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 over past two decades. The idea behind these algorithms is to transform constrained optimisation problems into multi-objective optimisation problems without any constraint, and then solve them. In this paper, we propose a new multi-objective method for constrained optimisation, which works by converting 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 does 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”. In a “wide gap” problem, an algorithm needs exponential time to cross between two fitness levels (a wide gap). We conduct a case study for answering this question. We prove that using helper and equivalent objectives can shorten the time of crossing the “wide gap”. We improve the performance of EAs for solving some combinatorial optimisation problems without any constraint, and then solve them. In this competition, we used multi-objective evolutionary algorithms on IEEE CEC2017 benchmark suite includes 50 and 100 dimensional functions. For 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 sum of each constraint violation degree and the degree of constraint violation. So far few theoretical analyses have been reported for answering this question.

Evolutionary algorithms (EAs) have been applied to 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.

$$\min \bar{f}(\bar{x}) = (f(\bar{x}), v(\bar{x})), \quad \bar{x} \in \Omega. \quad (2)$$

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

$$v(\bar{x}) = \sum_{i=1}^{q} v_i^f(\bar{x}) + \sum_{r=1}^p v_r^E(\bar{x}). \quad (3)$$

$$v_i^f(\bar{x})$$ is the degree of violating the $i$th inequality constraint.

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

$$v_r^E(\bar{x})$$ is the degree of violating the $r$th equality constraint.

$$v_r^E(\bar{x}) = \max\{0, |g_r^E(\bar{x}) - \epsilon\}, \quad i = 1, \cdots, r, \quad (5)$$

where $\epsilon$ is a user-defined tolerance allowed for the equality constraint.

Multi-objective EAs for constrained optimisation have been proposed over past two decades. Many empirical studies have demonstrated the efficiency of the multi-objective method [4]. 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 CEC2017/18 constrained optimisation competitions adopted multi-objective optimisation [5]. The competition benchmark suite includes 50 and 100 dimensional functions. 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.

The above questions motivate us to further study the multi-objective method for COPs. Our work is inspired by helper objectives [6]. The use of helper objectives has significantly improved the performance of EAs for solving some combinatorial optimisation problems, such as job shop scheduling, travelling salesman and vertex covering [6], [7]. Our work is

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

$$\min f(\vec{x}), \quad \vec{x} = (x_1, \cdots, x_D) \in \Omega,$n

subject to

$$\begin{cases} g_i^f(\vec{x}) \leq 0, & i = 1, \cdots, q, \\ g_r^E(\vec{x}) = 0, & i = 1, \cdots, r, \end{cases} \quad (1)$$

where $\Omega = \{ \vec{x} \mid L_j \leq x_j \leq U_j, j = 1, \cdots, D \}$ is a bounded domain in $\mathbb{R}^D$. $D$ is the dimension. $L_j$ and $U_j$ denote lower and upper boundaries respectively. $g_i^f(\vec{x}) \leq 0$ is an inequality constraint and $g_r^E(\vec{x}) = 0$ 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 $\Omega^f, \Omega^I$ and $\Omega^F$ respectively.

Manuscript received xx xx xxxx

This work was partially supported by EPSRC under Grant No. EP/1008909/1.

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 is with the School of Science and Technology, Nottingham Trent University, Nottingham NG11 8NS, U.K. Email: jun.he@ntu.ac.uk (corresponding author)
also inspired by objective decomposition, which was recently adopted in multi-objective EAs for COPs [8], [11]. 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 a new equivalent and helper objectives method for COPs. A COP is converted into an optimisation problem consisting of equivalent and helper objectives but without any constraint. Here an equivalent objective means its optimal solution set is identical to $\Omega^*$, but a helper objective does not. Then this problem is solved by a decomposition-based multi-objective EA.

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

1) In theory, the “wide gap” problem [12], [13] is regarded as a hard problem to EAs. We aim at proving using helper and equivalent objectives can 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 CEC2017/18 competitions.

This paper is a significant extension of our two-page poster presented in GECCO2019 [14]. The algorithm described in the current paper is a slightly revised version of HECO-DE in [14]. HECO-DE was ranked 1st in 2019 in IEEE CEC Competition on Constrained Real Parameter Optimization when compared with other eight state-of-art EAs [5].

