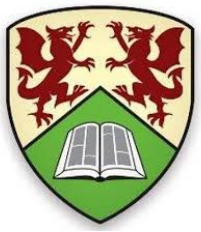
Multi-objective Evolutionary Algorithms for Single-objective Constrained Optimisation



Tao Xu

Supervisor: Dr. Jun He
Dr. Changjing Shang

Department of Computer Science
Aberystwyth University

This dissertation is submitted for the degree of
Doctor of Philosophy

Computer Science August 2021

PRIFYSGOL ABERYSTWYTH UNIVERSITY

Mandatory Layout of Declaration/Statements

Word Count of thesis: DECLARATION	43016
This work has not previously been accepted in substance for any degree and is not concurrently submitted in candidature for any degree.	
Candidate name	Tao Xu
Signature:	
Date	01/07/2021

STATEMENT 1

This thesis is the result of my own investigations, except where otherwise stated. Where *__correction services__ have been used, the extent and nature of the correction is clearly marked in a footnote(s).

Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signature:	
Date	01/07/2021

[*this refers to the extent to which the text has been corrected by others]

STATEMENT 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signature:	
Date	01/07/2021

NB: *Candidates on whose behalf a bar on access (hard copy) has been approved by the University should use the following version of Statement 2:*

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loans after expiry of a bar on access approved by Aberystwyth University.

Signature:	
Date	01/07/2021

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisor, Dr. Jun He, for his supervision, guidance, and encouragement during my PhD study at the Aberystwyth University. Without his continuous and illuminating instruction, this thesis could not have reached its present form. I also sincerely thank him for his enthusiastic support in my life.

I would also like to my second supervisor Dr. Changjing Shang and Prof. Qiang Shen for their insightful suggestions and comments on my thesis work.

Special acknowledgement is given to my group mates in the Department of Computer Science in Aberystwyth University, Dr. Christine Zarges, Dr. Thomas Jansen, Dr. Richard Jensen, Dr. Neil Mac Parthalain, Dr. Zhengpeng Li, Pu Zhang, Fangyi Li, Jing Yang, Kieran Stone and Muhammad Ismail. Discussions with them gave me lots of inspirations.

Finally, I wish to thank my parents for their unconditional love, support, and encouragement.

Abstract

In the real world there are a large number of optimization problems, especially in scientific research and engineering practice, which often have constraints and sometimes more than one objective. Due to the different characteristics of the problems themselves, traditional methods in operations research are no longer able to solve them independently. Evolutionary algorithms (EAs), as a global optimization method based on group search, are well suited for solving constrained optimization problems and multi-objective optimization problems. Therefore, evolutionary optimization has received increasing attention from researchers. The aim of this thesis is to design efficient multi-objective evolutionary algorithms (MOEAs) and strategies for constrained single-objective optimization problems (CSOPs) through in-depth exploration, and to conduct corresponding theoretical and experimental analysis. Specifically, the main research work of this thesis includes the following aspects.

Firstly, we consider a many-objective method for solving CSOPs. The method keeps the standard objectives: the original objective function and the sum of the degrees of constraint violation. Besides them, more objectives are added into the method. One objective is based on the feasible rule. The others come from the penalty function method. Then a multi-objective differential evolution algorithm is applied to solving multi-objective optimization problems with two, three and four objectives. An experimental study on thirteen benchmark functions from IEEE CEC2006 Competition is conducted. Experimental results confirm our expectation that adding more objectives could be useful and the solution quality is improved.

Secondly, we construct a new multi-objective evolutionary framework for solving CSOPs, which works by converting a CSOP into a problem with helper and equivalent objectives (HECO). 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 optimization 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 analyze 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 prove that using helper and equivalent objectives can shorten the time of crossing

the “wide gap”. A series of derivative algorithms are then designed based on HECO, such as HECO-DE, HECO-DEtch and HECO-DEm. Extensive experimental studies show these algorithms perform much better in solving benchmark problems in IEEE CEC2017/2018 Competition, and IEEE CEC2006 Competition than other state-of-the-art EAs.

Finally, apart from constraint handling techniques, we also contribute to new search operators which lead to further improvement our EAs for solving CSOPs. Two methods of studying valleys on a fitness landscape. Afterwards, the principle component analysis (PCA) could be used to characterize fitness landscapes. Based on this finding, a new search operator, called PCA-projection, is proposed. In order to verify the effectiveness of PCA-projection, we design two algorithms enhanced with PCA-projection for solving CSOPs. Experiment results indicate that the new search operator based on PCA-projection works very well.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Background

Optimization problem refers to finding the optimal solution or parameter value among many solutions or parameter values to make one or more functional indicators optimal, or to maximize or minimize some performance indicators of the system under certain conditions. Optimization problems exist widely in many fields such as signal processing, image processing, production scheduling, task allocation, pattern recognition, automatic control and mechanical design [106, 4, 78, 69]. Optimization methods are a mathematical-based application technique for solving various optimization problems. Various optimization methods have been widely used in the above-mentioned fields, and have produced great economic and social benefits. It has been proved that optimization methods can improve system efficiency, reduce energy consumption, and use resources rationally, and this effect becomes more obvious as the scale of the processing object increases.

Many complex optimization problems are constantly emerging in many disciplines such as electronics, communications, computers, automation, robotics, economics, and management. In the face of these large optimization problems, traditional optimization methods (e.g., Newton's method [93], simplex method [100], etc.) require traversing the entire search space, which cannot be completed in a short time and is prone to "combinatorial explosion" of the search. For example, many engineering optimization problems often require searching for optimal or quasi-optimal solutions in a complex and large search space. In view of the complexity, nonlinearity, constraint and difficulty of modeling, the search for efficient optimization algorithms has become one of the main research contents of related disciplines.

Inspired by human intelligence, the social nature of biological groups or the laws of natural phenomena, many intelligent optimization algorithms [53, 35, 68] have been invented to solve the above-mentioned complex optimization problems, mainly including: genetic

algorithms that mimic the evolutionary mechanism of organisms in nature; differential evolutionary algorithms that optimize search through cooperation and competition among individuals in a group; the particle swarm algorithm that simulates the group behavior of birds and fish; the simulated annealing algorithm that originates from the annealing process of solid matter; and so on. These algorithms have one thing in common, that is, they are developed by simulating or revealing certain phenomena and processes in nature or intelligent behaviors of biological groups; they are called intelligent optimization algorithms in the field of optimization, and they are characterized by simplicity, generality, and ease of parallel processing.

1.2 Motivation of Thesis

In real-world optimization problems, such as engineering design, operation scheduling, intelligent control, traffic optimization, financial investment, network communication, etc., there are often many constraints that pose great challenges for problem solving [90, 22]. Such optimization problems are called constrained optimization problems (COPs). COP is an important problem in the field of optimization, and the constraints in COP usually include upper and lower bound constraints on decision variables, equation/inequality constraints. According to the mathematical properties of the constraints, they can be classified as linear/nonlinear constraints. The existence of constraints leads to infeasibility domains in the search space of decision variables (the search space consists of two parts: feasible and infeasible domains). From unconstrained optimization to constrained optimization, the original single optimization objective must consider both the optimization objective and the constraint, but the effective unconstrained optimization method is at a loss when dealing with the constrained optimization problem. Therefore, it is of great theoretical significance and practical value to study the constrained optimization problem.

Constrained single-Objective optimization problem (CSOP) is the single-objective case of COP. It is of general interest to study CSOP. Traditional optimization algorithms solve such problems based on gradient information, which is only applicable to the case where the objective function and constraints are differentiable, and the solutions are mostly locally optimal. EA is a global optimization method that simulates a natural process, in which individuals are used to represent the solution of the problem to be solved, and a certain number of different individuals are formed into a population. Starting from the initial population, the population is guided to evolve by mutation, crossover, selection and other operations to mimic the evolutionary process in nature, so that the individuals in the population gradually approach the optimal solution of the problem. Compared with traditional optimization

algorithms, EAs are population-based search techniques, which have the characteristics of robustness, high search efficiency, and not easy to fall into the local optimum, so it is more suitable for solving CSOPs. However, EA is an unconstrained optimization technique, which must be combined with certain constraint handling mechanism to form a constrained optimization evolutionary algorithm.

Over the last two decades, there is an extensive study of using evolutionary multi-objective optimization (EMO) methods solving CSOPs, and this process is referred to as MOCO. Although those studies claimed that escaping from local optima with MOCO could be easier than with SOCO (evolutionary single-objective optimization method for solving CSOPs), and better results were obtained by MOCO when compared with SOCO [125]. Unfortunately, no EA in the most recent competitions (e.g. IEEE CEC2017/2018 on CSOPs [161]) is MOCO, but all EAs in the competition are SOCO conversely. In addition, MOCO is born with a defect that the optimum solution of a MOCO could be an infinite set while the optima of the original CSOP is just a single point. Thus, it is hard to explain why MOCO is efficient.

Based on the above observations, several issues are expected to be addressed in this thesis.

- Is MOCO more efficient?
- If not, can we construct one instead.

Essentially, both MOCO or SOCO devote themselves to handling constraints. However, besides constraint handling techniques, search operators as engines of optimization also have a vital role for solving CSOPs. Current search operators such as Mutation and Crossover in Differential Evolution [108] do not explicitly utilize features of fitness landscapes, which could be a very useful reference information for EAs search in high dimensional continuous space. As a consequence, we can also make contributions from this point of view, and enhance the performance of EAs for solving CSOPs afterwards.

1.3 Contributions

Based on the motivations, this thesis investigates solving CSOPs by MOEAs. The main contribution can be summarized as follows.

- The first contribution [167] of the thesis is a novel multi-objective method is proposed for solving CSOPs. The idea of the new problem formulation (Chapter4) is adding helper objectives besides the original objective and the degree of constraint violations. The new helper objectives are weighted sums of the normalized original objective

function and normalized degrees of constraint violation. Our experimental studies show EAs with three or four fitness functions obtain feasible solutions more quickly than that with the standard two fitness functions, which stands for our expectation that adding more helper functions could be useful.

- The second one [166] (Chapter 5) is an completely novel problem formulation compared to traditional MOCO. It reconstructs the objectives of the original problem into helper and equivalent objectives (HECO) with definitions in the thesis. Moreover, we have theoretically proven that for the "wide gap" problem, using helper and equivalent objectives may shorten hitting time of crossing the "wide gap". To the best of our knowledge, this might be the first theoretical work to explain the strengths of multi-objective EAs in performing CSOPs.

- Evolutionary algorithms, such as HECO-DE [166], HECO-DEtch, HECO-DEm [164] (Chapter 6) are constructed based on HECO. Extensive experiments show that HECO and its derived algorithms perform very well. It is worth mentioning that HECO-DE is ranked 1st in 2019 in IEEE CEC Competition on Constrained Real Parameter Optimization when compared with other eight state-of-art EAs [132].

- We also contribute to new search operators of EAs by exploiting the information of fitness landscape in search space. Firstly, we present two methods of studying valleys on a fitness landscape. The first method is based on the topological homeomorphism. It establishes a rigorous definition of a valley. A valley is regarded as a one-dimensional manifold. The second method takes a different view-point from statistics. It provides an algorithm of identifying the valley direction and location using principle component analysis. Based on the latter method, a new search operator, called PCA-projection, is proposed, in which PCA is used to project points along the maximal variance direction. Afterwards, we design two algorithms enhanced with PCA-projection for solving constrained optimization problems, called PMODE and HECO-PDE, respectively. From an experimental observation, we find that given a valley landscape, the maximal variance direction in a population can be regarded as the valley direction.

1.4 The Structure of the Thesis

The rest of this thesis is organized as follows.

Chapter 2 focuses on reviewing traditional SOCO methods, the most recent work on MOCO.

Chapter 3 introduces the benchmark functions from IEEE CEC2006 and IEEE CEC2017 competition on single-objective constrained real-parameter optimization tested in the thesis and their evaluation criteria.

Chapter 4 presents a novel MOCO method with experimental studies.

Chapter 5 presents a novel problem formulation, helper and equivalent objectives for solving CSOPs (HECO). HECO is also analyzed in theory.

Chapter 6 proposes several algorithms designed based on HECO, such as HECO-DE, HECO-DEtch and HECO-DEm. Extensive experimental studies are also presented in this chapter.

Chapter 7 presents two methods of studying valleys on a fitness landscape. In addition, principle component analysis (PCA) is used to characterize fitness landscapes. Based on this finding, a new search operator, called PCA-projection, is proposed in this chapter. In order to verify the effectiveness of PCA-projection, we design two algorithms enhanced with PCA-projection for solving CSOPs, called PMODE and HECO-PDE with experimental studies, respectively.

Chapter 8 gives the conclusion of the thesis and future work.

Chapter 2

Related Work

2.1 Traditional Approaches to Constrained Single Optimization

As unconstrained optimization problems, constrained optimization problems (COPs) are also a class of mathematical optimization problems. Then, most methods for dealing with unconstrained optimization problems can also be used to solve COPs with considering handling constraints simultaneously. The constrained single objective optimization problems (CSOPs) is the single objective case of COPs.

A constrained single-objective optimization problem (CSOP) is formulated in a mathematical form:

$$\begin{aligned} \min \quad & f(\vec{x}), \quad \vec{x} = (x_1, \cdots, x_D) \in \Omega, \\ \text{subject to} \quad & \begin{cases} g_i^I(\vec{x}) \leq 0, & i = 1, \cdots, q, \\ g_i^E(\vec{x}) = 0, & i = 1, \cdots, r, \end{cases} \end{aligned} \tag{2.1}$$

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. $g_i^I(\vec{x}) \leq 0$ is an inequality constraint and $g_i^E(\vec{x}) = 0$ is an equality constraint. A feasible solution satisfies all constraints, but an infeasible solution violating at least one. Ω^*, Ω_I, Ω_F denote the set of optimal feasible solution(s), infeasible solutions and feasible solutions respectively.

Classical constrained optimization methods include penalty function method, Lagrangian method [38] and Sequential Quadratic Programming [11]. They are all local search methods to find a local optimal solution. In many real-world CSOPs, on one hand, as the formulation of objective function is usually highly complicated, not only high dimensionality but also exist many local optima located on fitness landscape. On the other hand, the objective

functions in real-world problems are usually non-linear or non-differentiable, even without analytical expression.

Evolutionary algorithms (EAs), as global optimization algorithms, have requirements on the computability rather than analytic properties (e.g. continuity, differentiability, etc.) of the objective function itself. The key to solving CSOPs with EAs lies in how to handle constraints, that is, how to effectively balance the search between feasible and infeasible areas. Summarized in [87], the existing high impact constraint-handling techniques can be divided into the following categories:

- Penalty functions
- Feasibility rules
- Multi-objective method

2.1.1 Penalty Function Methods

General formula of a CSOP in equation 2.1 transformed by penalty function methods is the following:

$$\varphi(\vec{x}) = f(\vec{x}) + p(\vec{x}) \tag{2.2}$$

where $\varphi(\vec{x})$ is the transformed objective function including a penalty term $p(\vec{x})$ which can be calculated as follows:

$$p(\vec{x}) = \sum_{i=1}^{q} c_i^I \cdot \max(0, g_i^I(\vec{x})) + \sum_{i=1}^{r} c_i^E \cdot \max(0, |g_i^E(\vec{x})|) \tag{2.3}$$

where c_i^I and c_i^E are positive constants called “penalty factors".

It can be known from the formula of penalty function methods, the aim is to create selection preference for feasible solutions. In equation (2.2), a positive penalty value is added to the fitness of a infeasible solution as low values are preferred in minimization problem.

The simplest penalty function is “death-penalty”. In this case, infeasible solutions are simply eliminated from search process with worst fitness values [5, 123].

As the “death-penalty” always ignore the valuable information from infeasible solutions, penalty function is defined with penalty terms. Under this scheme, infeasible solutions are considered in the optimizing process. There are different methods for setting penalty factors. The first method is to keep penalty terms fixed during the search [92, 54, 52, 72]. The main drawback of this method is fixed penalty terms values are problem independent. The

second method is to get time (usually the generation counter) involved in construct penalty function [64, 67, 23, 89]. The main drawback of this method is that new parameters are also required to be tuned in penalty function. The last method improved the generalization, that is adaptive penalty functions [10, 110, 44, 45, 8, 18, 160]. However, this method still has drawbacks, i.e., the information for adaption is not necessarily useful, in another words, there is no certain trend for a algorithm in optimization process.

2.1.2 Feasibility Rules

The superiority of feasibility rule originally proposed by Deb [28] is one of the most popular constraint-handling techniques described as follows.

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

This simple constraint-handling scheme is popular due to its ability to be coupled to a variety of algorithms, without introducing new parameters. Mezura-Montes and Coello Coello [84] emphasized that it is important to combine feasibility rules with other mechanisms, such as archiving infeasible solutions which are near the feasible region. However, this approach needs an additional dynamic decreasing mechanism for the tolerance value (ε) for equality constraints.

Zielinski and Laur [176] combined DE with the feasibility rules with a greedy selection scheme between target and trial vectors. Instead of using penalty functions for constraint handling what is the most common approach, a method based on a modified selection procedure is adopted that favors feasible over infeasible individuals. No additional parameters are required for this constraint handling technique.

Self-adaptive mechanisms DE-based approaches also employ feasibility rules to choose among their variants, e.g., SaDE [12]. In their method, sequential quadratic programming (SQP) is used within some iterations to a subset of solutions in the population. Although the results of their algorithm are very competitive, SQP plays a major role there, and it lead to poor applicability.

PSO-based approaches also utilizes feasibility rules as constraint-handling technique. Zielinski and Laur [176] proposed premature convergence in test problems with a high number of equality constraints since there is no diversity maintenance mechanism.

In summary, the feasibility rules have been very popular in constrained optimization due to simplicity and flexibility, which makes them easily to be combined with selection mechanisms. However, they have drawbacks. For instance, feasibility rules inevitably cause premature convergence. It is on account of the fact that the rules still strongly favor feasible solutions. Then, this constraint handling technique will significantly increase the selection pressure if without further mechanisms adopted to preserve diversity [84]. Some approaches adopt special operators [9], which however can be considered as a secondary role since the main bias is provided by feasibility rules during the search. Feasibility rules are combined with self-adaptive variation operator selection mechanisms in DE [176, 12], PSO [176].

2.2 Multi-objective Methods for Constrained Optimization

In the real world, there exist many problems having two or more objectives to be optimized at the same time. We call them multi-objective optimization problems (MOPs). How to deal with MOPs has attracted the attention of researchers for many years. Due to the conflict among the objectives, solving an MOP produces a set of solutions representing the best possible trade-offs among the objectives. Hence, such solutions constitute the Pareto optimal set and the image of this set form the so-called Pareto front.

2.2.1 Basic Concepts

multi-objective optimization problem (MOP) is formulated as follows:

$$\begin{aligned} \min \quad & \vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \ldots, f_m(\vec{x}),)^T, \\ \text{subject to} \quad & \vec{x} = (x_1, \cdots, x_D) \in \Omega, \end{aligned} \tag{2.4}$$

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.

A few additional definitions are required to introduce the notion of optimality used in multi-objective optimization.

Definition 1. *Given two vectors $\vec{x}, \vec{y} \in \mathbb{R}^m$, we say that $\vec{x} \leq \vec{y}$ if $x_i \leq y_i$ for $i = 1, \ldots, m$, and that $\vec{x}$ dominates $\vec{y}$ (denoted by $\vec{x} \prec \vec{y}$) if $\vec{x} \leq \vec{y}$ and $\vec{x} \neq \vec{y}$*

Definition 2. *We say that a vector of decision variables $\vec{x} \in \mathscr{X} \subset \mathbb{R}^D$ is non-dominated with respect to $\mathscr{X}$, if there does not exist another $\vec{x}' \in \mathscr{X}$ such that $\vec{f}(\vec{x}') \prec \vec{f}(\vec{x})$*

Definition 3. *We say that a vector of decision variables $\vec{x}^* \in \mathscr{F} \subset \mathbb{R}^D$ ($\mathscr{F}$ is the feasible region) is Pareto-optimal if it is non-dominated with respect to $\mathscr{F}$.*

Definition 4. *The Pareto optimal set $\mathscr{P}^*$ is defined by:* $\mathscr{P}^* = \{\vec{x} \in \mathscr{F} \mid \vec{x} \text{ is Pareto-optimal } \}$

Definition 5. *The Pareto front $\mathscr{P}\mathscr{F}^*$ is defined by:* $\mathscr{P}\mathscr{F}^* = \left\{ \vec{f}(\vec{x}) \in \mathbb{R}^m \mid \vec{x} \in \mathscr{P}^* \right\}$

2.2.2 Three Schemes

A multi-objective method works by transforming the above CSOP into a multi-objective optimization problem without inequality and equality constraints and then, solving it by a multi-objective EA. Multi-objective EAs have been applied to CSOPs since the 1990s [135, 14]. Segura et al. [125] made a literature survey of the work up to 2016. Thus, most recent work will also be taken into account in this chapter. Following the taxonomy in [86, 125], 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.
2. Scheme with many objectives, which are the original objective f and degrees of violating each constraint v_i.
3. Scheme with other objective(s), for example, the penalty function [30], besides the original objective or the degree of constraint violation.

Bi-objective Methods

A popular implementation is to the bi-objective optimization: to minimize the original objective function f and the degree of constraint violation v simultaneously. This scheme is the most widely used one so far.

$$\min \vec{f}(\vec{x}) = (f(\vec{x}), v(\vec{x})), \qquad \vec{x} \in \Omega. \tag{2.5}$$

The constraint violation degree is often measured by the sum of constraint violation degrees:

$$v(\vec{x}) = \frac{\sum_{i=1}^{q} v_i^I(\vec{x}) + \sum_{i=1}^{r} v_i^E(\vec{x})}{q+r}, \tag{2.6}$$

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

$$v_i^I(\vec{x}) = \max\{0, g_i^I(\vec{x})\}, \quad i = 1, \cdots, q. \tag{2.7}$$

Equality constraints are transformed into the form:

$$|g_i^E(\vec{x})| - \varepsilon \leq 0, \quad i = 1, \cdots, r, \tag{2.8}$$

where ε is a small number.

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

$$v_i^E(\vec{x}) = \max\{0, |g_i^E(\vec{x})| - \varepsilon\}, \quad i = 1, \cdots, r, \tag{2.9}$$

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

The above two objectives are widely used in the existing multi-objective methods for constrained optimization [124].

Surry et al. [136, 135] proposed one of the most popular bi-objective methods COMOGA. This method sets each constraint as a separate criterion, after that, a form of pareto ranking is utilized to sort out solutions regarding their constraint violation. In the end, a self-adaptive form of Schaffer's VEGA scheme [121] is employed while the cost is used as one criterion and overall ranked performance against constraints is used as another. The COMOGA method utilizes the memory implicit in the population for achieving combinations of constraints and objectives. The population is then expected to construct not only a set of good solutions by evolutionary process, but also a context of the relative weighting of constraint satisfaction and cost minimisation.

Zhou et al. [175] presented an approach by using knowledge of Pareto-dominance. Then, a novel real-coded genetic algorithm is built upon Pareto strength and Minimal Generation Gap (MGG). Mezura-Montes [84] introduced an approach utilizing a simple diversity mechanism which reserves infeasible solutions to maintain the diversity, without a penalty function. The approach could guide the search towards global optimum in spite of relatively slowly reaching feasible region in the search space. However, a simple feasibility-based comparison mechanism is employed for pushing search toward feasible region. In addition, evolutionary strategy searches locally with small steps in the beginning. Also, a combined (discrete/intermediate) panmictic recombination technique is to promote the capabilities of exploitation.

Cai and Wang et al. [13] firstly proposed a method called CW. This method updates population by doing replacement between dominated parent solutions and non-dominated offspring solutions. Additionally, three models of a population-based algorithm-generator and an infeasible solution archiving and replacement mechanism are also the main contributions. The search operator is just a very simple crossover. Then, Wang and Cai [151] introduced an improved version of CW, named CMODE. CMODE has two differences: 1) DE, instead of a

simplex crossover, is adopted as search engine. 2) A novel infeasible solution replacement mechanism is built upon multi-objective techniques.

Wang et al. [152] proposed a novel algorithm which effectively combines multi-objective optimization with global and local search models. Regarding to the global search, a niching genetic algorithm based on tournament selection was proposed. In addition, a parallel local search operator that implements a clustering partition of the population and multi-parent crossover to generate the offspring population. As a result, dominated individuals in the parent population were replaced by non-dominated offspring. Simultaneously, the best infeasible individual replacement scheme provide the bias towards the feasible region in the search space. The global search model benefits the population diversity while the local search model contributes to the convergence.

However, the proposed method is highly sensitive to the problem-dependent expanding factor in the simplex crossover which could affect implementation of the algorithm in the real-world applications. In order to overcome aforementioned shortcomings, in Wang et al. [150] the global search and local search are executed dynamically according to the feasibility proportion of the current population. Venter and Haftka's [147] approach is also population-based which utilizes specialized multi-objective particle swarm optimization algorithm.

Wang et al. [154] proposed another algorithm which utilized orthogonal design to pairs of parent solutions to produce a set of representative offspring solutions. In each generation, only a few individuals are selected from the population as the parents. Then, paired parents are mated randomly and orthogonal crossover is implemented to reproduce a set of offspring. offspring is then also combined while non-dominated individuals are chosen as the potential offspring and utilized to complete the replacement by proposed mechanisms. Moreover, an improved BGA mutation operator is adopted to contribute the population diversity.

A method that also divides the search into several phases is introduced in [146]. In the first phase of the algorithm, only one objective constraint violation degree is considered while disregarding the original objective function. Then, it becomes a constraint satisfaction problem. The genetic search is forced to be guided to feasible region. Individuals are assigned with fitness value by a linear rank-based approach. In addition, the solution with smallest constraint violation degree value is archived. In the second phase, the original objective is evolved again. The problem then becomes a bi-objective optimization problem. Non-dominated ranking scheme helps exploration while elitist scheme helps exploitation.

Deb et al. also put effort on finding on the boundary of the feasible region [31]. The problem is firstly transformed into a standard bi-objective problem. A version of NSGA-II then includes the definition of a reference point from [33]. The reference point is dynamically

changed and used to find the nearby solutions. In each generation, the best found solution is considered as the reference point. ε-dominance [71] is used for the sake of maintaining diversity during the search process.

Ray et al. [113] presented an algorithm that explicitly keep a small amount of infeasible solutions near the constraint boundaries in the evolution process. Apart from approaching feasible boundaries from the feasible side by evolution of feasible solutions, these marginally infeasible solutions could approach the boundaries of feasible region from the infeasible side in the search space. "good" infeasible solutions are believed to contain more useful information of approaching a feasible optimum than some feasible solution. Thus, the ranks of solutions are adjusted not only by feasibility. The method has been further developed by involving local search [127], and also applied to a practical optimization problem [126].

Masuda and Kurihara [82] modified the standard Multi-objective Particle Swarm Optimizer (MOPSO) as: 1) limited number of Pareto optimal solutions as candidates survived into next generation comparing to unlimited number of candidates in MOPSO; 2) the global best solution is employed for updating Pareto optimal candidate set; 3) For the sake of maintaining diversity of population, particles are randomly moved to candidate set if it is not full yet.

Ji et al. [59] converted a berth allocation problem with constraints into problem (2.5) and solved it by a modified NSGA-II. An archive was designed as an efficient complementary mechanism to push the search toward the feasible solution. Superiority of feasibility rule [32] was employed to obtain this archive. Ji et al. [60] transformed a CSOP into problem (2.1) and solved it by a differential evolution (DE) algorithm. Different mutation operators were combined to improve the search ability and control the convergence. Then, multi-objective optimization techniques can be utilized in constrained optimization problem to balance population diversity and convergence. They combined multi-objective optimization with an ε-constrained method.

