{3}$$

$v_i^I(\vec{x})$ is the degree of violating the ith inequality constraint.

$$v_i^I(\vec{x})=\max\{0,g_i^I(\vec{x})\}, \quad i=1,\cdots,q. \tag{4}$$

$v_i^E(\vec{x})$ is the degree of violating the ith equal constraint.

$$v_i^E(\vec{x})=\max\{0,|g_i^E(\vec{x})|-\epsilon\}, \quad i=1,\cdots,r, \tag{5}$$

where ϵ is a user-defined tolerance allowed for the equality constraint.

Numerous multi-objective EAs for constrained optimisation have been proposed in last two decades. Many empirical studies have demonstrated the efficiency of the multi-objective method [4]. But intuitively, the more objectives a problem has, the more complicated it is. Thus, this raises a question why the multi-objective method could be superior to the single objective method. So far few theoretical analyses have been reported for answering this question.

In fact, none of EAs in the latest IEEE CEC 2017 and 2018 constrained optimisation competitions adopted multi-objective optimisation [5]. The competition benchmark suite includes 50 and 100 dimensional functions [5], [6]. For a multi-objective optimisation problem, the higher dimension, the more complex Pareto optimal set. This raises another question whether multi-objective EAs are able to compete with the state-of-art single-objective EAs in the competition. These research questions motivate us to further study the multi-objective method for COPs.

Our work is inspired by helper objectives [7]. A single-objective optimization problem is converted into a multi-objective one by adding helper objectives [7]. The use of helper objectives may significantly improve the performance of EAs for solving combinatorial optimisation problems, such as job shop scheduling, travelling salesman and vertex covering [7], [8]. Our work is also inspired by objective decomposition, which was recently adopted in multi-objective EAs for COPs [9]–[12]. Because the goal of COPs is to seek the optimal feasible solution(s) rather than a Pareto optimal set, decomposition-based multi-objective EAs with biased weights are flexible than those based on Pareto ranking.

This paper presents the equivalent and helper objectives method for COPs. Its idea is to convert a COP into an optimisation problem equivalent to the COP but without equality and inequality constraints, and also add several helper objectives. An equivalent objective means its optimal solution set is identical to Ω^*, but a helper objective not. Then this multi-objective optimisation problem is solved by a decomposition-based multi-objective EA.

Our research hypothesis is that the helper and equivalent objective method might outperform the single objective method on certain hard problems. We conduct both theoretical and empirical comparisons of these two methods.

1) In theory, the "wide gap" problem [13], [14] is regarded as a hard problem to EAs. We aim at proving using helper and equivalent objectives may shorten the hitting time of crossing such a "wide gap".
2) A case study is conducted for validating our theory. We aim at designing an EA with helper and equivalent objectives and demonstrating that it can outperform EAs in CEC 2017 and 2018 competitions.

The paper is organised as follows: Section III is literature review. Section IV describes the helper and equivalent objective method. Section IV theoretically analyses this method. Section V conducts a case study. Section VI reports experiments and results. Section VII concludes the work.

II. Literature Review

Multi-objective EAs have been applied to COPs since 1990s [15], [16]. Segura et al. [4] made a literature survey of the work up to 2016. Thus, this section focuses on reviewing most recent work. Following the taxonomy in [4], [17], a classification of these EAs is built upon the type of objectives.

1) Scheme with two objectives, which are the original objective f and a degree of violating constraints v [9], [11], [12], [18]–[21].
2) Scheme with many objectives, which are the original objective f and degrees of violating each constraint v_i [22], [23].
3) Scheme with other objective(s), for example, the penalty function [24], besides the original objective or the degree of constraint violation [24]–[27].

The first scheme is the most widely used one so far. Ji et al. [28] converted a berth allocation problem with constraints into problem (2) and solved it by a modified non-dominated sorting genetic algorithm II. Ji et al. [29] transformed a COP into problem (2) and solved it by a differential evolution (DE) algorithm. They combined multiobjective optimization with an ϵ-constrained method. X Recently, decomposition-based multi-objective EAs have applied to solving problem (2). u et al. [9] decomposed problem (2) into a tri-objective problem using the weighted sum method with static weights. They solved the multi-objective optimisation problem by a Pareto-ranking based DE algorithm. Wang et al. [12] decomposed problem (2) using the weighted sum method into a number of subproblems with dynamical weights. They solved the subproblems by DE. Peng et al. [11] decomposed problem (2) using the Chebyshev method. Weights are biased and adjusted dynamically for maintaining a balance between convergence and population diversity.

The second scheme converts a COP into a many-objective optimisation problem but is less used. Li et al. [23] solved the many-objective optimization problem by dynamical constraint handling.

The third scheme has an advantage of designing a new objective. Zeng et al. [10] designed a niche-count objective besides the original objective and a constraint-violation objective. The niche-count objective helps maintain population diversity. They applied three different multiobjective EAs (ranking-based, decomposition-based, and hype-volume) to the tri-objective optimisation problem. Jiao et al. [26] converted a COP into a dynamical bi-objective optimisation problem consisting of the original objective and a niche-count objective.

The helper and equivalent objective method proposed in this paper belongs to the third scheme. One objective is designed as an equivalent objective. The equivalent objective has the same optimal set as that to the original COP, but other objectives are also added as helper objectives for searching different directions. Under this framework, we have designed HECO-DE and HECO-PDE [27]. The latter is HECO-DE enhanced with principle component analysis. Experimental results show they were ranked top two when compared with EAs in CEC 2018 constrained optimisation competitions [27].

The theoretical analysis of multi-objective EAs for constrained optimisation is still rare and limited to combinatorial optimisation. He et al. [30] proved that a multi-objective EA with helper objectives is a 1/2-approximation algorithm for the knapsack problem. Recently, Neumann and Sutton [31] analysed the running time of a variant of Global Simple Evolutionary Multiobjective Optimizer on the knapsack problem.

III. The Helper and Equivalent Objective Method

A. Helper and Equivalent Objectives

We start from a problem existing in the classical bi-objective method for solving problem (2). The problem is that the Pareto optimal set to (2) is often significantly larger than Ω^*.

Example 1: Consider the following COP. Its optimal solution is a single point $\Omega^* = \{0\}$.

$$\begin{cases} \min & f(x) = x, \qquad x \in [-1000, 1000], \\ \text{subject to} & g(x) = \sin(\frac{x\pi}{1200}) \geq 0. \end{cases}$$

The degree of constrain violation is

$$v(x) = \max\{0, -\sin(x\pi/1200)\}. \tag{6}$$

The Pareto optimal set to the bi-objective problem $\min(f, v)$ is $\{-1000\} \cup [-200, 0]$, significantly larger than Ω^*. The Pareto front is shown in Fig. 1.

This example shows that using two objectives makes the problem more complicated. Thus, it is hard to explain why the multi-objective method is more efficient.

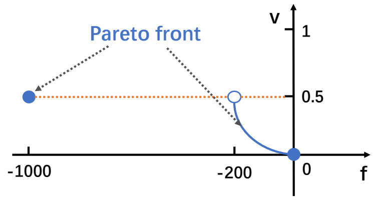

Fig. 1. Pareto front.

In order to develop a theory of understanding the multi-objective method for COPs, we introduce two concepts, equivalent and helper objectives. The term "helper objective" originates from [7].

Definition 1: A scalar function $g(\vec{x})$ defined on Ω is called an equivalent objective function with respect to the COP (1) if it satisfies the condition:

$$\arg\min\{f(\vec{x}); \vec{x} \in \Omega\} = \Omega^*. \tag{7}$$

A scalar function $g(\vec{x})$ is called a helper objective function if it does not satisfy the above condition.

Equivalent functions can be obtained from single objective methods for constrained optimisation. For example, a simple equivalent function is the death penalty function. Let Ω_F denote feasible solutions and Ω_I infeasible ones.

$$e(\vec{x}) = \begin{cases} f(\vec{x}), & \text{if } \vec{x} \in \Omega_F, \\ +\infty, & \text{if } \vec{x} \in \Omega_I. \end{cases} \tag{8}$$

But the objective function f is not an equivalent function unless all optimal solution(s) to $\min f$ are feasible. The constraint violation degree v is not an equivalent function unless all feasible solutions are optimal. Hence, except particular COPs, both f and v are only helper functions and the problem (2) is a two helper objective problem.

In practice, it is more convenient to construct an equivalent function $e(\vec{x})$ which is defined on population P, rather than Ω. In this case, the definition of helper and equivalent functions is modified as follows.

Definition 2: Given a population P such that $\Omega^* \cap P \neq \emptyset$, a scalar function $g(\vec{x})$ defined on P is called an equivalent objective function with respect to the COP (1) if it satisfies the following condition:

$$\arg\min\{f(\vec{x}); \vec{x} \in \Omega \cap P\} = \Omega^* \cap P. \tag{9}$$

A scalar function $g(\vec{x})$ defined on P is called a helper objective function if it does not satisfy the above condition. For a population P such that $\Omega^* \cap P = \emptyset$, we do not distinguish between equivalent and helper functions defined on P.

An example is the superiority of feasibility rule [32] which is described as follows. Given a population P,

1) A feasible solution with a smaller f value is better than one with a larger f value;
2) A feasible solution is better than an infeasible solution;
3) An infeasible solution with smaller constraint violation is better than one with larger constraint violation.

The above rule leads to an equivalent function on P as

$$e(\vec{x}) = \begin{cases} f(\vec{x}), & \text{if } \vec{x} \in \Omega_F \cap P, \\ v(\vec{x}) + f_F(P), & \text{if } \vec{x} \in \Omega_I \cap P, \end{cases} \tag{10}$$

where $f_F(P) = \max\{f(\vec{x}), \vec{x} \in \Omega_F \cap P\}$ if $\Omega_F \cap P \neq \emptyset$ or $f_F(P) = 0$ otherwise.

B. The Helper and Equivalent Objective Method

Once an equivalent objective function is obtained, the COP (1) can be converted to a single-objective optimisation problem without inequality and equality constraints.

$$\min e(\vec{x}), \quad \vec{x} \in P. \tag{11}$$

In practice, an EA generates a population sequence $\{P_t; t = 0, 1, \cdots\}$ and $e(\vec{x})$ relies on population P_t.

A single-objective EA (SOCO) for problem (11) is described as follows.

1: population $P_0 \leftarrow$ initialise a population of solutions;
2: **for** $t = 0, \cdots, T_{\max}$ **do**
3: population $C_t \leftarrow$ generate a population of solutions from P_t subject to a conditional probability $\Pr(C_t \mid P_t)$;
4: $P_{t+1} \leftarrow$ select optimal solution(s) to $\min e(\vec{x}), \vec{x} \in P_t \cup C_t$ and remove repeated solutions.
5: **end for**

$T_{\max}$ denotes the maximum number of generations. $\Pr(C_t \mid P_t)$ is a conditional probability determined by search operator(s). The population size $|P_t|$ is changeable so that P_t is able to contain all found best solutions.

Besides the equivalent function $e(\vec{x})$, we add several helper functions $h_i(\vec{x}), i = 1, \cdots, k$, and then obtain a helper and equivalent objective optimisation problem on population P.

$$\min \vec{f}(\vec{x}) = (e(\vec{x}), h_1(\vec{x}), \cdots, h_k(\vec{x})), \qquad \vec{x} \in P. \tag{12}$$

Furthermore, we decompose problem (12) into several single objective problem. Decomposition-based multi-objective EAs have been proven to be efficient in solving multiobjective optimisation problems [33], [34]. The decomposition method in the present work adopts the weighted sum approach, adding the helper objective onto the equivalent objective such that

$$\min w_0 e(\vec{x}) + \textstyle\sum_{j=1}^k w_j h_j(\vec{x}), \quad \vec{x} \in P, \tag{13}$$

where $w_j \geq 0$ are weights.

Problem (12) is transformed into λ single-objective optimisation subproblems by assigning λ tuples of weights $\vec{w}_i = (w_{0i}, w_{1i}, \cdots, w_{ki})$.

$$\min f_i = w_{0i} e + \textstyle\sum_{j=1}^k w_{ij} h_j, \quad i = 1, \cdots, \lambda. \tag{14}$$

One of f_i must be the equivalent function. We minimise all f_i simultaneously.

Since the ranges of e and h might be significantly different, one of them may play a dominant role in the weighted sum. It is therefore, helpful to normalise the values of each function to $[0, 1]$ so that none of them dominates others in the sum. The min-max normalisation method is adopted within a population P. Given a function $g(\vec{x})$, it is normalised to $[0, 1]$.

$$g(\vec{x}) \leftarrow \frac{g(\vec{x}) - \max_{i \in P} g(\vec{x}_i)}{\max_{i \in P} g(\vec{x}_i) - \min_{i \in P} g(\vec{x}_i)}. \tag{15}$$

A helper and equivalent objective EA (HECO) for problem (14) is described as follows.

population $P_0 \leftarrow$ initialise a population of solutions;
for $t = 0, \cdots, T_{\max}$ **do**
  adjust weights;
  population $C_t \leftarrow$ generate a population of solutions from P_t subject to a conditional probability $\Pr(C_t \mid P_t)$;
  $P_{t+1} \leftarrow$ select optimal solution(s) to $\min f_i(\vec{x}), \vec{x} \in P_t \cup C_t$ for $i = 1, \cdots, \lambda$ where f_i is calculated by formula (14); remove repeated solutions.
end for

HECO selects optimal solution(s) to $\min f_i(\vec{x}), \vec{x} \in P_t \cup C_t$ with respect to each function f_i (called elitist selection), but it does not select all non-dominated solutions with respect to $(e, h_1, \cdots, h_k)$ (no Pareto-based ranking).

Since our goal is to find the optimal solution(s) to $\min e(\vec{x})$ but not to $\min h_i(\vec{x})$, it is not necessary to generate solutions evenly spreading on the Pareto front. Thus, the decomposition mechanism proposed herein differs from that employed in traditional decomposition-based multi-objective EAs [33]. The weights are chosen dynamically over generations t so that each f_i eventually converges to an equivalent objective function. Thus, the adjustment of weights follows the principle:

$$\lim_{t \to +\infty} w_{0i} e(\vec{x}) + \sum_{j=1}^{k} w_{ij} h_j(\vec{x}) = e(\vec{x}). \qquad (16)$$

Compared with SOCO, HECO has two new features:

1) SOCO is one-dimension search along the direction e in the objective space. HECO is multi-dimensional search along several directions $(e, h_1, \cdots, h_k)$. e is the main search direction for SOCO, while $h_1, \cdots, h_k$ are auxiliary directions added by HECO. Intuitively, if SOCO encounters a "wide gap" along the direction e, HECO might bypass it through other auxiliary directions. This initiative discussion will be rigorously analysed in the next section.
2) The dynamically weighting ensures that at the beginning, HECO explores different directions $e, h_1, \cdots, h_k$, while at the end, HECO exploits the direction e for obtaining an optimal feasible solution.

HECO is a general framework which covers many variant algorithm instances. Equivalent and helper functions can be constructed in a different way, such as (8) and (10). Search operators can be chosen from evolutionary strategies, differential evolution, particle swarm optimisation and so on.