The paper is organised as follows: Section III is literature review. Section IV describes the helper and equivalent objective method. Section V theoretically analyses this method. Section VI conducts a case study. Section VII reports experiments and results. Section VIII 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 I with two objectives, the original objective $f$ and a degree of violating constraints $v$ [8], [10], [11], [18], [21].

2) Scheme II with many objectives, he original objective $f$ and degrees of violating each constraint $v_i$ [22], [23].

3) Scheme III with helper objective(s) besides the original objective or the degree of constraint violation [24]–[27]. For example, the penalty function forms helper objective.

The first scheme is the most widely used one so far. Ji et al. [28] converted a berth allocation problem with constraints into problem (1) 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 $\epsilon$-constrained method.

Recently, decomposition-based multi-objective EAs have applied to solving problem (2). Xu et al. [8] decomposed problem (2) into a tri-objective problem using the weighted sum method with static weights and solved it using a Pareto-ranking based DE algorithm. Wang et al. [11] decomposed problem (2) using the weighted sum method into a number of subproblems with dynamical weights and solved these subproblems by DE. Peng et al. [10] 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 helper objective. Zeng et al. [9] designed a niche-count objective besides the original objective and a constraint-violation objective and proposed an dynamic constrained multiobjective evolutionary algorithm (DCMOEA). 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. Recently, these EAs with dynamic constrained multi-objectives were further improved by adding the feasible-ratio control technique [30] and a dynamic constraint boundary [31].

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. Helper objectives are also used to add more search directions. Under this framework, we have designed HECO-DE and HECO-PDE [27]. HECO-PDE is an enhanced version of HECO-DE with principle component analysis. A multi-population implementation of HECO-DE is designed in [32] which is suitable for parallel processing.

In order to speed up the convergence of EAs for COPs, Deb and Datta [24] observed that the hybridisation of multi-objective EAs and local search can reduce the number of fitness evaluations by one or more orders of magnitude. However, the current paper will not discuss the benefit of hybridisation but only focus on using helper and equivalent objectives.

The theoretical analysis of multi-objective EAs for constrained optimisation is still rare and limited to combinatorial optimisation. He et al. [33] proved that a multi-objective EA with helper objectives is a 1/2-approximation algorithm for the knapsack problem. Recently, Neumann and Sutton [34] analysed the running time of a variant of Global Simple Evolutionary Multiobjective Optimizer on the knapsack problem. Nevertheless, no general theoretical analysis exists for the multi-objective EAs in continuous COPs.
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 \([\mathbb{1}]\). The Pareto optimal set to \([\mathbb{1}]\) is often significantly larger than \(\Omega^*\).

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

\[
\begin{align*}
\text{min } & f(x) = x, \quad x \in [-1000, 1000], \\
\text{subject to } & g(x) = \sin\left(\frac{x\pi}{1200}\right) \geq 0.
\end{align*}
\]

The degree of constrain violation is
\[
v(x) = \max\{0, -\sin(x\pi/1200)\}.
\]

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

![Pareto front](image)

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

In order to develop a theory of understanding the multi-objective method for constrained optimisation problems, \([\mathbb{36}], [\mathbb{37}]\), we introduce the concepts of equivalent and helper objectives. The term “helper” originates from \([\mathbb{6}]\).

**Definition 1:** A scalar function \(g(\mathbf{x})\) defined on \(\Omega\) is called an equivalent objective function with respect to the COP \([\mathbb{1}]\) if it satisfies the condition:
\[
\arg\min\{f(\mathbf{x}); \mathbf{x} \in \Omega\} = \Omega^*.
\]

A scalar function \(g(\mathbf{x})\) 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 \(\Omega_F\) denote feasible solutions and \(\Omega_I\) infeasible ones.
\[
\min e(\mathbf{x}) = \begin{cases} f(\mathbf{x}), & \text{if } \mathbf{x} \in \Omega_F, \\ +\infty, & \text{if } \mathbf{x} \in \Omega_I. \end{cases}
\]

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

In practice, it is more convenient to construct an equivalent function \(e(\mathbf{x})\) which is defined on population \(P\), rather than \(\Omega\). 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 \(g(\mathbf{x})\) defined on \(P\) is called an equivalent objective function with respect to the COP \([\mathbb{1}]\) if it satisfies the following condition:
\[
\arg\min\{f(\mathbf{x}); \mathbf{x} \in \Omega \cap P\} = \Omega^* \cap P.
\]

A scalar function \(g(\mathbf{x})\) 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 cannot distinguish equivalent and helper functions defined on the population.