Li and Zhang [74] proposed a mechanism of adding bias to standard dominance relationship. They claim that this control strategy based on biased threshold value could overcome the shortcomings brought by Pareto dominance which is lack of preference on constraints.

Runarsson and Yao [120] explained in depth to indicate the significance of search bias in constrained optimization based their previous research [119]. They studied why and when multi-objective approaches for CSOPs work or fail. Dong et al. [34] proposed a novel unbiased bi-objective optimization model which the original objective function and constraint violation degree are treated equally. In addition, the relationship between proposed unbiased model and existing biased model are analyzed in detail.

Another method related to biased mechanism is from Garza-Fabre et al. [42, 41]. A novel constraint-handling technique for hydrophobic-polar (HP) model was proposed. A basic genetic algorithm is performed on a large set of test instances for the bi-objective HP model (based on the square lattice). A search bias was incorporated with the feasibility of the individuals as a supplementary discrimination criterion.

Gao et al. [39] proposed a dual-population differential evolution (DPDE) model with coevolution. The CSOP is firstly transformed into a standard bi-objective. Then in each population, the evolution task is different: one is to optimise the original objective function, the other one is to optimise the constrained violation degree. Additionally, DPDE utilizes a information sharing mechanism to achieve the communication between two sub-populations.

Recently, decomposition-based multi-objective EAs have applied to solving problem (2.5). Xu et al. [167] decomposed problem (2.5) into a tri-objective problem using the weighted sum method with static weights. They solved the multi-objective optimization problem by a Pareto-ranking based DE algorithm. Wang et al. [148] decomposed problem (2.5) using the weighted sum method into a number of subproblems with dynamical weights. They solved the subproblems by DE. Peng et al. [105] decomposed problem (2.5) using the Chebycheff method. Weights are biased and adjusted dynamically for maintaining a balance between convergence and population diversity.

N-objective Methods

The second scheme converts a CSOP into a many-objective optimization problem but is less used. One of the most famous methods [104] was proposed by Parmee and Purchase based on the knowledge of Vector Evaluated Genetic Algorithm (VEGA) [121] by Schaffer. This approach combines multi-objective techniques with a greedy decoder. At the beginning, VEGA guides the search toward the feasible region. Then, a tailor-made operator instead of VEGA is employed to archive feasibility of solutions once a feasible solution is found.

Horn et al. [55] proposed a method called Niched–Pareto Genetic Algorithm (NPGA). A modified GA is served as the search operator while the Pareto dominance is employed as selection operator. In addition, a niching pressure is used to push the solutions along the Pareto Front. Based on the previous research, Coello et al. [19, 20] proposed a new version. There are two main differences in the new version comparing to NPGA. First, a simple random selection with low probability instead of niches is utilized. Second, only infeasible solutions are selected by dominance relationship, and feasible solutions are always ranked before infeasible solutions. It is worth to mention that by using random selection, the population diversity can be maintained, no need for an extra mechanism.

Jiménez [63] proposed a method which is constructed upon the use of goals and priorities. The objectives obtained from constraints are considered with higher priority comparing to the original objective function. In this case, feasible solutions are ranked before infeasible solutions. Moreover, no original function value is considered when comparing two infeasible solutions. A pre-selection scheme is also implemented to promote the offspring close to their parents.

Kukkonen and Lampinen [70] considered every single constraint as an objective solved by differential evolution. The DE/rand/1/bin [108] is used and feasible solutions are always considered before infeasible solutions. If both solutions are feasible, then compare their objective function value. If both solutions are infeasible, then compare them by the concept of weak dominance. Gong et al. [43] extended this scheme utilize the orthogonal design method to generate the initial population. A orthogonal based crossover operator is adopted to improve the local search capability. For constraint-handling, a relaxed form of Pareto dominance, named ε-dominance [71] is employed to update the archive which store the non-dominated solutions. However, most of the mentioned N-objective methods above could suffer from disconnected feasible regions in search space. Despite of this situation, several methods based on Schaffer's VEGA [121] could overcome these shortcomings. Cello proposed an application of VEGA considering many sub-populations. The first part of sub-populations are related to the violation degree of each constraint, while the second part of sub-populations are related to the original objective function value. Then, a feasible solution with high original objective function value would occur when combining solutions from two parts of sub-populations. However, this approach could suffer from the large number of sub-populations. Liang and Suganthan [77] presented a new mechanism which dynamically assigns objectives to sub-populations by the difficulty of each constraint.

Ray et al. [111] proposed a method ranks solution based on Pareto dominance by considering three aspects: the original objective function, the constraints, and the combination of aspects of objective function and constraints. Then, a solution is selected or updated depending on its rank value of three aspects. Additionally, the method also involves mating restrictions and niche mechanism according to Euclidean distance. Ray et al. [112] then improve the scheme for maintaining diversity of population. It is an optimization algorithm based on a society and civilization model. Clustering algorithms are employed to identify individuals of a certain society.

Hernández-Aguirre et al. [1] presented the Inverted Shrinkable Pareto Archived Evolution Strategy (IS-PAES) which is an extended version PAES. The main idea is utilizing shrinking mechanism to reduce the search space. At the beginning of the search, the whole search

space is taken into account. As the number of generation increase, the search space gradually shrinks to feasible region.

Angantyr et al. [2] gave a new way to balance objectives by promoting the oscillation between the search in feasible and infeasible regions. This method also considers two different ranks, one is objective function, the other is constraints. Adaptive weighting is also added to control the search bias especially increase the volume of constraints when there is only a few feasible solution in the population.

Churchill et al. [17] proposed an algorithm which focuses on promoting the search along the boundary regions. In this method, NSGA-II with modified crowding distance assignment mechanism is employed as multi-objective optimizer. However, the search time of directly using NSGA-II is too long. Thus, reference points are used and also guided elitism scheme is implemented.

Li et al. [73] solved the many-objective optimization problem by dynamical constraint handling. The idea is to change the constraint boundary over the states for the constraint problem. Then, the dynamic constraint handling can provide always feasible population for effective global search. The many-objective optimization is realized by the reference-point-based non-dominated sorting approach, which can keep the balance of objectives and constraints.

Other Methods

Some schemes cannot be classified as bi-objective neither N-objective. Therefore, they are classified into the third scheme. The third scheme has an advantage of designing a new objective.

Schoenauer and Xanthaki [122] handle constraints in a certain order. At the beginning, only one constraint is considered to be optimised. After a percentage of population is feasible for the constraint, another constraint is then taken into account. Despite of no multi-objective scheme involved, objectives are still simultaneously taken into account by this approach. In the end, death penalty comes in, and infeasible solution are disregarded.

Coello [21] gave a scheme that every single solution is compared with other solutions in the population. When comparing, feasible solutions are always better than infeasible solutions. In the case of comparing two infeasible solutions, the constraints are the first character to compare. For feasible solutions, the normalized original objective function value is the standard.

Watanabe and Sakakibara [156] proposed a non-feasible-compliant method by relaxing one of constraints. The method firstly converts the original problem into a bi-objective problem but not standard model. In this method, one objective is the original objective

function while the other is not constraint violation degree but equivalent to the original objective with relaxed constraints. Then, the first objective is optimised by penalty function. Moreover, NSGA-II is implemented. Murugan et al. [99] also modified NSGA-II for solving Transmission Constrained Generation Expansion Planning Problem. The first objective is the cost function while the second objective is normalised soft constraints. The hard constraints are just considered as constraints.

Deb and Datta [29, 30] proposed a hybrid approach combining bi-objective method with a penalty function method. The penalty factor is estimated by evolutionary process while the penalty function method provides an extra convergence direction to feasible region. Uniform adaptive scaling of equality and inequality constraints and local search is also added in [26].

Zeng et al. [170] 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 multi-objective EAs (ranking-based, decomposition-based, and hyper-volume) to the tri-objective optimization problem. Jiao et al. [62] converted a CSOP into a dynamical bi-objective optimization problem consisting of the original objective and a niche-count objective.

Discussions

As shown in previous subsections, there exists a huge number of proposals that consider multi-objective concepts. However, apart from introduced schemes above, there are several proposals that are just minor variants of the three schemes. Thus, they are included in the literature review. Unfortunately, among such a large number of proposals, no one has been found to be significantly superior to the others. This phenomenon can be possibly explained by No-Free-Lunch theorem by Wolpert and Macready [158]. Nonetheless, some studies claim that multi-objective is not always effective for some single-objective problems [87]. For instance, results of the only method using multi-objective concepts presented at the CEC2010 competition on constrained optimization [118] is much worse than those inspired by other schemes. In contrast, multi-objective schemes also perform very well to CSOPs [21, 153].

The existing implementation of Multi-objective EAs to CSOPs (MOCO) is balancing objective functions and constraints. It is also worth noticing that the whole set of solutions is usually not of interest to the user, because they are not exactly the solutions of original problems. In fact, MOCO is aiming at solving both the constrained and unconstrained problems simultaneously. Thus, searching in the infeasible region might cost too much time might. Especially, using Pareto Ranking might lead to a bias-free search where searching in the infeasible region costs most of the time. Additionally, the fitness landscape most likely

determines the destiny of the search, which can explain why some multi-objective schemes that require certain amount of bias in the process of the search.

Different rankings employed in certain searching periods shows its effectiveness [153]. As a consequence, the direct use of Pareto dominance concepts benefits the search in some stages of the optimization, while it increases the convergence time when it is used over the entire searching process. In the end, it is also worth mentioning that many of the EAs introduced above have only been tested on a few real-world applications or on a reduced number of benchmark problems. Therefore, their behaviors are unpredictable when solving different problems. Mezura Montes and Coello [85] did a comparative study, and disregarded several methods, which certainly show that the problem is more complex.

Chapter 3

Methodology

In this chapter, we introduce the benchmark functions on which our proposed algorithms are tested in the later experimental studies. The overall methodology to compare algorithms includes the evaluation criteria of performance of a single algorithm, and also includes the ranking schemes for comparing different algorithms.

3.1 Test Suites

In the thesis, different algorithms are tested on the same benchmark functions taken from IEEE CEC2006 and IEEE CEC2017/18 competitions on constrained single objective optimization problems.

The most widely used test suite are 24 benchmark functions collected in [76] and adopted by IEEE CEC2006 Competition. There exists an extensive studies adopting these 24 benchmark functions since 2006. These benchmark functions are different types, such as linear, non-linear, polynomial and quadric with different number of linear and non-linear constraints. These 24 test problems have 2-24 dimensions and are not easily scalable. In fact, the CEC 2006 benchmark has been successfully solved. Thus, it is increasingly harder for newly designed algorithms to demonstrate their superiority.

18 scalable benchmark functions were then presented in IEEE CEC2010. However, these benchmark functions are not adopted in the thesis since they have been solved satisfactorily as well. Thus, more recent scalable, higher dimensional CEC2017/18 benchmark [161] was constructed. CEC 2017/18 benchmark is an upgrade version of CEC 2010 benchmark. The former consists of 4×28 with dimension 10, 30, 50 and 100, the latter contains 2×18 with only dimension 10 and 30. In the context of Big Data, most optimization problems being considered contain few hundreds of variables, but neither CEC2006 nor CEC2010 benchmark contains problems with over 30 dimensions. However, CEC 2006 benchmark

has an advantage that many problems have many more constraints than CEC2010 and CEC2017/18 benchmark. CEC2006 and CEC2017/18 benchmark are hence employed as the test suite in the thesis.

3.2 Evaluation Criteria

Since there exists many differences of the performance evaluation criteria between CEC 2006 and CEC 2017/18 benchmark, the criteria are then introduced separately in this section.

3.2.1 Performance Evaluation Criteria of CEC 2006 Benchmark

The report [76] suggests that for each problem, one algorithm should run 25 times independently with $500,000$ maximum number of fitness evaluations (FES_{MAX}). Population size is free to set as long as the algorithm does not exceed the FES_{MAX}. ε in Equation (2.8) is set to 0.0001.

There are experimental data required to be recorded as follows.

- The best, median, worst result, mean value and standard deviation of function error value are required to be recorded. The function error value $(f(\vec{x}) - f(\vec{x}^*))$ is achieved best solution $\vec{x}$ after 5×10^5 fitness evaluations (*FES*).
- The number of violated constraints (including the number of violations by more than 1, 0.01 and 0.0001), and the mean violation $v(\vec{x})$ in Equation (2.6), and $v(\vec{x})$ at the median solution.
- The *FES* when finding a solution satisfying the condition $f(\vec{x}) - f(\vec{x}^*) \leq 0.0001$ and $\vec{x}$ is feasible.
- *Feasible Rate*, *Success Rate* and *Success Performance* for each problem are also required to be recorded. They can be formulated as follows:

$$\begin{aligned} \text{Feasible Rate} &= \frac{\text{Feasible Runs}}{\text{Total Runs}} \\ \text{Success Rate} &= \frac{\text{Success Runs}}{\text{Total Runs}} \\ \text{Success Rate} &= \frac{\text{Success Runs}}{\text{Total Runs}} \end{aligned} \tag{3.1}$$

  where *Feasible Run* represents a run during which at least one feasible solution is found under FES_{MAX}, *Success Run* represents a run during which the algorithm finds solution $\vec{x}$ satisfying the condition $f(\vec{x}) - f(\vec{x}^*) \leq 0.0001$.

- Convergence graph of each problem. The convergence speed is measured by the average convergence rate R_t defined as follows [47]:

$$R_t = 1 - \left| \frac{f(x_t) - f(\vec{x}^*)}{f(x_0) - f(\vec{x}^*)} \right|^{1/t} \tag{3.2}$$

  where R_t denotes the normalized convergence speed, t the number of current generation, $f(x_t)$ the objective value at t generation, and $f(\vec{x}^*)$ the objective value of the known optimal solution. In addition, R_t may take a negative value since the event $\mid f(x_t) - f(\vec{x}^*) \mid > \mid f(x_0) - f(\vec{x}^*) \mid$ could happen. This means, x_0 is an infeasible solution but its objective value is less than x_t which is a feasible solution. Using the average convergence rate R_t, we can easily evaluate and compare the convergence speed of different algorithms. It is better than the logarithmic rate $\log 10(f(x_t) - f(\vec{x}^*))$ used in many references [76] because the logarithmic rate itself does not provide any information about the convergence rate but only its slope does. However, the average convergence rate R_t provides a quantitative value of the convergence speed.

The sorting method for final results presented in [76] is shown as follows:

1) Sort feasible solutions in front of infeasible solutions;

2) Sort feasible solutions according to their function errors $f(\vec{x}) - f(\vec{x}^*)$;

3) Sort infeasible solutions according to their mean value of the violations of all constraints.

Experimental results are required to be compared with those in CEC 2006 Competition. Tenen EAs participated in the competition. Their characteristics were summarized by Barbosa et al. [7] as below.

- j-DE2 [56]: a DE algorithm with self-adaptive control parameters and the feasible rule: a feasible solution is better than an infeasible one and the latter are ranked according to the sum over all the constraint violations.

- DE [176]: the standard DE algorithm, with the same feasible rule constraint-handling method as jDE-2.

- SaDE [109]: an extension of the original SaDE. The constraint-handling method is similar to the feasible rule used by jDE-2 but the constraint violations are weighted.

- GDE [70]: this algorithm extends DE for constrained multiobjective optimization. The constraint-handling method is similar to the feasible rule constraint-handling method as jDE-2.
- DMS-PSO [77]: a dynamic and multiple PSO algorithm. The constraint-handling method is similar to SaDE.
- MDE [88]: a DE-based approach modified to solve constrained optimization problems. Its constraint-handling method is similar to the feasible rule constraint-handling method as jDE-2.
- PESO+ [97]: a PSO-based approach with topological organization and constraint handling similar to the feasible rule constraint-handling method as jDE-2.
- PCX [128]: it is derived from the population based algorithm-generator and uses the parent-centric recombination (PCX) operator and a stochastic remainder selection over three different constraint handling principles.
- ε_DE [137]: it uses the ε constraint-handling method and employs a gradient-based mutation/repair operator.
- MPDE [139]: a multi-populated DE algorithm with an adaptive penalty method to handle the constraint violations.

Evaluation criteria used in the competition are that for each algorithm. The test problem g20 is not considered in the comparison experiment because no feasible solution can be found.

3.2.2 Performance Evaluation Criteria of CEC 2017/18 Benchmark

The report [161] suggests for each of total 28 test problems, one algorithm should also run 25 times independently as same as testing CEC 2006 Benchmark, but up to $20000 \times D$ function evaluations (FES_{MAX}), where D is the dimensionality of optimization problems. ε in Equation (2.8) is also set to 0.0001.

Statistics have to be presented as follows:

- Feasibility Rate, which is calculated in the same way as in Equation (3.1) in CEC 2006 Benchmark.
- The best, median, worst result, mean value and standard deviation of function value are required to be recorded after $20000 \times D$ fitness evaluations (FES).

- The number of violated constraints (including the number of violations by more than 1, 0.01 and 0.0001), and the mean violation $v(\vec{x})$ in Equation (2.6), and $v(\vec{x})$ at the median solution of 25 runs.

The solution sorting method in CEC 2017/18 is similar to the one in CEC 2006.

1) Sort feasible solutions in front of infeasible solutions;

2) Sort feasible solutions according to their function value $f(\vec{x})$;

3) Sort infeasible solutions according to their mean value of the violations of all constraints.

A set of rules was provided for ranking all algorithms [161]. Our comparison follows the same rules, which are listed as below.

1. The procedure for ranking algorithms based on mean values:
   (a) Rank the algorithms based on feasibility rate;
   (b) Then rank the algorithms according to the mean violation amounts;
   (c) At last, rank the algorithms in terms of mean objective function value.
2. The procedure for ranking the algorithms based on the median solutions:
   (a) A feasible solution is better than an infeasible solution;
   (b) Rank feasible solutions based on their objective function values;
   (c) Rank infeasible solutions according to their constraint violation amounts.
3. Ranking all algorithms on multiple problems: for each problem, algorithms' ranks are determined in terms of the mean values and median solutions at maximum allowed number of evaluations, respectively. The total rank value of an algorithm is calculated as below:

$$\text{Rank value} = \sum_{i=1}^{28} rank_i(\text{using mean value}) + \sum_{i=1}^{28} rank_i(\text{using median solution}) \quad (3.3)$$

Experimental results are compared with eight EAs. The first seven EAs come from the CEC2017/18 constrained optimization competitions [132]. The last one, DeCODE [148], was a decomposition-based multi-objective EAs for constrained optimization published in 2018.

1. CAL-SHADE [169]: 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 [145]: 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 stop if a number of feasible solutions are found. (ii) minimize the function value until the stop condition is reached.

3. LSHADE44 [107]: Success-History based Adaptive Differential Evolution Algorithm including linear population size reduction, uses three different additional strategies compete, with the superiority of feasibility rule.

4. UDE [141]: 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 [51]: 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 [140]: 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 [36]: 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 [148]: 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.

3.2.3 Discussions

CEC competition ranking rules are instituted based on Friedman ranks from Friedman test. Friedman test is an important method of comparing multiple EAs on different benchmarks [40]. However, ranking in constrained optimization is complex. CEC rules have modified original Friedman ranks for fitting constrained optimization.

Both t-test and Wilcoxons test were directly applied to compare the objective function values of EAs in constrained optimization in [170, 148]. These tests work only if the found solutions are feasible. However, it is hard to compare results including infeasible solutions. For example,

- EA 1: infeasible solutions with better objective function values; EA 2: feasible solutions with worse objective function value;
- EA 1: solutions with worse objective function values but higher feasibility rate; EA 2: better objective function values but lower feasibility rate.

In other words, feasibility rate and constraint violation degree must be considered besides a statistical test as current CEC competition rules. But designing such new rules with an existing statistical test is not the goal of this thesis.

From the above argument, the CEC competition rules probably are the most appropriate methods for comparing multiple EAs on a set of benchmarks because they are specifically designed for constrained optimization.

Chapter 4

A New Multi-objective Problem Formulation for Constrained Single-objective Optimization

This chapter introduces our very beginning work on solving CSOPs. We consider a many-objective problem formulation for solving constrained optimization problems. The problem formulation keeps the standard objectives: the original objective function and the sum of the degrees of constraint violation. Besides them, more objectives are added into the problem formulation. One obejctive is based on the feasible rule. The others come from the penalty function method. Then a multi-objective differential evolution algorithm is applied to solving multi-objective optimization problems with two, three and four objectives. A multi-objective differential evolution algorithm, called CMODE [151], is applied to solving for multi-objective optimization problems. We also conduct an experimental study on thirteen benchmark functions from IEEE CEC2006 benchmark functions. Experimental results confirm our expectation that adding more objectives could be useful and the solution quality is improved.

4.1 A Many-objective Problem Formulation for Constrained Optimization

4.1.1 A Many-Objective Problem Formulation

Besides the above two objectives functions, we may construct many other objectives or helper functions [163]. According to penalty function methods, more helper functions with

different penalty coefficients can be constructed in the form

$$f_i(\vec{x}) = f(\vec{x}) + c_i v(\vec{x}), \qquad i = 3, \cdots, K, \tag{4.1}$$

where $c_i \in (0, +\infty]$ (where $i = 1, \cdots, K$) are penalty coefficients. If let $c_i = +\infty$, then $f_i(\vec{x})$ represents the death penalty to infeasible solutions.

According to the feasible rule [28], we construct another objective. The feasible rule means that: during pairwise-comparing individuals,

1. when two feasible solutions are compared, the one with a better objective function profit is chosen;

2. when one feasible solution and one infeasible solution are compared, the feasible solution is chosen;

3. when two infeasible solutions are compared, the one with smaller constraint violation is chosen.

According to the feasible rule, the third objective is constructed as follows: for an individual x in a population P,

$$\min f_{K+1}(\vec{x}) = \begin{cases} f(\vec{x}), & \text{if } \vec{x} \text{ is feasible;} \\ f^\sharp + v(\vec{x}), & \text{otherwise.} \end{cases} \tag{4.2}$$

In the above, $f^\sharp$ is the "worst" fitness of feasible individuals in population P, given by

$$f^\sharp = \begin{cases} \max\{f(\vec{x}); \vec{x} \in P \text{ and } \vec{x} \text{ is feasible } \}; \\ 0, \quad \text{otherwise.} \end{cases} \tag{4.3}$$

Since the reference point $f^\sharp$ depends on population P, thus for the same x, the values of $f_{k+1}(\vec{x})$ in different populations P might be different. However the optimal feasible solution to minimizing $f_{K+1}(\vec{x})$ always is the best in any population. Thus the optimal feasible solution to minimizing $f_{K+1}(\vec{x})$ is exactly the same as that to the constrained optimization problem. Based on this reason, $f_{k+1}(\vec{x})$ is called an equivalent fitness function.

In summary, the original constrained optimization problem is transferred into a many-objective optimization problem:

$$\min(f_i(\vec{x}), \quad i = 1, \cdots, K+1). \tag{4.4}$$

The problem formulation potentially may include many objectives inside.

4.2 Multi-objective Differential Evolution for Constrained Optimization

4.2.1 MOEAs

Many MOEAs have been proposed for solving multi-objective optimization problem. They can be classified into two categories: one aims to evolve the non-domination set and eventually to reach Pareto optimal set, such as the non-dominate sorting genetic algorithm [32] and strength Pareto evolutionary algorithm [177]. Another category focuses on solving a series of scalar optimization problems, such as the vector evaluated genetic algorithm [121] and MOEAs based on decomposition [172]. The algorithm below gives a general description for the MOEAs based on the dominance relation.

1: initialize a set of solutions;
2: evaluate the values of $f_i, i = 1, \cdots, K+1$ for each solution;
3: select non-dominated solutions and construct an initial population P_0;
4: **for** $t = 0, 1, 2 \cdots, \cdots$ **do**
5: generate a children population C_t from the parent population P_t;
6: evaluate the values of $f_i, i = 1, \cdots, K+1$ for each solution;
7: select non-dominated solutions in $P_t \cup C_t$ and obtain the next generation population P_{t+1}.
8: **end for**

4.2.2 Constrained Multi-objective Differential Evolution

A MOEA based on differential evolution (DE), called CMODE [151], is chosen to solve the above multi-objective optimization problem (4.4). Different from normal MOEAs, CMODE is specially designed for solving constrained optimization problems. Hence it is expected that CMODE is efficient in solving the multi-objective optimization problem (4.4). CMODE [151] originally is applied to solving a bi-objective optimization problem which consists of only two objectives: f_1 and f_2. However, it is easy to extend the existing CMODE to multi-objective optimization problems. The algorithm is described as follows.

Input: μ: population size;
1: λ: the number of individuals involved in DE operations
2: $FES_{\max}$: the maximum number of fitness evaluations
3: randomly generate an initial population P_0 with population size μ;

4: evaluate the values of f and v for each individual in the initial population, and then calculate the value of f_i where $i = 1, \cdots, m$;
5: set $FES = \mu$; // FES denotes the number of fitness evaluations;
6: set $A = \emptyset$; //A an archive to store the infeasible individual with the lowest degree of constraint violation;
7: **for** $t = 1, \cdots, FES_{\max}$ **do**
8: choose λ individuals (denoted by Q) from population P_t;
9: let $P' = P_t \setminus Q$;
10: for each individual in set Q, an offspring is generated by using DE mutation and crossover operations as explained in Section 4.2.3. Then λ children (denoted by C) are generated from Q;
11: evaluate the values of f and v for each individual in C and then obtain the value of f_i where $i = 1, \cdots, m$;
12: set $FES = FES + \lambda$;
13: identify all nondominated individuals in C (denoted by R);
14: **for** each individual $\vec{x}$ in R **do**
15: find all individual(s) in Q dominated by $\vec{x}$;
16: randomly replace one of these dominated individuals by $\vec{x}$;
17: **end for**
18: let $P_{t+1} = P' \cup Q$;
19: **if** no feasible solution exists in R **then**
20: identify the infeasible solution $\vec{x}$ in R with the lowest degree of constraint violation and add $\vec{x}$ to A;
21: **end if**
22: **if** $\mod(t,k) = 0$ **then**
23: execute the infeasible solution replacement mechanism and set $A = \emptyset$;
24: **end if**
25: **end forreturn** the best found solution

The algorithm is explained step-by-step in the following. At the beginning, an initial population P_0 is chosen at random, where all initial vectors are chosen randomly from $[L_i, U_i]^n$.

At each generation, parent population P_t is split into two groups: one group with λ parent individuals that are used for DE operations (set Q) and the other group (set P') with $\mu - \lambda$ individuals that are not involved in DE operations. DE operations are applied to λ selected children (set Q) and then generate λ children (set C).

Selection is based on the dominance relation. First nondominated individuals (set R) are identified from children population C. Then these individual(s) will replace the dominated individuals in Q (if exists). As a result, population set Q is updated. Merge population set Q with those parent individuals that are involved in DE operation (set P') together and form the next parent population P_{t+1}. The procedure repeats until reaching the maximum number of evaluations. The output is the best found solution by DE.

The infeasible solution replacement mechanism is that, provided that a children population is composed of only infeasible individuals, the "best" child, who has the lowest degree of constraint violation, is stored into an archive. After a fixed interval of generations, some randomly selected infeasible individuals in the archive will replace the same number of randomly selected individuals in the parent population.

4.2.3 Differential Evolution

optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. Such methods are commonly known as metaheuristics as they make few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. However, metaheuristics such as DE do not guarantee an optimal solution is ever found.