C. Implicit Equivalent Objective

Without the help of an equivalent objective, a decomposition-based multi-objective EA for COPs will face a problem. The solution set found by the algorithm is often larger than Ω^*. This claim is shown through Example 1. We assign λ pairs of weights in objective decomposition: $(1,0), (w_i, 1-w_i), (0,1)$ where $i = 2, \cdots, \lambda-1$ and $w_i > 0$ and obtain λ subproblems with a bounded constraint $x \in [-1000, 1000]$.

$$\begin{cases} \min f_1(x) = x, \\ \min f_i(x) = w_i f(x) + (1-w_i) v(x), \\ \min f_\lambda(x) = v(x). \end{cases}$$

The optimal solution to $\min f$ is $x = -1000$. The optimal solution to any $\min w_i f + (1-w_i) v$ is infeasible. The optimal solution to $\min v$ is $[0, 500]$. The solution set to the λ subproblems consists of infinite solutions, much larger than $\Omega^* = \{0\}$. Using dynamical adjustment of weights does not help here.

However, in practice, it is common to utilise the superiority of feasibility rule to select solutions. Using the rule, an infeasible solution such as $x = -1000$ is not selected. Among feasible solutions $x \in [0, 500]$, only the minimal point $x = 0$ is selected. Since the superiority of feasibility rule is an equivalent objective, it implies that many multi-objective EAs for COPs implicitly utilise an equivalent objective (10). Based on this argument, multi-objective EAs for COPs are classified into three types:

1) Scheme which is to optimise helper objectives only;
2) Scheme which is to optimise helper objectives but select solutions by the superiority of feasibility rule (an implcit equivalent objective);
3) Scheme which is to explicitly optimise both helper and equivalent objectives.

In this paper, the notation HECO refers to the third scheme. It has some advantages: an explicit equivalent objective is utilised and the objective can be designed more general beyond the superiority of feasibility rule.

IV. A Theoretical Analysis

A. Preliminary Definitions and Lemma

Intuitively, an equivalent objective ensures a primary search direction towards Ω^* and avoid an enlarged Pareto optimal set. Helper objectives provide auxiliary search directions. If there exists an obstacle like a "wide gap" on the primary direction, auxiliary directions may help bypass it. In theory, we aim at mathematically proving the conjecture: using helper and equivalent objectives might be useful on the "wide-gap" problem. First we introduce several preliminary definitions and a lemma.

For the sake of analysis, the search space Ω is regarded as a finite set. This simplification is made due to two reasons. First, any computer can only represent a finite set of real numbers subject to a precision. Secondly, population P_t consists of finite individuals (points). But the probability of P_t at a single point or several points always equals to 0 in a continuous space Ω. To handle this dilemma, we need to assume that possible values of P_t are finite.

Let $\vec{f}(\vec{x}) = (f_1(\vec{x}), \cdots, f_k(\vec{x}))$ be a scalar function ($k = 1$) or a vector-valued function ($k > 1$). Consider a minimisation problem with bounded constraints:

$$\min \vec{f}(\vec{x}), \qquad \vec{x} \in \Omega. \qquad (17)$$

If $k = 1$, it degenerates into a single-objective problem.

Definition 3: Given the optimisation problem (17), $\vec{f}(\vec{x})$ is said to *dominate* $\vec{f}(\vec{y})$ (written as $\vec{f}(\vec{x}) \succ \vec{f}(\vec{y})$) if

1) $\forall i \in \{1, \cdots, k\} : f_i(\vec{x}) \le f_i(\vec{y})$;
2) $\exists i \in \{1, \cdots, k\} : f_i(\vec{x}) < f_i(\vec{y})$.

If $k = 1$, the two conditions degenerate into one inequality $f(\vec{x}) < f(\vec{y})$.

Based on the domination relationship, the non-dominated set and Pareto optimal set are defined as follows.

Definition 4: A set $S \subset S'$ is called *a non-dominated set in the set* S' if and only if $\forall \vec{x} \in S$, $\forall \vec{y} \in S'$, $\vec{x}$ is not dominated by $\vec{y}$. A set S is called *a Pareto optimal set* if and only if it is a non-dominated set in Ω.

Given a target set, the hitting time is the number of generations for an EA to reach the set [35]. The hitting time of an EA from one set to another is defined as follows.

Definition 5: Let $\{P_t; t = 0, 1, \cdots\}$ be a population sequence of an EA. Given two sets S_1 and S_2, the expected hitting time of the EA from S_1 to S_2 is defined by

$$T(S_2 \mid S_1) := \sum_{t=0}^{+\infty} \Pr(P_0 \subset \overline{S_2}, \cdots, P_t \subset \overline{S_2}),$$

where the notation $\overline{S}$ denotes the complement set of S.

From the definition, it is straightforward to derive a lemma for comparing the hitting time of two EAs.

Lemma 1: Let $\{P_t; t = 0, 1, \cdots\}$ and $\{P'_t; t = 0, 1, \cdots\}$ be two population sequences and S_1 and S_2 two sets such that $S_1 \cap S_2 = \emptyset$. Let $P_0 = P'_0 = S_1$. If for any t,

$$\Pr(P_0{=}S_1{\subset}\overline{S_2},\cdots,P_t{\subset}\overline{S_2}) \geq \Pr(P'_0{=}S_1{\subset}\overline{S_2},\cdots,P'_t{\subset}\overline{S_2}), \tag{18}$$

then $T(S_2 \mid S_1) \geq T'(S_2 \mid S_1)$. Furthermore, if the inequality (18) holds strictly for some t, then $T(S_2 \mid S_1) > T'(S_2 \mid S_1)$.

This lemma provides a criterion to determine whether an EA has a shorter hitting time than another EA. The comparison is qualitative because no estimation of the hitting time is involved. For a quantitative comparison, it is necessary to utilise more advanced tools such as drift analysis [35].

B. Fundamental Theorem

Now we compare SOCO for the single-objective problem (11) and HECO for the helper and equivalent objective problem (14). In order to make a fair comparison, a natural premise is that both EAs use identical search operator(s).

The main purpose of using HECO is to tackle hard problems facing SOCO. Yet, what kind of problems are hard to SOCO? According to [13], [14], hard problems to EAs can be classified into two types: the "wide gap" problem and the "long path" problem. The concept of "wide gap" is established on fitness levels. In the helper and equivalent objective method, the equivalent function $e(\vec{x})$ plays the role of "fitness". In constrained optimisation, function $f(\vec{x})$ is not suitable as "fitness" because the minimum value of f might be obtained by an infeasible solution.

The values of $e(\vec{x})$ are split into fitness levels: $FL_0 < FL_1 < \cdots < FL_m$ and the search space Ω is split into disjoint level sets: $\Omega = \cup_{i=0}^{m} L_i$ where $L = \{\vec{x}; e(\vec{x}) = FL\}$. Given a fitness level FL and its corresponding point set L, let L^b denote points at better levels $L^b := \{\vec{x}; e(\vec{x}) < FL\}$. A "wide gap" between L and L^b is defined as follows.

Definition 6: Given an EA, we say a wide gap existing between L and L^b if for a subset $A \subset L$, the expected hitting time $T(L^b \mid A \subset L)$ is an exponential function of the dimension D.

Several conditions are established for mathematically analysing SOCO and HECO. Let $\{P_t; t = 0, 1, \cdots\}$ represent the population sequence from SOCO and $\{P'_t; t = 0, 1, \cdots\}$ from HECO. Assume $P_0 = P'_0$ are chosen from the fitness level FL. For SOCO, thanks to elitist selection, its offspring are either at the level FL or better fitness levels. For HECO, because of selection on both equivalent and helper function directions, offspring may include points from worse fitness levels too. This observation is summarised as a condition.

Condition 1: Assume that $P_0 = P'_0 \subset L$. For SOCO, $P_t \subset L \cup L^b$ for ever. Provided that $P_t = X = (\vec{x}_1, \cdots, \vec{x}_m) \subset L$, there is a one to many mapping from P_t to P'_t where P'_t is in the set

$$Map(X){=}\{X'{=}(\vec{x}_1,\cdots,\vec{x}_m,*) \mid *{=}\emptyset \text{ or } *{\subset}\overline{L{\cup}L^b}\}.$$

The event of $P_t = (\vec{x}_1, \cdots, \vec{x}_m) \subset L$ requires $\vec{x}_1 \in L, \cdots, \vec{x}_m \in L$. The probability of this event happening is larger than that of the event $P'_t = (\vec{x}_1, \cdots, \vec{x}_m, *)$ where $* = \emptyset$ or $* \subset \overline{L \cup L^b}\}$ because the latter event requires $\vec{x}_1 \in L, \cdots, \vec{x}_m \in L$ and also $* \subset \overline{L \cup L^b}$. This leads to the following conditions.

Condition 2: Let $P_0 = P'_0 = A \subset L$. For any t, it holds

$$\begin{aligned}&\Pr(P_0{=}A{\subset}L,\cdots,P_t{=}Z{\subset}L)\\ \geq &\textstyle\sum_{*\subset\overline{L^b}}\cdots\sum_{*\subset\overline{L^b}} \Pr(P'_0{=}A'{\subset}\overline{L^b},\cdots,P'_t{=}Z'{\subset}\overline{L^b}).\end{aligned}$$

Condition 3: For some t, the above inequality is strict.

Thanks to elitist selection and equivalent objective, Conditions 1 and 2 are always true. Condition 3 could be true, for example, if the transition probability from $*$ to L^b is greater than 0. Using the above conditions, we prove a fundamental theorem of comparing HECO and SOCO.

Theorem 1: Consider SOCO for the single objective problem (11) and HECO for the helper and equivalent objective problem (14) using elitist selection and identical search operator(s). Assume that SOCO faces a wide gap, that is, $T(L^b \mid A \subset L)$ is an exponential function of D for a subset A. Let initial population $P_0 = P'_0 = A$. Under Conditions 1 and 2, the expected hitting time $T(L^b \mid A) \geq T'(L^b \mid A)$. Furthermore, under Condition 3, $T(L^b \mid A) > T'(L^b \mid A)$.

Proof: From Conditions 1 and 2, it follows for any t,

$$\begin{aligned}&\Pr(P_0{\subset}\overline{L^b},\cdots,P_t{\subset}\overline{L^b}){=}\textstyle\sum_{A\subset L}\cdots\sum_{Z\subset L}\Pr(P_0{=}A,\cdots,P_t{=}Z)\\ \geq &\Pr(P'_0{\subset}\overline{L^b},\cdots,P'_t{\subset}\overline{L^b})\\ = &\textstyle\sum_{A\subset L}\cdots\sum_{Z\subset L}\sum_{*\subset\overline{L^b}}\cdots\sum_{*\subset\overline{L^b}}\Pr(P_0{=}A,\cdots,P_t{=}Z').\end{aligned} \tag{19}$$

From Lemma 1, it is known $T(L^b \mid A) \geq T'(L^b \mid A)$. The second conclusion is drawn from Condition 3. ■

Theorem 1 proves that the hitting time of HECO crossing a wide gap is not more than SOCO under Conditions 1 and 2 (always true) and shorter than SOCO under Condition 3 (sometimes true). In Conditions 2 and 3, the part $* \cdots *$ is a path of searching along helper directions and intuitively is regarded as a bypass over the wide gap. Theorem 1 reveals if such a bypass exists, HECO may shorten the hitting time of crossing the wide gap. Nevertheless, Theorem 1 is inapplicable to the multi-helper objective method, because the one-to-many mapping in Condition 1 cannot be established.

Example 2: Consider the COP below,

$$\begin{cases} \min \quad f(x) = x, \qquad x \in [-500, 3000] \\ \text{subject to} \quad g(x) = \sin(\frac{x\pi}{1000}) \geq 0. \end{cases} \tag{20}$$

Its optimal solution is $x = 0$. The feasible region is $\Omega_F = [0, 1000] \cup [2000, 3000]$. The objective function $f(\vec{x})$ is not an equivalent function because its minimal point is $x = -500$, an infeasible solution.

First, we analyse a SOCO algorithm using elitist selection and the equivalent objective from the superiority of feasibility rule.

$$\min e(x) = \begin{cases} f(x), & \text{if } x \in \Omega_F, \\ v(x) + 3000, & \text{if } x \in \Omega_I. \end{cases} \tag{21}$$

where $v(x) = \max\{0, -\sin(\frac{x\pi}{1000})\}$.

Mutation is $y = x + U(-1, 1)$, where x is the parent and y its child. $U(-1, 1)$ is a uniform random number in $(-1, 1)$.

Assume that SOCO starts at $L = \{2000\}$. Then $L^b = [0, 1000]$. Because of elitist selection, the EA cannot accept a worse solution. Then it cannot cross the infeasible region $(1000, 2000)$, a wide gap to SOCO. Thus, $P_t \in L$ for ever and cannot crossover the wide gap.

Secondly, we analyse a HECO algorithm employing elitist selection, identical mutation but two objectives.

$$\min \vec{f}(x) = (e(x), f(x)), \qquad x \in [-500, 3000]. \tag{22}$$

Its Pareto front is displayed in Fig. 2.

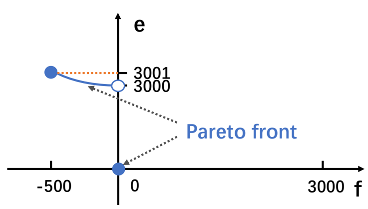

Fig. 2. Pareto front to the two-objective optimisation problem (22)

We assign two pairs of weights: $\vec{w}_1 = (1, 0)$ and $\vec{w}_2 = (0, 1)$ on (e, f). Assume that SOCO starts at $L = \{2000\}$. For any $x \in P_t \cap [1000, 2000]$, after mutation, some point y such that $y < x - \frac{1}{2}$ is generated with a positive probability. Since $f(y) < f(x)$, y is selected to P'_t. Thus, P'_t makes a downhill-search along the direction f. Repeating this procedure for 2000 generations, P_t can reach the set $L^b = [0, 1000]$ with a positive probability. This implies for $t \geq 2000$,

$$\Pr(P'_0 \subset \overline{L^b}, \cdots, P'_t \subset \overline{L^b}) < 1.$$

According to Theorem 1, $T'(L^b \mid L) < T(L^b \mid L)$. Fig. 3 visualises the bypass in the objective space.

V. A Case Study

A. Search Operators from LSHADE44

In order to validate our theory, we must construct a HECO algorithm from a SOCO algorithm such that their search operators are identical but one is with a single objective and the other with helper and equivalent objectives. For comparative purpose, LSHADE44 [36] is chosen as the SOCO algorithm because it is ranked only 4th in the CEC 2017/18 competition [5]. If the constructed HECO algorithm outperforms

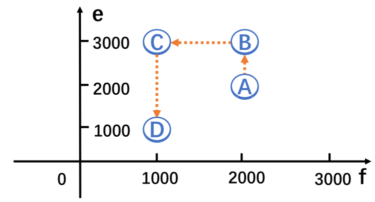

Fig. 3. A bypass in objective space: $A(2000, 2000) \to B(2000-\epsilon_1, 3000+\epsilon_2) \to C(1000-\epsilon_3, 3000+\epsilon_4) \to D(1000, 1000)$ where $\epsilon_i \in (0, 1)$ over the wide gap between fitness levels $e(x) = 2000$ and $e(x) = 1000$.