An example is the superiority of feasibility rule \([\mathbb{35}]\) 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
\[
e(\mathbf{x}) = \begin{cases} f(\mathbf{x}), & \text{if } \mathbf{x} \in \Omega_F \cap P, \\ v(\mathbf{x}) + f_F(P), & \text{if } \mathbf{x} \in \Omega_I \cap P, \\ 0, & \text{otherwise.} \end{cases}
\]

where \(f_F(P) = \max\{f(\mathbf{x}), \mathbf{x} \in \Omega_F \cap P\}\) 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 \([\mathbb{1}]\) can be converted to a single-objective optimisation problem without any constraint.
\[
\min e(\mathbf{x}), \quad \mathbf{x} \in P.
\]

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

A single-objective EA (SOCO) for problem \([\mathbb{11}]\) is described as follows.

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

\(T_{\text{max}}\) is 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 \(e(\mathbf{x})\), we add several helper functions \(h_i(\mathbf{x}), i = 1, \cdots, k\), and then obtain a helper and equivalent objective optimisation problem on population \(P\).
\[
\min \tilde{f}(\mathbf{x}) = (e(\mathbf{x}), h_1(\mathbf{x}), \cdots, h_k(\mathbf{x})), \quad \mathbf{x} \in P.
\]

Furthermore, we decompose problem \([\mathbb{12}]\) into several single objective problems. Decomposition-based multi-objective EAs have been proven to be efficient in solving multiobjective optimisation problems \([\mathbb{36}], [\mathbb{37}]\). The decomposition method
in the present work adopts the weighted sum approach, adding
the helper objective onto the equivalent objective such that
\[ \min w_0 e(\vec{x}) + \sum_{j=1}^{k} w_j h_j(\vec{x}), \quad \vec{x} \in P, \] (13)
where \( w_j \geq 0 \) are weights.

Problem (12) is transformed into \( \lambda \) single-objective optimisation subproblems by assigning \( \lambda \) tuples of weights \( \vec{w}_i = (w_{i0}, w_{i1}, \ldots, w_{ik}) \).
\[ \min f_i = w_i e + \sum_{j=1}^{k} w_j h_j, \quad i = 1, \ldots, \lambda. \] (14)
At least one \( f_i \) is chosen to an equivalent objective 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 \([0, 1]\).
\[ g(\vec{x}) \leftarrow \frac{g(\vec{x}) - \max_{\vec{y} \in P} g(\vec{y})}{\max_{\vec{y} \in P} g(\vec{y}) - \min_{\vec{y}} g(\vec{y})}. \] (15)

A helper and equivalent objective EA (HECO) for problem (14) is described as follows.
1: population \( P_0 \leftarrow \) initialise a population of solutions;
2: for \( t = 0, \ldots, T_{\text{max}} \) do
3: adjust weights;
4: population \( C_t \leftarrow \) generate a population of solutions from \( P_t \) subject to a conditional probability \( \Pr(C_t \mid P_t) \);
5: \( P_{t+1} \leftarrow \) select optimal solution(s) to \( \min f_i(\vec{x}), \vec{x} \in P_t \cup C_t \) for \( i = 1, \ldots, \lambda \) where \( f_i \) is calculated by formula (14); remove repeated solutions.
6: 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 \((e, h_1, \ldots, h_k)\) (no Pareto-based ranking).

Since our goal is to find the optimal solution(s) to \( \min e(\vec{x}) \) but not to \( \min h(\vec{x}) \), 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 \([36]\). 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:
\[ \lim_{t \to +\infty} w_{hi,t} > 0 \quad \text{and} \quad \lim_{t \to +\infty} w_{ji,t} = 0 \quad \text{for} \quad j > 0. \] (16)

HECO has two characteristics:
1) SOCO is one-dimension search along the direction \( e \) in the objective space. HECO is multi-dimensional search along several directions \((e, h_1, \ldots, h_k)\). \( e \) is the main search direction for SOCO, while \( h_1, \ldots, h_k \) 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 later.
2) The dynamically weighting ensures that at the beginning, HECO explores different directions \((e, h_1, \ldots, h_k)\), 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 aid of an equivalent objective, a decomposition-based multi-objective EA for COPs faces a problem. The solution set found by the algorithm is often larger than \( \Omega^* \). This claim is shown through Example 1. We assign \( \lambda \) pairs of weights in objective decomposition: \((1, 0), (w_i, 1 - w_i), (0, 1)\) where \( i = 2, \ldots, \lambda - 1 \) and \( w_i > 0 \) and obtain \( \lambda \) subproblems with a bounded constraint \( x \in [-1000, 1000]\).