DE is used for multidimensional real-valued functions but does not use the gradient of the problem being optimized, which means DE does not require the optimization problem to be differentiable, as is required by classic optimization methods such as gradient descent and quasi-newton methods. DE can therefore also be used on optimization problems that are not even continuous, are noisy, change over time, etc.[1]

DE optimizes a problem by maintaining a population of candidate solutions and creating new candidate solutions by combining existing ones according to its simple formulae, and then keeping whichever candidate solution has the best score or fitness on the optimization problem at hand. In this way the optimization problem is treated as a black box that merely provides a measure of quality given a candidate solution and the gradient is therefore not needed.

DE is originally due to Storn and Price.[2][3] Books have been published on theoretical and practical aspects of using DE in parallel computing, multiobjective optimization, constrained optimization, and the books also contain surveys of application areas.[4][5][6][7] Surveys on the multi-faceted research aspects of DE can be found in journal articles .[8][9]

DE is arguably one of the most powerful stochastic real-parameter optimization algorithms in current use [25]. There exist several variants of DE. The original DE algorithm [131]

is utilized in this chapter. A population P_t is represented by μ n-dimensional vectors:

$$P_t = \{\vec{x}_{1,t}, \cdots, \vec{x}_{\mu,t}\}, \tag{4.5}$$

$$\vec{x}_{i,t} = (x_{i,1,t}, x_{i,2,t}, \cdots, x_{i,n,t}), i = 1, 2, \cdots, \mu, \tag{4.6}$$

where t represents the generation counter. Population size μ does not change during the minimisation process. The initial vectors are chosen randomly from $[L_i, U_i]^n$. The formula below shows how to generate an initial individual $\vec{x} = (x_1, \cdots, x_n)$ at random:

$$x_i = L_i + (U_i - L_i) \times rand, \quad i = 1, \cdots, n, \tag{4.7}$$

where *rand* is the random number $[0, 1]$.

Three operations are used in the DE [131]: mutation, crossover and selection, which are described as follows.

- *Mutation:* for each target $\vec{x}_{i,t}$ where $i = 1, \cdots, n$, a mutant vector

  $$\vec{v}_{i,t} = (v_{i,1,t}, v_{i,2,t}, \cdots, v_{i,n,t})$$

  is generated by

  $$\vec{v}_{i,t} = \vec{x}_{r1,t} + F \cdot (\vec{x}_{r2,t} - \vec{x}_{r3,t}) \tag{4.8}$$

  where random indexes $r1, r2, r3 \in \{1, \cdots, \mu\}$ are mutually different integers. They are also chosen to be different from the running index i. F is a real and constant factor from $[0, 2]$ which controls the amplification of the differential variation $(\vec{x}_{r2,t} - \vec{x}_{r3,t})$. In case $\vec{v}_{i,t}$ is out of the interval $[L_i, U_i]$, the mutation operation is repeated until $\vec{v}_{i,t}$ falls in $[L_i, U_i]$.

- *Crossover:* For the sake of increasing diversity of the population, the target vector $\vec{x}_{i,t}$ is mixed with mutant vector $\vec{v}_{i,j}$, then generate the trial vector $\vec{u}_{i,t}$. The crossover methods can be classified into two categories: *exponential* and *binomial* [108], which are listed as follows:

  1. Exponential crossover:

  $$\vec{u}_{i,j,t} = \begin{cases} \vec{v}_{i,j,t}, & \text{for } j = \langle l \rangle_D, \langle l+1 \rangle_D, \ldots, \langle l+L-1 \rangle_D, \\ \vec{x}_{i,j,t}, & \text{otherwise.} \end{cases} \tag{4.9}$$

where $\langle\rangle_D$ denotes a modulo function with modulus D, l denotes the starting integer number randomly chosen from $[0, D-1]$. The integer L is drawn from $[0, D-1]$ with probability $P_r(L >= v) = CR_i^{v-1}, (v >= 0)$.

2. Binomial crossover:

$$\vec{u}_{i,j,t} = \begin{cases} \vec{v}_{i,j,t}, & \text{if } \text{rand}_j(0,1) <= CR_i or\, j = j_{rand}, \\ \vec{x}_{i,j,t}, & \text{otherwise.} \end{cases} \tag{4.10}$$

where j_{rand} is integer randomly chosen from $[1, D]$, and $\text{rand}_j(0,1)$ is random number falling in $[0,1]$

- *Selection:* a greedy criterion is used to decide which offspring generated by mutation and crossover should be selected to population P_{t+1}. Trial vector $\vec{u}_{i,t}$ is compared to target vector $\vec{x}_{i,t}$, then the better one will be reserved to the next generation.

4.3 Experiments and Results

4.3.1 Experimental Settings

CMODE is used to solve the following four multi-objective optimization problems in which the numbers of objectives are two, three, three and four respectively.

$$\min \begin{cases} f_1(\vec{x}) = f(\vec{x}), \\ f_2(\vec{x}) = v(\vec{x}), \end{cases} \tag{4.11}$$

$$\min \begin{cases} f_1(\vec{x}) = f(\vec{x}), \\ f_2(\vec{x}) = v(\vec{x}), \\ f_3(\vec{x}) = \begin{cases} f(\vec{x}), & \text{if } \vec{x} \text{ is feasible,} \\ +\infty, & \text{otherwise.} \end{cases} \end{cases} \tag{4.12}$$

$$\min \begin{cases} f_1(\vec{x}) = f(\vec{x}), \\ f_2(\vec{x}) = v(\vec{x}), \\ f_4(\vec{x}) = f(\vec{x}) + 1000v(\vec{x}). \end{cases} \tag{4.13}$$

$$\min \begin{cases} f_1(\vec{x}) = f(\vec{x}), \\ f_2(\vec{x}) = v(\vec{x}), \\ f_3(\vec{x}) = \begin{cases} f(\vec{x}), & \text{if } \vec{x} \text{ is feasible,} \\ +\infty, & \text{otherwise.} \end{cases} \\ f_4(\vec{x}) = f(\vec{x}) + 1000v(\vec{x}), \end{cases} \tag{4.14}$$

For any of the above four multi-objective optimization problems, our ultimate aim is to find an optimal feasible solution through finding the Pareto front. It is obvious that the optimal feasible solution to the original CSOP (2.1) is on the Pareto front.

In order to compare the performance of CMODE on the four multi-objective optimization problems, thirteen benchmark functions were employed as the instances to perform experiments. These benchmarks and evaluation criteria is introduced in Chapter 3.

CMODE contains several parameters which are the population size μ, the scaling factor F in mutation, the crossover control parameter Cr. Usually, F is set within $[0,1]$ and mostly from 0.5 to 0.9; Cr is also chosen from $[0,1]$ and higher values can produce better results in most cases. In our experiments, set F as 0.6, Cr as 0.95. The population size $\mu = 180$. The tolerance value δ for the equality constraints was set to 0.0001. The maximum number of fitness evaluations $FES_{\max}$ is set to two values: $5 \cdot 10^3$ and $FES_{\max} = 5 \cdot 10^4$.

4.3.2 Experimental Results

Table 4.1 shows the result of function error values achieved by CMODE on two helper functions f_1, f_2 on thirteen benchmark functions. In the table, NA means that no feasible solution was found. Within $5 \cdot 10^3$ fitness evaluations, CMODE can not find a feasible solution on six benchmark functions g01, g03, g05, g10, g11 and g13.

Table 4.2 is the result of function error values achieved by CMODE using three helper functions f_1, f_2, f_3 on thirteen benchmark functions. Within $5 \cdot 10^3$ fitness evaluations, CMODE may not find a feasible solution on four benchmark functions g03, g05, g11 and g13, and not always on g10. This is better than that using only two fitness functions f_1 and f_2. Within $5 \cdot 10^4$ fitness evaluations, the result achieved by CMODE with three helper functions is similar to that with two helper functions. Therefore using three helper fitness functions may find a feasible solution more quickly.

Table 4.3 shows the result of function error values achieved by CMODE with three helper functions f_1, f_2, f_4 on thirteen benchmark functions. The result is very similar to that with with three helper functions f_1, f_2, f_3. Within $5 \cdot 10^3$ fitness evaluations, CMODE may not

Table 4.1 Function Error Values Achieved by CMODE with two fitness functions f_1 and f_2 When $FES = 5 \times 10^3$, $FES = 5 \times 10^4$ for Test Functions g01-g13

FES	5×10^3					5×10^4				
	Best	Median	Worst	Mean	Std	Best	Median	Worst	Mean	Std
g01	6.2996E+00	NA	NA	NA	NA	3.5476E-07	1.9686E-03	1.9686E-03	1.4583E-03	8.7898E-04
g02	2.4311E-01	2.4311E-01	3.1059E-01	2.6387E-01	3.1146E-02	2.2807E-02	1.4556E-01	1.4556E-01	1.2195E-01	4.9337E-02
g03	NA	NA	NA	NA	NA	5.4382E-04	5.4382E-04	6.5021E-02	1.5423E-02	2.7703E-02
g04	1.1760E+01	3.4800E+01	3.4800E+01	2.7427E+01	1.0747E+01	2.6993E-08	1.1079E-05	1.1079E-05	9.3109E-06	4.1353E-06
g05	NA	NA	NA	NA	NA	1.3954E-03	1.3954E-03	1.3954E-03	1.3054E-03	4.6893E-12
g06	4.9363E+00	8.6183E+00	8.6183E+00	7.3900E+00	1.7349E+00	3.6702E-08	3.6702E-08	1.3304E-07	4.4410E-08	2.6676E-08
g07	5.5401E+00	6.6050E+00	6.6050E+00	6.4821E+00	3.4018E-01	6.8830E-03	1.2144E-01	1.2144E-01	9.3948E-02	4.9945E-02
g08	3.0724E-06	1.0893E-04	1.0893E-04	9.2652E-05	3.8953E-05	5.5511E-17	5.5511E-17	5.5511E-17	5.5511E-17	0
g09	6.0244E+00	7.8532E+00	7.8532E+00	7.4312E+00	7.8579E-01	1.8475E-06	1.8475E-06	8.2849E-05	2.9886E-05	3.9299E-05
g10	NA	NA	NA	NA	NA	2.6119E+00	1.1829E+01	1.1829E+01	9.7027E+00	3.9606E+00
g11	NA	NA	NA	NA	NA	9.9990E-05	9.9990E-05	9.9990E-05	9.9990E-05	2.8445E-14
g12	6.8496E-12	2.4866E-10	2.4866E-10	1.5910E-10	1.1899E-10	0	0	0	0	0
g13	NA	NA	NA	NA	NA	8.3328E-06	8.3328E-06	8.3328E-06	8.3328E-06	0

Table 4.2 Function Error Values Achieved by CMODE with three fitness functions f_1, f_2 and f_3 When $FES = 5 \times 10^3$, $FES = 5 \times 10^4$ for Test Functions g01-g13

FES	5×10^3					5×10^4				
	Best	Median	Worst	Mean	Std	Best	Median	Worst	Mean	Std
g01	7.1822E+00	7.7277E+00	7.7277E+00	7.6622E+00	1.8090E-01	1.4820E-07	1.4820E-07	1.0231E-04	2.4669E-05	4.4535E-05
g02	2.4311E-01	2.4311E-01	3.1059E+01	2.4850E-01	1.8651E-02	2.2807E-02	1.4556E-01	1.4556E-01	1.1119E-01	5.6253E-02
g03	NA	NA	NA	NA	NA	2.7793E-03	1.0114E-02	1.0114E-02	5.1265E-03	3.4921E-03
g04	1.1760E+01	3.4800E+01	3.4800E+01	3.2036E+01	7.6416E+00	2.6993E-08	1.1079E-05	1.1079E-05	1.0229E-05	3.0034E-06
g05	NA	NA	NA	NA	NA	1.3954E-03	1.3954E-03	1.3954E-03	1.3954E-03	5.6231E-11
g06	7.9681E+00	3.3066E+01	3.3066E+01	2.8239E+01	1.0087E+01	2.3017E-07	2.3017E-07	2.7603E-05	7.0734E-06	1.3686E-05
g07	5.4591E+00	5.6860E+00	5.6860E+00	5.5952E+00	1.1342E-01	5.8314E-03	5.8314E-03	8.2143E-02	5.8314E-03	3.4519E-02
g08	2.3534E-09	9.5024E-09	2.3534E-09	3.7282E-09	2.8733E-09	2.7755E-17	4.1633E-17	4.1633E-17	3.8857E-17	5.6655E-18
g09	1.8694E+00	8.5045E+00	8.4045E+00	6.8964E+00	2.8079E+00	8.2568E-06	8.2568E-06	1.8555E-04	5.7900E-05	9.1248E-05
g10	1.6339E+03	1.6339E+03	NA	NA	NA	2.7483E+00	1.8283E+01	1.8283E+01	1.5893E+01	5.7160E+00
g11	NA	NA	NA	NA	NA	9.9990E-05	9.9990E-05	9.9990E-05	9.9990E-05	2.3519E-14
g12	4.1179E-10	4.2982E-09	4.2982E-09	3.1023E-09	1.8292E-09	0	0	0	0	0
g13	NA	NA	NA	NA	NA	8.3328E-06	8.3328E-06	8.3328E-06	8.3328E-06	0

find a feasible solution only on four benchmark functions g03, g05, g11 and g13. This is better than that with only two fitness functions f_1 and f_2. Within $5 \cdot 10^4$ fitness evaluations, the result achieved by CMODE with three helper functions is similar to that with two helper functions. Therefore it also confirms that using three helper fitness functions may find a feasible solution more quickly.

Table 4.4 is the result of function error values achieved by CMODE with four helper functions f_1, f_2, f_3, f_4 on thirteen benchmark functions. Within $5 \cdot 10^3$ fitness evaluations, the result is similar to that with three helper function f_1, f_2, f_3. CMODE may not find a feasible solution on four benchmark functions g03, g05, g11 and g13, and not always on g10. This is better than that with only two fitness functions f_1 and f_2. But it is not as good as that with the three helper functions f_1, f_2, f_4. Within $5 \cdot 10^4$ fitness evaluations, the result achieved by CMODE with four helper functions is the same or better than that with two helper functions.

Table 4.3 Function Error Values Achieved with three fitness functions f_1, f_2 and f_4 When $FES = 5 \times 10^3$, $FES = 5 \times 10^4$ for Test Functions g01-g13

FES	5×10^3					5×10^4				
	Best	Median	Worst	Mean	Std	Best	Median	Worst	Mean	Std
g01	7.1822E+00	7.7277E+00	7.7277E+00	7.6840E+00	1.5100E-01	1.4820E-07	1.4820E-07	1.0231E-04	1.3331-05	3.4817E-05
g02	2.4311E-01	2.4311E-01	3.1059E-01	2.6200E-01	3.0924E-02	2.2807E-02	1.4556E-01	1.4556E-01	1.2439E-01	4.7190E-02
g03	NA	NA	NA	NA	NA	2.7793E-03	2.7793E-03	1.0114E-02	4.1377E-03	2.9035E-03
g04	1.2710E+01	1.2710E+01	4.2288E+01	2.2909E+01	1.4307E+01	3.6379E-12	7.2759E-12	7.2759E-12	4.5474E-12	1.8189E-11
g05	NA	NA	NA	NA	NA	1.3954E-03	1.3954E-03	1.3954E-03	1.3954E-03	6.1932E-11
g06	1.0451E+01	1.6199E+01	1.6199E+01	1.1601E+01	2.3467E+00	1.6370E-11	4.9386E-05	4.9386E-05	3.9509E-05	2.0161E-05
g07	5.4591E+00	5.6960E+00	5.6860E+00	5.6315E+00	9.8877E-02	5.8314E-03	5.8314E-03	8.2143E-02	1.1938E-02	2.1129E-02
g08	2.3534E-09	9.5024E-09	2.3534E-09	3.2113E-09	2.3710E-09	2.7755E-17	4.1633E-17	4.1633E-17	3.5761E-17	6.9920E-18
g09	1.8694E+00	8.4045E+00	8.4045E+00	7.0975E+00	2.6679E+00	8.2568E-06	8.2568E-06	1.8555E-04	6.4992E-05	8.4410E-5
g10	1.6339E+03	1.6339E+03	1.6339E+03	1.6339E+03	0	2.7483E+00	1.8283E+01	1.8283E+01	1.4399E+01	6.8714E+00
g11	NA	NA	NA	NA	NA	9.9990E-05	9.9990E-05	9.9990E-05	9.9990E-05	1.4939E-14
g12	4.0929E-10	4.2981E-09	4.2981E-09	3.4007E-09	1.6709E-09	0	0	0	0	0
g13	NA	NA	NA	NA	NA	8.3328E-06	8.3328E-06	8.3328E-06	8.3328E-06	7.0710E-13

Table 4.4 Function Error Values Achieved with four fitness functions f_1, f_2, f_3 and f_4 When $FES = 5 \times 10^3$, $FES = 5 \times 10^4$ for Test Functions g01-g13

FES	5×10^3					5×10^4				
	Best	Median	Worst	Mean	Std	Best	Median	Worst	Mean	Std
g01	7.1822E+00	7.7277E+00	7.7277E+00	7.5531E+00	2.5968E-01	1.4820E-07	1.4820E-07	1.0231E-04	3.6929E-05	5.0053E-05
g02	2.4311E-01	2.4311E-01	3.1059E-01	2.5660E-01	2.7549E-02	2.2807E-02	1.4556E-01	1.4556E-01	1.1251E-01	5.5527E-02
g03	NA	NA	NA	NA	NA	2.7793E-03	2.7793E-03	1.0114E-02	3.9529E-02	2.7446E-03
g04	1.2710E+01	1.2710E+01	4.2288E+01	1.9048E+01	1.2359E+01	3.6379E-12	7.2759E-12	7.2759E-12	5.1204E-12	1.4424E-11
g05	NA	NA	NA	NA	NA	1.3954E-03	1.3954E-03	1.3954E-03	1.3954E-03	5.1610E-11
g06	1.0451E+01	1.0451E+01	1.6199E+01	1.1477E+01	2.2419E+00	1.6370E-11	4.9386E-05	4.9386E-05	3.5558E-05	2.2631E-05
g07	5.4591E+00	5.6960E+00	5.6860E+00	5.6043E+00	1.1112E-01	5.8314E-03	5.8314E-03	8.2143E-02	2.4146E-02	3.3263E-02
g08	2.3534E-09	9.5024E-09	2.3534E-09	4.3551E-09	3.2760E-09	2.7755E-17	4.1633E-17	4.1633E-17	3.6082E-17	6.9388E-18
g09	1.8694E+00	8.4045E+00	8.4045E+00	6.8361E+00	2.8486E+00	8.2568E-06	8.2568E-06	1.8555E-04	2.9532E-05	5.8803E- 05
g10	1.6339E+03	1.6339E+03	NA	NA	NA	2.7483E+00	1.8283E+01	1.8283E+01	1.4100E+01	7.0271E+00
g11	NA	NA	NA	NA	NA	9.9990E-05	9.9990E-05	9.9990E-05	9.9990E-05	1.7579E-14
g12	4.0929E-10	4.2981E-09	4.2981E-09	3.9990E-09	1.0568E-09	0	0	0	0	
g13	NA	NA	NA	NA	NA	8.3328E-06	8.3328E-06	8.3328E-06	8.3328E-06	0

In summary, our experimental results confirm that the performance of CMODE with more helper functions is better than that of that with only two helper functions in some cases and almost the same in other cases.

4.4 Summary

This chapter proposes a new multi-objective problem formulation for solving constrained optimization problems. Besides the standard problem formulation with two objectives: to minimise the original objective function and the sum of degrees of constraint violation, other helper fitness functions are constructed from weighted sums of the normalised original objective and the normalised degree of constraint violation.

The new problem formulation is compared with the standard problem formulation using the same CMODE for solving MOPs. Experimental results show that CMODE with three fitness functions obtains remarkable better performance than that with the standard two fitness

functions [151] on most benchmark functions (12/13) from CEC2006 benchmark functions. This confirms our expectation that adding more helper functions may significantly improve the performance of MOEAs for CSOPs. The new problem formulation is extremely encouraging since our method is able to compete with other leading methods [13, 152, 80, 151].

However, according to the experiment study on IEEE CEC2017/2018 benchmark functions [161] in Chapter 7, this problem formulation cannot compete with other EAs participate in the competition [132, 133]. Therefor, we decide to develop another problem formulation in the next chapter.

Chapter 5

Helper and Equivalent Objectives: An Efficient Approach to Constrained Optimization

Multi-objective EAs for constrained optimization have been proposed over past two decades. Many empirical studies have demonstrated the efficiency of the multi-objective method [125]. 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 optimization competitions adopted multi-objective optimization [132]. The competition benchmark suite includes 50 and 100 dimensional functions. For a multi-objective optimization 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 CSOPs. The notation of 'helper objectives' was firstly introduced in [58]. 'helper objective' is an additional objective besides the original objective to be optimized. It can help diverse the population and form good building blocks. A helper objective is called an auxiliary objective in some references [3].

Most studies of 'helper objectives' are based on the observation of experimental tests. There are only a few theoretical analyses of EAs using 'helper objectives'. As early as in 2005, Neumann and Wegener [102] investigated the problem of minimum spanning trees by multi-objective EAs and prove that a single-objective problem might be solved more easily via a multi-objective model.

Friedrich et al. [37] proves that for the vertex cover problem, EAs using a multi-objective model can generate optimal solutions more quickly than EAs using the single-objective one; for the more general SETCOVER problem, EAs using a multi-objective model can generate some good approximation solution, while EAs using the single-objective approach cannot generate solutions with a guaranteed approximation ratio. He et al. [48] also proves that by using helper objectives, an evolutionary algorithm may produce $1/2$-approximation solutions to the 0-1 knapsack problem.

Recently, Antipov and Buzdalova use EA+RL method dynamically chooses two 'auxiliary objectives' (same meaning as 'helper objective') for optimization of the non-monotonic JUMP function. EA+RL method is analyzed whether it is effective to relearn which objective is helpful when helpfulness of 'auxiliary objectives' changes during optimization. Neumann and Sutton [101] analyzed the running time of a variant of global simple evolutionary multi-objective optimizer on the Knapsack problem. However, none of these theoretical analysis is based on multi-objective evolutionary algorithms for constrained optimization.

In this chapter, we propose a new multi-objective method for constrained optimization, which is to convert a constrained optimization problem into a problem with helper and equivalent objectives [166]. 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 optimization 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 analyze 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".

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 [49, 15] 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 chapter is a significant extension of our two-page poster in GECCO2019 [165]. The algorithm described in the current chapter is a slightly revised version of HECO-DE in [165]. 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 [132].

5.1 The Helper and Equivalent Objective Method

5.1.1 Helper and Equivalent Objectives

We start from a problem existing in the classical bi-objective method for solving problem (2.5). The Pareto optimal set to (2.5) is often significantly larger than Ω^*.

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

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

The degree of constrain violation is

$$v(x) = \max\{0, -\sin(x\pi/1200)\}. \tag{5.1}$$

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. 7.1.

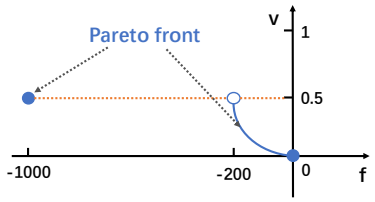


Fig. 5.1 Pareto front.

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 CSOPs, we introduce two concepts, equivalent and helper objectives. The term "helper objective" originates from [58].

Definition 6. *A scalar function $g(\vec{x})$ defined on Ω is called an equivalent objective function with respect to the CSOP (2.1) if it satisfies the condition:*

$$\arg\min\{f(\vec{x});\vec{x}\in\Omega\}=\Omega^*. \tag{5.2}$$

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 optimization. For example, a simple equivalent function is the death penalty function. Let Ω_F denote feasible solutions and Ω_I infeasible ones.

$$\min e(\vec{x})=\begin{cases} f(\vec{x}), & \text{if } \vec{x}\in\Omega_F,\\ +\infty, & \text{if } \vec{x}\in\Omega_I. \end{cases} \tag{5.3}$$

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 CSOPs, $\min(f,v)$ 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 7. *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 CSOP (2.1) if it satisfies the following condition:*

$$\arg\min\{f(\vec{x});\vec{x}\in\Omega\cap P\}=\Omega^*\cap P. \tag{5.4}$$

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 can not distinguish between equivalent and helper functions defined on the population.

An example is the superiority of feasibility rule [28] 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

$$e(\vec{x}) = \begin{cases} f(\vec{x}), & \text{if } \vec{x} \in \Omega_F \cap P, \\ v(\vec{x}) + f_F(P), & \text{if } \vec{x} \in \Omega_I \cap P, \end{cases} \tag{5.5}$$

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.

5.1.2 The Helper and Equivalent Objective Method

Once an equivalent objective function is obtained, the CSOP (2.1) can be converted to a single-objective optimization problem without any constraint.

$$\min e(\vec{x}), \quad \vec{x} \in P. \tag{5.6}$$

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 (5.6) 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 $\min e(\vec{x}), \vec{x} \in P_t \cup C_t$; remove repeated solutions.
5: **end for**

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

$$\min \vec{f}(\vec{x}) = (e(\vec{x}), h_1(\vec{x}), \cdots, h_k(\vec{x})), \qquad \vec{x} \in P. \tag{5.7}$$

Furthermore, we decompose problem (5.7) into several single objective problem. Decomposition-based multi-objective EAs have been proven to be efficient in solving multiobjective optimization problems [172, 142]. The decomposition method in the present work adopts the weighted sum approach, adding the helper objective onto the equivalent objective such that

$$\min w_0 e(\vec{x}) + \sum_{j=1}^{k} w_j h_j(\vec{x}), \quad \vec{x} \in P, \tag{5.8}$$

where $w_j \geq 0$ are weights.

Problem (5.7) is transformed into λ single-objective optimization subproblems by assigning λ tuples of weights $\vec{w}_i = (w_{0i}, w_{1i}, \cdots, w_{ki})$.

$$\min f_i = w_{0i}e + \sum_{j=1}^{k} w_{ji}h_j, \quad i = 1, \cdots, \lambda. \tag{5.9}$$

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 $[0,1]$.

$$g(\vec{x}) \leftarrow \frac{g(\vec{x}) - \max_{\vec{y}\in P} g(\vec{y})}{\max_{\vec{y}\in P} g(\vec{y}) - \min_{\vec{y}} g(\vec{y})}. \tag{5.10}$$

A helper and equivalent objective EA (HECO) for problem (5.9) is described as follows.

1: population $P_0 \leftarrow$ initialise a population of solutions;
2: **for** $t = 0, \cdots, T_{\max}$ **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 $i = 1, \cdots, \lambda$ where f_i is calculated by formula (5.9); 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 $(e, h_1, \cdots, h_k)$ (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 [172]. 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:

$$\lim_{t\to+\infty} w_{0i,t} > 0 \text{ and } \lim_{t\to+\infty} w_{ji,t} = 0 \text{ for } j > 0. \tag{5.11}$$

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 $(e, h_1, \cdots, h_k)$. 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 later.

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 (5.3) and (5.5). Search operators can be chosen from evolutionary strategies, differential evolution, particle swarm optimization and so on.

5.1.3 Implicit Equivalent Objective

Without the aid of an equivalent objective, a decomposition-based multi-objective EA for CSOPs faces 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 $x \in [-1000, 1000]$.

$$\begin{cases} \min f_1(x) = f, \\ \min f_i(x) = w_i f + (1-w_i)v, \quad i = 2, \cdots, \lambda - 1, \\ \min f_\lambda(x) = v. \end{cases}$$

The optimal solution to $\min f$ is $x = -1000$. The optimal solution to $\min f_i, i = 2, \cdots, \lambda - 1$ 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. But the superiority of feasibility rule is an equivalent objective (5.5), thus, many multi-objective EAs for CSOPs

implicitly utilise an equivalent objective. Based on this argument, multi-objective EAs for CSOPs 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 chapter, 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.