LSHADE44 and also top three EAs in the competition, then we have a good reason to claim the helper and equivalent objective method works.

For the sake of a self-contained presentation, search operators in LSHADE44 are summarised as follows.

LSHADE44 employs two mutation operators. The first one is current-to-pbest/1 mutation (see (6) in [37]). Mutant point $\vec{u}_i$ is generated from target point $\vec{x}_i$ by

$$\vec{u}_i = \vec{x}_i + F(\vec{x}_{pbest} - \vec{x}_i) + F(\vec{x}_{r_1} - \vec{x}_{r_2}), \tag{23}$$

where $\vec{x}_{pbest}$ is chosen at random from the top $100p\%$ of population P where $p \in (0, 1)$. $\vec{x}_{r_1}$ is chosen at random from population P, while $\vec{x}_{r_2}$ at random from $P \cup A$ where A represents an archive. Mutation factor $F \in (0, 1)$.

The second mutation is randrl/1 mutation (see (3) in [38]).

$$\vec{u}_i = \vec{x}_{r_1} + F(\vec{x}_{r_2} - \vec{x}_{r_3}), \tag{24}$$
$$\vec{u}_i = \vec{x}_{r_1^*} + F(\vec{x}_{r_2^*} - \vec{x}_{r_3^*}). \tag{25}$$

In (24), mutually distinct $\vec{x}_{r_1}$, $\vec{x}_{r_2}$ and $\vec{x}_{r_3}$ are randomly chosen from population P. They are also different from $\vec{x}_i$. In (25), $\vec{x}_{r_1}$, $\vec{x}_{r_2}$ and $\vec{x}_{r_3}$ are chosen as that in (24) but then are ranked. $\vec{x}_{r_1^*}$ denotes the best, while $\vec{x}_{r_2^*}$ and $\vec{x}_{r_3^*}$ denote the other two.

LSHADE44 employs two crossover operators. The first one is binomial crossover (see (4) in [39]). Trial point $\vec{y}_i$ is generated from target point $\vec{x}_i$ and mutant $\vec{u}_i$ by

$$y_{i,j} = \begin{cases} u_{i,j}, & \text{if} \quad rand_j(0,1) \leq CR \text{ or } j = j_{rand}, \\ x_{i,j}, & \text{otherwise}, \end{cases} \tag{26}$$

where integer j_{rand} is chosen at random from $[1, D]$. $rand_j(0, 1)$ is chosen at random from $(0, 1)$. Crossover rate $CR \in [0, 1]$.

The second crossover is the exponential crossover (see (3) in [40]).

The combination of a mutation operator and a crossover operator forms a search strategy. Thus, four search strategies (combinations) can be produced. LSHADE44 employs a mechanism of competition of strategies [41], [42] to create trial points. The kth strategy is chosen subject to a probability q_k. All q_k are initially set to the same value, i.e., $q_k = 1/4$. The kth strategy is considered successful if a generated trial point y is better than the original point x. The probability q_k is adapted according to its success counts:

$$q_k = \frac{n_k + n_0}{\sum_{i=1}^{4}(n_i + n_0)}, \tag{27}$$

where n_k is the count of the kth strategys successes, and $n_0 > 0$ is a constant.

LSHADE44 adapts parameters F and CR in each strategy based on previous successful values of F and CR [36]. Each strategy has its own pair of memories MF and MC for saving F and CR values. The size of a historical memory is H.

LSHADE44 uses an archive A for the current-to-pbest/1 mutation [36]. The maximal size of archive A is set to $|A|_{\max}$. At the beginning of search, the archive is empty. During a generation, each point which is rewritten by its successful trial point is stored into the archive. If the archive size exceeds the maximum size $|A|_{\max}$, then $|A| - |A|_{\max}$ individuals are randomly removed from A.

LSHADE44 takes a mechanism to linearly decrease the population size [36], [43]. For population P_t, its size must equal to a required size N_t. Otherwise its size is reduced. The required initial size is set to N_0 and the finial size to $N_{T_{\max}}$. The required size at the tth generation is set by the formula:

$$N_t = round\left(N_0 - \tfrac{t}{T_{\max}}(N_0 - N_{T_{\max}})\right). \tag{28}$$

If $|P_t| > N_t$, then $|P_t| - N_t$ worst individuals are deleted from the population.

B. A New Equivalent Objective Function

Two equivalent functions (8) and (10) have been constructed from the death penalty method and the superiority of feasibility rule respectively. However, measured by these functions, a feasible solution always dominates any infeasible one. To reduce the effect of such heavily imposed preference of feasible solutions, we construct a new equivalent function.

Let $x^*(P)$ be the best individual's fitness in population P,

$$x^*(P) = \begin{cases} \arg\min\{v(\vec{x}); \vec{x} \in P\}, & \text{if } P \cap \Omega_F = \emptyset, \\ \arg\min\{f(\vec{x}); \vec{x} \in P \cap \Omega_F\}, & \text{if } P \cap \Omega_F \neq \emptyset. \end{cases}$$

Let $\tilde{e}(\vec{x})$ denote the fitness difference between a point $f(\vec{x})$ and $f(x^*(P))$.

$$\tilde{e}(\vec{x}) = |f(\vec{x}) - f(x^*(P))| \tag{29}$$

$\tilde{e}$ is not an equivalent function because the fitness of an infeasible solution sometimes equals to $f^*(P)$ too. An equivalent function on population P is defined as

$$e(\vec{x}) = w_1 \tilde{e}(\vec{x}) + w_2 v(\vec{x}), \tag{30}$$

where $w_1, w_2 \geq 0$ are weights. The number of such equivalent functions is infinite.

Theorem 2: Function $e(\vec{x})$ given by (30) is an equivalent objective function for any weights $w_1 > 0, w_2 > 0$.

Proof: Given a P satisfying $\Omega^* \cap P \neq \emptyset$, $\min e(\vec{x}) = 0$. On one hand, for any $\vec{x} \in \Omega^* \cap P$, it holds $e(\vec{x}) = 0$. On the other hand, for $\vec{x} \in P$ such that $e(\vec{x}) = 0$, it holds $v(\vec{x}) = 0$, then $\vec{x} \in \Omega^*$. ■

According to (30), an infeasible solution $\vec{x}$ might be better than a feasible solution $\vec{y}$ in terms of e if they satisfy the condition

$$w_1 \tilde{e}(\vec{x}) + w_2 v(\vec{x}) < \tilde{e}(\vec{y}).$$

We may adjust weights w_1, w_2 to control the contribution of $\tilde{e}$ and v to the equivalent function e. This feature may help search the infeasible region.

We choose f as a helper function and then obtain a problem with helper and equivalent objectives.

$$\min \vec{f}(\vec{x}) = (e(\vec{x}), f(\vec{x})), \qquad \vec{x} \in P, \tag{31}$$

The problem is decomposed into λ single objective subproblems through the weighted sum method: for $i = 1, \cdots, \lambda$,

$$\min f_i(\vec{x}) = w_{1i}\tilde{e}(\vec{x}) + w_{2i} v(\vec{x}) + w_{3i} f(\vec{x}). \tag{32}$$

An extra term $\tilde{e}$ is added besides the original objective function f and constraint violation degree v. So problem (31) is equivalent to a triple-objective optimisation problem.

$$\min \vec{f}(\vec{x}) = (\tilde{e}(\vec{x}), f(\vec{x}), v(\vec{x})), \quad \vec{x} \in P, \tag{33}$$

where the weighted sum of e and f forms an equivalent function. The weighted sum of f and v forms a penalty function but may not be an equivalent function.

C. A New multi-objective EA for Constrained Optimisation

A HECO algorithm is designed which reuses search operators from LSHADE44 [36]. We call it HECO-DE because it is built upon HECO and DE. Different from the single-objective method LSHADE44, HECO-DE has three new multi-objective features: helper and equivalent objectives, objective decomposition and dynamical adjustment of weights. The procedure of HECO-DE is described in detail as below.

Initialise algorithm parameters, including the required initial population sizes N_0 and final size $N_{T_{\max}}$, the maximum number of fitness evaluations $FES_{\max}$, circle memories for parameters F and CR, the size of historical memories H; initial probabilities q_k of four strategies, and external archive A;
Set the counter of fitness evaluations FES to 0, and the counter of generations t to 0;
Randomly generate N_0 solutions and form an initial population P_0;
Evaluate the value of $f(\vec{x})$ and $v(\vec{x})$ for each $\vec{x} \in P_0$;
Increase counter FES by N_0;
while $FES \leq FES_{\max}$ (or $t \leq T_{\max}$) **do**
    Adjust weights in objective decomposition.
    Assign sets S_F and S_{CR} to $\emptyset$ for each strategy. The sets are used to preserve successful values of F and CR for each search strategy respectively. The set C (used for saving children population) is also set to $\emptyset$.
    Randomly select λ individuals (denoted by Q) from P and then denote the rest individuals $P \setminus Q$ by P';
    for x_i in Q, $i = 1, \ldots, \lambda$ **do**
        Select one strategy (say k) with probability q_k and generate mutation factor F and crossover rate CR from respective circle memories;

12: Generate a trail point $\vec{y_i}$ by applying the selected strategy;
13: Evaluate the value of $f(\vec{y_i})$ and $v(\vec{y_i})$;
14: Add $\vec{y_i}$ to subpopulation Q, resulting in an enlarged subpopulation Q';
15: Normalise $\tilde{e}(\vec{x})$, $f(\vec{x})$ and $v(\vec{x})$ for each individual $\vec{x}$ in Q'.
16: Calculate f_i value for $\vec{x}_i$ and $\vec{y_i}$ according to formula (32).
17: **if** $f_i(\vec{y_i}) < f_i(\vec{x_i})$ **then**
18: Add $\vec{y_i}$ into children C and $\vec{x}_i$ into archive A;
19: Save values of F and CR into respective sets S_F and S_{CR} and increase respective success count;
20: **end if**
21: **end for**
22: Update circle memories M_F and M_{CR} using respective sets S_F and S_{CR} for each strategy (see its detail in LSHADE44 [36]);
23: Merge subpopulation P' (not involved in mutation and crossover) and children C and form new population P;
24: Calculate the required population size N_t;
25: **if** $N_t < |P|$ **then**
26: Randomly delete $|P| - N_t$ individuals from P;
27: **end if**
28: Calculate the required archive size $|A|_{\max} = 4N_t$;
29: **if** $|A| > |A|_{\max}$ **then**
30: Randomly delete $|A| - |A|_{\max}$ individuals from archive A;
31: **end if**
32: Increase counter FES by λ and counter t by 1;
33: **end while**

There are several major differences between HECO-DE and LSHADE44 which are listed as below.

Lines 12: in HECO-DE, mutation is applied to subpopulation Q, rather than the whole population P. Thus, current-to-pbest/1 mutation and randr1/1 mutation must be modified because the ranking of individuals is restricted to subpopulation Q. Given target x_i and subpopulation Q, x_{Qbest} is chosen to be the individual in Q with the lowest value of $f_i(\vec{x})$. Hence, current-to-pbest/1 mutation (23) is modified as

$$\vec{u}_i = \vec{x}_i + F_k(\vec{x}_{Qbest} - \vec{x}_i) + F_k(\vec{x}_{r_1} - \vec{x}_{r_2}), \tag{34}$$

This new mutation is called current-to-Qbest/1 mutation. For randr1/1 mutation (25), $\vec{x}_{r_1}$, $\vec{x}_{r_2}$ and $\vec{x}_{r_3}$ are not compared but just randomly selected from subpopulation Q. Thus it returns to the original rand/1 mutation (24).

Lines 12 and 16: ranking individuals is used in both mutation (23) and calculation of the equivalent function (30). Because ranking is restricted within subpopulation Q and its size λ is a small constant, the time complexity of ranking is a constant. This is different from LSHADE44 in which individuals in the whole population P are ranked. Its time complexity if a function of dimension D.

Lines 17-20: if $f_i(\vec{y_i}) < f_i(\vec{x_i})$, then $\vec{y_i}$ is accepted and added into children population C. HECO-DE minimises λ functions f_i simultaneously. In *Line 7*, the weights on each f_i are dynamically adjusted (detail in Subsection V-D). This is the most important difference from LSHADE44.

Since λ is a small constant, the number of operations in HECO-DE is only changed by a constant when compared with LSHADE44. Thus, the time complexity of HECO-DE in each generation is the same as LSHADE44 [36].

D. A New Mechanism of Dynamical Adjustment of Weights

We propose a special mechanism for dynamically adjusting weights. Function f_i in subproblem (32) is a weighted sum of helper and equivalent functions:

$$f_i(\vec{x}) = w_{1i}\tilde{e}(\vec{x}) + w_{2i}v(\vec{x}) + w_{3i}f(\vec{x}), \tag{35}$$

where w_{1i}, w_{2i}, w_{3i} are the weights on functions $\tilde{e}, v$ and f respectively. Weights are adjusted according to the following principle: each f_i converges to an equivalent function. Thus,

$$\lim_{t\to+\infty} w_{1i,t} > 0, \lim_{t\to+\infty} w_{2i,t} > 0, \lim_{t\to+\infty} w_{3i,t} = 0.$$

In HECO-DE, weights are designed to linearly increase (for w_1, w_2) or decrease (for w_3) over t and also linearly increase (for w_1, w_2) or decrease (for w_3) over i. In more detail, weights are given by

$$w_{1i,t} = \frac{t}{T_{\max}} \cdot \frac{i}{\lambda}, \tag{36}$$

$$w_{2i,t} = \frac{t}{T_{\max}} \cdot \frac{i}{\lambda} + \gamma, \tag{37}$$

$$w_{3i,t} = \left(1 - \frac{t}{T_{\max}}\right) \cdot \left(1 - \frac{i}{\lambda}\right), \tag{38}$$

where λ is the number of subproblems. $i = 1, 2, \dots, \lambda$. $T_{\max}$ is the maximal number of generations. $\gamma \in (0, 1)$ is a bias constant which is linked to the number of constraints. The more constraints, the larger γ and w_2.

Figures 4 and 5 depict the change of normalised weights over $t/T_{\max}$. For λth individual, weights $w_{1\lambda} > 0, w_{2\lambda} > 0$ but $w_{3\lambda} = 0$,. This individual minimises an equivalent function f_λ. For 1st individual, weight w_{31} initially is set to a large value. Thus, at the beginning of search, this individual focuses on minimising a helper function f_1. Subsequently w_{31} decreases to 0. It turns to minimise an equivalent function f_1 at the end of search.

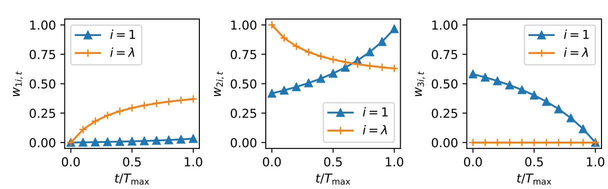

Fig. 4. The change of weights for 1st and λth individuals on CEC 2006 benchmark functions. $\gamma = 0.7$.