The optimal solution to \( \min f \) is \( x = -1000 \). The optimal solution to \( \min f_i, i = 2, \ldots, \lambda - 1 \) is infeasible. The optimal solution to \( \min v \) is \([0, 500]\). The solution set to the \( \lambda \) 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. But the superiority of feasibility rule is an equivalent objective \([10]\), thus, many multi-objective EAs for COPs implicitly utilise an equivalent objective. Based on this argument, multi-objective EAs for COPs are classified into three types.

1) Type I is to optimise helper objectives only;
2) Type II is to optimise helper objectives but select solutions by the superiority of feasibility rule (an implicit equivalent objective);
3) Type III is to explicitly optimise both helper and equivalent objectives.

In this paper, the notation HECO refers to type III. It has some advantages: an explicit equivalent objective is utilised and it can be designed more flexibly 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 \( \Omega^* \) 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 can help bypass it. In theory, we aim at mathematically proving the conjecture: using helper and equivalent objectives can shorten the time of crossing the “wide gap”. First we introduce several preliminary definitions and a lemma.
If $k = 1$, it degenerates into a single-objective problem.

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

1. $\forall i \in \{1, \ldots, k\}$: $f_i(\mathbf{x}) \leq f_i(\mathbf{y})$;
2. $\exists i \in \{1, \ldots, k\}$: $f_i(\mathbf{x}) < f_i(\mathbf{y})$.

Definition 4: A set $\mathcal{S}$ is called a non-dominated set in $\mathcal{S}$ if and only if it is not dominated by $\mathcal{x}$. A set $\mathcal{S}$ is called a Pareto optimal set if and only if it is a non-dominated set in $\Omega$.

Given a target set, the hitting time is the number of generations for an EA to reach the set $\mathcal{S}$. The hitting time of an EA from one set to another is defined as follows.

Definition 5: Let $\{I_t; t = 0, 1, \ldots\}$ be a population sequence of an EA. Given two sets $\mathcal{S}_1$ and $\mathcal{S}_2$, the expected hitting time of the EA from $\mathcal{S}_1$ to $\mathcal{S}_2$ is defined by

$$T(\mathcal{S}_2 | \mathcal{S}_1) := \sum_{t=0}^{+\infty} \Pr(P_0 \subset \mathcal{S}_2, \ldots, P_t \subset \mathcal{S}_2),$$

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

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

Lemma 1: Let $\{P_t; t = 0, 1, \ldots\}$ and $\{P'_t; t = 0, 1, \ldots\}$ be two population sequences and $\mathcal{S}_1$ and $\mathcal{S}_2$ two sets such that $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$. Let $P_0 = P'_0 = \mathcal{S}_1$. If for any $t$,

$$\Pr(P_0 = S_1 \subset \mathcal{S}_2, \ldots, P_t \subset \mathcal{S}_2) \geq \Pr(P'_0 = S_1 \subset \mathcal{S}_2, \ldots, P'_t \subset \mathcal{S}_2),$$

then $T(\mathcal{S}_2 | \mathcal{S}_1) \geq T'(\mathcal{S}_2 | \mathcal{S}_1)$. Furthermore, if the inequality holds strictly for some $t$, then $T(\mathcal{S}_2 | \mathcal{S}_1) > T'(\mathcal{S}_2 | \mathcal{S}_1)$.

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 average drift analysis. This will not be discussed in the current paper.

### B. Fundamental Theorem

Now we compare SOCO for the single-objective problem and HECO for the helper and equivalent objective problem. 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 [12], [13], 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 $e(\mathbf{x})$ plays the role of “fitness”. In constrained optimisation, function $f(\mathbf{x})$ is not suitable as “fitness” because the minimum value of $f$ might be obtained by an infeasible solution.