5.2 A Theoretical Analysis

5.2.1 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 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.

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 with a limited precision. Secondly, population P_t consists of finite individuals (points). But the probability of P_t at finite points always equals to 0 in a continuous space. To handle this issue, we 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:

$$\min \vec{f}(\vec{x}), \qquad \vec{x} \in \Omega. \tag{5.12}$$

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

Definition 8. *Given the optimization problem (5.12), $\vec{f}(\vec{x})$ is said to* dominate *$\vec{f}(\vec{y})$ (written as $\vec{f}(\vec{x}) \succ \vec{f}(\vec{y})$) if*

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

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 9. *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 [50]. The hitting time of an EA from one set to another is defined as follows.

Definition 10. *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,*

$$\Pr(P_0 = S_1 \subset \overline{S_2}, \cdots, P_t \subset \overline{S_2}) \geq \Pr(P'_0 = S_1 \subset \overline{S_2}, \cdots, P'_t \subset \overline{S_2}), \tag{5.13}$$

then $T(S_2 \mid S_1) \geq T'(S_2 \mid S_1)$. Furthermore, if the inequality (5.13) 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 average drift analysis [50]. This will not be discussed in the current chapter.

5.2.2 Fundamental Theorem

Now we compare SOCO for the single-objective problem (5.6) and HECO for the helper and equivalent objective problem (5.9). 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 [49, 15], 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 optimization, 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 11. *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 needed for mathematically comparing 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

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

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

Thanks to elitist selection and equivalent objective(s), 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 (5.6) and HECO for the helper and equivalent objective problem (5.9) 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,

$$\begin{aligned}
&\Pr(P_0 \subset \overline{L^b}, \cdots, P_t \subset \overline{L^b}) = \sum_{A \subset L} \cdots \sum_{Z \subset L} \Pr(P_0 = A, \cdots, P_t = Z) \\
\geq &\Pr(P_0' \subset \overline{L^b}, \cdots, P_t' \subset \overline{L^b}) \\
= &\sum_{A \subset L} \cdots \sum_{Z \subset L} \sum_{* \subset \overline{L^b}} \cdots \sum_{* \subset \overline{L^b}} \Pr(P_0 = A, \cdots, P_t = Z').
\end{aligned} \tag{5.14}$$

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.

The example presented below is a simple problem in constrained optimization. The purpose is to help understand our fundamental theorem.

In the theory of EAs, it is common to discuss extremely simple examples such as OneMax or Jump functions, like Antipov and Buzdalova [3] made their analysis on the Jump function. We follow a similar way to present the example.

However, our fundamental theorem is more general. It can be taken as the foundation of HECO algorithms for constrained optimization because the wide gap may exists in many problems.

Example 2. *Consider the CSOP below,*

$$\begin{cases} \min \quad f(x) = x, \qquad x \in [-500, 3000] \\ \textit{subject to} \quad g(x) = \sin(\frac{x\pi}{1000}) \geq 0. \end{cases} \tag{5.15}$$

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.

$$\min e(x) = \begin{cases} f(x), & \text{if } x \in \Omega_F, \\ v(x) + 3000, & \text{if } x \in \Omega_I. \end{cases} \tag{5.16}$$

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.

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

$$\min \vec{f}(x) = (e(x), f(x)), \qquad x \in [-500, 3000]. \tag{5.17}$$

Its Pareto front is displayed in Fig. 7.2.

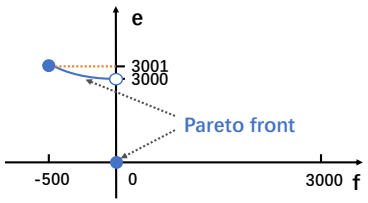


Fig. 5.2 Pareto front to the two-objective optimization problem (22)

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. 7.3 visualises the bypass in the objective space.

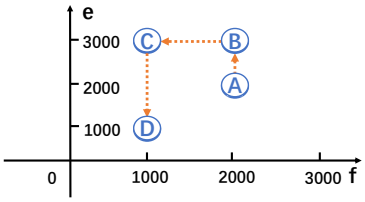


Fig. 5.3 A bypass in objective space: $A(2000,2000) \to B(2000-\varepsilon_1, 3000+\varepsilon_2) \to C(1000-\varepsilon_3, 3000+\varepsilon_4) \to D(1000,1000)$ where $\varepsilon_i \in (0,1)$ over the wide gap between fitness levels $e(x) = 2000$ and $e(x) = 1000$.

5.3 Summary

This chapter proposes the helper and equivalent objective method for constrained optimization. Its idea is to convert a CSOP into an equivalent and helper objective optimization problem, then to decompose it into several single objective subproblems using the weighted sum approach.

It has been theoretically proven that for the "wide gap" problem, using helper and equivalent objectives may shorten hitting time of crossing the "wide gap". To the best of our knowledge, this might be the first theoretical work to explain the strengths of multi-objective EAs in performing CSOPs.

Our fundamental theorem is general because it can be applicable to any HECO algorithm if it satisfies three conditions. There are no specific requirements of mutation, crossover, and selection, or helper and equivalent objectives. Among the three conditions, Conditions 1 and 2 are always true. Condition 3 is mild.

This theory has guided the design of HECO-DE, as an instance of HECO algorithm. Without this theory, it is impossible for us to design an equivalent objective.

Chapter 6

HECO-DE: A Case Study

In this chapter, we conduct a case study for validating our proposed HECO model. 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. The case study proves the efficiency of the helper and equivalent objective method for constrained optimization.

6.1 A Case Study

6.1.1 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 [107] is chosen as the SOCO algorithm because it is ranked only 4th in the CEC2017/18 competition [132]. If the constructed HECO algorithm outperforms LSHADE44 and winer 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 [171]). Mutant point $\vec{u}_i$ is generated from target point $\vec{x}_i$ by

$$\vec{u}_i = \vec{x}_i + F(\vec{x}_{pbest} - \vec{x}_i) + F(\vec{x}_{r_1} - \vec{x}_{r_2}), \tag{6.1}$$

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 [66]).

$$\vec{u}_i = \vec{x}_{r_1} + F(\vec{x}_{r_2} - \vec{x}_{r_3}), \tag{6.2}$$

$$\vec{u}_i = \vec{x}_{r_1^*} + F(\vec{x}_{r_2^*} - \vec{x}_{r_3^*}). \tag{6.3}$$

In (6.2), 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 (6.3), $\vec{x}_{r_1}$, $\vec{x}_{r_2}$ and $\vec{x}_{r_3}$ are chosen as that in (6.2) 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 [57]). Trial point $\vec{y}_i$ is generated from target point $\vec{x}_i$ and mutant $\vec{u}_i$ by

$$y_{i,j} = \begin{cases} u_{i,j}, & \text{if} \quad rand_j(0,1) \leq CR \text{ or } j = j_{rand}, \\ x_{i,j}, & \text{otherwise}, \end{cases} \tag{6.4}$$

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 [168]).

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 [143, 144] 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:

$$q_k = \frac{n_k + n_0}{\sum_{i=1}^{4}(n_i + n_0)}, \tag{6.5}$$

where n_k is the count of the kth 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 [107]. 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 [107]. 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 [107, 138]. 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:

$$N_t = round\left(N_0 - \frac{t}{T_{\max}}(N_0 - N_{T_{\max}})\right). \tag{6.6}$$

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

6.1.2 A New Equivalent Objective Function

Two equivalent functions (5.3) and (5.5) 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 $\vec{x}_P^*$ be the best individual in population P,

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

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

$$\tilde{e}(\vec{x}) = |f(\vec{x}) - f(\vec{x}_P^*)| \tag{6.7}$$

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

$$e(\vec{x}) = w_1\tilde{e}(\vec{x}) + w_2 v(\vec{x}), \tag{6.8}$$

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 $e(\vec{x})$ given by (6.8) 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 $\min\{e(\vec{x}); \vec{x} \in P\} = 0$. On one hand, for any $\vec{x} \in \Omega^* \cap P$, $\hat{e}(\vec{x}) = 0$ and $v(\vec{x}) = 0$, then $e(\vec{x}) = 0$. On the other hand, for $\vec{x} \in P$ such that $e(\vec{x}) = 0$, it holds $v(\vec{x}) = 0$, then $\vec{x} \in \Omega^*$. □

If two solutions $\vec{x}_1$ (infeasible) and $\vec{x}_2$ (feasible) in population P satisfy

$$w_1|f(\vec{x}_1) - f(\vec{x}_P^*)| + w_2 v(\vec{x}_1) < w_1|f(\vec{x}_2) - f(\vec{x}_P^*)|, \tag{6.9}$$

then under the equivalent objective function e, infeasible $\vec{x}_1$ is better than feasible $\vec{x}_2$. This feature may help search the infeasible region. For example, in Fig. 7.4, assume that $f(\vec{x}_1) - f(\vec{x}_P^*) = 0$ and $f(\vec{x}_2) - f(\vec{x}_P^*) = 1$, $v(\vec{x})_1 = 0.5$ and $w_1 = w_2$. Then we have $e(\vec{x}_1) = 0.5e(\vec{x}_2)$. Starting from $\vec{x}_1$, it is much easier to reach the left feasible region in which the optimal feasible solution $\vec{x}_P^*$ locates.

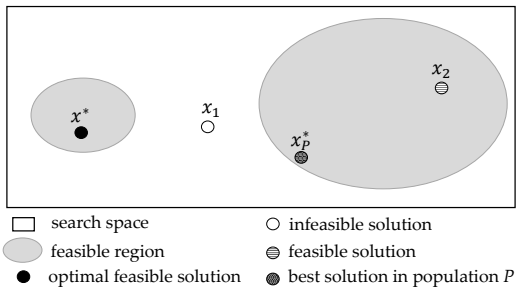


Fig. 6.1 There exist two feasible regions. An infeasible $\vec{x}_1$ satisfying (6.9) is better than $\vec{x}_2$ under the equivalent objective function e. This may help population $P = (\vec{x}_1, \vec{x}_2, \vec{x}_P^*)$ move from the right feasible region to the left feasible region in which the optimal feasible solution $\vec{x}^*$ locates.

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

$$\min \vec{f}(\vec{x}) = (e(\vec{x}), f(\vec{x})), \qquad \vec{x} \in P, \tag{6.10}$$

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

$$\min f_i(\vec{x}) = w_{1i}\tilde{e}(\vec{x}) + w_{2i}v(\vec{x}) + w_{3i}f(\vec{x}). \tag{6.11}$$

An extra term $\tilde{e}$ is added besides the original objective function f and constraint violation degree v.

6.1.3 A New multi-objective EA for Constrained Optimisation

A HECO algorithm is designed which reuses search operators from LSHADE44 [107]. 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 $N_{T_{\max}}$, 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;
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 $f(\vec{x})$ and $v(\vec{x})$ for each $\vec{x} \in P_0$;
5: Increase counter FES by N_0;
6: **while** $FES \leq FES_{\max}$ (or $t \leq T_{\max}$) **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 λ 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 trial 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 (6.11).
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 [107]);
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 (6.1) is modified as

$$\vec{u}_i = \vec{x}_i + F_k(\vec{x}_{Qbest} - \vec{x}_i) + F_k(\vec{x}_{r_1} - \vec{x}_{r_2}), \tag{6.12}$$

This new mutation is called current-to-Qbest/1 mutation. For randr1/1 mutation (6.3), $\vec{x}_{r_1}$, $\vec{x}_{r_2}$ and $\vec{x}_{r_3}$ are not compared but just randomly selected from subpopulation Q. Thus it returns to the original rand/1 mutation (6.2).

Lines 12 and 16: ranking individuals is used in both mutation (6.1) and calculation of the equivalent function (6.8). 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 (details in Subsection 6.1.4). 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 [107].

6.1.4 A New Mechanism of Dynamical Adjustment of Weights

We propose a special mechanism for dynamically adjusting weights. Function f_i in subproblem (6.11) is a weighted sum of helper and equivalent functions:

$$f_i(\vec{x}) = w_{1i}\tilde{e}(\vec{x}) + w_{2i}v(\vec{x}) + w_{3i}f(\vec{x}), \tag{6.13}$$

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

$$w_{1i,t} = \frac{t}{T_{\max}} \cdot \frac{i}{\lambda}, \tag{6.14}$$

$$w_{2i,t} = \frac{t}{T_{\max}} \cdot \frac{i}{\lambda} + \gamma, \tag{6.15}$$

$$w_{3i,t} = \left(1 - \frac{t}{T_{\max}}\right) \cdot \left(1 - \frac{i}{\lambda}\right), \tag{6.16}$$

where λ is the number of subproblems. $T_{\max}$ 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 6.2 and 6.3 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.

6.2 Comparative Experiments and Results

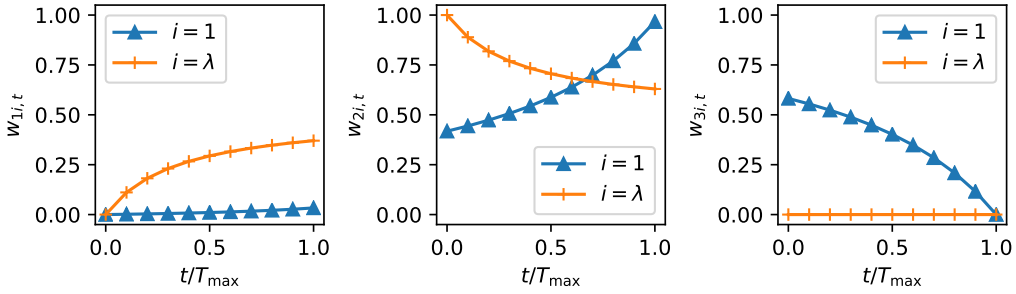


Fig. 6.2 The change of weights for 1st and λth individuals on CEC2006 benchmark functions. $\gamma = 0.7$.

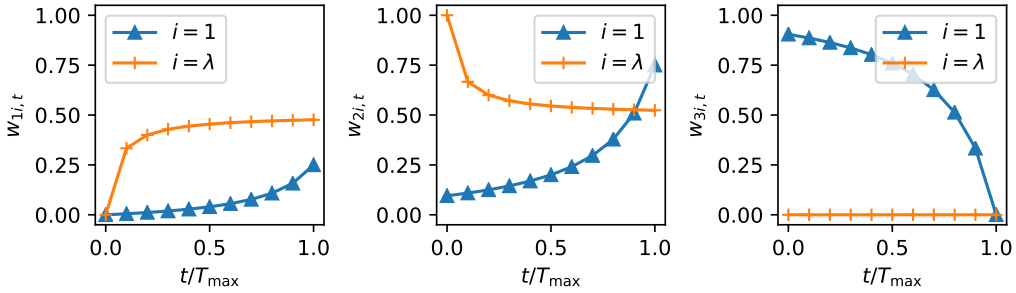


Fig. 6.3 The change of weights for 1st and λth individuals on CEC2017/2018 benchmarks. $\gamma = 0.1$.

6.2.1 Experimental Setting

HECO-DE was tested on two well-known benchmark sets from IEEE CEC2017/2018 and the IEEE CEC2006 introduced in Chapter 3.

Tables 6.1 and 6.2 list the parameter setting used in HECO-DE. In Table 6.1, parameters inherited from LSHADE44 are set to values similar to LSHADE44 [107].

Table 6.1 Parameters inherited from LSHADE44

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	$\sigma = 0.0001$

In Table 6.2, population size N_0, the number of subproblems λ and constraint violation bias γ are set to different values on CEC2006 and CEC2017 benchmarks. Since CEC2006

benchmarks include more constraints, both the values of λ and γ are set higher on CEC2006 benchmarks than that on CEC2017. The initial population size N_0 is set to a constant on CEC2006 benchmarks, while it is set to $12D$ on CEC2017 benchmarks because the dimension D ranges from 10 to 100. As required by the competitions, twenty five independent runs were taken on each benchmark.

Table 6.2 Different parameter setting in CEC2006 and CEC2017

CEC2006	
$FES_{\max}$ from CEC2006 benchmarks	$FES_{\max} = 500,000$
required population sizes	$N_0 = 450$, $N_{T_{\max}} = \lambda$
population size of Q	$\lambda = 45$
constraint violation bias	$\gamma = 0.7$
CEC2017	
$FES_{\max}$ from CEC2017 benchmarks	$FES_{\max} = 20000D$
required population sizes	$N_0 = 12D$, $N_{T_{\max}} = \lambda$
population size of Q	$\lambda = 20$
constraint violation bias	$\gamma = 0.1$

6.2.2 Experimental results on IEEE CEC2017 benchmarks

HECO-DE was compared with eight EAs introduced in Section 3.2. 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 (6.13), $\tilde{e}(\vec{x})$ 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 weighted sum (6.13), $\tilde{e}(\vec{x})$ is replaced by $e(\vec{x})$ given by (5.5). We call it HECO-DE(FR). The three algorithms adopt same parameter setting.

Table 6.3 summarizes the ranks of EAs on four dimensions and total ranks. HECO-DE is the top-ranked among 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 6.4 and 6.5 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.

6.2.3 Experimental results on IEEE CEC2006 benchmarks

HECO-DE was compared with five EAs, which are CMODE [151], NSES [61], FROFI [149], DW [105] and DeCODE [148], on IEEE CEC2006 benchmarks.

Table 6.3 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**

Table 6.6 summarises experiment results, where "Mean" and "Std Dev" denote the mean and standard deviation of objective function values, respectively. As suggested in [76], 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 6.6, "*" denotes that the algorithm satisfies this successful rule in 25 runs for a test problem.

As shown in Table 6.6, 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 6.7 shows that HECO-DE always find the optimum on all test functions. But without 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 (6.8).

Table 6.4 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

6.2.4 Convergence Speed of HECO-DE on IEEE CEC2006 Benchmark Suit

Fig. 6.4 plots the convergence speed at the median run of HECO-DE. The convergence speed is measured by the average convergence rate R_t defined as follows [47]:

$$R_t = 1 - \left| \frac{f_t - f^*}{f_0 - f^*} \right|^{1/t} \tag{6.17}$$

where R_t denotes the normalised convergence speed, t is the counter of the current generation, f_t is the objective value at t generation, and f^* the objective value of the known optimal solution.

Ten typical test function chosen from CEC2006 Benchmark are classified into five groups: quadratic, polynomial, linear, nonlinear and cubic. In each type of problems, we choose one function with relatively large feasible region and one function with very tiny feasible region.

As shown in Fig. 6.4, the convergence speed on all test functions is within the range around $[0.002, 0.01]$ after 50,000 generations. The case of g12 is special. At the beginning, the convergence speed is negative. This implies an infeasible solution with $|f_t - f^*| > |f_0 - f^*|$ is generated and accepted.

Fig. 6.4 shows that HECO-DE need more FES on test functions with tiny feasible region (g18, g03, g21 and g05) than test functions with large feasible region (g04, g09, g24 and

Table 6.5 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

g06) for satisfying the success criteria. However, this observation does not hold on nonlinear functions (g13 and g02).

6.2.5 Fine-tuning parameters on CEC2006 benchmark

CEC2006 benchmarks have more constraints than CEC2017 benchmarks. Thus the size of subpopulation λ and constraint violation bias in CEC2006 are set to different values from CEC2017. Fine-tuning of parameters λ and γ was conducted on IEEE CEC2006 benchmark functions. For brevity, only performance on g02, g10, g17, g21, and g23 are shown in Tables 6.8 and 6.9 while other functions share the same performance with different value of parameter λ and γ. As shown in Tables 6.8, the value $\lambda = 45$ is the best because HECO-DE can always solve all tested benchmark functions 100% successfully. As shown in Tables 6.9, $\gamma = 0.7$ gives the best performance. The λ and γ values are larger than those used in CEC2017 ($\lambda = 20$ and $\gamma = 0.1$). This is due to CEC2006 benchmarks are strongly constrained.

6.2.6 Detailed experimental results and ranking of HECO-DE on CEC2017 benchmarks

In terms of IEEE CEC2017 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 A.1-A.4.

Table 6.6 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 6.7 Comparison of HECO-DE with HCO-DE and HECO-DE(FR) on functions g02, g10, g21, and g23

CEC2006	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]**

As shown in Table A.1-A.4, 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.

6.2.7 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 6.10-6.17 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 6.10 and 6.11, 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

Table 6.8 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 6.9 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]

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 6.12 and 6.13, 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 6.14 and 6.15, 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

Table 6.10 Ranks of HECO-DE and Other EAs based on mean solution on the 28 functions of 10*D* 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 6.11 Ranks of HECO-DE and Other EAs based on median solution on the 28 functions of 10*D* 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**

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 6.16 and 6.17 record rank values based on mean values and median solution on the 28 test functions on 100*D*. 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. 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 6.12 Ranks of HECO-DE and Other EAs based on mean solution on the 28 functions of 30*D* 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 6.13 Ranks of HECO-DE and Other EAs based on median solution on the 28 functions of 30*D* 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 6.14 Ranks of HECO-DE and Other EAs based on mean solution on the 28 functions of 50*D* 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 6.15 Ranks of HECO-DE and Other EAs based on median solution on the 28 functions of 50*D* 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 6.16 Ranks of HECO-DE and Other EAs 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 6.17 Ranks of HECO-DE and Other EAs 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

6.3 Two Attempts to improve HECO-DE

In this section, we tried two measures to improve the performance of HECO-DE. One is using Tchebycheff decomposition approach instead of weighted sum approach. The other one is using three-population model.

6.3.1 Tchebycheff Decomposition Approach

In decomposition mechanism, apart from weighted sum approach in (6.13), Tchebycheff approach [91] was also tested in HECO-DE. Tchebycheff decomposition approach with HECO-DE is denoted as HECO-DE/tch.

Similar to MOEA/D [172], Tchebycheff approach is used to the optimise the three-objective optimization problem. The three-objective optimization problem is then decomposed into λ subproblems and the objective function of ith subproblem is

$$\min[f_i(x|w_i,z^*) = \max\{w_{1,i}|\tilde{e}(\vec{x}) - z_1^*|, w_{2,i}|v(\vec{x}) - z_2^*|, w_{3,i}|f(\vec{x}) - z_3^*|\}] \tag{6.18}$$

where z_1^*, z_2^* and z_3^* are the minimum value of $\tilde{e}(\vec{x})$, $v(\vec{x})$ and $f(\vec{x})$ respectively in each sub-population Q.

6.3.2 Comparative Experiments

For fair comparison, HECO-DEtch was tested under the same experiment setting.

Table 6.18 summarises the ranks of EAs on four dimensions and total ranks. HECO-DEtch is the top-ranked amongst all compared. This result clearly demonstrates that HECO-DEtch consistently outperforms other EAs on all dimensions. Without the equivalent function, HCO-DEtch is worse than HECO-DEtch and HECO-DEtch(FR). HECO-DEtch(FR) which uses the superiority of feasibility rule as the equivalent objective is slightly worse than HECO-DEtch. Tables 6.19 and 6.20 provide a sensitivity analysis of parameters λ and γ. HECO-DEtch with all five λ and γ values had obtained lower total ranks than other EAs.

Table 6.18 Total Ranks of HECO-DEtch and Other EAs on IEEE CEC2017 Benchmarks

Algorithm/Dimension	10D	30D	50D	100D	Total
CAL_LSAHDE(2017)	420	421	467	475	1783
LSHADE44+IDE(2017)	310	395	422	391	1518
LSAHDE44(2017)	330	344	343	344	1361
UDE(2017)	340	370	376	436	1522
MA_ES(2018)	269	260	272	282	1083
IUDE(2018)	197	260	261	330	1048
LSAHDE_IEpsilon(2018)	217	279	320	363	1179
DeCODE(2018)	233	296	294	325	1148
HCO-DEtch	277	254	252	222	1005
HECO-DEtch(FR)	163	195	197	**195**	750
HECO-DEtch	**157**	**151**	**173**	221	**702**

6.3.3 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 6.21-6.28 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 6.21 and 6.22, respectively. In terms of mean of solutions, HECO-DEtch had the lowest rank values on 11 of 28 problems (functions C01-C06, C08, C09, C16 and C22). However, HECO-DEtch got relatively poor performance on C07, C11 and C12. HECO-DEtch got the second lowest total rank value 82 which was slighter worse than the rank values obtained by HECO-DEtch(FR). In terms of median solution, HECO-DEtch got the lowest rank value on 12 of 28 problems (functions C01-C06, C08, C09, C13, C16, C22, and C25). However, its performance is poor on functions C11 and C21. HECO-DEtch was ranked first with a total rank value 75. The overall performance

Table 6.19 Total Ranks of HECO-DEtch with varying λ and Other EAs on IEEE CEC2017 Benchmarks

Algorithm/Dimension	10D	30D	50D	100D	Total
CAL_LSAHDE(2017)	506	507	569	568	2150
LSHADE44+IDE(2017)	379	484	517	470	1850
LSAHDE44(2017)	406	431	430	417	1684
UDE(2017)	427	477	469	527	1900
MA_ES(2018)	323	321	339	349	1332
IUDE(2018)	249	339	328	415	1331
LSAHDE_IEpsilon(2018)	278	352	412	444	1486
DeCODE(2018)	283	380	378	394	1435
HECO-DEtch($\lambda = 15$)	180	202	253	273	908
HECO-DEtch($\lambda = 20$)	**178**	184	222	255	839
HECO-DEtch($\lambda = 25$)	181	198	214	269	862
HECO-DEtch($\lambda = 30$)	196	**175**	**187**	**238**	**796**
HECO-DEtch($\lambda = 35$)	209	183	226	240	858

of HECO-DEtch 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 6.23 and 6.24, respectively. HECO-DEtch had the lowest rank values on 9 of 28 problems (functions C01, C02, C06, C09, C10, C13, C15, 20, C21 and C24). However, HECO-DEtch got relatively poor performance on functions C05 and C11. In terms of median solution, HECO-DEtch got the lowest rank value on 10 of 28 problems (functions C01-C03, C05, C06, C13, C15, C20, C21 and C27). However, its performance is poor on functions C11. Total rank values of HECO-DEtch were the lowest ones, 79 in terms of mean of solutions and 72 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 6.25 and 6.26, respectively. HECO-DEtch had the lowest rank values on 6 of 28 problems (functions C01, C02, C10, C14, C15 and C20). However, HECO-DEtch got relatively poor performance on functions C05 and C11. In terms of median solution, HECO-DEtch got the lowest rank value on 8 of 28 problems (functions C01, C02, C05, C10, C12, C13, C20 and C27). But its performance is poor on functions C11. Total rank values of HECO-DEtch were the lowest ones, 95 in terms of mean values and 78 in terms of median solution, respectively.