VI. Comparative Experiments and Results

A. Experimental Settings

HECO-DE was tested on two well-known benchmark sets. The first set is from IEEE CEC 2017 Competition and Special Session on Constrained Single Objective Real-Parameter Optimization [6] which consists of 28 scalable functions with

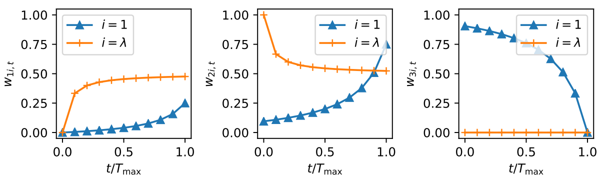

Fig. 5. The change of weights for 1st and λth individuals on CEC 2017 benchmarks. $\gamma = 0.1$.

dimension $D = 10, 30, 50, 100$ (total 4×28 benchmarks). The second set is from the IEEE CEC2006 Special Session on Constrained Real-parameter Optimization [44] which consists of 24 functions with their dimension. According to [44], there is no feasible solutions for function g20 and it is extremely difficult to find the optimum of function g22. Thus, these two functions are excluded in the comparison.

Tables I and II list the parameter setting on CEC 2017 and 2006 benchmarks respectively. Parameters from LSHADE44 are set to similar values to LSHADE44 [36]. Population size N_0, number of subproblems λ and constraint violation bias γ are set to different values on the two benchmark sets. Since CEC 2006 benchmarks include more constraints, its γ value is set higher than that in CEC 2017 benchmarks. As same as the competitions, twenty five independent runs were taken on each benchmarks.

TABLE I
PARAMETER SETTING ON IEEE CEC2006 BENCHMARKS

maximal number of fitness evaluations	$FES_{\max} = 500,000$
required population sizes	$N_0 = 450$, $N_{T_{\max}} = \lambda$
population size of Q	$\lambda = 45$
constraint violation bias	$\gamma = 0.7$
historical memory size	$H = 5$
number of strategies	$K = 4$
constant in strategy adaption	$n_0 = 2$
threshold in strategy adaption	$\delta = 1/20$
the maximum size of archive A	$\lvert A \rvert_{\max} = 4N_t$
tolerance for equivalent constraints	$\sigma = 0.0001$

TABLE II
PARAMETER SETTING ON IEEE CEC2017 BENCHMARKS

number of fitness evaluations	$FES_{\max} = 20000D$
required population sizes	$N_0 = 12 \times D$, $N_{T_{\max}} = \lambda$
population size of Q	$\lambda = 20$
constraint violation bias	$\gamma = 0.1$
historical memory size	$H = 5$
number of strategies	$K = 4$
constant in strategy adaption	$n_0 = 2$
threshold in strategy adaption	$\delta = 1/20$
the maximum size of archive A	$\lvert A \rvert_{\max} = 4N_t$
tolerance for equivalent constraints	$\sigma = 0.0001$

B. Experimental results on IEEE CEC2017 benchmarks

HECO-DE was compared with seven single-objective EAs in CEC 2017/18 constrained optimisation competitions [5], which are CAL-SHADE [45], LSHADE44+IDE [46], LSHADE44 [36], UDE [47], MA-ES [48], IUDE [49], LSHADE-IEpsilon [50], and one decomposition-based multi-objective EA, DeCODE [12].

HECO-DE was also compared with its two variants. The first variant is to remove the equivalent function from HECO-DE. In the tri-objective problem (33), $\tilde{e}(\vec{x})$ is replaced by $f(\vec{x})$. We call it HCO-DE. The second variant is to choose the superiority of feasibility rule as the equivalent function. In the tri-objective problem (33), $\tilde{e}(\vec{x})$ is replaced by $e(\vec{x})$ given by (10). We call it HECO-DE(FR). The three algorithms adopt same parameter setting.

According to the CEC 2018 competition rules [5], EAs under comparison were ranked on the experimental results against the use of 28 benchmarks under $D = 10, 30, 50, 100$, in terms of the mean values and median solution. All results were compared at the precision level of $1e-8$ in the same way as the official ranking source code [5], [6]. The rank value of each algorithm on each dimension was calculated as below:

$$\begin{aligned} \text{Rank value} = & \textstyle\sum_{i=1}^{28} \text{rank}_i(\text{by mean value}) \\ & + \textstyle\sum_{i=1}^{28} \text{rank}_i(\text{by median solution}). \end{aligned} \tag{39}$$

The total rank value is the sum of rank values on four dimensions.

Table III summarises the ranks of EAs on four dimensions and total ranks. HECO-DE is the top-ranked amongst all compared. This result clearly demonstrates that HECO-DE consistently outperforms other EAs on all dimensions. Without the equivalent function, HCO-DE is worse than HECO-DE and HECO-DE(FR). HECO-DE(FR) which uses the superiority of feasibility rule as the equivalent objective is slightly worse than HECO-DE. Tables IV and V provide a sensitivity analysis of parameters λ and γ. HECO-DE with all five λ and γ values had obtained lower total ranks than other EAs.

Due to the paper length restriction, more experimental results is provided in the supplement.

TABLE III
TOTAL RANKS OF HECO-DE AND OTHER EAS ON IEEE CEC2017 BENCHMARKS

Algorithm/Dimension	$10D$	$30D$	$50D$	$100D$	Total
CAL_LSAHDE(2017)	421	420	469	478	1788
LSHADE44+IDE(2017)	310	394	422	392	1518
LSAHDE44(2017)	332	344	342	342	1360
UDE(2017)	341	372	377	438	1528
MA_ES(2018)	271	261	273	282	1087
IUDE(2018)	198	261	269	327	1055
LSAHDE_IEpsilon(2018)	222	278	324	372	1196
DeCODE(2018)	239	297	302	328	1166
HCO-DE	282	253	255	219	1009
HECO-DE(FR)	158	194	186	**202**	740
HECO-DE	**154**	**139**	**156**	205	**654**

C. Experimental results on IEEE CEC2006 benchmarks

HECO-DE was compared with five EAs, which are CMODE [20], NSES [51], FROFI [52], DW [11] and DeCODE [12], on IEEE CEC2006 benchmarks.

Table VI summarises experiment results, where "Mean" and "Std Dev" denote the mean and standard deviation of objective function values, respectively. As suggested in [44], a successful run is a run during which an algorithm finds

TABLE IV
TOTAL RANKS OF HECO-DE WITH VARYING λ AND OTHER EAS ON IEEE CEC2017 BENCHMARKS

Algorithm/Dimension	$10D$	$30D$	$50D$	$100D$	Total
CAL_LSAHDE(2017)	507	508	569	582	2166
LSHADE44+IDE(2017)	381	486	524	483	1874
LSAHDE44(2017)	409	431	431	422	1693
UDE(2017)	431	479	480	537	1927
MA_ES(2018)	326	321	341	347	1335
IUDE(2018)	250	343	345	424	1362
LSAHDE_IEpsilon(2018)	277	354	420	472	1523
DeCODE(2018)	301	381	390	410	1482
HECO-DE($\lambda = 15$)	**172**	199	218	261	850
HECO-DE($\lambda = 20$)	194	**149**	**181**	242	**766**
HECO-DE($\lambda = 25$)	177	174	197	241	789
HECO-DE($\lambda = 30$)	195	192	204	**210**	801
HECO-DE($\lambda = 35$)	189	208	200	222	819

TABLE V
TOTAL RANKS OF HECO-DE WITH VARYING γ VALUES AND OTHER EAS ON IEEE CEC2017 BENCHMARKS

Algorithm/Dimension	$10D$	$30D$	$50D$	$100D$	Total
CAL_LSAHDE(2017)	508	511	572	583	2174
LSHADE44+IDE(2017)	373	485	518	482	1858
LSAHDE44(2017)	405	428	427	422	1682
UDE(2017)	423	471	465	532	1891
MA_ES(2018)	329	320	334	349	1332
IUDE(2018)	249	317	315	419	1300
LSAHDE_IEpsilon(2018)	276	341	415	475	1507
DeCODE(2018)	296	362	370	398	1426
HECO-DE($\gamma = 0.0$)	254	207	243	287	991
HECO-DE($\gamma = 0.1$)	186	**177**	**186**	234	**783**
HECO-DE($\gamma = 0.2$)	**182**	186	197	223	788
HECO-DE($\gamma = 0.3$)	190	220	229	**210**	849
HECO-DE($\gamma = 0.4$)	209	262	283	261	1015

a feasible solution $\vec{x}$ satisfying $f(\vec{x}_{best}) - f(\vec{x}^*) \leq 0.0001$, where $f(\vec{x}_{best})$ is the best solution found by the algorithm and $f(\vec{x}^*)$ is the optimum. In Table VI, "*" denotes that the algorithm satisfies this successful rule in 25 runs for a test problem.

As shown in Table VI, the performance of HECO-DE is similar to NSES, FROFI, DeCODE, which can always find optimum of all test problems. HECO-DE performs better than CMODE and DW. CMODE cannot find the optimum of problem g21 and DW cannot find the optimum of g17 with 100% success rate.

HECO-DE was also compared with HCO-DE and HECO-DE(FR) on four functions g02, g10, g21, and g23. As we can see in Table VII, HECO-DE always find the optimum on all test functions. Due to lack of an equivalent objective, HCO-DE has a lower success rate or feasible rate. HECO-DE(FR) faces performance degradation on g10, g21, and g23, probably because the superiority of feasibility rule has a higher selection pressure than the equivalent function (30).

VII. CONCLUSIONS

This paper proposes the helper and equivalent objective method for constrained optimisation. It is theoretically proven that for a hard problem called "wide gap", using helper and equivalent objectives may shorten the time of crossing a "wide gap". To the best of our knowledge, this might be the first general theoretical result to show the strengths of multi-objective EAs in performing COPs.

A case study is conducted for validating our method. An algorithm, called HECO-DE, is implemented which employs helper and equivalent objectives and reuses search operators from LSHADE44 [36]. A new equivalent function and a new mechanism of dynamically weighting are used in HECO-DE. Experimental results show that the overall performance of HECO-DE is ranked first when compared with LSHADE44, other EAs in the CEC 2017/18 competition [5] and DeCODE [12]. HECO-DE also performs well on IEEE CEC2006 benchmarks. This case study proves the efficiency of the helper and equivalent objective method for constrained optimisation.

For future work, we will design multi-objective EAs for constrained optimisation using different equivalent and helper objectives and search operators.

REFERENCES

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

[2] C. A. Coello 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.

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

[4] C. Segura, C. A. C. Coello, G. Miranda, and C. León, "Using multi-objective evolutionary algorithms for single-objective constrained and unconstrained optimization," *Annals of Operations Research*, vol. 240, no. 1, pp. 217–250, 2016.

[5] P. Suganthan. (2018) The CEC 2018 competition on constrained real-parameter optimization. Accessed on 1 May 2019. [Online]. Available: http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/SharedDocuments/CEC-2018

[6] ——. (2017) The CEC 2017 competition on constrained real-parameter optimization. [Online]. Available: https://www.researchgate.net/publication/317228117

[7] M. T. Jensen, "Helper-objectives: Using multi-objective evolutionary algorithms for single-objective optimisation," *Journal of Mathematical Modelling and Algorithms*, vol. 3, no. 4, pp. 323–347, 2004.

[8] T. Friedrich, J. He, N. Hebbinghaus, F. Neumann, and C. Witt, "Analyses of simple hybrid algorithms for the vertex cover problem," *Evolutionary Computation*, vol. 17, no. 1, pp. 3–19, 2009.

[9] T. Xu, J. He, C. Shang, and W. Ying, "A new multi-objective model for constrained optimisation," in *Advances in Computational Intelligence Systems: the 16th UK Workshop on Computational Intelligence*, P. Angelov, A. Gegov, C. Jayne, and Q. Shen, Eds. Springer, 2017, pp. 71–85.

[10] S. Zeng, R. Jiao, C. Li, X. Li, and J. S. Alkasassbeh, "A general framework of dynamic constrained multiobjective evolutionary algorithms for constrained optimization," *IEEE transactions on Cybernetics*, vol. 47, no. 9, pp. 2678–2688, 2017.

[11] C. Peng, H.-L. Liu, and F. Gu, "A novel constraint-handling technique based on dynamic weights for constrained optimization problems," *Soft Computing*, vol. 22, no. 12, pp. 3919–3935, 2018.

[12] B.-C. Wang, H.-X. Li, Q. Zhang, and Y. Wang, "Decomposition-based multiobjective optimization for constrained evolutionary optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018.

[13] J. He and X. Yao, "Towards an analytic framework for analysing the computation time of evolutionary algorithms," *Artificial Intelligence*, vol. 145, no. 1-2, pp. 59–97, 2003.

[14] T. Chen, J. He, G. Chen, and X. Yao., "Choosing selection pressure for wide-gap problems," *Theoretical Computer Science*, vol. 411, no. 6, pp. 926–934, 2010.

[15] P. D. Surry and N. J. Radcliffe, "The COMOGA method: constrained optimisation by multi-objective genetic algorithms," *Control and Cybernetics*, vol. 26, pp. 391–412, 1997.