The values of $e(\mathbf{x})$ are split into fitness levels: $FL_0 < FL_1 < \cdots < FL_m$ and the search space $\Omega$ is split into disjoint level sets: $\Omega = \cup_{i=0}^{m} L_i$ where $L_i = \{ x; e(\mathbf{x}) = FL_i \}$. Given a fitness level $FL$ and its corresponding point set $L$, let $L^b$ denote points at better levels $L^b := \{ x; e(\mathbf{x}) < FL \}$. 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 $T(L^b | A \subset L)$ is an exponential function of the dimension $D$.

Several conditions are needed for mathematically comparing SOCO and HECO. Let $\{P_t; t = 0, 1, \ldots\}$ represent the population sequence from SOCO and $\{P'_t; t = 0, 1, \ldots\}$ from HECO. Assume $P_0 = P'_0$ are chosen from the fitness level $FL$. For SOCO, thanks to elitist selection, its offspring are either $L$ or $L^b$. 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 $P_t = \mathbf{x} = (x_1, \ldots, x_m) \subset L$, there is a one-to-many mapping from $P_t$ to $P'_t$ where $P'_t$ is in the set

$$\{ \mathbf{x}' = (x_1, \ldots, x_m, \ast) | \ast = \emptyset \text{ or } \ast \in L \cup L^b \}.$$
Its Pareto front is displayed in Fig. 2.

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

$$
\Pr(P_0 \subset \overline{L^t}, \ldots, P_t \subset \overline{L^t}) = \sum_{A \subset L} \ldots \sum_{Z \subset L} \Pr(P_0 = A, \ldots, P_t = Z) \\
\geq \Pr(P'_0 \subset \overline{L^t}, \ldots, P'_t \subset \overline{L^t}) \\
= \sum_{A \subset L} \ldots \sum_{Z \subset L} \Pr(P_0 = A, \ldots, P_t = Z').
$$

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 path $* \cdots *$ 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,

$$
\begin{align*}
\min f(x) &= x, & x \in [-500, 3000] \\
\text{subject to } g(x) &= \sin(\frac{x}{100}) \geq 0.
\end{align*}
$$

Its optimal solution is $x = 0$. The feasible region is $\Omega_E = [0, 1000] \cup [2000, 3000]$. The objective function $f(x)$ 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.

$$
\min e(x) = \begin{cases} 
 f(x), & \text{if } x \in \Omega_E, \\
 u(x) + 3000, & \text{if } x \in \Omega_I.
\end{cases}
$$

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

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.

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

$$
\min \bar{f}(x) = (e(x), f(x)), \quad x \in [-500, 3000].
$$

Its Pareto front is displayed in Fig. 2.

We assign two pairs of weights: $\bar{w}_1 = (1, 0)$ and $\bar{w}_2 = (0, 1)$ 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 $t \geq 2000$,

$$
\Pr(P'_0 \subset \overline{L^t}, \ldots, P'_t \subset \overline{L^t}) < 1.
$$

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 follow Occam’s razor, that is to construct a HECO algorithm from a SOCO algorithm such that their search operators are identical but their objectives are different. No extra operation is added to HECO. For comparative purpose, LSHADE44 [39] is chosen as the SOCO algorithm because it is ranked only 4th in the CEC2017/18 competition [5]. If the constructed HECO algorithm outperforms LSHADE44 and winner 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 [40]). Mutant point $\bar{u}_i$ is generated from target point $\bar{x}_i$ by

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

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

The second mutation is randll/1 mutation (see (3) in [41]).

$$
\bar{u}_i = \bar{x}_{r_1} + F(\bar{x}_{r_2} - \bar{x}_{r_3}),
$$

$$
\bar{u}_i = \bar{x}_{r_1} + F(\bar{x}_{r_2} - \bar{x}_{r_3}).
$$

In (24), mutually distinct $\bar{x}_{r_1}, \bar{x}_{r_2}$ and $\bar{x}_{r_3}$ are randomly chosen from population $P$. They are also different from $\bar{x}_i$. In (25), $\bar{x}_{r_1}, \bar{x}_{r_2}$ and $\bar{x}_{r_3}$ are chosen as that in (24) but then are ranked. $\bar{x}_{r_1}^\star$ denotes the best, while $\bar{x}_{r_2}^\star$ and $\bar{x}_{r_3}^\star$ denote the other two.
resher and each point which is rewritten by its successful trial
At the beginning of search, the archive is empty. During a
required initial size is set to

where \( q_k \) is the count of the \( k \)-th strategy's successes, and \( n_0 > 0 \) is a constant.