Table 6.27 and 6.28 record rank values based on mean values and median solution on the 28 test functions on 100D. HECO-DEtch had the lowest rank values on 4 of 28 problems (functions C01, C02, C05 and C20). But HECO-DEtch got relatively poor performance on

Table 6.20 Total Ranks of HECO-DEtch with Varying γ Values and Other EAs on IEEE CEC2017 Benchmarks

Algorithm/Dimension	10D	30D	50D	100D	Total
CAL_LSAHDE(2017)	500	507	567	568	2142
LSHADE44+IDE(2017)	374	480	517	474	1845
LSAHDE44(2017)	402	421	425	419	1667
UDE(2017)	419	468	465	521	1873
MA_ES(2018)	327	320	336	339	1322
IUDE(2018)	240	314	310	408	1272
LSAHDE_IEpsilon(2018)	269	346	410	451	1476
DeCODE(2018)	283	365	368	389	1405
HECO-DEtch($\gamma = 0.0$)	274	288	273	353	1188
HECO-DEtch($\gamma = 0.1$)	202	186	**199**	260	847
HECO-DEtch($\gamma = 0.2$)	**184**	**167**	210	**242**	**803**
HECO-DEtch($\gamma = 0.3$)	200	206	223	224	853
HECO-DEtch($\gamma = 0.4$)	197	249	243	233	922

Table 6.21 Ranks of HECO-DEtch and It's Two Variants and Other EAs 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	9	10	10	10	6	1	11	8	8	11	10	11	1	9	11	11	10	10	11	11	9	1	231
LSHADE44+IDE(2017)	1	1	9	8	1	11	8	1	1	2	1	3	1	10	4	9	7	9	4	4	2	5	9	7	9	6	10	9	152
LSAHDE44(2017)	1	1	10	6	1	10	7	1	9	2	2	10	1	9	9	10	8	8	2	1	6	8	8	6	10	7	8	10	171
UDE(2017)	1	1	8	9	10	8	6	1	7	2	10	1	10	7	5	8	9	7	8	11	9	10	7	3	8	10	7	7	190
MA_ES(2018)	1	1	1	10	1	6	4	1	1	2	4	11	8	11	10	1	11	1	10	10	10	7	11	5	1	9	4	8	160
IUDE(2018)	1	1	6	4	1	1	11	1	1	2	5	9	1	3	3	1	4	5	4	8	3	9	1	3	1	3	6	6	104
LSAHDE_IEpsilon(2018)	1	1	7	5	1	9	5	1	1	2	3	5	1	2	6	1	3	2	3	7	8	1	4	8	1	5	3	11	107
DeCODE(2018)	1	1	1	7	1	1	3	9	7	1	9	4	9	1	1	7	2	10	9	6	1	1	6	1	7	1	11	4	122
HCO-DEtch	1	1	1	1	1	1	2	11	11	11	11	6	1	3	11	1	6	6	11	3	4	6	5	11	1	8	5	5	145
HECO-DEtch(FR)	1	1	1	1	1	1	1	1	1	2	7	8	1	3	7	1	1	3	4	5	7	1	3	9	1	4	1	2	**79**
HECO-DEtch	1	1	1	1	1	1	10	1	1	2	8	7	1	6	2	1	5	4	4	2	5	1	2	2	6	2	2	2	82

functions C08, C11, C12, C13 and C21. HECO-DEtch got the second lowest total rank value 111 here, worse than HECO-DEtch(FR) with rank value 102. In terms of median solution, HECO-DEtch got the lowest rank value on 7 of 28 problems (functions C01, C02, C05, C14, C15, C20, C23 and C27). But it had a poor performance on functions C08, C11-C13 and C21. HECO-DEtch got the second lowest total rank value 110 which was only worse than the rank values obtained by HECO-DEtch(FR) with rank value 93.

According to the competition rules, HECO-DEtch got the lowest total rank values on dimension 10, 30 and 50, and second lowest total rank values on dimension 100. This means that HECO-DEtch 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 6.22 Ranks of HECO-DEtch and It's Two Variants and Other EAs based on median solution on the 28 functions of 10*D* 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	4	1	1	2	2	5	1	3	10	1	9	1	8	10	6	1	3	7	1	8	1	9	109
IUDE(2018)	1	1	1	1	1	1	11	1	1	2	1	10	1	3	5	1	1	6	4	8	8	1	4	5	1	1	6	6	93
LSAHDE_IEpsilon(2018)	1	1	8	5	1	1	6	1	1	2	6	7	1	3	4	1	6	2	3	7	11	1	8	2	1	6	3	11	110
DeCODE(2018)	1	1	1	7	1	1	3	1	9	2	7	1	1	1	1	7	5	11	11	6	1	1	1	1	7	3	11	8	111
HCO-DEtch	1	1	1	1	1	1	2	11	1	11	11	8	1	6	11	1	2	5	9	3	9	1	7	11	1	5	5	5	132
HECO-DEtch(FR)	1	1	1	1	1	1	1	1	1	2	8	9	1	6	2	1	2	3	4	5	10	1	6	4	1	4	4	2	84
HECO-DEtch	1	1	1	1	1	1	5	1	1	2	9	6	1	6	2	1	2	4	4	2	7	1	5	3	1	2	2	2	**75**

Table 6.23 Ranks of HECO-DEtch and It's Two Variants and Other EAs based on mean solution on the 28 functions of 30*D* 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	5	11	1	11	10	4	11	11	11	9	11	11	10	1	6	11	11	11	10	11	11	9	1	233
LSHADE44+IDE(2017)	1	1	10	7	1	11	7	10	1	9	3	8	8	10	7	8	8	9	4	8	9	8	10	8	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	7	9	7	8	10	164
UDE(2017)	1	1	6	10	6	3	4	9	7	8	6	9	9	7	6	7	9	8	10	10	5	7	5	5	6	9	7	7	187
MA_ES(2018)	1	1	1	9	1	2	2	2	1	1	1	10	1	8	10	1	10	2	11	11	10	1	7	6	1	10	1	5	127
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	2	5	5	6	8	138
LSAHDE_IEpsilon(2018)	1	1	8	3	1	9	5	2	1	1	9	7	10	6	3	6	6	1	3	4	4	10	1	4	7	6	2	11	132
DeCODE(2018)	1	1	1	8	10	4	9	7	9	1	11	1	1	2	4	5	5	11	9	7	6	3	9	3	4	2	11	6	151
HCO-DEtch	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	1	5	4	130
HECO-DEtch(FR)	1	1	4	2	7	6	6	8	1	1	7	4	1	3	2	10	3	3	4	5	3	2	4	9	10	3	3	3	116
HECO-DEtch	1	1	5	6	7	1	3	2	1	1	8	2	1	3	1	3	2	4	4	1	1	5	2	1	3	4	4	2	**79**

Table 6.24 Ranks of HECO-DEtch and It's Two Variants and Other EAs based on median solution on the 28 functions of 30*D* 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	4	1	11	9	10	2	10	3	2	1	11	7	9	9	10	4	8	10	8	10	9	9	6	10	9	194
LSAHDE44(2017)	1	1	9	6	1	10	8	2	2	2	2	8	9	10	8	10	8	6	2	3	9	9	9	7	10	9	9	10	180
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	3	2	2	2	1	11	1	9	10	1	7	1	11	11	11	1	8	8	1	7	3	5	133
IUDE(2018)	1	1	7	7	1	9	7	2	8	2	5	6	1	1	3	5	4	7	8	9	4	1	1	1	6	1	6	8	122
LSAHDE_IEpsilon(2018)	1	1	8	3	1	8	6	2	2	2	8	7	10	3	5	8	6	2	3	4	8	10	6	5	8	5	4	11	147
DeCODE(2018)	1	1	1	7	1	6	10	2	8	2	11	1	1	1	3	6	5	11	10	7	4	1	11	4	5	8	11	6	145
HCO-DEtch	1	1	1	1	1	1	1	11	11	11	10	4	1	4	11	1	1	5	7	2	2	6	5	11	1	4	5	4	124
HECO-DEtch(FR)	1	1	1	1	1	1	2	8	2	2	7	5	1	4	1	3	3	3	4	5	3	5	3	2	3	3	2	2	79
HECO-DEtch	1	1	1	4	1	1	4	2	2	2	9	3	1	4	1	3	2	4	4	1	1	4	4	2	4	2	1	3	**72**

Table 6.25 Ranks of HECO-DEtch and It's Two Variants and Other EAs based on mean solution on the 28 functions of 50*D* 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	5	11	11	10	8	4	8	11	11	10	11	11	9	1	11	7	10	11	10	11	11	8	1	254
LSHADE44+IDE(2017)	10	1	10	3	1	10	8	10	7	10	1	5	8	10	7	8	9	8	4	7	10	7	9	9	9	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	8	8	7	9	166
UDE(2017)	1	1	6	9	11	4	5	9	1	9	5	7	10	7	6	6	10	10	10	8	4	9	3	6	6	10	10	6	189
MA_ES(2018)	1	1	1	8	1	1	3	2	8	1	3	11	9	8	9	1	7	1	11	10	9	6	8	7	1	6	1	5	140
IUDE(2018)	1	1	7	7	1	11	10	5	1	1	6	4	6	4	3	5	3	6	9	9	5	5	2	3	5	3	6	8	137
LSAHDE_IEpsilon(2018)	1	1	8	2	7	8	6	6	6	6	9	9	5	6	5	7	6	2	7	4	6	11	1	5	7	7	2	11	161
DeCODE(2018)	1	1	1	5	1	3	9	4	9	5	11	6	1	5	4	4	5	11	2	6	8	1	10	4	3	5	11	7	143
HCO-DEtch	1	1	1	11	1	7	1	3	11	11	10	1	2	1	11	1	1	5	8	2	1	2	6	11	10	4	5	4	133
HECO-DEtch(FR)	1	1	4	6	7	6	2	7	3	7	7	3	3	1	2	10	2	3	4	5	2	4	4	1	2	1	4	2	104
HECO-DEtch	1	1	5	4	9	2	4	8	3	1	8	2	4	1	1	3	4	4	4	1	3	3	5	2	4	2	3	3	**95**

Table 6.26 Ranks of HECO-DEtch and It's Two Variants and Other EAs 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	5	8	10	6	10	11	11	8	1	213
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	2	7	6	10	9	6	187
MA_ES(2018)	1	1	1	9	1	3	3	2	8	1	2	10	1	9	8	1	6	1	10	11	9	6	8	6	1	5	3	5	132
IUDE(2018)	1	1	7	8	1	11	10	4	1	1	6	5	1	5	1	6	2	6	2	10	5	5	2	1	4	4	6	8	124
LSAHDE_IEpsilon(2018)	1	1	8	2	1	8	6	7	3	6	9	9	7	1	5	8	7	2	8	4	6	11	5	5	7	7	4	11	159
DeCODE(2018)	1	1	1	6	1	5	9	6	10	5	11	6	1	5	2	5	5	11	3	6	7	1	11	4	4	6	11	7	151
HCO-DEtch	1	1	1	1	1	1	1	2	11	11	10	3	1	2	10	1	1	5	9	2	3	3	7	11	10	1	5	4	119
HECO-DEtch(FR)	1	1	5	7	1	2	2	8	3	7	7	2	1	2	3	3	4	3	5	7	4	4	1	2	2	2	2	2	93
HECO-DEtch	1	1	4	3	1	4	4	5	3	1	8	1	1	2	3	4	3	4	5	1	2	2	4	2	2	3	1	3	**78**

Table 6.27 Ranks of HECO-DEtch and It's Two Variants and Other EAs 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	7	5	11	10	7	9	5	7	10	11	10	11	11	10	1	11	11	8	11	11	11	11	8	1	250
LSHADE44+IDE(2017)	11	11	9	2	3	10	7	5	2	7	1	2	5	10	5	9	9	8	3	7	8	7	7	8	9	9	9	8	191
LSAHDE44(2017)	1	1	8	1	2	9	8	1	1	1	4	11	6	9	8	10	7	7	2	6	10	6	9	7	8	7	7	10	167
UDE(2017)	9	9	6	10	11	6	5	9	10	8	6	3	11	6	7	6	10	9	10	9	5	11	1	9	5	10	10	6	217
MA_ES(2018)	1	1	1	8	10	2	4	2	9	4	2	10	2	8	9	1	6	1	11	10	9	5	8	5	1	6	1	5	142
IUDE(2018)	1	1	10	9	6	11	10	3	5	5	3	4	7	4	4	7	4	6	8	8	3	9	5	6	6	4	6	9	164
LSAHDE_IEpsilon(2018)	8	8	7	3	9	8	6	7	6	6	7	5	4	5	6	8	8	2	6	5	4	10	2	10	7	8	4	11	180
DeCODE(2018)	1	1	1	6	8	4	9	8	3	10	11	8	1	7	3	5	5	11	7	3	2	2	10	3	4	5	11	7	156
HCO-DEtch	1	1	1	4	4	7	1	4	8	11	10	1	3	1	11	1	1	5	9	2	1	1	6	1	10	3	5	4	117
HECO-DEtch(FR)	1	1	4	7	4	1	2	6	11	3	8	6	8	2	1	3	2	3	3	4	6	3	3	2	2	1	2	3	**102**
HECO-DEtch	1	1	5	5	1	3	3	11	4	2	9	9	9	2	2	4	3	4	3	1	7	4	4	4	3	2	3	2	111

Table 6.28 Ranks of HECO-DEtch and It's Two Variants and Other EAs 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	10	7	9	5	5	10	4	10	11	11	10	1	3	4	8	6	11	11	11	10	1	225
LSHADE44+IDE(2017)	11	11	10	1	4	9	8	5	2	8	4	3	5	11	6	9	8	7	3	8	9	7	8	9	10	8	7	9	200
LSAHDE44(2017)	1	1	9	2	5	8	9	1	1	1	2	11	6	10	9	10	6	8	2	7	10	6	10	10	9	5	8	10	177
UDE(2017)	9	9	6	10	11	5	5	9	9	10	6	2	11	7	8	6	10	9	10	10	6	11	3	7	6	9	9	6	219
MA_ES(2018)	1	1	1	8	10	2	3	2	11	5	1	8	3	9	4	1	5	1	11	11	8	5	9	5	1	6	3	5	140
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	7	8	6	7	6	4	6	4	8	7	2	6	6	5	10	4	8	8	7	5	11	183
DeCODE(2018)	1	1	1	6	7	4	10	8	4	7	11	9	2	8	3	5	9	11	8	5	1	2	11	2	5	10	11	7	169
HCO-DEtch	1	1	1	4	1	11	1	4	4	11	10	1	1	1	11	1	1	5	9	2	2	1	7	1	2	3	4	4	105
HECO-DEtch(FR)	1	1	4	7	1	1	2	6	3	3	8	7	8	1	1	3	2	3	3	4	7	3	2	3	3	1	2	3	**93**
HECO-DEtch	1	1	5	4	1	3	4	11	4	2	9	10	9	1	1	4	3	4	3	1	11	4	1	4	4	2	1	2	110

6.3.4 Three-population Model

However, it is observed that computational time of HECO-DE increases quickly as the dimension D. In fact, this phenomenon happens in all algorithms. A solution to this shortcoming is parallel computation. This chapter considers a multi-population implementation of HECO-DE called HECO-DEm which has great potential in parallel processing. Its idea is to divide a population into several subpopulations and evolve them separately. A new feature of HECO-DEm is its potential in parallel processing because each subpopulation may run on a CPU core. Another new feature of HECO-DEm is different search directions in different subpopulations.

Based on the HECO framework (Chapter 5), we designed a multi-objective EA called HECO-DE for CSOPs. This section presents its multi-population implementation (named HECO-DEm) for potentially parallel computation. It is expected that using multi-population can maintain population diversity and increase search directions, then it may improve the capacity of HECO-DE to handle complex fitness landscapes especially in a high dimensional search space.

HECO-DE aims at solving λ subproblems (6.11) simultaneously. As shown in Fig.6.6, it utilises only one population P. At each generation, λ individuals are sampled from P for minimising f_i ($i = 1, \cdots, \lambda$) respectively. This implies λ different directions. Each individual in P has the same chance for searching one of directions f_i. The adaption of setting such as parameter values, weights and strategies is on the whole population P. This idea probably is not good. If individuals in P distribute over a wide area in several different local landscapes, a better idea is to search one local landscape by a group of individuals. Furthermore one population is not suitable for parallel computation.

A potential improvement is to split P into several subpopulations and maintain a certain level of search independence in each population. In this chapter, a new three-population model (Fig. 6.5) is proposed which split P into three subpopulations $P_n, n = 1, 2, 3$. We do not split the population to many due to the limitation of computational resource. Each population is used to solve a specific set of subproblems with different weights. Adaption happens within a subpopulation, rather than in the whole population.

$$
\begin{aligned}
&\min f_i^{P_1}(\vec{x}) = w_{1i}^{P_1}\tilde{e}(\vec{x}) + w_{2i}^{P_1}v(\vec{x}) + w_{3i}^{P_1}f(\vec{x}), \quad i = 1, \cdots, \lambda/3. \\
&\min f_i^{P_2}(\vec{x}) = w_{1i}^{P_2}\tilde{e}(\vec{x}) + w_{2i}^{P_2}v(\vec{x}) + w_{3i}^{P_1}f(\vec{x}), \quad i = \lambda/3+1, \cdots, 2\lambda/3. \\
&\min f_i^{P_3}(\vec{x}) = w_{1i}^{P_3}\tilde{e}(\vec{x}) + w_{2i}^{P_3}v(\vec{x}) + w_{3i}^{P_3}f(\vec{x}), \quad i = 2\lambda/3+1, \cdots, \lambda.
\end{aligned}
$$

The splitting is based on the superiority of feasibility rule [28]. Individuals in P_1 are better than those in P_2, while individuals in P_2 are better than those in P_3. Intuitively, each subpopulation searches different fitness levels on the landscape.

Another benefit of the multi-population model is its potential in parallel processing. Current experiments show that computation time on CEC 2017 competition benchmarks is about 10-20 hours. Thus running EAs on multi-core CPU may significantly shorten computation time.

6.3.5 Comparative Experiments and Results

The CEC 2017 constrained optimization competition benchmark suit [132] was used in the experiment. The suit consists of 4×28 functions with dimension $D = 10, 30, 50, 100$. The parameter setting in HECO-DEm are listed in Table 6.29.

Table 6.29 Parameter setting

number of fitness evaluations	$FES_{\max} = 20000D$
required population sizes	$N_0 = 12 \times D$, $N_{T_{\max}} = \lambda$
population size of Q	$\lambda = 10$
historical memory size	$H = 5$
number of strategies	$K = 4$
constant in strategy adaption	$n_0 = 2$
threshold in strategy adaption	$\delta = 1/20$

According to the evaluation criteria in the CEC 2018 competition [133], for each problem and each dimension, an EA must run 25 independent times.

6.3.6 Comparative Experiment Results

In order to evaluate the performance of HECO-DEm, it was compared with seven state-of-art single-objective EAs and one multi-objective EA. Seven single-objective EAs comes from CEC 2017 and 2018 constrained optimization competitions, which are CAL-SHADE [169], LSHADE44+IDE [145], LSHADE44 [107], UDE [141], MA-ES [51], IUDE [140], LSHADE-IEpsilon [36]. One multi-objective EA is DeCODE [148] which was published in 2018. HECO-DEm was also compared with HECO-DE. Hence, ten EAs participated in ranking. The detail of these algorithms is described in the supplement.

The ranking result of ten EAs on each dimension and the total ranks is presented in Table 6.30. HECO-DE and HECO-DEm are the top two algorithms. Both clearly outperform seven single-objective EAs in the CEC 2017 and 2018 competition and one most recent

MOEA. The performance of HECP-DEm and HECO-DE is similar. HECO-DEm performs slightly better than HECO-DE on high dimensions such as 50D and 100D but slightly worse than on low dimensions such as 10D and 30D. This result validates our initial conjecture, that is, using multi-population may improve the ability of HECO to handle complex fitness landscapes in a high dimensional search space. It is not surprised that HECO-DE performs better on low dimensions 10D and 30D because one population is sufficient.

Table 6.30 Ranking result of HECO-DE, DeCODE and EAs in CEC 2018 constrained optimization competition

Algorithm/Dimension	10D	30D	50D	100D	Total
CAL_LSAHDE(2017)	386	381	424	431	1622
LSHADE44+IDE(2017)	285	358	380	351	1374
LSAHDE44(2017)	298	306	307	310	1221
UDE(2017)	305	338	338	390	1371
MA_ES(2018)	247	236	244	250	977
IUDE(2018)	178	226	223	293	920
LSAHDE_IEpsilon(2018)	192	239	288	330	1049
DeCODE	213	266	264	288	1031
HECO-DE	**158**	**160**	171	181	**670**
HECO-DEm	182	166	**167**	**165**	680

6.3.7 Detailed Experimental Results of HECO-DEm

The best, median, worst, mean, standard deviation and feasibility rate of the function values tested by HECO-DEm on 10D, 30D, 50D and 100D are recorded in Table A.17-A.20 in Appendix A.

6.3.8 Detailed Ranking Results of Nine EAs

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 6.31,6.32,6.33,6.34,6.35,6.36,6.37 and 6.38 respectively.

6.4 Summary

In this chapter, a case study is conducted for validating this theory. A new algorithm, called HECO-DE, is designed which employs helper and equivalent objectives and reuses search operators from LSHADE44 [107]. Experimental results show that with the aid of helper and

Table 6.31 Ranks of HECO-DEm and Other EAs based on mean values on the 28 functions of 10 dimensions

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	10	10	10	6	7	10	10	10	6	1	10	7	8	10	9	10	1	8	10	10	9	10	10	10	8	1	213
LSHADE44+IDE(2017)	1	1	8	7	1	10	6	1	1	2	1	3	1	9	5	8	6	8	4	4	2	5	8	8	8	6	9	8	141
LSAHDE44(2017)	1	1	9	5	1	9	5	1	9	2	2	9	1	8	9	9	7	7	2	1	4	7	7	7	9	7	7	9	155
UDE(2017)	1	1	7	8	9	7	4	1	7	2	9	1	9	6	6	7	8	6	8	10	8	9	6	4	7	9	6	6	172
MA_ES(2018)	1	1	1	9	1	5	2	1	1	2	4	10	7	10	10	1	10	1	10	9	9	6	10	6	1	8	2	7	145
IUDE(2018)	1	1	5	3	1	1	8	1	1	2	5	6	1	3	4	1	5	5	4	7	3	8	1	4	1	4	5	5	96
LSAHDE_IEpsilon(2018)	1	1	6	4	1	8	3	1	1	2	3	5	1	2	7	1	4	2	3	6	6	1	2	9	1	5	1	10	97
DeCODE	1	1	1	6	1	1	1	9	7	1	8	4	8	1	1	6	3	9	9	5	1	1	5	1	6	2	10	4	113
HECO-DE	1	1	1	1	1	1	10	1	1	2	7	8	1	4	2	1	1	3	4	3	7	1	4	2	1	1	3	2	**75**
HECO-DEm	1	1	1	1	1	1	9	1	6	2	10	7	1	5	3	1	1	4	7	2	5	1	3	3	1	3	4	3	88

Table 6.32 Ranks of HECO-DEm and Other EAs based on median solution on the 28 functions of 10 dimensions

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	10	10	1	8	5	10	10	1	4	3	1	8	8	10	10	9	1	8	4	1	9	10	10	10	9	1	173
LSHADE44+IDE(2017)	1	1	9	5	1	10	6	1	1	2	2	4	1	10	7	9	6	8	4	4	5	1	10	9	9	8	8	2	144
LSAHDE44(2017)	1	1	8	7	1	9	7	1	1	2	4	10	1	9	9	8	7	6	2	1	1	1	8	8	8	6	7	9	143
UDE(2017)	1	1	6	8	1	7	4	1	8	2	9	1	1	1	5	7	9	7	9	10	1	1	1	5	7	9	6	5	133
MA_ES(2018)	1	1	1	9	1	1	2	1	1	2	2	5	1	3	10	1	8	1	8	9	6	1	3	7	1	7	1	8	102
IUDE(2018)	1	1	1	1	1	1	8	1	1	2	1	7	1	3	5	1	1	5	4	7	7	1	4	5	1	1	5	5	82
LSAHDE_IEpsilon(2018)	1	1	7	4	1	1	3	1	1	2	6	6	1	3	4	1	5	2	3	6	10	1	5	2	1	5	2	10	95
DeCODE	1	1	1	6	1	1	1	1	8	2	8	1	1	1	1	6	4	10	10	5	1	1	1	1	6	3	10	7	100
HECO-DE	1	1	1	1	1	1	9	1	1	2	7	9	1	6	2	1	2	3	4	3	8	1	6	3	1	2	3	2	**83**
HECO-DEm	1	1	1	1	1	1	10	1	1	2	10	8	1	6	2	1	2	4	4	2	9	1	7	4	1	4	4	4	94

equivalent objectives, HECO-DE has outperformed LSHADE44, and a latest decomposition-based MOEA, DeCODE [148]. This case study proves the efficiency of the helper and equivalent objective method for constrained optimization.

A simplified dynamic weight adjustment mechanism and Tchebycheff decomposition approach are employed to improve the performance of HECO-DE. The experiment results show that that using the new dynamic weight adjustment mechanism not only simplifies the formulation but also improve HECO-DE's performance comprehensively. However, Tchebycheff decomposition approach might not be as effective as weighted sum approach in HECO-DE according to the experiment study. a multi-population version of HECO-DE, called HECO-DEm. It divides a population into three subpopulations. Different from HECO-DE, different weights are assigned to objective decomposition in each subproblems. This makes search directions in one subpopulation are different from another subpopulation. Unlike HECO-DE, HECO-DEm is suitable for parallel computation. It is expected that using multi-population can maintain population diversity and increase search directions, then it may improve the capacity of HECO-DE to handle complex fitness landscapes especially in a high dimensional search space. Experiment results show that its overall performance is the same as HECO-DE and much better than other eight algorithms under comparison. The multi-population version performs slightly better than HECO-DE in 50 and 100 dimensional functions.