TABLE VI

COMPARATIVE EXPERIMENT RESULTS ON IEEE CEC2006 BENCHMARKS. * DENOTES THE NUMBER OF SATISFYING SUCCESSFUL RULE

	CMODE Mean±Std Dev	NSES Mean±Std Dev	DW Mean±Std Dev	FROFI Mean±Std Dev	DeCODE Mean±Std Dev	HECO-DE Mean±Std Dev
g01	-1.5000E+01±0.00E+00*	-1.5000E+01±4.21E-30*	-1.5000E+01±5.02E-14*	-1.5000E+01±0.00E+00*	-1.5000E+01±0.00E+00*	-1.5000E+01±0.00E+00*
g02	-8.0362E-01±2.42E-08*	-8.0362E-01±2.41E-32*	-8.0362E-01±9.99E-08*	-8.0362E-01±1.78E-07*	-8.0362E-01±3.12E-09*	-8.0362E-01±1.21E-06*
g03	-1.0005E+00±5.29E-10*	-1.0005E+00±5.44E-19*	-1.0005E+00±4.27E-12*	-1.0005E+00±4.49E-16*	-1.0005E+00±4.00E-16*	-1.0005E+00±3.54E-09*
g04	-3.0666E+04±2.64E-26*	-3.0666E+04±2.22E-24*	-3.0666E+04±0.00E+00*	-3.0666E+04±3.71E-12*	-3.0666E+04±3.71E-12*	-3.0666E+04±0.00E+00*
g05	5.1265E+03±1.24E-27*	5.1265E+03±0.00E+00*	5.1265E+03±4.22E-10*	5.1265E+03±2.78E-12*	5.1265E+03±2.78E-12*	5.1265E+03±0.00E+00*
g06	-6.9618E+03±1.32E-26*	-6.9618E+03±0.00E+00*	-6.9618E+03±0.00E+00*	-6.9618E+03±0.00E+00*	-6.9618E+03±0.00E+00*	-6.9618E+03±0.00E+00*
g07	2.4306E+01±7.65E-15*	2.4306E+01±.37E-09*	2.4306E+01±5.28E-10*	2.4306E+01±6.32E-15*	2.4306E+01±8.52E-12*	2.4306E+01±1.77E-14*
g08	-9.5825E+02±6.36E-18*	-9.5825E+02±2.01E-34*	-9.5825E+02±2.78E-18*	-9.5825E+02±1.42E-17*	-9.5825E+02±1.42E-17*	-9.5825E+02±0.00E+00*
g09	6.8063E+02±4.96E-14*	6.8063E+02±1.10E-25*	6.8063E+02±2.23E-11*	6.8063E+02±2.23E-11*	6.8063E+02±2.54E-13*	6.8063E+02±5.57E-14*
g10	7.0492E+03±2.52E-13*	7.0492E+03±2.07E-24*	7.0492E+03±4.43E-08*	7.0492E+03±3.26E-12*	7.0492E+03±6.34E-10*	7.0492E+03±1.35E-06*
g11	7.499E-01±0.00E+00*	7.499E-01±0.00E+00*	7.499E-01±1.06E-16*	7.499E-01±1.13E-16*	7.499E-01±1.13E-16*	7.499E-01±0.00E+00*
g12	-1.00E+00±0.00E+00*	-1.00E+00±0.00E+00*	-1.00E+00±0.00E+00*	-1.00E+00±0.00E+00*	-1.00E+00±0.00E+00*	-1.00E+00±0.00E+00*
g13	5.3942E-02±1.04E-17*	5.3942E-02±1.98E-34*	5.3942E-02±6.03E-14*	5.3942E-02±2.41E-17*	5.3942E-02±2.13E-17*	5.3942E-02±1.30E-17*
g14	-4.7765E+01±3.62E-15*	-4.7765E+01±0.00E+00*	-4.7765E+01±3.47E-10*	-4.7765E+01±2.34E-14*	-4.7765E+01±2.93E-14*	-4.7765E+01±2.60E-15*
g15	9.6172E+02±0.00E+00*	9.6172E+02±0.00E+00*	9.6172E+02±4.47E-13*	9.6172E+02±5.80E-13*	9.6172E+02±5.80E-13*	9.6172E+02±0.00E+00*
g16	-1.9052E+00±2.64E-26*	-1.9052E+00±2.62E-30*	-1.9052E+00±0.00E+00*	-1.9052E+00±4.53E-16*	-1.9052E+00±4.53E-16*	-1.9052E+00±0.00E+00*
g17	8.8535E+03±1.24E-27*	8.8535E+03±2.51E-23*	**8.8802E+03±3.63E+01**	8.8535E+03±0.00E+00*	8.8535E+03±3.23E-08*	8.8535E+03±2.98E-08*
g18	-8.6603E-01±6.51E-17*	-8.6603E-01±4.62E-33*	-8.6603E-01±3.30E-07*	-8.6603E-01±6.94E-16*	-8.6603E-01±2.47E-16*	-8.6603E-01±0.00E+00*
g19	3.2656E+01±1.07E-10*	3.2656E+01±1.52E-05*	3.2656E+01±3.37E-07*	3.2656E+01±2.18E-14*	3.2656E+01±2.25E-14*	3.2656E+01±4.17E-10*
g21	**2.6195E+01±5.34E+01**	1.9372E+02±1.62E-22*	1.9372E+02±3.66E-09*	1.9372E+02±2.95E-11*	1.9372E+02±4.82E-10*	1.9372E+02±5.17E-11*
g23	-4.0006E+02±7.33E-11*	-4.0006E+02±9.08E-26*	-4.0006E+02±6.49E-06*	-4.0006E+02±1.71E-13*	-4.0006E+02±1.66E-05*	-4.0006E+02±4.37E-09*
g24	-5.5080E+00±.24E-28*	-5.5080E+00±0.00E+00*	-5.5080E+00±0.00E+00*	-5.5080E+00±9.06E-16*	-5.5080E+00±9.06E-16*	-5.5080E+00±0.00E+00*
*	**21**	**22**	**21**	**22**	**22**	**22**

TABLE VII

COMPARISON OF HECO-DE WITH HCO-DE AND HECO-DE(FR) ON FUNCTIONS G02, G10, G21, AND G23

CEC 2006	Mean (Success Rate%)[Feasible Rate%]		
	HCO-DE	HECO-DE(FR)	HECO-DE
g02	-0.8032(96)[100]	**-0.8036(100)[100]**	**-0.8036(100)[100]**
g10	6815.3984(76)[80]	7013.3762(80)[84]	**7049.2480(100)[100]**
g21	23.2469(12)[12]	7.4898(4)[4]	**193.7245(100)[100]**
g23	-376.0544(96)[100]	-376.0436(92)[100]	**-400.0551(100)[100]**

[16] E. Camponogara and S. N. Talukdar, "A genetic algorithm for constrained and multi-objective optimization," in *3rd Nordic Workshop on Genetic Algorithms and Their Applications (3NWGA)*, Vaasa, Finland, 1997.

[17] E. Mezura-Montes and C. A. C. Coello, "Constrained optimization via multiobjective evolutionary algorithms," in *Multiobjective Problem Solving from Nature*, J. Knowles, D. Corne, K. Deb, and D. Chair, Eds. Springer Berlin Heidelberg, 2008, pp. 53–75.

[18] Y. Zhou, Y. Li, J. He, and L. Kang, "Multi-objective and MGG evolutionary algorithm for constrained optimisation," in *Proceedings of 2003 IEEE Congress on Evolutionary Computation*. Canberra, Australia: IEEE Press, 2003, pp. 1–5.

[19] Z. Cai and Y. Wang, "A multiobjective optimization-based evolutionary algorithm for constrained optimization," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 658–675, 2006.

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

[21] ——, "A dynamic hybrid framework for constrained evolutionary optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 1, pp. 203–217, 2012.

[22] C. A. C. Coello and E. Mezura-Montes, "Handling constraints in genetic algorithms using dominance-based tournaments," in *Adaptive Computing in Design and Manufacture V*. Springer, 2002, pp. 273–284.

[23] X. Li, S. Zeng, C. Li, and J. Ma, "Many-objective optimization with dynamic constraint handling for constrained optimization problems," *Soft Computing*, vol. 21, no. 24, pp. 7435–7445, 2017.

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

[25] R. Datta and K. Deb, "Uniform adaptive scaling of equality and inequality constraints within hybrid evolutionary-cum-classical optimization," *Soft Computing*, vol. 20, no. 6, pp. 2367–2382, 2016.

[26] R. Jiao, S. Zeng, J. S. Alkasassbeh, and C. Li, "Dynamic multi-objective evolutionary algorithms for single-objective optimization," *Applied Soft Computing*, vol. 61, pp. 793–805, 2017.

[27] W. Huang, T. Xu, K. Li, and J. He, "Multiobjective differential evolution enhanced with principle component analysis for constrained optimization," *Swarm and Evolutionary Computation*, p. 100571, 2019.

[28] B. Ji, X. Yuan, and Y. Yuan, "Modified NSGA-II for solving continuous berth allocation problem: Using multiobjective constraint-handling strategy," *IEEE Transactions on Cybernetics*, vol. 47, no. 9, pp. 2885–2895, 2017.

[29] J.-Y. Ji, W.-J. Yu, Y.-J. Gong, and J. Zhang, "Multiobjective optimization with -constrained method for solving real-parameter constrained optimization problems," *Information Sciences*, vol. 467, pp. 15–34, 2018.

[30] J. He, B. Mitavskiy, and Y. Zhou, "A theoretical assessment of solution quality in evolutionary algorithms for the knapsack problem," in *Proceedings of 2014 IEEE Congress on Evolutionary Computation*. IEEE, 2014, pp. 141–148.

[31] F. Neumann and A. M. Sutton, "Runtime analysis of evolutionary algorithms for the knapsack problem with favorably correlated weights," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2018, pp. 141–152.

[32] 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.

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

[34] A. Trivedi, D. Srinivasan, K. Sanyal, and A. Ghosh, "A survey of multiobjective evolutionary algorithms based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 3, pp. 440–462, 2017.

[35] J. He and X. Yao, "Average drift analysis and population scalability," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 3, pp. 426–439, 2017.

[36] R. Poláková, "L-shade with competing strategies applied to constrained optimization," in *Evolutionary Computation (CEC), 2017 IEEE Congress on*. IEEE, 2017, pp. 1683–1689.

[37] J. Zhang and A. C. Sanderson, "Jade: adaptive differential evolution with optional external archive," *IEEE Transactions on evolutionary computation*, vol. 13, no. 5, pp. 945–958, 2009.

[38] P. Kaelo and M. Ali, "A numerical study of some modified differential evolution algorithms," *European journal of operational research*, vol. 169, no. 3, pp. 1176–1184, 2006.

[39] S. M. Islam, S. Das, S. Ghosh, S. Roy, and P. N. Suganthan, "An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 482–500, 2011.

[40] D. Zaharie, "Influence of crossover on the behavior of differential evolution algorithms," *Applied soft computing*, vol. 9, no. 3, pp. 1126–1138, 2009.

[41] J. Tvrdık, "Competitive differential evolution," in *MENDEL*, 2006, pp. 7–12.
[42] J. Tvrdík, "Adaptation in differential evolution: A numerical comparison," *Applied Soft Computing*, vol. 9, no. 3, pp. 1149–1155, 2009.
[43] R. Tanabe and A. S. Fukunaga, "Improving the search performance of shade using linear population size reduction," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*. IEEE, 2014, pp. 1658–1665.
[44] J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. N. Suganthan, C. C. Coello, and K. Deb, "Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization," *Journal of Applied Mechanics*, vol. 41, no. 8, pp. 8–31, 2006.
[45] A. Zamuda, "Adaptive constraint handling and success history differential evolution for cec 2017 constrained real-parameter optimization," in *Evolutionary Computation (CEC), 2017 IEEE Congress on*. IEEE, 2017, pp. 2443–2450.
[46] J. Tvrdík and R. Poláková, "A simple framework for constrained problems with application of l-shade44 and ide," in *Evolutionary Computation (CEC), 2017 IEEE Congress on*. IEEE, 2017, pp. 1436–1443.
[47] A. Trivedi, K. Sanyal, P. Verma, and D. Srinivasan, "A unified differential evolution algorithm for constrained optimization problems," in *Evolutionary Computation (CEC), 2017 IEEE Congress on*. IEEE, 2017, pp. 1231–1238.
[48] M. Hellwig and H.-G. Beyer, "A matrix adaptation evolution strategy for constrained real-parameter optimization," in *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2018, pp. 1–8.
[49] D. S. Anupam Trivedi and N. Biswas, "An improved unified differential evolution algorithm for constrained optimization problems," in *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2018, pp. 1–10.
[50] Z. Fan, Y. Fang, W. Li, Y. Yuan, Z. Wang, and X. Bian, "Lshade44 with an improved ϵ constraint-handling method for solving constrained single-objective optimization problems," in *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2018, pp. 1–8.
[51] L. Jiao, L. Li, R. Shang, F. Liu, and R. Stolkin, "A novel selection evolutionary strategy for constrained optimization," *Information Sciences*, vol. 239, pp. 122–141, 2013.
[52] Y. Wang, B.-C. Wang, H.-X. Li, and G. G. Yen, "Incorporating objective function information into the feasibility rule for constrained evolutionary optimization," *IEEE Transactions on Cybernetics*, vol. 46, no. 12, pp. 2938–2952, 2015.

SUPPLEMENT

This supplement provides further details of the benchmark problems used for comparative experimental investigations and of experimental results and comparisons.

A. Description of EAs under comparison on CEC 2018 benchmarks

The first seven EAs come from the CEC 2018 constrained optimisation competition [5]. The last one, DeCODE [12], was a decomposition-based multi-objective EAs for constrained optimisation published in 2018.

1) CAL-SHADE [45]: Success-History based Adaptive Differential Evolution Algorithm including liner population size reduction, enhanced with adaptive constraint violation handling, i.e. adaptive ϵ-constraint handling.
2) LSHADE+IDE [46]: A simple framework for cooperation of two advanced adaptive DE variants. The search process is divided into two stages: (i) search feasible solutions via minimizing the mean violation and stopped if a number of feasible solutions are found. (ii) minimize the function value until the stop condition is reached.
3) LSHADE44 [36]: Success-History based Adaptive Differential Evolution Algorithm including liner population size reduction, uses three different additional strategies compete, with the superiority of feasibility rule.
4) UDE [47]: Uses three trial vector generation strategies and two parameter settings. At each generation, UDE divides the current population into two sub-populations. In the first population, UDE employs all the three trial vector generation strategies on each target vector. For another one, UDE employs strategy adaption from learning experience from evolution in first population.
5) MA-ES [48]: Combines the Matrix Adaptation Evolution Strategy for unconstrained optimization with well-known constraint handling techniques. It handles box-constraints by reflecting exceeding components into the predefined box. Additional in-/equality constraints are dealt with by application of two constraint handling techniques: ϵ-level ordering and a repair step that is based on gradient approximation.
6) IUDE [49]: An improved version of UDE. Different from UDE, local search and duplication operators have been removed, it employs a combination of ϵ-constraint handling technique and the superiority of feasibility rule.
7) LSHADE-IEpsilon [50]: An improved ϵ-constrained handling method (IEpsilon) for solving constrained single-objective optimization problems. The IEpsilon method adaptively adjusts the value of ϵ according to the proportion of feasible solutions in the current population. Furthermore, a new mutation operator DE/randr1*/1 is proposed.
8) DeCODE [12]: A recent decomposition-based EA made use of the weighted sum approach to decompose the transformed bi-objective problem into a number of scalar optimisation subproblems and then applied differential evolution to solve them. They designed a strategy of adjusting weights and a restart strategy to tackle COPs with complicated constraints.

B. The IEEE CEC2006 benchmark suit

TABLE VIII
DESCRIPTION OF 24 BENCHMARK FUNCTIONS FROM IEEE CEC2006 WHERE D DENOTES DIMENSION, ρ THE ESTIMATED RATIO BETWEEN THE FEASIBLE AREA AND THE SEARCH SPACE, $f(\vec{x}^*)$ THE OPTIMUM OBJECTIVE FUNCTION VALUE

Function	D	Type	ρ	$f(\mathbf{x}^*)$
g01	13	Quadratic	0.0111%	-15.00000000
g02	20	Nonlinear	99.9971%	-0.8036191041
g03	10	Polynomial	0.0000%	-1.0005001000
g04	5	Quadratic	51.1230%	-30665.5386717833
g05	4	Cubic	0.0000%	5126.4967140071
g06	2	Cubic	0.0066%	-6961.8138755802
g07	10	Quadratic	0.0003%	24.3062090682
g08	2	Nonlinear	0.8560%	-0.0958250414
g09	7	Polynomial	0.5121%	680.6300573744
g10	8	Linear	0.0010%	7049.2480205287
g11	2	Quadratic	0.0000%	0.7499000000
g12	3	Quadratic	4.7713%	-1.0000000000
g13	5	Nonlinear	0.0000%	0.0539415140
g14	10	Nonlinear	0.0000%	-47.7648884595
g15	3	Quadratic	0.0000%	961.7150222900
g16	5	Nonlinear	0.0204%	-1.9051552585
g17	6	Nonlinear	0.0000%	8853.5338748065
g18	9	Quadratic	0.0000%	-0.8660254038
g19	15	Nonlinear	33.4761%	32.6555929502
g20	24	Linear	0.0000%	0.2049794002
g21	7	Linear	0.0000%	193.7245100697
g22	22	Linear	0.0000%	236.4309755040
g23	9	Linear	0.0000%	-400.055100000
g24	2	Linear	79.6556%	5.5080132716

C. The IEEE CEC2017 Benchmark Suit

IEEE CEC 2018 constrained optimisation competition [5] adopted the same benchmark suit as the IEEE CEC2017 competition [6]. The suit is listed in Table IX which consists of 4×28 problems with the dimension $D = 10, 30, 50, 100$.