LSHADE44 employs a mechanism of competition of strategies [44, 45] to create trial points. The \( k \)-th strategy is chosen subject to a probability \( q_k \). All \( q_k \)-s are initially set to the same value, i.e., \( q_k = 1/4 \). The \( k \)-th 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:

\[
q_k = \frac{n_k + n_0}{\sum_{i=1}^{4} (n_i + n_0)},
\]

where \( n_k \) is the count of the \( k \)-th strategy's 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 \( CR \) [39]. 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 [39]. The maximal size of archive \( A \) is set to \( |A|_{\text{max}} \). 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 \( |A|_{\text{max}} \), then \( |A| - |A|_{\text{max}} \) individuals are randomly removed from \( A \).

LSHADE44 takes a mechanism to linearly decrease the population size [39, 46]. 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 \( N_{T_{\text{max}}} \). The required size at the \( t \)-th generation is set by the formula:

\[
N_t = \text{round} \left( N_0 - \frac{t}{T_{\text{max}}}(N_0 - N_{T_{\text{max}}}) \right).
\]

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 in population \( P \),

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

For each \( x \in P \), \( \tilde{e}(x) \) denotes the fitness difference between \( f(x) \) and \( f(x_p^*) \).

\[
\tilde{e}(x) = |f(x) - f(x_p^*)| \tag{29}
\]

\( \tilde{e} \) itself is not an equivalent function because in some problems, the fitness of an infeasible solution is equal to \( f(x_p^*) \) too. An equivalent function on population \( P \) is defined as

\[
e(x) = w_1\tilde{e}(x) + w_2v(x), \tag{30}
\]

where \( w_1, w_2 > 0 \) are weights, which are used to control the contribution of \( \tilde{e} \) and \( v \) to the equivalent function \( e \). The number of such equivalent functions is infinite because \( w_1 \in (0, +\infty), w_2 \in (0, +\infty) \).

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

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

If two solutions \( x_1 \) (infeasible) and \( x_2 \) (feasible) in population \( P \) satisfy

\[
w_1|f(x_1) - f(x_p^*)| + w_2v(x_1) < w_1|f(x_2) - f(x_p^*)|, \tag{31}
\]

then under the equivalent objective function \( e \), infeasible \( x_1 \) is better than feasible \( x_2 \). This feature may help search the infeasible region. For example, in Fig. 4 assume that \( f(x_1) - f(x_p^*) = 0 \) and \( f(x_2) - f(x_p^*) = 1 \), \( v(x_1) = 0.5 \) and \( w_1 = w_2 \). Then we have \( e(x_1) = 0.5e(x_2) \). Starting from \( x_1 \), it is much easier to reach the right feasible region in which the optimal feasible solution \( x_p^* \) locates.

![Fig. 4](image)

There exist two feasible regions. An infeasible \( x_1 \) satisfying [33] is better than \( x_2 \) under the equivalent objective function \( e \). This may help population \( P = \{x_1, x_2, x_p^*\} \) move from the right feasible region to the left feasible region in which the optimal feasible solution \( x_p^* \) locates.

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

\[
\min \bar{f}(\bar{x}) = (e(\bar{x}), f(\bar{x})), \quad \bar{x} \in P, \tag{32}
\]

The problem is decomposed into \( \lambda \) single objective subproblems through the weighted sum method: for \( i = 1, \ldots, \lambda \),

\[
\min f_i(\bar{x}) = w_1\tilde{e}(\bar{x}) + w_2v(\bar{x}) + w_{3i}f(\bar{x}). \tag{33}
\]

An extra term \( \tilde{e} \) is added besides the original objective function \( f \) and constraint violation degree \( v \).
C. A New multi-objective EA for Constrained Optimization

A HECO algorithm is designed which reuses search operators from LSHADE44 [39]. 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.