Table 6.33 Ranks of HECO-DEm and Other EAs based on mean values on the 28 functions of 30 dimensions

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	10	10	10	6	8	1	10	10	4	10	10	10	9	10	10	9	1	4	10	10	10	10	10	10	8	1	213
LSHADE44+IDE(2017)	1	1	9	6	1	10	4	10	1	9	3	7	7	9	7	8	7	8	4	6	8	7	9	9	8	7	9	8	183
LSAHDE44(2017)	1	1	8	4	1	9	5	2	1	1	2	5	6	8	8	9	6	6	2	1	7	8	7	8	9	6	7	9	147
UDE(2017)	1	1	5	9	7	4	2	8	7	8	7	8	8	6	6	7	8	7	9	9	4	6	5	6	6	8	6	6	174
MA_ES(2018)	1	1	1	8	1	3	1	2	1	1	1	9	1	7	10	1	9	2	10	10	9	1	6	7	1	9	1	4	118
IUDE(2018)	1	1	6	5	1	7	7	2	7	1	5	2	5	1	5	4	3	5	7	8	6	5	4	3	5	3	5	7	121
LSAHDE_IEpsilon(2018)	1	1	7	2	1	8	3	2	1	1	8	6	9	4	3	6	5	1	3	2	3	9	1	5	7	5	2	10	116
DeCODE	1	1	1	7	9	5	6	7	9	1	10	1	1	2	4	5	4	10	8	5	5	4	8	4	4	2	10	5	139
HECO-DE	1	1	1	3	8	1	9	2	1	1	6	4	1	3	1	1	2	3	4	7	2	3	3	1	1	4	3	2	**79**
HECO-DEm	1	1	1	1	1	1	10	9	1	7	9	3	1	5	1	1	1	4	6	3	1	2	2	2	1	1	4	3	83

Table 6.34 Ranks of HECO-DEm and Other EAs based on median solution on the 28 functions of 30 dimensions

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	10	10	1	6	8	1	1	1	4	8	10	7	9	10	10	8	1	4	3	10	6	10	10	10	7	1	168
LSHADE44+IDE(2017)	1	1	9	3	1	10	6	10	2	10	3	2	1	10	7	8	8	9	4	6	9	7	9	9	8	5	9	8	175
LSAHDE44(2017)	1	1	8	5	1	9	5	2	2	2	2	7	8	9	8	9	7	5	2	1	8	8	8	7	9	8	8	9	159
UDE(2017)	1	1	5	9	1	4	2	9	8	9	7	9	7	6	6	5	9	7	7	9	3	6	1	6	6	9	6	6	164
MA_ES(2018)	1	1	1	8	1	3	1	2	2	2	1	10	1	8	10	1	6	1	10	10	10	1	7	8	1	6	1	4	118
IUDE(2018)	1	1	6	6	1	8	4	2	8	2	5	3	1	1	3	4	3	6	7	8	3	1	1	1	5	2	5	7	105
LSAHDE_IEpsilon(2018)	1	1	7	2	1	7	3	2	2	2	8	5	9	3	5	7	5	2	3	2	7	9	3	5	7	3	2	10	123
DeCODE	1	1	1	6	1	5	7	2	8	2	10	1	1	1	3	5	4	10	9	5	3	1	10	4	4	7	10	5	127
HECO-DE	1	1	1	3	1	1	9	2	2	2	6	6	1	4	1	1	2	3	4	7	2	5	4	2	1	4	3	2	**81**
HECO-DEm	1	1	1	1	1	1	10	8	2	2	9	4	1	5	1	1	1	4	4	3	1	4	5	3	1	1	4	3	83

Table 6.35 Ranks of HECO-DEm and Other EAs based on mean values on the 28 functions of 50 dimensions

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	10	10	9	5	8	9	10	8	4	7	10	10	10	10	10	8	1	10	6	9	10	10	10	10	7	1	232
LSHADE44+IDE(2017)	9	1	9	5	1	9	5	8	7	10	1	4	7	9	7	8	8	7	4	4	9	5	8	9	9	8	8	9	188
LSAHDE44(2017)	1	1	8	3	1	8	4	1	3	1	2	9	6	8	8	9	7	6	3	1	10	6	6	8	8	7	6	8	149
UDE(2017)	1	1	5	9	10	4	2	7	1	9	5	6	9	6	6	6	9	9	9	7	3	8	5	6	6	9	9	5	172
MA_ES(2018)	1	1	1	8	1	2	1	2	8	1	3	10	8	7	9	1	6	1	10	9	8	4	7	7	1	5	1	4	127
IUDE(2018)	1	1	6	7	1	10	7	4	1	1	6	3	5	1	3	5	3	5	8	8	4	3	4	3	5	2	5	7	119
LSAHDE_IEpsilon(2018)	1	1	7	4	8	7	3	6	6	7	7	8	4	4	5	7	5	2	7	2	5	10	2	5	7	6	2	10	148
DeCODE	1	1	1	6	1	3	6	3	9	5	10	5	1	2	4	4	4	10	2	3	7	2	9	4	4	4	10	6	127
HECO-DE	1	1	1	2	1	6	9	5	3	4	8	1	3	5	1	1	2	3	4	6	1	7	3	1	1	3	3	2	88
HECO-DEm	1	1	1	1	1	1	10	10	3	6	9	2	1	3	1	1	1	4	6	5	2	1	1	2	1	1	4	3	**83**

Table 6.36 Ranks of HECO-DEm and Other EAs based on median solution on the 28 functions of 50 dimensions

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	10	10	1	6	8	10	7	8	4	7	10	6	10	10	10	9	1	3	7	9	6	10	10	10	7	1	192
LSHADE44+IDE(2017)	1	1	9	3	1	9	5	9	9	10	3	3	8	10	7	9	8	8	5	5	9	7	9	9	9	8	9	9	192
LSAHDE44(2017)	1	1	8	5	1	8	4	1	3	1	1	10	7	9	9	8	7	6	4	1	10	8	8	8	8	7	6	8	158
UDE(2017)	1	1	5	9	10	5	2	8	1	9	5	5	9	7	6	6	9	7	10	8	1	6	1	7	6	9	8	5	166
MA_ES(2018)	1	1	1	8	1	3	1	2	8	1	2	9	1	8	8	1	5	1	9	10	8	5	7	6	1	4	1	4	117
IUDE(2018)	1	1	6	7	1	10	7	3	1	1	6	4	1	3	1	5	2	5	2	9	4	4	1	1	4	2	5	7	104
LSAHDE_IEpsilon(2018)	1	1	7	2	1	7	3	6	3	7	7	8	6	1	5	7	6	2	8	2	5	10	5	5	7	6	2	10	140
DeCODE	1	1	1	6	1	4	6	5	10	6	10	5	1	3	4	4	4	10	3	4	6	3	10	4	4	5	10	6	137
HECO-DE	1	1	1	3	1	2	9	4	3	4	8	1	1	5	2	1	3	3	5	7	2	2	3	2	1	3	3	2	**83**
HECO-DEm	1	1	1	1	1	1	10	7	3	5	9	2	1	2	2	1	1	4	5	6	3	1	4	3	1	1	4	3	84

Table 6.37 Ranks of HECO-DEm and Other EAs based on mean values on the 28 functions of 100 dimensions

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)	9	9	10	10	6	4	9	10	8	9	5	6	9	10	10	10	10	9	1	10	10	7	10	10	10	10	7	1	229
LSHADE44+IDE(2017)	10	10	8	4	4	9	4	5	2	7	1	1	6	9	5	8	8	7	3	4	6	6	6	7	9	8	8	7	172
LSAHDE44(2017)	1	1	7	3	3	8	5	1	1	1	4	10	7	8	8	9	6	6	2	3	9	5	8	6	8	6	6	9	151
UDE(2017)	8	8	5	9	10	5	2	8	10	8	6	3	10	5	7	5	9	8	9	8	5	10	2	8	5	9	9	5	196
MA_ES(2018)	1	1	1	7	9	2	1	2	9	4	2	9	4	7	9	1	5	1	10	9	8	2	7	4	1	5	2	4	127
IUDE(2018)	1	1	9	8	5	10	7	3	6	5	3	4	8	1	4	6	3	5	8	7	2	8	5	5	6	3	5	8	146
LSAHDE_IEpsilon(2018)	7	7	6	5	8	7	3	6	7	6	7	5	5	4	6	7	7	2	6	2	3	9	4	9	7	7	4	10	166
DeCODE	1	1	1	6	7	3	6	7	3	10	10	7	3	6	3	4	4	10	7	1	1	1	9	3	4	4	10	6	138
HECO-DE	1	1	1	2	1	6	8	4	3	2	8	8	2	3	1	1	2	3	3	6	7	4	3	1	3	2	1	2	89
HECO-DEm	1	1	1	1	1	1	10	9	5	3	9	2	1	2	1	1	1	4	5	5	4	3	1	2	2	1	3	3	**83**

Table 6.38 Ranks of HECO-DEm and Other EAs based on median solution on the 28 functions of 100 dimensions

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)	9	9	10	10	7	5	9	10	6	9	5	5	9	1	10	10	10	9	1	1	4	7	6	10	10	10	9	1	202
LSHADE44+IDE(2017)	10	10	9	1	3	9	5	5	2	8	4	3	6	10	6	8	7	6	3	5	8	6	7	8	9	7	6	8	179
LSAHDE44(2017)	1	1	8	4	4	8	6	1	1	1	2	10	7	9	9	9	5	7	2	4	9	5	9	9	8	4	7	9	159
UDE(2017)	8	8	5	9	10	4	2	8	8	10	6	2	10	6	8	5	9	8	9	9	6	10	2	6	5	8	8	5	194
MA_ES(2018)	1	1	1	7	9	2	1	2	10	5	1	7	4	8	4	1	4	1	10	10	7	4	8	4	1	5	1	4	123
IUDE(2018)	1	1	7	8	5	10	4	3	9	4	3	4	8	2	7	6	3	5	7	8	3	8	5	5	6	3	5	7	147
LSAHDE_IEpsilon(2018)	7	7	6	5	8	6	3	6	7	6	7	6	5	5	4	7	6	2	6	3	5	9	4	7	7	6	4	10	164
DeCODE	1	1	1	6	6	3	7	7	3	7	10	8	3	7	3	4	8	10	8	2	1	2	10	3	4	9	10	6	150
HECO-DE	1	1	1	2	1	7	8	4	3	2	8	9	1	4	1	1	1	3	3	7	10	3	1	1	3	2	2	2	92
HECO-DEm	1	1	1	3	1	1	10	9	3	3	9	1	2	3	1	1	2	4	3	6	2	1	3	2	2	1	3	3	**82**

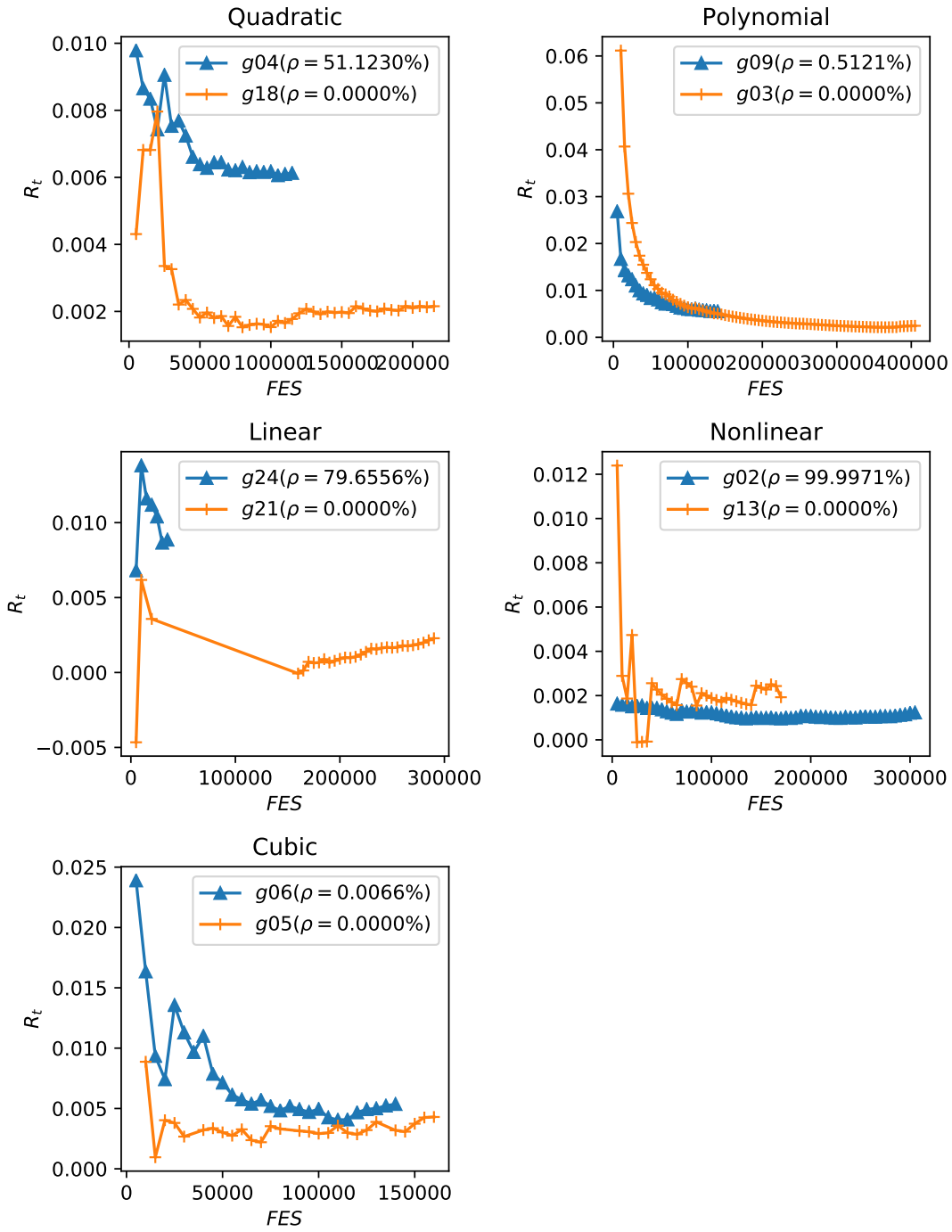


Fig. 6.4 Average convergence rates on ten typical CEC2006 benchmark functions which are divided into five groups, such as quadratic, polynomial, linear, nonlinear and cubic, where ρ denotes the estimated percentage of feasible region in the search space.

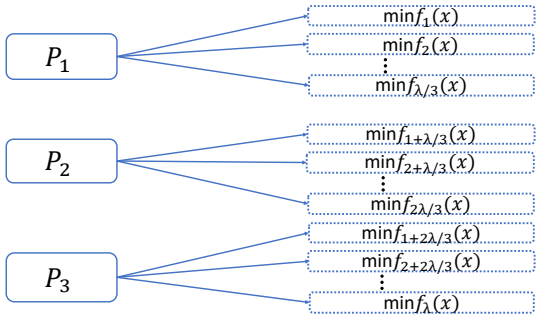


Fig. 6.5 Three-population $P_n, n = 1, 2, 3$ in HECO-DEm

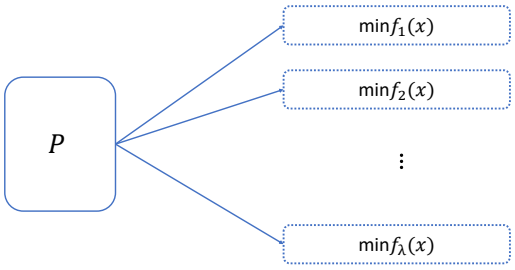


Fig. 6.6 One population P in HECO-DE

Chapter 7

HECO-PDE: An Enhanced version of HECO-DE with Principal Component Analysis

The word "valley" is a popular term used in intuitively describing fitness landscapes. What is a valley on a fitness landscape? How to identify the direction and location of a valley if it exists? However, such questions are seldom rigorously studied in evolutionary optimization especially when the search space is a high dimensional continuous space. This chapter presents two methods of studying valleys on a fitness landscape. The first method is based on the topological homeomorphism. It establishes a rigorous definition of a valley. A valley is regarded as a one-dimensional manifold. The second method takes a different viewpoint from statistics. It provides an algorithm of identifying the valley direction and location using principle component analysis.

Multiobjective evolutionary algorithms (MOEAs) have been successfully applied to a number of constrained optimization problems. Many of them adopt mutation and crossover operators from differential evolution. However, these operators do not explicitly utilise features of fitness landscapes. To improve the performance of algorithms, this chapter aims at designing a search operator adapting to fitness landscapes. Through an observation, we find that principle component analysis (PCA) can be used to characterise fitness landscapes. Based on this finding, a new search operator, called PCA-projection, is proposed. In order to verify the effectiveness of PCA-projection, we design two algorithms enhanced with PCA-projection for solving constrained optimization problems, called PMODE and HECO-PDE, respectively. Experiments have been conducted on the IEEE CEC 2017 competition benchmark suite in constrained optimization. PMODE and HECO-PDE are compared with the algorithms from the IEEE CE2017/2018 competition and another recent MOEA

for constrained optimization. Experimental results show that an algorithm enhanced with PCA-projection performs better than its corresponding opponent without this operator. Furthermore, HECO-PDE is ranked first on all dimensions according to the competition rules. This study reveals that decomposition-based MOEAs, such as HECO-PDE, are competitive with best single-objective and multiobjective evolutionary algorithms for constrained optimization, but MOEAs based on non-dominance, such as PMODE proposed in this chapter, may not.

7.1 Previous Work

In evolutionary optimization, the term "fitness landscape" is a metaphor [115] to intuitively describe the relationship between individuals (solutions) and their fitness values (solution quality). The landscape metaphor originates from population genetics which was first used by Wright [159] to visualize the relationship between biological genotypes and reproductive success. Currently fitness landscapes become a valuable concept in evolutionary biology and combinatorial optimization [117, 79].

A fitness landscape can be viewed as a mapping from a configuration space into a real space, while the configuration space is equipped with a distance measure or a neighborhood structure. Landscapes may change under different search operators or different distance mesurements [114]. For combinatorial fitness landscape, a formal landscape theory was proposed by Stadler [129] and then was further developed [130, 117].

The mathematical analysis of landscapes usually is a challenging task, thus several statistical analyzing tool were introduced for learn about the nature of landscapes. One of the earliest statistical measures of a landscape was the auto-correlation function proposed by Weinberger [157]. Davidor [27] suggests a simple statistic, called epistasis variance, as a mean to measure the amount of nonlinearity. Jones and Forrest [65] introduces the fitness distance correlation to classify easy and hard fitness landscapes. Reeves and Eremeev [116] took the number of optima as a statistical measure of a fitness landscapes. Merz [83] introduced the random walks technique for analyzing the fitness landscapes of combinatorial problems. Recently Moser et al. [94] proposed predictive diagnostic optimization as a means of characterizing combinatorial fitness landscapes.

So far a lot of work has contributed to combinatorial fitness landscapes, but continuous fitness landscapes still receive less analyses. Munoz et al. [96] introduced an information content-based method for continuous fitness landscapes and their method generates four measures related to the landscape features. This chapter focuses on studying a special landscape: valleys. It aims to provide a rigorous analysis of valleys and answer two questions:

1)What is a valley on a continuous fitness landscape specially when the search space is a high dimension space? 2)How to identify the direction and location of a valley if it exists?

Many MOEAs have been proposed for solving constrained optimization problems [125]. Most of them adopt mutation and crossover operators from differential evolution (DE). However, these operators do not explicitly utilise characteristics of fitness landscapes. Intuitively, a search operator which adapts to fitness landscapes may be more efficient than those without adaptation. A recent theoretical study also claims that in terms of the time-based fitness landscape, unimodal functions are the easiest and deceptive functions to EAs are the hardest [46]. Therefore, it is important to design a landscape-adaptive search operator which may improve the performance of EAs.

Our idea is to enhance EAs with principle component analysis (PCA). There exist a suite of studies which have shown PCA may improve the performance of EAs. Munteanu and Lazarescu [98] designed a PCA-mutation operator and claimed that PCA-mutation is more successful in maintaining population diversity during search. Because of PCA's inherent capability of rebuilding a new coordinate system, Li et al. [75] applied PCA to the design of crossover for reducing correlations among variables. PCA was used in particle swam optimization (PSO) to mine population information for promising principal component directions [16, 174, 103]. This information is utilized in velocity vectors of particles. Because PCA is a powerful tool in dimensional reduction, it helped EAs solve high dimensional optimization problems [162, 24]. Besides its application in designing search operators, local principal component analysis is used for building a regularity model in multiobjective estimation of distribution algorithms [173, 155]. However, the number of references of applying PCA to EAs is still very small and it is worth making further investigations.

In this chapter, we design a new search operator adapting to fitness landscapes with the aid of PCA. PCA is used to identify the maximal variance direction in a population. Given a "valley" fitness landscape in the 3-dimensional space, we observe that the direction obtained by PCA is consistent with the valley direction. Based on this observation, we design a new search operator, called PCA-projection. The research question of this chapter is whether a MOEA enhanced with PCA-projection is able to outperform its rival without this operator or other state-of-arts EAs. To answer this question, we design two MOEAs enhanced with PCA-projection, conduct experiments on the IEEE CEC 2017 benchmark suit for constrained optimization competition [132] and compare them with EAs from the CEC 2018 competition [133].

7.2 A Topological Method for Studying Valley Landscapes

Valleys is a popular term used in intuitively describing landscapes. But what is a valley on a fitness landscape especially in a high dimensional spaces? This section aims to provide a rigorous definition of valley and ridge landscapes from the topological viewpoint.

Continuous optimization problems can be roughly classified into two categories: minimization and maximization. For the sake of convenience, this chapter only considers the single-objective minimization problem without a constraint, which is given as follows:

$$\min f(x), \quad x \in \mathbb{R}^d, \tag{7.1}$$

where $f : \mathbb{R}^d \to \mathbb{R}$ is a continuous function and $\mathbb{R}^d$ is the d-dimension real space.

A global fitness landscape is the set of triples $\{(x, f, d) \mid x \in \mathbb{R}^d\}$ where d is the Euclidean distance in $\mathbb{R}^d$. A complex global landscape usually consists of several local landscapes such as valley, ridge and plateau landscapes. A local fitness landscape is a set of triples $\mathscr{L} = \{(x, f, d) \mid x \in \mathscr{S} \subset \mathbb{R}^d\}$ where $\mathscr{S}$ is a subset of $\mathbb{R}^d$. The core question in this section is under what kind of conditions, a landscape is called a valley? According to Oxford Online English Dictionary, a valley is "a low area of land between hills or mountains, typically with a river or stream flowing through it". This definition is applicable to $\mathbb{R}^2$. However, it becomes difficult to imagine a valley in a higher dimensional space. The meaning of "low area", "hills" and 'mountain" needs formalization.

What is the difference between a valley landscape and a non-valley landscape? Let's explain their difference by two simple non-valley and valley landscapes in the 2-dimensional space. The first example is a non-valley landscape:

$$\mathscr{L}_s = \{(x, f_s, d) \mid x \in \mathbb{R}^2\}, \tag{7.2}$$

where f_s is a sphere function, given as follows:

$$f_s(x) = x_1^2 + x_2^2, . \tag{7.3}$$

Figure 7.1 shows the contour and 3D graphs of the sphere landscape $\mathscr{L}_s$ in the domain $[-10, 10]^2$. Since $f_s(x)$ is a sphere function, it is a common sense that no valley exists on the sphere landscape. The sphere function can be taken as a natural benchmark landscape to decide whether any other landscape contains a valley or not.

Given any $x \in \mathbb{R}^2$, a point x' is said in the lower fitness area than $f(x)$ if $f(x') < f(x)$ and in the higher fitness area than $f(x)$ if $f(x') > f(x)$. The δ-neighbour of x is a hyper-cube,

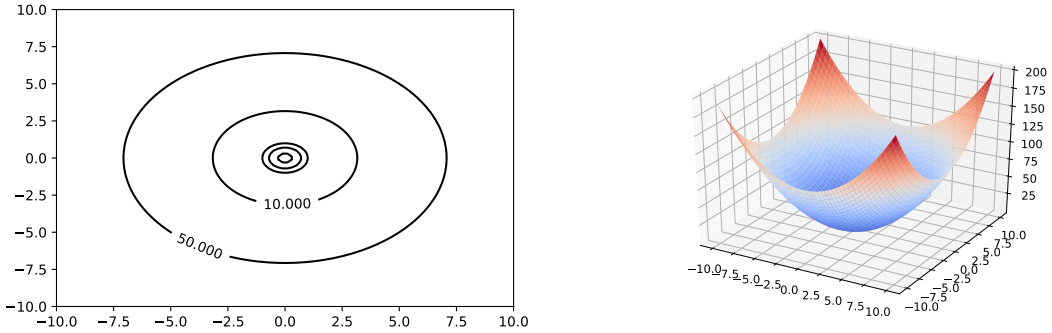


Fig. 7.1 The sphere landscape $\mathscr{L}_s$ where $f_s(x) = x_1^2 + x_2^2$.

given by

$$\mathrm{N}_\delta(x) = \{y \mid y \in [x_i - \delta, x_i + \delta]^2\}. \tag{7.4}$$

The area ratio between the lower fitness area and higher fitness area of the neighbor $\mathrm{N}_\delta(x)$ is calculated by

$$\frac{\mathrm{Area}(x' \in \mathrm{N}_\delta(x) \mid f_s(x') < f_s(x))}{\mathrm{Area}(x' \in \mathrm{N}_\delta(x) \mid f_s(x') > f_s(x))}. \tag{7.5}$$

where the area of a subset S is given by

$$\mathrm{Area}(S) = \int_{x \in S} d(x). \tag{7.6}$$

The second example is a simple valley landscape:

$$\mathscr{L}_e = \{(x, f_e, d) \mid x \in \mathbb{R}^2\}, \tag{7.7}$$

where $f_e(x)$ is an elliptic function, given as follows:

$$f_e(x) = x_1^2 + (0.1x_2)^2. \tag{7.8}$$

Figure 7.2 shows the contour graph of the elliptic landscape $\mathscr{L}_e$ in the domain $[-10, 10]^2$. Different from the sphere landscape $\mathscr{L}_s$, there is a valley on the elliptic landscape $\mathscr{L}_e$ which

is the line:

$$\mathscr{V}_e = \{x \mid x_2 = 0\}. \tag{7.9}$$

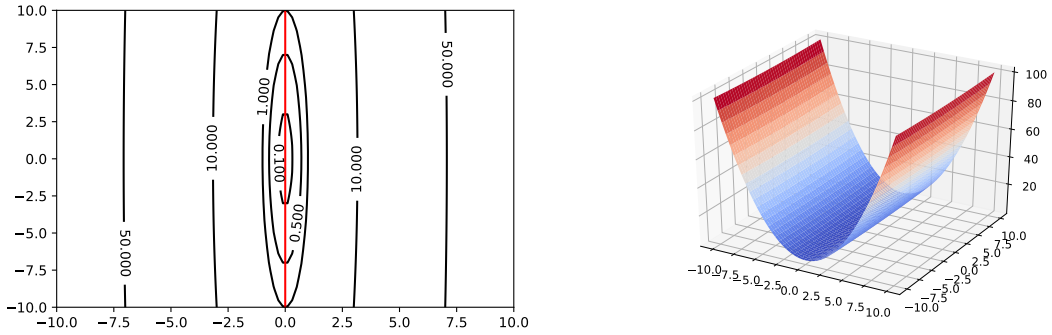


Fig. 7.2 The elliptic landscape $\mathscr{L}_e$ where $f_e(x) = x_1^2 + (0.1x_2)^2$.

The valley $\mathscr{V}_e$ satisfies two characteristics :

- $\mathscr{V}_e$ is a 1-dimensional manifold;
- $\mathscr{V}_e$ follows the gradient descent direction.

But these two characteristics are not sufficient for $\mathscr{V}_e$ to be a valley. Another important characteristic is observed from Figure 7.3, that is, for the elliptic function, the area ratio between the lower fitness area and higher fitness area of the neighbor $\mathrm{N}_\delta(x)$ is smaller than the area ratio for the sphere function.

Taking the sphere function as a benchmark, the above characteristic can be formalized as follows:

- $\exists \alpha > 0$ (e.g. set $\alpha = 10$ for $f_e(x)$), $\forall \delta \leq \alpha$ and $\forall x \in \mathscr{V}_e$, it holds

$$\frac{\mathrm{Area}(x' \in \mathrm{N}_\delta(x) \mid f_e(x') < f_e(x))}{\mathrm{Area}(x' \in \mathrm{N}_\delta(x) \mid f_e(x') > f_e(x))} < \frac{\mathrm{Area}(x' \in \mathrm{N}_\delta(x) \mid f_s(x') < f_s(x))}{\mathrm{Area}(x' \in \mathrm{N}_\delta(x) \mid f_s(x') > f_s(x))}. \tag{7.10}$$

The parameter α represents a degree of the valley width. Furthermore let

$$\beta = \max_{x \in \mathscr{L}} \frac{\mathrm{Area}(y \in \mathrm{N}_\delta(x)), f_e(y) < f_e(x))}{\mathrm{Area}(y \in \mathrm{N}_\delta(x)), f_e(y) > f_e(x))}. \tag{7.11}$$

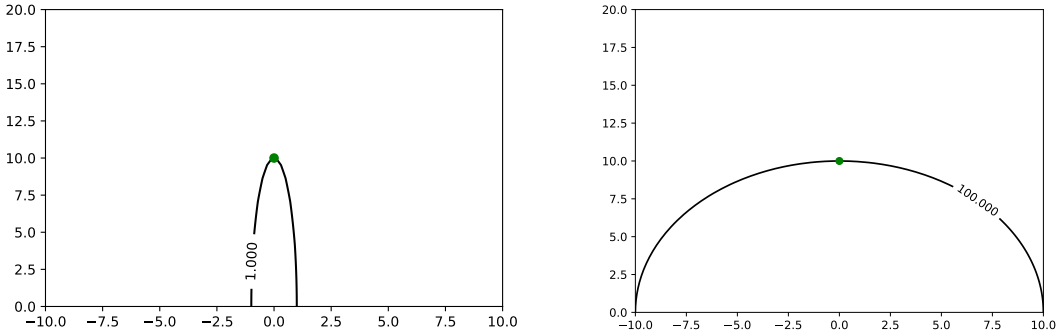


Fig. 7.3 A comparison between the elliptic landscape (left) and sphere landscape (right figure).