D. Fine-tuning parameters on CEC 2006 benchmark

CEC 2006 benchmarks has more constraints than CEC 2017 benchmarks. Thus the size of subpopulation λ and constraint violation bias in CEC 2006 are set to different values from CEC 2017. Fine-tuning of parameters λ and γ was conducted on IEEE CEC2006 benchmark functions g02, g10, g17, g21, and g23. Experimental results in Tables X and XI show that the best value of λ is 45 and the best value of γ is 0.7. These values are used on other benchmarks. The λ and γ values are larger than those used in CEC 2017 ($\lambda = 20$ and $\gamma = 0.1$). This is due to CEC 2006 benchmarks are strongly constrained.

E. Detailed experimental results and ranking of HECO-DE on CEC 2017 benchmarks

In terms of IEEE CEC 2017 benchmark functions, the best, median, worst, mean, standard deviation and feasibility rate of the function values tested by HECO-DE on $10D$, $30D$, $50D$ and $100D$ are recorded in Table XII-XXI.

TABLE IX
DETAILS OF 28 TEST PROBLEMS FROM IEEE CEC2018. I IS THE NUMBER OF INEQUALITY CONSTRAINTS, E IS THE NUMBER OF EQUALITY CONSTRAINTS

Problem Search Range	Type of Objective	Number of Constraints	
		E	I
C01 $[-100,100]^D$	Non Separable	0	1 Separable
C02 $[-100,100]^D$	Non Separable, Rotated	0	1 Non Separable, Rotated
C03 $[-100,100]^D$	Non Separable	1 Separable	1 Separable
C04 $[-10,10]^D$	Separable	0	2 Separable
C05 $[-10,10]^D$	Non Separable	0	2 Non Separable, Rotated
C06 $[-20,20]^D$	Separable	6	0 Separable
C07 $[-50,50]^D$	Separable	2 Separable	0
C08 $[-100,100]^D$	Separable	2 Non Separable	0
C09 $[-10,10]^D$	Separable	2 Non Separable	0
C10 $[-100,100]^D$	Separable	2 Non Separable	0
C11 $[-100,100]^D$	Separable	1 Non Separable	1 Non Separable
C12 $[-100,100]^D$	Separable	0	2 Separable
C13 $[-100,100]^D$	Non Separable	0	3 Separable
C14 $[-100,100]^D$	Non Separable	1 Separable	1 Separable
C15 $[-100,100]^D$	Separable	1	1
C16 $[-100,100]^D$	Separable	1 Non Separable	1 Separable
C17 $[-100,100]^D$	Non Separable	1 Non Separable	1 Separable
C18 $[-100,100]^D$	Separable	1	2
C19 $[-50,50]^D$	Separable	0	2 Non Separable
C20 $[-100,100]^D$	Non Separable	0	2
C21 $[-100,100]^D$	Rotated	0	2 Rotated
C22 $[-100,100]^D$	Rotated	0	3 Rotated
C23 $[-100,100]^D$	Rotated	1 Rotated	1 Rotated
C24 $[-100,100]^D$	Rotated	1 Rotated	1 Rotated
C25 $[-100,100]^D$	Rotated	1 Rotated	1 Rotated
C26 $[-100,100]^D$	Rotated	1 Rotated	1 Rotated
C27 $[-100,100]^D$	Rotated	1 Rotated	2 Rotated
C28 $[-50,50]^D$	Rotated	0	2 Rotated

- c is the number of violated constraints at the median solution where three figures indicate the number of violations (including inequality and equality) by more than 1.0, in the range $[0.01, 1.0]$ and in the range $[0.0001, 0.01]$ respectively.
- $\overline{v}$ denotes the mean value of the constraint violations of all constraints at the median solution.
- SR is the feasibility rate of the solutions obtained in 25 runs.
- $\overline{vio}$ denotes the mean constraint violation value of all the solutions in 25 runs.

As shown in Table XII-XXI, HECO-DE got high accuracy results with high feasibility rate on most test problems. However, no feasible solution was found in functions C17, C19, C26 and C28 on any dimensions. This is a common issue faced by all EAs when solving these problems. For functions C08, C11, C18, c22 and C27, a feasible solution sometimes was not found.

F. Detailed ranking results of EAs on 2017 benchmarks

For the 28 test problems in $10D$, $30D$, $50D$ and $100D$, the ranks of each algorithm in terms of mean values and median solution are listed in Table XIII-XXIII respectively.

Regarding the test functions with $10D$, rank values based on mean values and median solution on the 28 test functions are reported in Table XIII and XIV, respectively. In terms of mean of solutions, HECO-DE had the lowest rank values on 8 of 28 problems (functions C01-C03, C05-C09). However, HECO-DE got relatively poor performance on C11, C13, C16 and C25. HECO-DE got the second lowest total rank value 83 which was slighter worse than the rank values obtained by HECO-DE(FR). In terms of median solution, HECO-DE got the lowest rank value on 13 of 28 problems (functions C01-C09, C13, C16, C21, and C24). But its performance is not good on functions C11, C12 and C14. HECO-DE was ranked first with a total rank value 71. The overall performance of HECO-DE is also the best among all nine EAs on $10D$ by summing up the two rank values in terms of mean values and median solution together.

Regarding the test functions with $30D$, rank values based on mean values and median solution on the 28 test functions are listed in Table XVI and XVII, respectively. HECO-DE had the lowest rank values on 11 of 28 problems (functions C01-C03, C06, C09, C10, C13, C15, 20, C21 and C24). However, HECO-DE got relatively poor performance on functions C05 and C11. In terms of median solution, HECO-DE got the lowest rank value on 9 of 28 problems (functions C01-C03, C05, C06, C13, C15, C20 and C21). But its performance was not good on functions C11. Total rank values of HECO-DE were the lowest ones, 70 in terms of mean of solutions and 69 in terms of median solution, respectively.

Regarding the test functions with $50D$, rank values based on mean values and median solution on the 28 test functions are reported in Table XIX and XX, respectively. HECO-DE had the lowest rank values on 5 of 28 problems (functions C01-C05, C12, C15-C17, C21, C24 and C25). However, HECO-DE got relatively poor performance on functions C05 and C11. In terms of median solution, HECO-DE got the lowest rank value on 9 of 28 problems (functions C01-C03, C05, C10, C12, C13, C20 and C23). But its performance was not good on functions C11. Total rank values of HECO-DE were the lowest ones, 88 in terms of mean values and 68 in terms of median solution, respectively.

Table XXII and XXIII record rank values based on mean values and median solution on the 28 test functions on $100D$. HECO-DE had the lowest rank values on 4 of 28 problems (functions C01, C02, C15 and C20). But HECO-DE got relatively poor performance on functions C05, C08, C11, C13 and C21. HECO-DE got the lowest total rank value 108 here.

TABLE X
MEAN OBJECTIVE FUNCTION VALUE, SUCCESS RATE, FEASIBLE RATE ON IEEE CEC2006 BENCHMARK FUNCTIONS G02, G10, G17, G21, AND G23 WITH VARIED λ.

Prob.	Mean (Success Rate%)[Feasible Rate%]				
	35	40	45	50	55
g02	-0.8032(92)[100]	**-0.8036(100)[100]**	**-0.8036(100)[100]**	**-0.8036(100)[100]**	-0.8032(96)[100]
g03	**-1.0005(100)[100]**	**-1.0005(100)[100]**	**-1.0005(100)[100]**	**-1.0005(100)[100]**	-1.00047(96)[100]
g10	**7049.2480(100)[100]**	**7049.2480(100)[100]**	**7049.2480(100)[100]**	**7049.2480(100)[100]**	7049.2481(96)[100]
g13	**0.0539(100)[100]**	**0.0539(100)[100]**	**0.0539(100)[100]**	**0.0539(100)[100]**	0.0539(96)[100]
g17	8856.5008(96)[100]	**8853.5339(100)[100]**	**8853.5339(100)[100]**	**8853.5339(100)[100]**	8853.7232(96)[100]
g21	**193.7245(100)[100]**	**193.7245(100)[100]**	**193.7245(100)[100]**	**193.7245(100)[100]**	**193.7245(100)[100]**
g23	-388.0548(96)[100]	-376.0544(92)[100]	**-400.0551(100)[100]**	-376.0544(92)[100]	**-400.0551(100)[100]**

TABLE XI
MEAN OBJECTIVE FUNCTION VALUE, SUCCESS RATE, FEASIBLE RATE ON IEEE CEC2006 BENCHMARK FUNCTIONS G02, G10, G17, G21, AND G23 WITH VARIED γ.

Prob.	Mean (Success Rate%)[Feasible Rate%]				
	0.5	0.6	0.7	0.8	0.9
g02	**-0.8036(100)[100]**	-0.8034(96)[100]	**-0.8036(100)[100]**	**-0.8036(100)[100]**	-0.8036(96)[100]
g03	**-1.0005(100)[100]**	**-1.0005(100)[100]**	**-1.0005(100)[100]**	**-1.0005(100)[100]**	**-1.0005(100)[100]**
g10	10384.6108(8)[92]	7049.2986(80)[100]	**7049.2480(100)[100]**	**7049.2480(100)[100]**	**7049.2480(100)[100]**
g13	**0.0539(100)[100]**	**0.0539(100)[100]**	**0.0539(100)[100]**	**0.0539(100)[100]**	0.0615(92)[100]
g17	8854.9176(52)[100]	8853.9032(80)[100]	**8853.5339(100)[100]**	**8853.5339(100)[100]**	8856.5699(92)[100]
g21	23.2469(12)[12]	131.7327(68)[68]	**193.7245(100)[100]**	**193.7245(100)[100]**	**193.7245(100)[100]**
g23	-387.5537(80[100])	-387.4869(88)[100]	**-400.0551(100)[100]**	-376.0544(92)[100]	-376.0544(92)[100]

In terms of median solution, HECO-DE got the lowest rank value on 5 of 28 problems (functions C01, C02, C15 and C20). But it had a poor performance on functions C11-C13 and C21. HECO-DE got the second lowest total rank value 97 which was only worse than the rank values obtained by HECO-DE(FR).

According to the competition rules, HECO-DE got the lowest or at least comparable total rank values on each dimension. This means that HECO-DE had an overall better performance than other eights algorithms on the IEEE CEC2017 benchmark suit. However, the ranking tables also show that no algorithm could perform better than other algorithms on all problems.

TABLE XII
FUNCTION VALUES OF HECO-DE ACHIEVED FOR 10D ($FES_{\max} = 20000 \times D$) ON IEEE CEC2017 BENCHMARKS

problem	C01	C02	C03	C04	C05	C06	C07
Best	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	-9.91896e+02
Median	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	-9.68180e+02
c	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0
$\overline{v}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
mean	0.00000e+00	0.00000e+00	0.00000e+00	5.42911e+00	8.69719e-31	0.00000e+00	-9.56102e+02
Worst	0.00000e+00	0.00000e+00	0.00000e+00	1.35728e+01	2.17430e-29	0.00000e+00	-8.80241e+02
std	0.00000e+00	0.00000e+00	0.00000e+00	6.64928e+00	4.26074e-30	0.00000e+00	3.35462e+01
SR	100	100	100	100	100	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
Problem	C8	C9	C10	C11	C12	C13	C14
Best	-1.34840e-03	-4.97525e-03	-5.09647e-04	-1.68818e-01	3.98790e+00	0.00000e+00	2.37633e+00
Median	-1.34840e-03	-4.97525e-03	-5.09647e-04	-1.66490e-01	3.98790e+00	0.00000e+00	2.37633e+00
c	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0
$\overline{v}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
mean	-1.34840e-03	-4.97525e-03	-5.09647e-04	-1.04491e+00	3.98791e+00	1.59463e-01	2.37633e+00
Worst	-1.34840e-03	-4.97525e-03	-5.09647e-04	-5.03190e+00	3.98796e+00	3.98658e+00	2.37633e+00
std	3.82639e-16	0.00000e+00	0.00000e+00	1.34370e+00	1.05948e-05	7.81207e-01	1.33227e-15
SR	100	100	100	56	100	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	2.41023e-05	0.00000e+00	0.00000e+00	0.00000e+00
Problem	C15	C16	C17	C18	C19	C20	C21
Best	2.35612e+00	0.00000e+00	1.08553e-02	1.00000e+01	0.00000e+00	5.59892e-02	3.98790e+00
Median	2.35612e+00	0.00000e+00	1.08553e-02	5.04203e+01	0.00000e+00	2.94245e-01	3.98790e+00
c	0 0 0	0 0 0	1 0 0	0 0 0	1 0 0	0 0 0	0 0 0
$\overline{v}$	0.00000e+00	0.00000e+00	4.50000e+00	0.00000e+00	6.63359e+03	0.00000e+00	0.00000e+00
mean	2.35612e+00	6.28263e-02	1.08347e-02	3.43142e+01	0.00000e+00	3.01877e-01	3.98790e+00
Worst	2.35612e+00	1.57066e+00	1.03418e-02	5.19710e+01	0.00000e+00	5.23269e-01	3.98791e+00
std	1.06951e-15	3.07785e-01	1.00610e-04	1.98547e+01	0.00000e+00	1.30553e-01	2.61846e-06
SR	100	100	0	100	0	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	4.54000e+00	0.00000e+00	6.63359e+03	0.00000e+00	0.00000e+00
Problem	C22	C23	C24	C25	C26	C27	C28
Best	6.17530e-30	2.37633e+00	2.35612e+00	3.09207e-86	1.08553e-02	9.05515e+01	3.74160e-15
Median	6.17530e-30	2.37633e+00	2.35612e+00	1.63062e-74	1.08553e-02	9.57215e+01	1.01234e-10
c	0 0 0	0 0 0	0 0 0	0 0 0	1 0 0	0 0 0	1 0 0
$\overline{v}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	4.50000e+00	0.00000e+00	6.63359e+03
mean	1.59463e-01	2.37633e+00	2.35612e+00	4.39784e-01	1.93244e-02	9.38603e+01	2.33151e-08
Worst	3.98658e+00	2.37633e+00	2.35612e+00	1.57066e+00	2.27203e-01	9.57221e+01	2.01317e-07
std	7.81207e-01	3.52023e-07	4.59998e-08	7.05223e-01	4.24337e-02	2.48162e+00	5.59317e-08
SR	100	100	100	100	0	100	0
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	4.90000e+00	0.00000e+00	6.63359e+03