1: Initialise algorithm parameters, including the required initial population sizes $N_0$ and final size $N_{r_{\text{max}}}$, the maximum number of fitness evaluations $FES_{\text{max}}$, circle memories for parameters $F$ and $CR$, the size of historical memories $H$; initial probabilities $q_k$ of four strategies, and external archive $A$;
2: Set the counter of fitness evaluations $FES$ to 0, and the counter of generations $t$ to 0;
3: Randomly generate $N_0$ solutions and form an initial population $P_0$;
4: Evaluate the value of $f(x)$ and $v(x)$ for each $x \in P_0$;
5: Increase counter $FES$ by $N_0$;
6: while $FES \leq FES_{\text{max}}$ (or $t \leq T_{\text{max}}$) do
7:   Adjust weights in objective decomposition.
8:   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 $\emptyset$.
9:   Randomly select $\lambda$ individuals (denoted by $Q$) from $P$ and then denote the rest individuals $P \setminus Q$ by $P'$;
10:  for $x_i$ in $Q$, $i = 1, \ldots, \lambda$ do
11:   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 $y_i$ by applying the selected strategy;
13:  Evaluate the value of $f(y_i)$ and $v(y_i)$;
14:  Add $y_i$ to subpopulation $Q$, resulting in an enlarged subpopulation $Q'$;
15:  Normalise $\tilde{e}(x)$, $f(x)$ and $v(x)$ for each individual $x$ in $Q'$;
16:  Calculate $f_i$ value for $\tilde{x}_i$ and $y_i$ according to formula (33).
17:  if $f_i(y_i) < f_i(\tilde{x}_i)$ then
18:   Add $y_i$ into children population $C$ and $\tilde{x}_i$ 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: Update circle memories $M_F$ and $M_{CR}$ using respective sets $S_F$ and $S_{CR}$ for each strategy (see its detail in LSHADE44 [39]);
22: Merge subpopulation $P'$ (not involved in mutation and crossover) and children $C$ and form new population $P$;
23: Calculate the required population size $N_i$;
24: if $N_i < |P|$ then
25: Randomly delete $|P| - N_i$ individuals from $P$;
26: end if
27: Calculate the required archive size $|A|_{\text{max}} = 4N_t$;
28: end while

In HECO-DE, weights are designed to linearly increase (for $w_1$, $w_2$) or decrease (for $w_3$) over $t$ and also linearly

D. A New Mechanism of Dynamical Adjustment of Weights

We propose a special mechanism for dynamically adjusting weights. The new mutation is called current-to-Qbest/1 mutation. For rand1/1 mutation (25), $\tilde{x}_i$, $\tilde{x}_r_1$, $\tilde{x}_r_2$, and $\tilde{x}_r_3$ are not compared but just randomly selected from subpopulation $Q$. Thus it returns to the original rand1/1 mutation (24).

This new mutation is called current-to-Qbest/1 mutation. For rand1/1 mutation (25), $\tilde{x}_i$, $\tilde{x}_r_1$, $\tilde{x}_r_2$, and $\tilde{x}_r_3$ are not compared but just randomly selected from subpopulation $Q$. Thus it returns to the original rand1/1 mutation (24).

Lines 12-16: ranking individuals is used in both mutation (25) and calculation of the equivalent function (30). Because ranking is restricted within subpopulation $Q$ and its size $\lambda$ 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 $f_i(y_i) < f_i(\tilde{x}_i)$, then $y_i$ is accepted and added into children population $C$. HECO-DE minimises $\lambda$ 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 $\lambda$ 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 [39].
increase (for \( w_1, w_2 \)) or decrease (for \( w_3 \)) over \( i \). In more detail, weights are given by

\[
\begin{align*}
    w_{1i,t} &= \frac{t}{T_{\text{max}}} \frac{i}{\lambda}, \\
    w_{2i,t} &= \frac{t}{T_{\text{max}}} \frac{i}{\lambda} + \gamma, \\
    w_{3i,t} &= \left(1 - \frac{t}{T_{\text{max}}} \right) \left(1 - \frac{i}{\lambda}\right),
\end{align*}
\]

where \( \lambda \) is the number of subproblems, \( T_{\text{max}} \) 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 \( \gamma \) and \( w_2 \).