The parameter β represent a degree of the valley narrowness. It should be mentioned that the parameter β could take the value 0 in some extreme situation. For example,

$$\mathscr{L}_z = \{(x, f_z, d)\}, \tag{7.12}$$

where $f_z(x)$ is given as follows:

$$f_z(x) = x_1^2, \quad x \in \mathbb{R}^2. \tag{7.13}$$

Figure 7.4 shows the contour and 3D graphs of this special elliptic landscape. It is clear that a valley exists which is the line:

$$\mathscr{V}_z = \{x \mid x_1 = 0\}. \tag{7.14}$$

Beyond simple elliptic valley landscapes, there are many different and complex valley landscapes. It is impossible to list them one by one. A question is how to extend simple valley landscapes to a more general valley landscape. The extension can be implemented using the homeomorphism from topology. Given two topological spaces X and Y, a function $f : X \to Y$ is called a homeomorphism if it satisfies the following properties: f is an injection from X to Y, both f and its inverse function f^{-1} are continuous [95].

Given an elliptic function f_e and its simple valley $\mathscr{V}_e$, a general valley landscape can be topologically constructed using the homeomorphism technique. Let $h : \mathbb{R}^2 \to \mathbb{R}^2$ be a

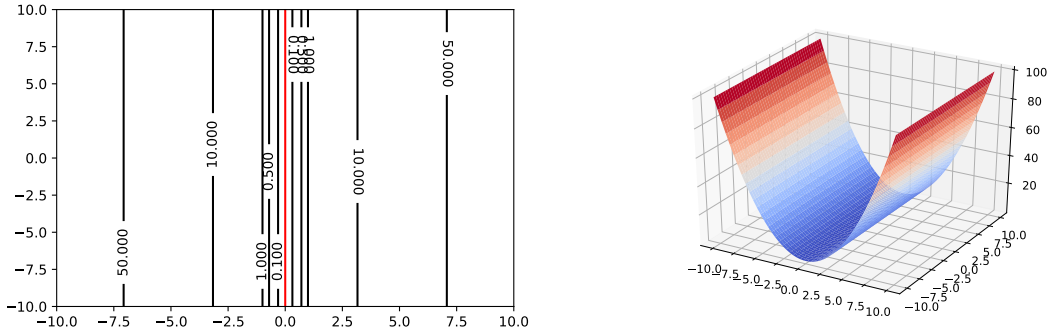


Fig. 7.4 The landscape $\mathscr{L}_z$ where $f_z(x) = x_1^2$.

homeomorphism and denote

$$y = h(x), \tag{7.15}$$

$$g(y) = f_e(h^{-1}(x)), \tag{7.16}$$

$$h(\mathscr{V}) = \{(y, h(y)) \mid h^{-1}(y) \in \mathscr{V}\}. \tag{7.17}$$

$h(\mathscr{V})$ is called a valley if the homeomorphism h satisfies the following two conditions: let $y = h(x)$ and $y' = h(x')$,

- the fitness order is preserved, i.e. $f(x) < f(x') \Longleftrightarrow g(y) < g(y')$;
- the area ratio related to the function g is smaller than the area ratio related to the sphere function f_s, i.e. $\exists \alpha > 0$ and $\alpha^\sharp > 0$, $\forall \delta \leq \alpha$ and $\delta^\sharp \leq \alpha^\sharp$, $\forall x \in \mathscr{V}$ and $y = h(x)$,

$$\frac{\text{Area}(y' \in \text{N}_{\delta^\sharp}(y) \mid g(y') < g(y))}{\text{Area}(y' \in \text{N}_{\delta^\sharp}(y) \mid g(y') > g(y))} < \frac{\text{Area}(x' \in \text{N}_{\delta}(x) \mid f_s(x') < f_s(x))}{\text{Area}(x' \in \text{N}_{\delta}(x) \mid f_s(x') > f_s(x))}. \tag{7.18}$$

The simplest homeomorphism which satisfies the about conditions is the linear transformation

$$y_1 = a_1 x_1, \tag{7.19}$$

$$y_2 = a_2 x_2 \tag{7.20}$$

where $a_1 \neq a_2$ are two constants.

Homeomorphism can be used to construct a well-known valley landscape which is generated from Rosenbrock function. Consider the simple elliptic function

$$f(x_1,x_2) = (x_1)^2 + 100(x_2)^2. \tag{7.21}$$

Let the homeomorphism $h(x_1,x_2) : \mathbb{R}^2 \to \mathbb{R}^2$ be

$$y_1 = 1 - x_1, \tag{7.22}$$
$$y_2 = x_2 + (1 - x_1)^2. \tag{7.23}$$

Then the Rosenbrock function is generated as follows:

$$g(y_1,y_2) = (1 - y_1)^2 + 100(y_2 - (y_1)^2)^2. \tag{7.24}$$

After studying valley landscapes in the 2-dimensional space $\mathbb{R}^2$, a general valley landscape in any d-dimensional space $\mathbb{R}^d$ can be defined in a similar way for any dimensionality $d \geq 2$.

Definition 12. *A simple elliptic valley landscape is*

$$\mathscr{L}_e = \{(x, f_e(x))\}, \tag{7.25}$$

where $f_e : \mathbb{R}^d \to \mathbb{R}$ is an elliptic function, given as follows:

$$f_e(x) = \sum_{i=1}^{d-1} (x_i)^2 + \gamma(x_d)^2, \quad x \in \mathbb{R}^d. \tag{7.26}$$

where the parameter $\gamma < 1$.

Although it is difficult to visualize a fitness landscape if $d > 3$, it still is possible to imagine the valley on this landscape $\mathscr{L}_e$ which is

$$\mathscr{V}_e = \{x \mid x_d = 0\}. \tag{7.27}$$

It is easy to verify the valley satisfying the following characteristics:

1. the valley is a 1-dimensional manifold;
2. the valley follows the gradient descent direction;

3. $\exists \alpha > 0$ (e.g. $\alpha = 10$ for $f_e(x)$), $\forall \delta \leq \alpha$ and $\forall x \in \mathscr{V}_e$, it holds

$$\frac{\text{Area}(x' \in \text{N}_\delta(x) \mid f_e(x') < f_e(x))}{\text{Area}(x' \in \text{N}_\delta(x) \mid f_e(x') > f_e(x))} < \frac{\text{Area}(x' \in \text{N}_\delta(x) \mid f_s(x') < f_s(x))}{\text{Area}(x' \in \text{N}_\delta(x) \mid f_s(x') > f_s(x))}. \tag{7.28}$$

where $f_s(x)$ is a sphere function in the $(d+1)$-dimensional space, given by

$$f_s(x) = \sum_{i=1}^{d} (x_i)^2, \quad x \in \mathbb{R}^d. \tag{7.29}$$

Based on the simple elliptic valley landscape, a general valley landscape is defined as below.

Definition 13. *A general valley landscape $\mathscr{V} = \{(x, g, d)\}$ is constructed from a simple elliptic $\mathscr{V}_e = \{(x, f, d)\}$ using the homeomorphism technique in the following way: let $h : \mathbb{R}^d \to \mathbb{R}^d$ be a homeomorphism and denote*

$$y = h(x), \tag{7.30}$$
$$g_e(y) = f_e(h^{-1}(x)), \tag{7.31}$$
$$h(\mathscr{V}) = \{(y, h(y)) \mid h^{-1}(y) \in \mathscr{V}\}. \tag{7.32}$$

$h(\mathscr{V})$ is called a valley if the homeomorphism h satisfies the following two conditions: let $y = h(x)$ and $y' = h(x')$,

- *the fitness order is unchanged, i.e. $f(x) < f(x') \Longleftrightarrow g(y) < g(y')$;*
- *the area ratio related to the function g is smaller than the area ratio related to the sphere function f_s, i.e. $\exists \alpha > 0$ and $\alpha^\sharp > 0$, $\forall \delta \leq \alpha$ and $\delta^\sharp \leq \alpha^\sharp$, $\forall x \in \mathscr{V}$ and $y = h(x)$,*

$$\frac{\text{Area}(y' \in \text{N}_{\delta^\sharp}(y) \mid g(y') < g(y))}{\text{Area}(y' \in \text{N}_{\delta^\sharp}(y) \mid g(y') > g(y))} < \frac{\text{Area}(x' \in \text{N}_\delta(x) \mid f_s(x') < f_s(x))}{\text{Area}(x' \in \text{N}_\delta(x) \mid f_s(x') > f_s(x))}. \tag{7.33}$$

At the end, the homeomorphism method of defining a valley can be generalized to define a ridge straightforward. The analysis is almost identical except that a valley represents a lower area but a ridge represents a higher area. Let's show this link by a simple elliptic landscape.

$$\mathscr{L}_e = \{(x, f_e, d) \mid x \in \mathbb{R}^2\}, \tag{7.34}$$

where $f_e(x)$ is an elliptic function, given as follows:

$$f_e(x) = -x_1^2 - (0.1x_2)^2, \quad x \in \mathbb{R}^2. \tag{7.35}$$

Figure 7.5 shows the contour and 3-D graphs of the landscape $\mathscr{L}_e$. The ridge on $\mathscr{L}_e$ which is the line:

$$\mathscr{V} = \{(x_1, x_2) \mid x_1 = 0\}. \tag{7.36}$$

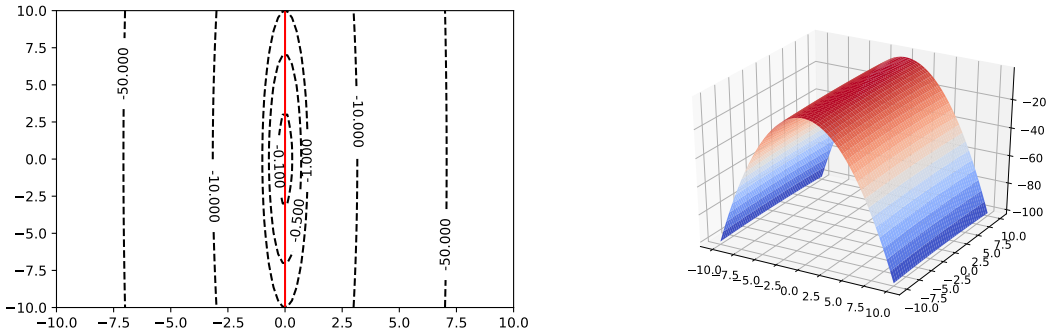


Fig. 7.5 The fitness landscape $\mathscr{L}_e$ where $f_e(x) = -x_1^2 - (0.1x_2)^2$.

The same topological method can be applied to studying a ridge on a fitness landscape because a ridge on the fitness landscape (x, f, d) is equivalent to a valley on the fitness landscape $(x, -f, d)$.

The topological method provides a rigorous definition of a valley or a ridge on a fitness landscape. Because the method is based on topology, it potentially may lead to a rigorous study of valleys and ridges.

7.3 A Statistical Method for Studying Valley and Ridge Landscapes

So far the definition of valleys has been established in the previous section. It is regarded as a one-dimensional manifold in a two or high dimensional space. But a big question still exists, that is how to identify its location and direction of a valley or a ridge if it exists in a fitness landscape. The topological method does not provide too much help. This section

presents a statistical method for studying the valley and ridge landscapes. The purpose is to find a practical method of identifying the location and direction of a valley or a ridge.

Let's still start from an intuitive observation of the simple sphere and elliptic landscapes discussed in the previous section:

$$\mathscr{L}_s = \{(x, f_s, d) \mid x \in \mathbb{R}^2\}, \tag{7.37}$$

$$\mathscr{L}_e = \{(x, f_e, d) \mid x \in \mathbb{R}^2\}, \tag{7.38}$$

where f_s is a sphere function and f_e is an elliptic function, given as follows respectively:

$$f_s(x) = x_1^2 + x_2^2, \tag{7.39}$$

$$f_e(x) = x_1^2 + (0.1x_2)^2. \tag{7.40}$$

For the elliptic landscape $\mathscr{L}_e$, Figure 7.6 shows the location of a valley is at the line $\mathscr{V} = \{x \mid x_0 = 0\}$. It is observed that the variance of the contour along the direction $x_0 = 0$ is much larger than that along the direction $x_1 = 0$. This leads to an important characteristic of the valley: the variance of the contour along the valley direction is maximal.

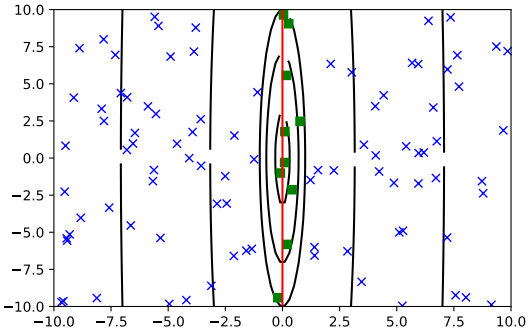


Fig. 7.6 The elliptic landscape with a valley

Based on the above observation, a statistical method is proposed for identifying the valley direction. The idea behind this method is statistical sampling. Suppose that a valley is located in a domain, that is $[-10, 10]^2$ in Figure 7.7. Sample a population of points from this domain at random. There are 100 points in Figure 7.7. The fitness value of these 100 points are evaluated and then the best 10 points are selected which are marked by "x". Figure 7.7 shows the best 10 points distribute along the valley. Therefore the valley direction can be regarded as a direction along which the variance of the selected points is maximal.

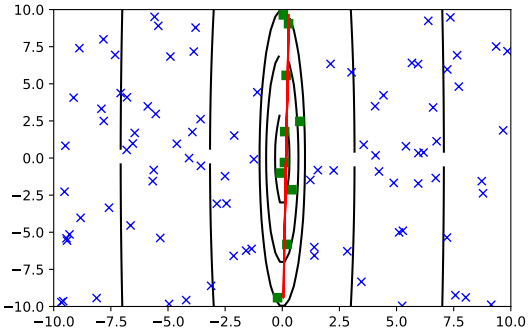


Fig. 7.7 The valley direction and location identified by PCA-projection.

The task of identifying the direction with the maximal variance in a data exactly can be implemented by the principle component analysis (PCA) [6]. Assume that the valley direction is linear, the valley direction and location then can be approximated by the first principle component found by linear PCA. Project the 10 selected points onto the first principle component. Figure 7.7 shows that the projected points (labeled by a red line) approximately represent the valley direction. This procedure is called PCA projection which is described by Algorithm 1.

Algorithm 1 PCA projection

1: Sample a population P of points from a domain;
2: Select M individuals $\{\mathbf{x}_1, \cdots, \mathbf{x}_M\}$ with smaller fitness values from the population P. Denote these individuals by $\mathbf{X}$.
3: Calculate the $d \times 1$ mean vector $\mathbf{m}$ and $d \times d$ covariance matrix ■:

$$\mathbf{m} = \frac{1}{M}\sum_{i=1}^{M}\mathbf{x}_i, \qquad \blacksquare = \frac{1}{M-1}\sum_{i=1}^{M}(\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^T. \tag{7.41}$$

4: Calculate the eigenvectors $\mathbf{v}_1, \cdots, \mathbf{v}_d$ of the covariance matrix ■, sorted them so that the eigenvalues of $\mathbf{v}_i$ is larger than $\mathbf{v}_j$ for $i < j$. Choose the first principle component $\mathbf{V} = [\mathbf{e}_1$.
5: Project $\mathbf{x}_i$ onto the first principle component:

$$\mathbf{y}_i = \mathbf{V}^T(\mathbf{x} - \mathbf{m}). \tag{7.42}$$

6: Reconstruct the projected point $\mathbf{x}_i$ in the original space:

$$\mathbf{x}_i' = \mathbf{m} + \mathbf{V}\mathbf{y}_i. \tag{7.43}$$

It should be pointed out that PCA-projection can be applied to any fitness landscape. Consider the application of PCA-projection to the sphere function. Sample 100 points from this domain at random and select the best 10 points. Project the 10 selected points onto the first principle component. Figure 7.8 shows that the projected points (labeled by a red line) with two different random seeds used in the sampling. Since the distribution of points along each direction through the original point should be the same, the direction of the projected points generated by PCA-projection could be any direction.

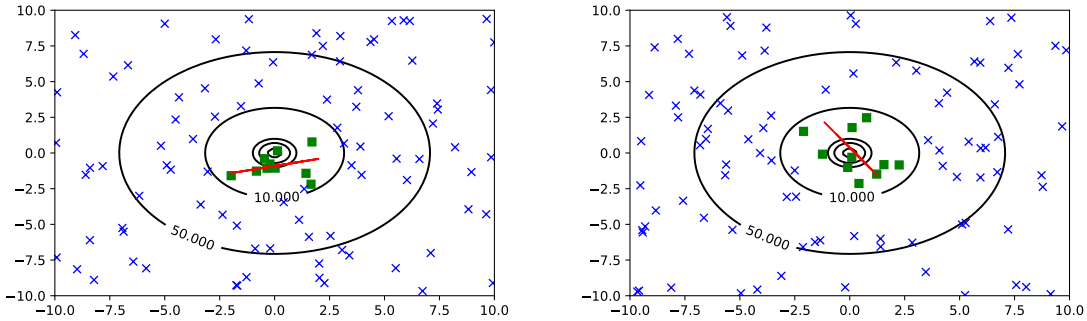


Fig. 7.8 The projected points after PCA-projection with two different random seeds.

At the end, PCA-projection is applied to a well-known valley landscape, called Rosenbrock function:

$$f_r(x_1,x_2) = (1-x_1)^2 + 100(x_2-x_1^2), \quad -1 < x_1 < 2, -1 < x_2 < 2 \tag{7.44}$$

Its minimum point is at $(1,1)$ with $f(1,1) = 0$. There exists a deep valley on the fitness landscape generated by Rosenbrock function. Sample 100 points from $[-1,2]^2$ at random and select the best 10 points. Project the 10 selected points onto the first principle component. Figure 7.9 shows that the projected points (labeled by dotted points) approximately represent the valley direction and location.

7.4 Application of Principle Component Analysis in Evolutionary algorithms

This chapter presents two methods of studying valley and ridge fitness landscapes. The first method is based on the topological homeomorphism. A rigorous definition of a valley and a ridge has been established. The second method is based on principle component analysis.

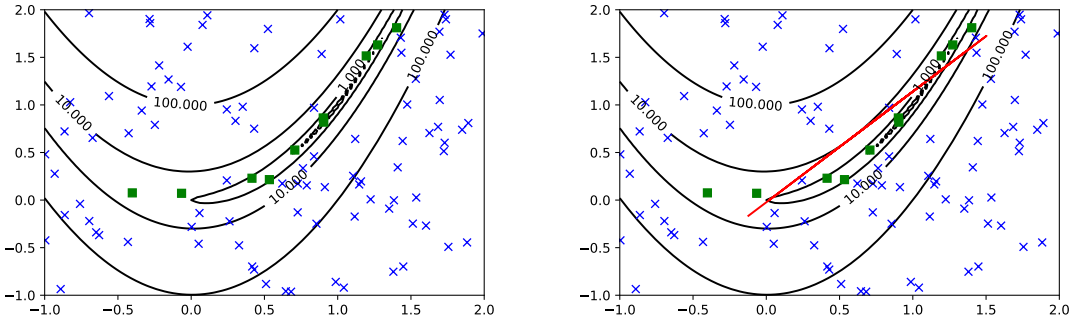


Fig. 7.9 PCA and the valley landscape

It provides an algorithm of identifying the direction and location of a valley or a ridge if it exists.

It is an interesting idea to apply PCA to the design of EAs but so far only a few research chapters can be found on this topic. Munteanu and Lazarescu's work [98] designed a mutation operator based on PCA. They claimed that a PCA-mutation genetic algorithm (GA) is more successful in maintaining population diversity during search. Their experimental results show that a GA with the PCA-mutation obtained better solutions compared to solutions found using GAs with classical mutation operators for a filter design problem.

Munteanu and Lazarescu [98] designed a new mutation operator on a projection search space generated by PCA, rather than the original space. Their PCA mutation is described as follows. A population with μ individuals is represented by an $n \times \mu$ matrix $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_\mu]$ where n is the space dimension and μ the population size. Each $\mathbf{x}$ is an individual represented by a column vector.

1: From the data set $\mathbf{X}$, calculate the $n \times n$ covariance matrix $\blacksquare$:

$$\blacksquare = \mathbb{E}[(\mathbf{x} - \mathbf{m})(\mathbf{x} - \mathbf{m})^T] \tag{7.45}$$

where $\mathbf{m} = \mathbb{E}[\mathbf{x}]$ which is the mean over $\mathbf{x}_1, \cdots, \mathbf{x}_\mu$.

2: Given the co-variance matrix $\blacksquare$, compute its eigenvectors $\mathbf{v}_1, \cdots, \mathbf{v}_n$ and sort them in the order of the corresponding eigenvalues of these eigenvectors from high to low. Form a $n \times n$ matrix $\mathbf{V} = [\mathbf{v}_1, \cdots, \mathbf{v}_n]$.

3: Calculate the projection of the data set $\mathbf{X}$ using the orthogonal basis $\mathbf{v}_1, \cdots, \mathbf{v}_n$ and obtain a projected population, represented by the matrix $\mathbf{Y} = [\mathbf{y}_1, \cdots, \mathbf{y}_n]^T$:

$$\mathbf{y}_i = \mathbf{V}^T(\mathbf{x}_i - \mathbf{m}). \tag{7.46}$$

4: Compute the squared length of the projections along each direction $\mathbf{v}_i$, that is,

$$\| L_i(x_j) \|^2 = y_{i,j}^2, \quad i = 1, \cdots, n, \quad j = 1, \cdots, \mu. \tag{7.47}$$

5: Choose quantities $c_{i,j}$ randomly between 0 and $c_{\max}$ where $c_{\max}$ is a constant parameter of the mutation operator such that $c_{i-1,j} \leq c_{i,j}$ for $i = 1, \cdots, n$.

6: The mutation operator adds the quantities $c_{i,j}$ to each projected squared coordinate as follows:

$$\| L_i'(x_j) \|^2 = \| L_i(x_j) \|^2 + c_{i,j}. \tag{7.48}$$

7: Compute the sign of each element in the matrix $\mathbf{Y}$, which is represented by the matrix $signum(\mathbf{Y})$.

8: Generate the child $\mathbf{y}_i'$ from $\mathbf{y}_i$ as follows: $y_{i,j}'$ equals to the square roots of the mutated square projections $\| L_i'(x_j) \|^2$ multiplied by the corresponding sign $signum(y_{i,j})$.

9: Obtain the mutated point in the original search space:

$$\mathbf{x}_i' = \mathbf{V}\mathbf{y}_i' + \mathbf{m}. \tag{7.49}$$

Notice that the above PCA-mutation does not reduce the data set $\mathbf{X}$ into a lower dimension space, instead $\mathbf{X}$ and $\mathbf{Y}$ have the same dimension. This PCA-mutation aims to conduct mutation in the projection space rather than the original space. However the dimensions of the projection space and original space are the same.

7.5 A New Search Operator: PCA-projection

In order to improve the performance of EAs, we propose a new search operator, called PCA-projection, which is able to adapt to fitness landscapes.

7.5.1 Principle Component Analysis and Valley Direction

Although PCA-mutation proposed in [98] was efficient for a filter design problem, it has a disadvantage. PCA-mutation still acts on the same dimension space as the original search space. Thus, as the population size increases, the calculation of eigenvalues and eigenvectors in PCA becomes more and more expensive. In this chapter, we propose a different PCA-search operator in which PCA is only applied to several selected points. A question is how to

select points from a population for implementing PCA? The solution relies on the "valley" concept.

7.5.2 Proposed PCA Projection

Based on the observation in the above subsection, we propose a new search operator as Algorithm 1. Here is our idea: Given a population, we select a group of points with smaller function values from the population and apply PCA to calculate principle components; then project the points onto the principle components; at the end, reconstruct the projected points in the original search space. These points are taken as the children. We call the search operator PCA-projection, rather than PCA-mutation [98], because it does not include a mutation step.

7.5.3 Characteristics of PCA-projection

PCA-projection is a nonlinear mapping from points in $\mathbb{R}^n$ to points in $\mathbb{R}^m$ which are assigned along the first m principle components. It can be regarded as a multi-parent recombination operator. Like other recombination operators, it works only if the parent population keeps a degree of diversity. Otherwise, it might degenerate. For example, if k points are identical (say $\mathbf{x}$), then after PCA-projection, the projected points are still $\mathbf{x}$. If k points distribute on the same line (say $\mathbf{y} = \mathbf{ax} + \mathbf{b}$), then the first principle component is $\mathbf{y} = \mathbf{ax} + \mathbf{b}$. After PCA-projection, there is no change to these k points.

PCA-projection generates the direction along which the distribution of points has the maximal variance. It is not the gradient direction. Let's show the difference through a simple example. Consider the minimisation problem

$$\min f(x,y) = |x| + 0 \times y.$$

If k points distribute on the same line (say $x_i = 1, y_i = i$, where $i = 1, \cdots, k$), then the first component direction found by PCA-projection is $x = 1$. But the gradient direction is perpendicular to the line $x = 1$.

Compared with PCA-mutation in [98], PCA-projection has three new characteristics:

1. The time complexity of PCA-projection is less than PCA-mutation in [98]. Given k points in $\mathbb{R}^n$, the covariance matrix computation in PCA is $O(k^2 n)$ and its eigenvalue decomposition is $O(n^3)$. So, the complexity of PCA is $O(k^2 n + n^3)$. In PCA-projection, only k good points are sampled from the population. For example, $k = 8$ in this chapter, so, its time complexity is $O(8^2 n + n^3)$. But in PCA-mutation, the number of

points equals to the population size. For example, population size is $12n$, so, its time complexity is $O(144n^3 + n^3)$.

2. The PCA-projection has an intuitive explanation. Given a valley landscape, it projects an individual to a new position along the valley direction.

3. It also takes the advantage of compressing a higher dimensional data into a lower dimension space. It projects a point into a lower dimensional space.

7.6 Two New Algorithms Enhanced by PCA-Projection

In this section, we present two MOEAs enhanced with PCA-projection for CSOPs.

7.6.1 New Algorithm 1: PMODE = CMODE + PCA-projection

We design the first algorithm though adding PCA-projection to CMODE, an algorithm proposed by Cai and Wang [13, 151]. It combines multiobjective optimization with differential evolution. CMODE is used to solve the standard bi-objective problem in equation 2.5

CMODE belongs to the family of MOEAs based on non-dominance. At each generation, it identifies non-dominant solutions under functions (f, v) and replaces those dominated solutions. It is straightforward to add PCA-projection into CMODE through the mixed strategy, that is to apply PCA-projection with probability p and normal mutation and crossover operations with probability $1 - p$. After PCA-projection is added into CMODE, we name the new algorithm PMODE.

The difference between PMODE and CMODE is PCA-projection. There is no other change in other parts. This design aims to evaluate the effectiveness of PCA-project without the interface of other factors.