TABLE XIII
RANKS BASED ON **MEAN SOLUTION** ON THE 28 FUNCTIONS OF $10D$ ON IEEE CEC2017 BENCHMARKS

Problem	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	Total
CAL_LSAHDE(2017)	1	1	11	11	11	7	10	10	10	10	6	1	11	8	8	11	10	11	1	9	11	11	10	10	11	11	9	1	232
LSHADE44+IDE(2017)	1	1	9	8	1	11	9	1	1	2	1	3	1	10	4	9	7	9	4	4	2	4	9	7	9	6	10	9	152
LSAHDE44(2017)	1	1	10	6	1	10	8	1	9	2	2	10	1	9	9	10	8	8	2	1	7	8	8	6	10	7	8	10	173
UDE(2017)	1	1	8	9	10	8	7	1	7	2	10	1	10	7	5	8	9	7	8	11	9	10	7	3	8	10	7	7	191
MA_ES(2018)	1	1	1	10	1	6	5	1	1	2	4	11	7	11	10	1	11	3	10	10	10	7	11	5	1	9	2	8	160
IUDE(2018)	1	1	6	3	1	1	11	1	1	2	5	9	1	3	3	1	5	5	4	8	4	9	1	3	1	2	6	6	104
LSAHDE_IEpsilon(2018)	1	1	7	5	1	9	6	1	1	2	3	6	1	2	6	1	3	4	3	7	8	1	4	9	1	5	1	11	110
DeCODE(2018)	1	1	1	7	1	1	4	9	7	1	9	5	9	1	1	7	2	10	9	6	1	1	6	1	7	1	11	4	124
HCO-DE	1	1	1	1	1	1	3	11	11	11	11	7	1	3	11	1	6	6	11	3	6	6	5	11	1	8	5	5	149
HECO-DE(FR)	1	1	1	1	1	1	2	1	1	2	7	8	1	3	7	1	1	1	4	5	5	1	3	8	5	3	3	2	**80**
HECO-DE	1	1	1	4	1	1	1	1	1	2	8	4	7	3	2	6	4	2	4	2	3	4	2	2	6	4	4	2	83

TABLE XIV
RANKS BASED ON **MEDIAN SOLUTION** ON THE 28 FUNCTIONS OF $10D$ ON IEEE CEC2017 BENCHMARKS

Problem	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	Total
CAL_LSAHDE(2017)	1	1	11	11	1	9	8	10	11	1	4	3	1	9	8	11	11	10	1	9	4	1	10	10	11	11	10	1	189
LSHADE44+IDE(2017)	1	1	10	6	1	11	9	1	1	2	2	4	1	11	7	10	7	9	4	4	5	1	11	9	10	9	9	2	158
LSAHDE44(2017)	1	1	9	8	1	10	10	1	1	2	4	11	1	10	9	9	8	7	2	1	1	1	9	8	9	7	8	10	159
UDE(2017)	1	1	7	9	1	8	7	1	9	2	10	1	1	1	5	8	10	8	10	11	1	1	1	5	8	10	7	6	150
MA_ES(2018)	1	1	1	10	1	1	5	1	1	2	2	5	1	3	10	1	9	2	8	10	6	1	3	7	1	8	1	9	111
IUDE(2018)	1	1	1	1	1	1	11	1	1	2	1	10	1	3	5	1	1	6	4	8	9	1	4	5	1	1	6	6	94
LSAHDE_IEpsilon(2018)	1	1	8	5	1	1	6	1	1	2	6	8	1	3	3	1	6	3	3	7	11	1	8	4	1	6	2	11	112
DeCODE(2018)	1	1	1	7	1	1	4	1	9	2	9	1	1	1	1	7	5	11	11	6	1	1	1	1	7	4	11	8	115
HCO-DE	1	1	1	1	1	1	3	11	1	11	11	9	1	6	11	1	2	5	9	3	10	1	7	11	1	5	3	5	133
HECO-DE(FR)	1	1	1	1	1	1	2	1	1	2	8	7	1	6	4	1	2	1	4	5	8	1	6	2	1	2	5	2	78
HECO-DE	1	1	1	1	1	1	1	1	1	2	7	6	1	6	2	1	2	4	4	2	7	1	5	2	1	3	4	2	**71**

TABLE XV

FUNCTION VALUES OF HECO-DE ACHIEVED FOR $30D$ ($FES_{\max} = 20000 \times D$) ON IEEE CEC2017 BENCHMARKS

problem	C01	C02	C03	C04	C05	C06	C07
Best	1.24862e-29	2.12623e-29	2.36658e-30	1.35728e+01	0.00000e+00	0.00000e+00	-2.19162e+03
Median	5.23113e-29	4.96613e-29	9.01274e-29	1.35728e+01	0.00000e+00	0.00000e+00	-1.91485e+03
c	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0
$\overline{v}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
mean	5.63040e-29	5.57256e-29	1.07822e-28	1.35728e+01	1.59465e-01	4.92467e+01	-1.90807e+03
Worst	1.86036e-28	1.35548e-28	2.96612e-28	1.35728e+01	3.98662e+00	1.50462e+02	-1.59126e+03
std	3.48173e-29	2.56870e-29	7.57292e-29	5.65094e-15	7.81216e-01	6.11356e+01	1.59817e+02
SR	100	100	100	100	100	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
Problem	C8	C9	C10	C11	C12	C13	C14
Best	-2.83981e-04	-2.66551e-03	-1.02842e-04	-1.23626e+01	3.98253e+00	0.00000e+00	1.40852e+00
Median	-2.83981e-04	-2.66551e-03	-1.02842e-04	-2.81552e+02	3.98253e+00	5.38003e-27	1.40852e+00
c	0 0 0	0 0 0	0 0 0	1 0 0	0 0 0	0 0 0	0 0 0
$\overline{v}$	0.00000e+00	0.00000e+00	0.00000e+00	5.15602e+00	0.00000e+00	0.00000e+00	0.00000e+00
mean	-2.83981e-04	-2.66551e-03	-1.02842e-04	-2.88422e+02	3.98253e+00	1.09425e-26	1.40852e+00
Worst	-2.83981e-04	-2.66551e-03	-1.02842e-04	-7.24259e+02	3.98253e+00	6.34654e-26	1.40852e+00
std	1.12431e-14	8.70234e-17	3.72014e-13	2.14245e+02	6.74301e-07	1.54346e-26	9.65830e-16
SR	100	100	100	0	100	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	7.95513e+00	0.00000e+00	0.00000e+00	0.00000e+00
Problem	C15	C16	C17	C18	C19	C20	C21
Best	2.35612e+00	1.57066e+00	3.08555e-02	4.22186e+01	0.00000e+00	9.86676e-01	3.98253e+00
Median	2.35612e+00	1.57066e+00	3.15387e-02	5.34725e+01	0.00000e+00	1.21870e+00	3.98253e+00
c	0 0 0	0 0 0	1 0 0	0 0 0	1 0 0	0 0 0	0 0 0
$\overline{v}$	0.00000e+00	0.00000e+00	1.55000e+01	0.00000e+00	2.13749e+04	0.00000e+00	0.00000e+00
mean	2.35612e+00	1.57066e+00	1.46237e-01	5.18854e+01	0.00000e+00	1.24525e+00	3.98253e+00
Worst	2.35612e+00	1.57066e+00	2.23668e-01	5.48137e+01	0.00000e+00	1.72676e+00	3.98253e+00
std	1.14778e-15	6.49635e-07	2.18583e-01	3.80234e+00	0.00000e+00	2.01869e-01	9.88227e-07
SR	100	100	0	100	0	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	1.52200e+01	0.00000e+00	2.13749e+04	0.00000e+00	0.00000e+00
Problem	C22	C23	C24	C25	C26	C27	C28
Best	2.72944e-04	1.40852e+00	2.35612e+00	1.57066e+00	9.65986e-02	2.08760e+02	2.01800e-01
Median	1.37656e-01	1.40852e+00	2.35612e+00	6.28305e+00	6.65154e-01	2.08770e+02	2.93281e+00
c	0 0 0	0 0 0	0 0 0	0 0 0	1 0 0	0 0 0	1 0 0
$\overline{v}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	1.55000e+01	0.00000e+00	2.13848e+04
mean	1.99711e-01	1.43633e+00	2.35612e+00	4.58659e+00	5.51798e-01	2.24098e+02	3.78446e+00
Worst	1.29494e+00	1.49544e+00	2.35612e+00	6.28305e+00	8.34109e-01	2.51377e+02	1.25840e+01
std	2.51084e-01	4.05468e-02	1.14658e-14	2.26195e+00	2.68652e-01	2.04449e+01	3.89563e+00
SR	100	100	100	100	0	100	0
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	1.54600e+01	0.00000e+00	2.13870e+04

TABLE XVI

RANKS BASED ON **MEAN SOLUTION** ON THE 28 FUNCTIONS OF $30D$ ON IEEE CEC2017 BENCHMARKS

Problem	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	Total
CAL_LSAHDE(2017)	1	1	11	10	11	5	11	1	11	10	4	11	11	11	9	11	11	10	1	6	11	11	11	10	11	11	9	1	232
LSHADE44+IDE(2017)	1	1	10	6	1	11	7	10	1	9	3	8	8	10	7	8	8	9	4	8	9	8	10	9	8	8	10	9	201
LSAHDE44(2017)	1	1	9	4	1	10	8	2	1	1	2	6	7	9	8	9	7	7	2	3	8	9	8	8	9	7	8	10	165
UDE(2017)	1	1	6	9	6	3	5	9	7	8	7	9	9	7	6	7	9	8	10	10	5	7	5	6	6	9	7	7	189
MA_ES(2018)	1	1	1	8	1	2	4	2	1	1	1	10	1	8	10	1	10	2	11	11	10	1	7	7	1	10	1	5	129
IUDE(2018)	1	1	7	5	1	8	10	2	7	1	5	5	6	1	5	4	4	6	8	9	7	6	3	3	5	5	6	8	139
LSAHDE_IEpsilon(2018)	1	1	8	2	1	9	6	2	1	1	9	7	10	6	3	6	6	1	3	4	4	10	1	5	7	6	2	11	133
DeCODE(2018)	1	1	1	7	10	4	9	8	9	1	11	1	1	2	4	5	5	11	9	7	6	3	9	4	4	3	11	6	153
HCO-DE	1	1	1	1	7	7	1	11	10	11	10	3	1	3	11	1	1	5	7	2	2	4	6	11	2	2	5	4	131
HECO-DE(FR)	1	1	5	11	7	6	2	2	1	1	6	4	1	3	2	10	3	4	4	5	3	5	2	2	10	1	4	2	108
HECO-DE	1	1	1	3	7	1	3	2	1	1	8	2	1	3	1	3	2	3	4	1	1	2	4	1	3	4	3	3	**70**

TABLE XVII

RANKS BASED ON **MEDIAN SOLUTION** ON THE 28 FUNCTIONS OF $30D$ ON IEEE CEC2017 BENCHMARKS

Problem	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	Total
CAL_LSAHDE(2017)	1	1	11	11	1	7	11	1	1	1	4	9	11	8	9	11	11	9	1	6	4	11	7	10	11	11	8	1	188
LSHADE44+IDE(2017)	1	1	10	3	1	11	9	10	2	10	3	2	1	11	7	9	9	10	4	8	10	8	10	9	9	6	10	9	193
LSAHDE44(2017)	1	1	9	5	1	10	8	2	2	2	2	8	9	10	8	10	8	6	2	3	9	9	9	7	10	9	9	10	179
UDE(2017)	1	1	6	10	1	5	5	9	8	9	6	10	8	7	6	6	10	8	8	10	4	7	1	6	7	10	7	7	183
MA_ES(2018)	1	1	1	9	1	4	4	2	2	2	1	11	1	9	10	1	7	1	11	11	11	1	8	8	1	7	1	5	132
IUDE(2018)	1	1	7	6	1	9	7	2	8	2	5	6	1	1	3	5	3	7	8	9	4	1	1	1	6	3	6	8	122
LSAHDE_IEpsilon(2018)	1	1	8	2	1	8	6	2	2	2	9	7	10	3	5	8	6	2	3	4	8	10	6	5	8	5	2	11	145
DeCODE(2018)	1	1	1	6	1	6	10	2	8	2	11	1	1	1	3	6	5	11	10	7	4	1	11	4	5	8	11	6	144
HCO-DE	1	1	1	1	1	1	1	11	11	11	10	4	1	4	11	1	1	5	7	2	2	4	5	11	1	4	5	4	122
HECO-DE(FR)	1	1	1	8	1	1	2	2	2	2	8	5	1	4	2	3	4	4	4	5	3	6	4	2	3	1	4	2	86
HECO-DE	1	1	1	3	1	1	3	2	2	2	7	3	1	4	1	3	2	3	4	1	1	5	3	2	4	2	3	3	**69**

TABLE XVIII

FUNCTION VALUES OF HECO-DE ACHIEVED FOR $50D$ ($FES_{\max} = 20000 \times D$) ON IEEE CEC2017 BENCHMARKS

problem	C01	C02	C03	C04	C05	C06	C07
Best	3.26897e-28	2.74240e-28	6.07817e-28	1.35728e+01	1.99828e-28	1.37141e+02	-3.27891e+03
Median	7.56234e-28	6.20353e-28	2.16503e-27	1.35728e+01	1.07226e-27	3.18299e+02	-2.70042e+03
c	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0
$\overline{v}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
mean	8.56437e-28	6.88503e-28	6.18894e+00	1.38003e+01	4.78395e-01	3.29350e+02	-2.71356e+03
Worst	3.02740e-27	1.76074e-27	7.74180e+01	1.69142e+01	3.98662e+00	4.76668e+02	-1.74763e+03
std	5.13174e-28	3.12836e-28	2.09877e+01	7.84266e-01	1.29550e+00	8.49272e+01	3.98328e+02
SR	100	100	100	100	100	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
Problem	C8	C9	C10	C11	C12	C13	C14
Best	-1.34534e-04	-2.03709e-03	-4.82664e-05	-7.94134e+02	3.98145e+00	3.25024e-26	1.09995e+00
Median	-1.34527e-04	-2.03709e-03	-4.82653e-05	-1.09580e+03	3.98145e+00	1.28055e-25	1.09995e+00
c	0 0 0	0 0 0	0 0 0	1 0 0	0 0 0	0 0 0	0 0 0
$\overline{v}$	0.00000e+00	0.00000e+00	0.00000e+00	4.42331e+01	0.00000e+00	0.00000e+00	0.00000e+00
mean	-1.34500e-04	-2.03709e-03	-4.82635e-05	-1.55921e+03	4.47573e+00	3.18930e-01	1.09995e+00
Worst	-1.34278e-04	-2.03709e-03	-4.82524e-05	-1.31990e+03	7.07354e+00	3.98662e+00	1.10000e+00
std	6.63405e-08	6.32044e-16	4.45902e-09	4.27039e+02	1.13254e+00	1.08154e+00	8.45949e-06
SR	100	100	100	0	100	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	4.23783e+01	0.00000e+00	0.00000e+00	0.00000e+00
Problem	C15	C16	C17	C18	C19	C20	C21
Best	2.35612e+00	1.57066e+00	3.10112e-01	4.42174e+01	0.00000e+00	2.03570e+00	3.98145e+00
Median	2.35612e+00	1.57066e+00	4.13497e-01	4.61633e+01	0.00000e+00	2.49861e+00	3.98145e+00
c	0 0 0	0 0 0	1 0 0	0 0 0	1 0 0	0 0 0	0 0 0
$\overline{v}$	0.00000e+00	0.00000e+00	2.55000e+01	0.00000e+00	3.61162e+04	0.00000e+00	0.00000e+00
mean	2.35612e+00	1.75915e+00	5.11704e-01	4.70125e+01	0.00000e+00	2.51270e+00	4.59984e+00
Worst	2.35612e+00	6.28305e+00	9.93719e-01	4.42149e+01	0.00000e+00	3.06595e+00	7.08178e+00
std	1.32633e-15	9.23436e-01	2.82294e-01	4.37480e+00	0.00000e+00	2.93609e-01	1.23677e+00
SR	100	100	0	72	0	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	2.54200e+01	1.70218e+00	3.61162e+04	0.00000e+00	0.00000e+00
Problem	C22	C23	C24	C25	C26	C27	C28
Best	2.25269e+01	1.09995e+00	2.35612e+00	6.28305e+00	5.84432e-01	2.47657e+02	2.79454e+00
Median	2.65138e+01	1.09995e+00	2.35612e+00	6.28305e+00	9.12631e-01	2.47695e+02	8.12215e+00
c	0 0 0	0 0 0	0 0 0	0 0 0	1 0 0	0 0 0	1 0 0
$\overline{v}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	2.55000e+01	0.00000e+00	3.61449e+04
mean	4.73690e+01	1.11255e+00	3.23577e+00	8.35650e+00	8.91498e-01	2.53009e+02	9.59562e+00
Worst	2.10530e+02	1.15245e+00	5.49772e+00	2.51326e+01	1.04807e+00	2.64434e+02	1.65574e+01
std	5.09604e+01	2.24184e-02	1.41057e+00	4.23171e+00	1.45244e-01	7.75164e+00	5.88050e+00
SR	100	100	100	100	0	100	0
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	2.55000e+01	0.00000e+00	3.61467e+04