Figures 5 and 6 depict the change of normalised weights over \( t/T_{\text{max}} \). For \( \lambda \)th individual, weights \( w_{1\lambda} > 0, w_{2\lambda} > 0 \) but \( w_{3\lambda} = 0 \). This individual minimises an equivalent function \( f_{\lambda} \). 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. 5. The change of weights for 1st and \( \lambda \)th individuals on CEC2006 benchmark functions. \( \gamma = 0.7 \).

Fig. 6. The change of weights for 1st and \( \lambda \)th individuals on CEC2017 benchmarks. \( \gamma = 0.1 \).

VI. COMPARATIVE EXPERIMENTS AND RESULTS

A. Experimental Setting

HECO-DE was tested on two well-known benchmark sets. The first set is from IEEE CEC2017 Competition and Special Session on Constrained Single Objective Real-Parameter Optimization [5] which consists of 28 scalable functions with dimension \( D = 10, 30, 50, 100 \) (total \( 4 \times 28 \) benchmarks). The second set is from the IEEE CEC2006 Special Session on Constrained Real-parameter Optimization [47] which consists of 24 functions. According to [47], 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 used in HECO-DE. In Table I parameters inherited from LSHADE44 are set to values similar to LSHADE44 [39].

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda )</td>
<td>( \lambda = 15 )</td>
</tr>
<tr>
<td>constraint violation bias</td>
<td>( \gamma = 0.7 )</td>
</tr>
<tr>
<td>( N_0 )</td>
<td>( N_0 = 450, N_{T_{\text{max}}} = \lambda )</td>
</tr>
<tr>
<td>( \text{FES}_{\text{max}} ) from CEC2006 benchmarks</td>
<td>( \text{FES}_{\text{max}} = 500,000 )</td>
</tr>
<tr>
<td>( \text{FES}_{\text{max}} ) from CEC2017 benchmarks</td>
<td>( \text{FES}_{\text{max}} = 20000D )</td>
</tr>
</tbody>
</table>

B. Experimental results on IEEE CEC2017 benchmarks

HECO-DE was compared with seven single-objective EAs in CEC2017/18 constrained optimisation competitions, which are CAL-SHADE [48], LSHADE44+IDE [49], LSHADE44 [39], UDE [50], MA-ES [51], IUDE [52], LSHADE-Epsilon [53], and one decomposition-based MOEA, DeCODE [11].

HECO-DE was also compared with its two variants. The first variant is to remove the equivalent function from HECO-DE. In the weighted sum (35), \( \hat{e}(\vec{x}) \) 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 weighted sum (35), \( \hat{e}(\vec{x}) \) 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 CEC2017/18 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]. The rank value of each algorithm on each dimension was calculated as below:

\[
\text{Rank value} = \sum_{i=1}^{28} \frac{\text{rank}_i}{\text{mean value}} + \sum_{i=1}^{28} \text{rank}_i \text{ (by median solution)}.
\]

\( (39) \)

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 \( \lambda \) and \( \gamma \). HECO-DE with all five \( \lambda \) and \( \gamma \) values had obtained lower total ranks than other EAs.

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

**TABLE III**

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

**TABLE IV**

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

**TABLE V**

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

C. Experimental results on IEEE CEC2006 benchmarks

HECO-DE was compared with five EAs, which are CMODE [20], NSES [54], FROFI [55], DW [10] and DeCODE [11], on IEEE CEC2006 benchmarks.

Table VII summarises experiment results, where “Mean” and “Std Dev” denote the mean and standard deviation of objective function values, respectively. As suggested in [47], a successful run is a run during which an algorithm finds 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 VII, \( \ast \) denotes that the algorithm satisfies this successful rule in 25 runs for a test problem.

As shown in Table VII 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. Table VII shows that HECO-DE always find the optimum on all test functions. But without an equivalent objective, HECO-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 has proposed a 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 can shorten the time of crossing the “wide gap”. This general theoretical result shows the strengths of multi-objective EAs in solving COPs.

A case study has been conducted for validating our method. An algorithm, called HECO-DE, has been implemented which employs both helper and equivalent objectives and reuses search operators from LSHADE44 [39]. A new equivalent function and a new mechanism of dynamically weighting are designed in HECO-DE. Experimental results show that the overall performance of HECO-DE is ranked first when compared with other state-of-art EAs on CEC2017 benchmarks. HECO-DE also performs well on CEC2006 benchmarks.

For future work, we will consider each constraint violation pressure than the equivalent function (30).
REFERENCES