TABLE XIX

RANKS BASED ON **MEAN SOLUTION** ON THE 28 FUNCTIONS OF $50D$ ON IEEE CEC2017 BENCHMARKS

Problem	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	Total
CAL_LSAHDE(2017)	11	11	11	10	10	6	11	11	10	8	4	8	11	11	10	11	11	9	1	11	7	10	11	10	11	11	8	1	255
LSHADE44+IDE(2017)	10	1	10	4	1	10	8	10	7	10	1	5	8	10	7	8	9	8	4	7	10	7	9	9	8	9	9	10	209
LSAHDE44(2017)	1	1	9	1	1	9	7	1	3	1	2	10	7	9	8	9	8	7	3	3	11	8	7	8	7	8	7	9	165
UDE(2017)	1	1	6	9	11	5	5	9	1	9	5	7	10	7	6	6	10	10	10	8	4	9	3	6	5	10	10	6	189
MA_ES(2018)	1	1	1	8	1	2	4	2	8	1	3	11	9	8	9	1	7	1	11	10	9	6	8	7	1	6	1	5	142
IUDE(2018)	1	1	7	7	1	11	10	7	1	1	8	4	6	4	3	5	4	6	9	9	5	5	2	3	3	3	6	8	140
LSAHDE_IEpsilon(2018)	1	1	8	2	7	8	6	8	6	7	9	9	5	6	5	7	6	2	7	5	6	11	1	5	6	7	2	11	164
DeCODE(2018)	1	1	1	5	1	4	9	6	9	6	11	6	1	5	4	4	5	11	2	6	8	1	10	4	2	5	11	7	146
HCO-DE	1	1	1	11	1	7	1	3	11	11	10	1	4	1	11	1	1	5	8	2	1	2	6	11	10	4	5	4	135
HECO-DE(FR)	1	1	5	6	7	1	2	5	3	5	6	3	2	1	2	10	2	3	4	4	3	4	5	1	9	1	4	2	102
HECO-DE	1	1	4	3	9	3	3	4	3	4	7	2	3	3	1	3	3	4	4	1	2	3	4	1	4	2	3	3	**88**

TABLE XX

RANKS BASED ON **MEDIAN SOLUTION** ON THE 28 FUNCTIONS OF $50D$ ON IEEE CEC2017 BENCHMARKS

Problem	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	Total
CAL_LSAHDE(2017)	1	1	11	11	1	7	11	11	7	8	4	8	11	7	11	11	11	10	1	6	8	10	6	10	11	11	8	1	214
LSHADE44+IDE(2017)	1	1	10	3	1	10	8	10	9	10	3	4	9	11	7	10	9	9	5	8	10	8	10	9	9	9	10	10	213
LSAHDE44(2017)	1	1	9	5	1	9	7	1	3	1	1	11	8	10	9	9	8	7	4	3	11	9	9	8	8	8	7	9	177
UDE(2017)	1	1	6	10	11	6	5	9	1	9	5	6	10	8	6	7	10	8	11	9	1	7	3	7	6	10	9	6	188
MA_ES(2018)	1	1	1	9	1	3	4	2	8	1	2	10	1	9	8	1	6	1	10	11	9	6	8	6	1	5	1	5	131
IUDE(2018)	1	1	7	8	1	11	10	5	1	1	7	5	1	5	1	6	4	6	2	10	5	5	3	1	4	4	6	8	129
LSAHDE_IEpsilon(2018)	1	1	8	2	1	8	6	8	3	7	9	9	7	1	5	8	7	2	8	5	6	11	5	5	7	7	2	11	160
DeCODE(2018)	1	1	1	6	1	5	9	7	10	6	11	6	1	5	4	5	5	11	3	7	7	1	11	4	4	6	11	7	156
HCO-DE	1	1	1	1	1	1	1	2	11	11	10	3	1	2	10	1	1	5	9	2	3	2	7	11	10	3	5	4	120
HECO-DE(FR)	1	1	5	7	1	2	2	6	3	5	6	2	1	2	2	3	2	3	5	4	4	4	2	2	2	1	4	2	84
HECO-DE	1	1	1	3	1	4	3	2	3	1	8	1	1	2	2	3	3	4	5	1	2	3	1	2	2	2	3	3	**68**

TABLE XXI

FUNCTION VALUES OF HECO-DE ACHIEVED FOR $100D$ ($FES_{\max} = 20000 \times D$) ON IEEE CEC2017 BENCHMARKS

problem	C01	C02	C03	C04	C05	C06	C07
Best	5.47366e-21	7.19883e-21	1.41057e+02	1.69142e+01	2.66253e-17	5.05460e+02	-5.00027e+03
Median	3.21897e-19	6.57495e-19	3.36356e+02	4.97476e+01	1.26216e-14	9.07974e+02	-1.75646e+03
c	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0
$\overline{v}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
mean	7.51270e-19	4.62157e-18	3.24358e+02	6.02942e+01	1.59465e+00	8.88484e+02	-2.04944e+03
Worst	4.86463e-18	9.19137e-17	4.95920e+02	2.27844e+02	3.98662e+00	1.09726e+03	-3.68566e+02
std	1.05527e-18	1.78522e-17	9.03104e+01	4.07022e+01	1.95304e+00	1.11626e+02	1.17219e+03
SR	100	100	100	100	100	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
Problem	C8	C9	C10	C11	C12	C13	C14
Best	2.91098e-04	0.00000e+00	-1.70891e-05	-3.83253e+03	3.98064e+00	3.37712e+01	7.84202e-01
Median	4.89414e-04	0.00000e+00	-1.68744e-05	-3.94619e+03	1.46028e+01	2.35304e+02	7.84445e-01
c	0 0 0	0 0 0	0 0 0	1 0 0	0 0 0	0 0 0	0 0 0
$\overline{v}$	0.00000e+00	0.00000e+00	0.00000e+00	7.81370e+01	0.00000e+00	0.00000e+00	0.00000e+00
mean	1.08808e-03	0.00000e+00	-1.68459e-05	-4.31676e+03	1.70060e+01	2.76821e+02	7.85919e-01
Worst	1.52346e-02	0.00000e+00	-1.62139e-05	-4.53746e+03	3.17614e+01	7.99908e+02	7.94379e-01
std	2.89065e-03	0.00000e+00	1.70733e-07	3.00520e+02	7.87982e+00	2.27727e+02	2.76686e-03
SR	96	100	100	0	100	100	100
$\overline{vio}$	3.20930e-07	0.00000e+00	0.00000e+00	9.41717e+01	0.00000e+00	0.00000e+00	0.00000e+00
Problem	C15	C16	C17	C18	C19	C20	C21
Best	5.49772e+00	6.28305e+00	6.61552e-01	4.62791e+01	0.00000e+00	5.47915e+00	5.02861e+00
Median	5.49772e+00	6.28305e+00	1.02926e+00	1.48540e+02	0.00000e+00	6.20755e+00	1.46029e+01
c	0 0 0	0 0 0	1 0 0	1 0 0	1 0 0	0 0 0	0 0 0
$\overline{v}$	0.00000e+00	0.00000e+00	5.05000e+01	2.11920e+01	7.29695e+04	0.00000e+00	0.00000e+00
mean	6.25170e+00	6.28305e+00	9.95776e-01	9.42757e+01	0.00000e+00	6.24532e+00	1.97529e+01
Worst	8.63931e+00	6.28305e+00	1.04035e+00	8.02027e+01	0.00000e+00	7.20779e+00	3.96546e+01
std	1.34172e+00	3.80293e-07	7.94962e-02	4.07627e+01	0.00000e+00	4.75486e-01	1.02325e+01
SR	100	100	0	12	0	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	5.05000e+01	1.44421e+01	7.29695e+04	0.00000e+00	0.00000e+00
Problem	C22	C23	C24	C25	C26	C27	C28
Best	8.86509e+01	7.84204e-01	5.49772e+00	1.88494e+01	9.60614e-01	2.84846e+02	1.33704e+01
Median	1.15222e+03	7.84208e-01	5.49772e+00	2.51326e+01	1.01425e+00	2.84969e+02	2.89869e+01
c	0 0 0	0 0 0	0 0 0	0 0 0	1 0 0	0 0 0	1 0 0
$\overline{v}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	5.05000e+01	0.00000e+00	7.30625e+04
mean	1.23905e+03	7.88427e-01	6.37736e+00	2.70176e+01	1.01549e+00	2.96309e+02	3.19541e+01
Worst	5.32177e+03	8.10585e-01	8.63931e+00	4.39822e+01	1.09605e+00	2.74868e+02	5.66894e+01
std	1.03932e+03	9.66942e-03	1.41057e+00	6.75260e+00	2.80498e-02	1.71702e+01	8.36820e+00
SR	84	100	100	100	0	84	0
$\overline{vio}$	3.71000e-01	0.00000e+00	0.00000e+00	0.00000e+00	5.05000e+01	3.50173e-05	7.30614e+04

TABLE XXII

RANKS BASED ON **MEAN SOLUTION** ON THE 28 FUNCTIONS OF $100D$ ON IEEE CEC2017 BENCHMARKS

Problem	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	Total
CAL_LSAHDE(2017)	10	10	11	11	6	5	11	11	7	9	5	8	10	11	10	11	11	10	1	11	11	8	11	11	11	11	8	1	251
LSHADE44+IDE(2017)	11	11	9	2	2	10	7	6	2	7	1	2	5	10	5	8	9	8	3	7	8	7	7	8	8	9	9	8	189
LSAHDE44(2017)	1	1	8	1	1	9	8	1	1	1	4	11	6	9	8	9	7	7	2	6	10	6	9	7	7	7	7	10	164
UDE(2017)	9	9	6	10	11	6	5	10	10	8	6	3	11	6	7	5	10	9	10	9	5	11	1	9	4	10	10	6	216
MA_ES(2018)	1	1	1	8	10	2	3	2	9	4	2	10	2	8	9	1	6	1	11	10	9	4	8	5	1	6	2	5	141
IUDE(2018)	1	1	10	9	5	11	10	3	5	5	3	4	7	4	4	6	4	6	8	8	3	9	5	6	5	4	6	9	161
LSAHDE_IEpsilon(2018)	8	8	7	3	9	8	6	7	6	6	9	7	4	5	6	7	8	2	6	5	4	10	4	10	6	8	4	11	184
DeCODE(2018)	1	1	1	7	7	4	9	9	3	10	11	9	1	7	3	4	5	11	7	4	2	2	10	2	3	5	11	7	156
HCO-DE	1	1	1	4	4	7	1	4	8	11	10	1	3	1	11	1	1	5	9	2	1	1	6	1	9	2	5	4	115
HECO-DE(FR)	1	1	5	6	3	1	2	5	11	2	7	5	9	1	2	10	2	3	3	3	6	3	3	4	10	1	1	2	112
HECO-DE	1	1	4	5	8	3	4	8	3	3	8	6	8	3	1	3	3	4	3	1	7	5	2	3	2	3	3	3	**108**

TABLE XXIII

RANKS BASED ON **MEDIAN SOLUTION** ON THE 28 FUNCTIONS OF $100D$ ON IEEE CEC2017 BENCHMARKS

Problem	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	Total
CAL_LSAHDE(2017)	10	10	11	11	8	6	11	11	7	9	5	7	10	3	10	11	11	10	1	3	4	8	6	11	11	11	10	1	227
LSHADE44+IDE(2017)	11	11	10	1	4	9	8	7	2	8	4	3	5	11	6	9	8	7	3	8	10	7	8	9	10	8	7	9	203
LSAHDE44(2017)	1	1	9	2	5	8	9	1	1	1	2	11	6	10	9	10	6	8	2	7	11	6	10	10	9	5	8	10	178
UDE(2017)	9	9	6	10	11	5	5	10	9	10	6	2	11	7	8	6	10	9	10	10	8	11	3	7	6	9	9	6	222
MA_ES(2018)	1	1	1	8	10	3	3	2	11	5	1	9	3	9	4	1	5	1	11	11	9	5	9	5	1	6	1	5	141
IUDE(2018)	1	1	8	9	6	10	7	3	10	4	3	4	7	5	7	7	4	6	7	9	3	9	5	6	7	4	6	8	166
LSAHDE_IEpsilon(2018)	8	8	7	3	9	7	6	8	8	6	9	8	4	6	4	8	7	2	6	6	5	10	4	8	8	7	5	11	188
DeCODE(2018)	1	1	1	7	7	4	10	9	4	7	11	10	2	8	3	5	9	11	8	5	1	2	11	2	5	10	11	7	172
HCO-DE	1	1	1	5	1	11	1	4	4	11	10	1	1	1	11	1	1	5	9	1	2	1	7	1	2	2	4	4	104
HECO-DE(FR)	1	1	5	6	1	1	2	6	3	2	7	5	9	1	1	3	2	3	3	4	6	3	2	3	4	1	3	2	**90**
HECO-DE	1	1	4	4	1	2	4	5	4	3	8	6	8	4	1	3	3	4	3	2	7	4	1	3	3	3	2	3	97