7.6.2 New Algorithm 2: HECO-PDE = HECO-DE +PCA-projection

We design the second algorithm through adding PCA-projection to HECO-DE. PCA-projection is added into HECO-DE through a mixed strategy, that is to apply PCA-projection with probability p and normal mutation and crossover operations with probability $1 - p$. For the new algorithm enhanced with PCA-projection.

7.7 Comparative Experiments and Results

In order to demonstrate the effectiveness of PCA-projection, PMODE was tested on IEEE CEC2006 benchmark functions in Section **??**. HECO-PDE and PMODE are also tested on the IEEE CEC2017/2018 benchmark suite in constrained optimization competition in Section **??** and compared with the state-of-art EAs participated in the CEC 2018 competition [133].

Experimental settings is similar with HECO-DE. The parameters of HECO-PDE are set as follows. Although fine-tuning parameters of HECO-PDE may lead to better results, the setting is chosen as the same as that used in HECO-DE for a fair comparison.

1. the number of subproblem Q $\lambda = 12$;
2. in strategy competition, $n_0 = 2$, $K = 4$, $\delta = 1/20$;
3. the size of historical memories $H = 5$;
4. the initial and final required population sizes $\mu_0 = 12 \times D$, $\mu_{T_{\max}} = \lambda$.

7.7.1 General Performance of PMODE on CEC2006 benchmark

As shown in Tables 7.1−7.2, feasible solutions can always be found for 12 of 24 benchmark functions that are g01, g02, g04, g06, g07, g08, g09, g10, g12, g16, g19 and g24 within 5×10^3 FES. In 5×10^4 FES, feasible solutions can be found in every run for all benchmark functions apart from g20 and g22. g20 and g22 are very difficult for PMODE to solve because they are still far away from feasible region until 5×10^5 FES. However, within 5×10^5 FES, feasible solutions can be consistently found in all other 20 benchmark functions. Additionally, very close or equal to best known solution can be found in g01, g08, g10, g11, g12, g14, g16, g18, g19 and g24 in all runs, even better than best known solutions (shown as negative value) can always be found in g03, g04, g05, g06, g07, g09, g13, g15, g17 and g23. The result of the rest two benchmark functions g02 and g21 can also arrive at best known solutions in most runs.

Table 7.3 shows the number of FES in each successful run as suggested in CEC 2006 Competition: $| f(x) - f(\mathbf{x}^*) | \leq 0.0001$ and $\mathbf{x}$ is feasible. Feasible rate, the success rate, and the success performance are also recorded in Table 7.3. The feasible rate represents the percentage of runs where at least one feasible solution can be found by PMODE. The success rate denotes the percentage of runs where the PMODE can find a solution that satisfies the success condition. The success performance denotes the mean number of FES for successful runs.

As shown in Table 7.3, feasible solution with the probability 100% can be found in all benchmark functions except for g20 and g22, and no feasible solution found yet for these two function. For the success rate, PMODE can arrive 100% for all benchmark function apart from g02, g20, g21 and g22. However, the success rate of g02 and g21 are both over 90% which means the successful runs arise in a majority of trials for these two test functions. Regarding to the success performance, POMDE requires less than 1×10^5 FES for 16 test functions, less than 2×10^6 FES for 21 test functions and less than 2.7×10^6 FES for 22 test functions to achieve the target error accuracy level.

7.7.2 Experimental comparison of PMODE and CMODE

PMODE is compared with CMODE [151] on 24 benchmark test functions. 25 independent runs were executed on each test function and the maximum number of FES was 5×10^5.

Tables 7.4 reports the detailed comparative results of PMODE and CMODE on function error values and success performance. Additionally, a one-sample t-test [81] was implemented to verify the difference between success performance generated by PMODE and the results of COMDE. But the one-sample t-test was not used in function error values because the sample standard deviation s in function error values of PMODE sometimes equals to 0 and the t-test is invalid in this case. In the t-test, the null hypothesis is that the sample mean from 25 runs of PMODE equals to the population mean μ_0 whose value is taken from [151]. The statistic formula of one sample test is given as follows:

$$t = \frac{\overline{x} - \mu_0}{s/\sqrt{n}}, \tag{7.50}$$

where $\overline{x}$ denotes the sample mean from PMODE, s denotes the sample standard deviation of the sample and n denotes the sample size and μ_0 is the mean from [151].

Thus, the comparison of the success performance does not only depends on their values, but also should satisfies the statistic significance in the one-sample t-test, which means if p-value > 0.05, the results of success performance between PMODE and CMODE have no difference. As shown in Table 7.4, it can be observed that for $\overline{f^r(x)}$ (denotes function error values), PMODE clearly wins in 15 of 24 test functions (i.e., g03, g04, g06, g07, g08, g10, g13, g14, g15, g17, g18, g21, g23, g24) while CMODE is better in only 4 test functions (i.e., g01, g02, g09, g19). In the aspect of success performance, PMODE can achieve the target error accuracy level by fewer FES in 12 test functions (i.e., g02, g03, g05, g07, g09, g10, g14, g15, g17, g18, g21, g23) while CMODE have better performance in only 6 test functions (i.e., g01, g04, g06, g15, g19, g24). It can be observed that, although PMODE has smaller FES than CMODE, p-value by one-sample t-test > 0.05 in g11, g12 and g13. Thus, there are

no difference between the success performance of PMODE and CMODE on g11, g12 and g13 according to the one-sample t-test.

The test problem g20 is not listed in Table 7.5 since there is no feasible solution can be found. From Table 7.5, it can be seen that both PMODE and CMODE have good performance in all test functions but except g20 and g22. PMODE and CMODE have same performance for feasible rate in all test functions, where the average feasible rate are both 95.65%. However, PMODE wins again in success rate, although the success rate is not 100% in g02 and g21, PMODE can achieve an average 95.13% , whereas the success rate of CMODE is 94.78% on average.

7.7.3 Comparison of PMODE, CMODE and all EAs in CEC 2006 Competition

We also compare our experimental results with those EAs in CEC 2006 Competition. Table 7.6 lists the average feasible rate and success rate of all other twenty-three test functions tested by twelve EAs (PMODE, CMODE plus all 10 EAs participated in CEC 2006 Competition). DMS-PSO, ε_DE and SaDE can always get feasible solutions among all twenty-three test problems while PMODE and CMODE both arrive at a feasible rate 95.65%. DE, MDE and jDE-2 have the same performance with PMODE and CMODE in feasible rate. As shown by success rate, ε_DE achieves 95.65%, which is the highest score again. PMODE and CMODE also have a comparative performance in success rate with 95.13% and 94.78%, respectively.

Table 7.7 shows the success performance FEs divided by FEs of the best algorithm among the twelve EAs on twenty-three test problems. MDE, SaDE and DMS-PSO dominate among all competition algorithms including PMODE and CMODE on success performance, whereas PMODE and CMODE are ranked eighth and ninth respectively.

Table 7.8 lists the ranking of the twelve EAs in terms of $\overline{f^r(x)}$, feasible rate, success rate and success performance respectively. As a result, the final rank is calculated according to the overall ranking of all four measures. As we can see that, ε_DE and DMS-PSO win the first and second places among all twelve EAs respectively. It is worth mentioning that PMODE, proposed algorithm in this chapter, is in the third place while CMODE is only ranked seventh. Thus, PMODE gains a clear win against CMODE, and is among the top three EAs. This means PMODE is competitive with other types of EAs too.

7.7.4 Convergence Speed of PMODE

Fig. 7.10 describes the convergence speed of PMODE. The convergence speed is measured by the average convergence rate R_t defined as follows [47]:

$$R_t = 1 - \left| \frac{f(x_t) - f^*}{f(x_0) - f^*} \right|^{1/t} \tag{7.51}$$

where R_t denotes the normalized convergence speed, t the number of current generation, $f(x_t)$ the objective value at t generation, and f^* the objective value of the known optimal solution. In addition, R_t may take a negative value since the event $\mid f(x_t) - f^* \mid > \mid f(x_0) - f^* \mid$ could happen. This means, x_0 is an infeasible solution but its objective value is less than x_t which is a feasible solution. In this case, the convergence speed takes a negative value as shown by g23 in Fig. 7.10a.

Using the average convergence rate R_t, we can easily evalute and compare the convergence speed of different algorithms. It is better than the logarithmic rate $\log(f(x_t) - f_{\text{opt}})$ used in many references [76] because the logarithmic rate itself does not provide any information about the convergence rate but only its slop does. However, the average convergence rate R_t provides a quantitative value of the convergence speed.

Fig. 7.10 indicates the convergence speed of PMODE for 24 benchmark functions. In order to avoid stochastic distribution, the plotting stops at $f(x_t) - f^* \leq 10e^{-6}$. Since there is a large difference between convergence speed, test functions are divided into 8 groups by required FES, and each sub-figure contains two to four lines corresponding to their test functions. The horizontal axis represents FES, while the vertical axis represents R_t. As shown in Figs. 7.10a-7.10h, the convergence speed of all test functions follow the same rules: from high to low and become steady in the end. The average convergence rate R_t provides a quantitative value of the convergent speed. For example, $R_t = 0.0005$ means that the error $e_t = 0.9995^t e_0$ at the tth generation. Thus R_t provides an exact value of the convergent speed. However the index $\log(f(x_t) - f_{\text{opt}})$ cannot do it in this way.

For g23, g10 and g21 in Fig. 7.10a, 7.10d and 7.10g, the negative value of R_t means $f(x_t) > f(x_0)$. This means initially an infeasible solution is generated with a good function value $f(x_0)$ but later a feasible solution x_t is found with a worse function value $f(x_t)$.

In Fig. 7.10h, the function g22 is an intractable problem for PMODE which stops at 1.7×10^4 FES. The function error value doesn't make change after that FES.

7.7.5 Comparative Experimental Results on CEC2017 Benchmark

Table 7.9 summarises ranks of all algorithms on four dimensions and total ranks. HECO-PDE and PMODE got lower rank values than HECO-DE and CMODE, respectively. This result clearly demonstrates that HECO-DE and CMODE are improved by PCA-projection. Moreover, HECO-PDE got the lowest rank value among all compared algorithms. This result means that HECO-PDE enhanced with PCA-projection is the best in terms of the overall performance.

7.7.6 Detailed Experimental results of PMODE on CEC2017 benchmark

This subsection provides detailed experimental results of PMODE and HECO-PDE. 25 independent runs of PMODE and HECO-PDE are taken on each problem and dimension respectively. The maximum function evaluations is set to $20000 \times D$, where D is the dimension of an optimization problem.

Tables A.9−A.12 in Appendix A gives the experimental results of PMODE in terms of the best, median, worst, mean, standard deviation and feasibility rate of the function values $10D$, $30D$, $50D$ and $100D$. c is the number of violated constraints at the median solution: the sequence of three numbers 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 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 of 25 runs.

7.7.7 Experimental Results of HECO-PDE

Tables A.13−A.16 in Appendix A shows the experimental results of HECO-PDE in terms of the best, median, worst, mean, standard deviation and feasibility rate of the function values $10D$, $30D$, $50D$ and $100D$. c is the number of violated constraints at the median solution: the sequence of three numbers 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 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 of 25 runs.

The detailed rank values of all algorithms on mean values and median solutions on 28 test problems with the dimension of 10D, 30D, 50D and 100D are shown in Table 7.10-7.17, 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 7.10 and 7.11, respectively. As shown in Table 7.10, in terms of mean values with 10D, HECO-DE and CMODE got total rank values, with 81 and 236 respectively. By contrast, HECO-PDE got the lowest rank value, with 67 and PMODE also got a lower rank value than CMODE, with 218. As shown in Table 7.11, in terms of median solutions with 10D, HECO-DE and CMODE got total rank values, with 91 and 207 respectively. By contrast, HECO-PDE got the lowest rank value, with 87 while PMODE got the same rank value with CMODE, with 207.

Regarding the test functions with 30D, rank values based on mean values and median solution on the 28 test functions are reported in Table 7.12 and 7.13, respectively. As shown in Table 7.12, in terms of mean values with 30D, HECO-DE and CMODE got total rank values, with 82 and 307 respectively. By contrast, HECO-PDE got the lowest rank value, with 68 and PMODE got a slightly lower rank value than CMODE, with 305. As shown in Table 7.13, in terms of median solutions with 30D, HECO-DE and CMODE got total rank values, with 83 and 311 respectively. By contrast, HECO-PDE got the lowest rank value, with 71 and PMODE got lower rank value than CMODE, with 306.

Regarding the test functions with 50D, rank values based on mean values and median solution on the 28 test functions are reported in Table 7.14 and 7.15, respectively. As shown in Table 7.14, in terms of mean values with 50D, HECO-DE and CMODE got total rank values, with 90 and 314 respectively. By contrast, HECO-PDE got the lowest rank value, with 77 and PMODE got a slightly lower rank value than CMODE, with 312. As shown in Table 7.15, in terms of median solutions with 50D, HECO-DE and CMODE got total rank values, with 87 and 314 respectively. By contrast, HECO-PDE got the lowest rank value, with 75 while PMODE got a higher rank value than CMODE, with 318.

Regarding the test functions with 100D, rank values based on mean values and median solution on the 28 test functions are reported in Table 7.16 and 7.17, respectively. As shown in Table 7.16, in terms of mean values with 100D, HECO-DE got highest total rank value with 87 and CMODE got 315 respectively. By contrast, HECO-PDE got higher rank value than HECO-DE with 88 while PMODE got a lower rank value than CMODE, with 305. As shown in Table 7.17, in terms of median solutions with 100D, HECO-DE and CMODE got total rank values, with 96 and 315 respectively. By contrast, HECO-PDE got the lowest rank value with 89, while PMODE got a higher rank value than CMODE with 321.

According to the CEC 2018 competition rules, the ranks of HECO-PDE and HECO-DE are on the top two on each dimension but CMODE and PMODE on the bottom two. This result confirms our guess that MOEAs based on non-dominance, such as PMODE and CMODE, may not perform as good as decomposition-based MOEAs, such as HECO-PDE and HECO-DE for solving CSOPs.

7.8 summary

This chapter firstly presents two methods of studying valley and ridge fitness landscapes. The first method is based on the topological homeomorphism. A rigorous definition of a valley and a ridge has been established. The second method is based on principle component analysis. It provides an algorithm of identifying the direction and location of a valley or a ridge if it exists.

From an experimental observation, we find that given a valley landscape, the maximal variance direction in a population can be regarded as the valley direction. Based on this finding, a new search operator, called PCA-projection, is proposed, in which PCA is used to project points along the maximal variance direction. PCA-projection can be easily added into an existing MOEA through a mixed strategy. We design two MOEAs enhanced with PCA-projection, called HECO-PDE and PMODE, for evaluating the effectiveness of this new operator. Experimental results show that an EA enhanced with PCA-projection performs better than its corresponding opponent without this operator. The proposed PMODE not only has significantly improved the solution quality when compared with CMODE, an state-of-the art MOEA for CSOPs, but is also very competitive with the EAs in IEEE CEC 2006 Competition and is ranked third. Furthermore, HECO-PDE is ranked first on all dimensions when compared with the state-of-art single-objective EAs from the IEEE CEC 2017/2018 competition and another recent MOEA (DeCODE) for constrained optimization. This study also reveals that decomposition-based MOEAs, such as HECO-PDE and HECO-DE, are competitive with best single-objective and multi-objective EAs in constrained optimization, but MOEAs based on non-dominance, such as PMODE and CMODE, may not perform so well on the IEEE CEC 2017/2018 benchmark functions. For the future work, PCA-projection can be applied to other EAs.

Table 7.1 Function error values achieved when FES $= 5 \times 10^3$, FES $= 5 \times 10^4$, and FES $= 5 \times 10^5$ for test functions g01-g12

FES		g01	g02	g03	g04
5×10^3	Best	3.7476E+00 (0)	3.9521E-01 (0)	8.1970E-01 (0)	8.0248E+01 (0)
	Median	7.1483E+00 (0)	4.6885E-01 (0)	9.9764E-01 (0)	1.3409E+02 (0)
	Worst	8.3093E+00 (0)	5.8431E-01 (0)	1.0004E+00 (1)	2.1045E+02 (0)
	Mean	6.5899E+00	4.7402E-01	9.6788E-01	1.4122E+02
	Std	1.4271E+00	5.6480E-02	6.0624E-02	3.6330E+01
5×10^4	Best	8.7244E-02 (0)	1.4509E-01 (0)	2.6009E-05 (0)	5.3865E-03 (0)
	Median	2.1775E-01 (0)	2.4452E-01 (0)	3.6454E-04 (0)	1.9360E-02 (0)
	Worst	6.5961E-01 (0)	2.9740E-01 (0)	2.8050E-03 (0)	6.3210E-02 (0)
	Mean	2.5840E-01	2.4635E-01	4.3464E-04	2.3514E-02
	Std	1.3474E-01	3.2326E-02	5.4310E-04	1.5553E-02
5×10^5	Best	0.0000E+00 (0)	1.4432E-15 (0)	-2.8865E-15 (0)	-3.6379E-12 (0)
	Median	4.6185E-14 (0)	1.6653E-15 (0)	-2.6645E-15 (0)	-3.6379E-12 (0)
	Worst	1.8172E-12 (0)	8.7220E-03 (0)	-2.6645E-15 (0)	-3.6379E-12 (0)
	Mean	1.7209E-13	6.7092E-04	-2.7622E-15	-3.6379E-12
	Std	3.7931E-13	2.3701E-03	1.1249E-16	0.0000E+00
FES		g05	g06	g07	g08
5×10^3	Best	1.3619E+01 (0)	9.7204E+00 (0)	4.1846E+01 (0)	7.7709E-06 (0)
	Median	-7.0744E+00 (2)	3.6650E+01 (0)	6.6091E+01 (0)	2.1193E-04 (0)
	Worst	9.6255E+01 (3)	7.5112E+01 (0)	1.1750E+02 (0)	1.0131E-03 (0)
	Mean	5.1398E+01	3.8587E+01	7.0290E+01	2.9866E-04
	Std	1.1228E+02	1.7128E+01	2.2190E+01	2.8676E-04
5×10^4	Best	6.5827E-08 (0)	1.7270E-07 (0)	1.1916E-01 (0)	1.5668E-14 (0)
	Median	2.8887E-07 (0)	1.9594E-06 (0)	1.7154E-01 (0)	1.0883E-08 (0)
	Worst	1.0345E-06 (0)	1.3193E-05 (0)	3.2815E-01 (0)	6.2483E-07 (0)
	Mean	3.4680E-07	2.7423E-06	1.8937E-01	7.4641E-08
	Std	2.0430E-07	2.6518E-06	4.7690E-02	1.3977E-07
5×10^5	Best	-1.8189E-12 (0)	-1.6370E-11 (0)	-2.3803E-13 (0)	2.7755E-17 (0)
	Median	-1.8189E-12 (0)	-1.6370E-11 (0)	-2.2737E-13 (0)	4.1633E-17 (0)
	Worst	-1.8189E-12 (0)	-1.6370E-11 (0)	-2.1671E-13 (0)	4.1633E-17 (0)
	Mean	-1.8189E-12	-1.6370E-11	-2.2851E-13	4.1078E-17
	Std	0.0000E+00	0.0000E+00	4.6683E-15	2.7755E-18
FES		g09	g10	g11	g12
5×10^3	Best	2.3934E+01 (0)	4.1409E+03 (0)	2.8214E-05 (0)	2.1640E-05 (0)
	Median	4.9688E+01 (0)	5.9270E+03 (0	3.3153E-04 (0)	9.2278E-05 (0)
	Worst	8.6439E+01 (0)	1.1352E+04 (0)	2.6914E-03 (1)	3.9859E-04 (0)
	Mean	5.2651E+01	6.3646E+03	3.8302E-03	1.2505E-04
	Std	1.7102E+01	1.7857E+03	1.2484E-02	9.1642E-05
5×10^4	Best	1.8563E-04 (0)	7.2400E+00 (0)	9.4873E-11 (0)	0.0000E+00 (0)
	Median	7.3393E-04 (0)	1.1578E+01 (0)	4.4319E-10 (0)	0.0000E+00 (0)
	Worst	2.5274E-03 (0)	2.1756E+01 (0)	3.2906E-09 (0)	0.0000E+00 (0)
	Mean	8.0712E-03	1.2072E+01	7.9941E-10	0.0000E+00
	Std	5.5073E-04	3.2711E+00	8.2203E-10	0.0000E+00
5×10^5	Best	-2.2737E-13 (0)	-7.2759E-12 (0)	0.0000E+00 (0)	0.0000E+00 (0)
	Median	-2.2737E-13 (0)	-7.2759E-12 (0)	0.0000E+00 (0)	0.0000E+00 (0)
	Worst	-1.1368E-13 (0)	-7.2759E-12 (0)	0.0000E+00 (0)	0.0000E+00 (0)
	Mean	-2.0463E-13	-7.2759E-12	0.0000E+00	0.0000E+00
	Std	4.6412E-14	0.0000E+00	0.0000E+00	0.0000E+00

Table 7.2 Function error values achieved when FES $= 5 \times 10^3$, FES $= 5 \times 10^4$, and FES $= 5 \times 10^5$ for test functions g13-g24

FES		g13	g14	g15	g16
5×10^3	Best	9.2923E-01 (0)	-2.0579E+02 (3)	1.1259E-02 (0)	5.0879E-02 (0)
	Median	7.3286E-01 (2)	-1.2254E+02 (3)	1.1329E-01 (1)	9.8467E-02 (0)
	Worst	8.2007E-01 (3)	-3.7664E+01 (3)	7.5665E-01 (2)	2.1830E-01 (0)
	Mean	6.6186E-01	-1.2076E+02	4.7561E-01	1.0361E-01
	Std	3.2237E-01	3.7009E+01	5.5578E-01	3.4454E-02
5×10^4	Best	7.1483E-09 (0)	1.2165E-02 (0)	4.1154E-11 (0)	7.9541E-07 (0)
	Median	3.8363E-08 (0)	5.7221E-02 (0)	2.0634E-10 (0)	1.3600E-06 (0)
	Worst	3.6771E-07 (0)	3.0697E-01 (0)	1.4682E-09 (0)	3.2443E-06 (0)
	Mean	7.2797E-08	8.8248E-02	3.3435E-10	1.6191E-06
	Std	9.0395E-08	7.4330E-02	3.4142E-10	6.9377E-07
5×10^5	Best	-2.4286E-16 (0)	1.4210E-14 (0)	-1.1368E-13 (0)	3.7747E-15 (0)
	Median	-2.2204E-16 (0)	1.4210E-14 (0)	-1.1368E-13 (0)	3.7747E-15 (0)
	Worst	-1.9428E-16 (0)	2.1316E-14 (0)	-1.1368E-13 (0)	3.7747E-15 (0)
	Mean	-2.1954E-16	1.4779E-14	-1.1368E-13	3.7747E-15
	Std	1.0385E-17	1.9674E-15	0.0000E+00	0.0000E+00
FES		g17	g18	g19	g20
5×10^3	Best	2.1463E+02 (0)	6.7675E-01 (0)	1.2932E+02 (0)	1.2534E+01 (14)
	Median	1.0605E+02 (2)	8.7304E-01 (2)	3.0020E+02 (0)	1.0197E+01 (16)
	Worst	9.7101E+02 (3)	1.6762E-01 (5)	4.0959E+02 (0)	9.3376E+00 (19)
	Mean	1.2074E+02	7.5218E-01	2.8502E+02	1.0683E+01
	Std	1.3696E+02	1.8435E-01	7.4689E+01	1.8991E+00
5×10^4	Best	1.0381E-03 (0)	1.5288E-03 (0)	2.7929E+00 (0)	8.6285E-01 (14)
	Median	3.5125E-03 (0)	3.5573E-03 (0)	4.9162E+00 (0)	1.7833E+00 (16)
	Worst	2.8967E+00 (0)	6.2765E-03 (0)	1.0079E+01 (0)	5.4970E-01 (19)
	Mean	6.4163E-01	3.6941E-03	5.3814E+00	8.6565E-01
	Std	9.4554E-01	1.3026E-03	1.7217E+00	4.1305E-01
5×10^5	Best	-1.8189E-12 (0)	2.2204E-16 (0)	5.7661E-10 (0)	7.9138E-02 (10)
	Median	-1.8189E-12 (0)	2.2204E-16 (0)	3.2166E-09 (0)	8.3353E-02 (15)
	Worst	-1.8189E-12 (0)	2.2204E-16 (0)	1.3770E-09 (0)	7.3211E-02 (17)
	Mean	-1.8189E-12	2.2204E-16	3.9057E-09	8.4100E-02
	Std	8.2871E-25	0.0000E+00	3.0780E-09	2.5118E-02
FES		g21	g22	g23	g24
5×10^3	Best	-1.4412E+01 (1)	6.4219E+03 (4)	-4.6027E+02 (1)	1.7725E-03 (0)
	Median	6.0401E+02 (2)	4.1341E+03 (7)	-1.6899E+02 (3)	7.4986E-03 (0)
	Worst	1.6334E+02 (2)	2.0828E+03 (13)	-3.1643E+02 (5)	1.6723E-02 (0)
	Mean	1.7658E+02	6.6432E+03	-2.1998E+02	7.8850E-03
	Std	1.7532E+02	5.5802E+03	4.1690E+02	3.9429E-03
5×10^4	Best	1.4102E-02 (0)	-2.3478E+02 (6)	9.2163E+00 (0)	7.8120E-09 (0)
	Median	4.4718E-02 (0)	-1.9485E+02 (9)	2.1680E+01 (0)	1.1290E-07 (0)
	Worst	1.3106E+02 (0)	-2.2276E+02 (13)	4.9589E+01 (0)	5.9857E-07 (0)
	Mean	1.3327E+01	-2.2896E+02	2.3894E+01	1.3876E-07
	Std	3.7031E+01	1.1599E+01	1.0062E+01	1.4024E-07
5×10^5	Best	-3.0561E-10 (0)	-2.3643E+02 (8)	-5.6843E-13 (0)	3.2862E-14 (0)
	Median	-2.6631E-10 (0)	-2.3643E+02 (11)	-4.5474E-13 (0)	3.2862E-14 (0)
	Worst	1.3097E+02 (0)	-2.3414E+02 (14)	1.1368E-13 (0)	3.2862E-14 (0)
	Mean	5.2391E+00	-8.0649E+01	-3.4560E-13	3.2862E-14
	Std	2.6195E+01	6.0696E+02	2.0620E-13	0.0000E+00

Table 7.3 Number of FES to achieve the success condition, success rate, feasible rate, and success performance

Prob.	Best	Median	Worst	Mean	Std.	Feasible Rate	Success Rate	Success Performance
g01	134184	165520	224488	166889	19031.01	100%	100%	166889
g02	141048	179808	216256	179421	20577.24	100%	92%	179421
g03	44104	53376	66680	53123	5275.58	100%	100%	53123
g04	70328	76392	84688	76745	3366.5	100%	100%	76745
g05	23616	26560	28696	26497	1266.67	100%	100%	26497
g06	29272	38088	42992	37602	2938.38	100%	100%	37602
g07	117504	123496	135192	123904	3918.02	100%	100%	12394
g08	3008	5920	9064	5970	1610.98	100%	100%	5970
g09	51032	58264	64408	57842	3854.65	100%	100%	57842
g10	133384	137504	148248	138412	3881.14	100%	100%	138412
g11	2192	5888	8248	5655	1340.43	100%	100%	5655
g12	1240	4576	7784	4643	1926.71	100%	100%	4643
g13	22096	28048	40112	29042	4696.05	100%	100%	29042
g14	82040	92016	100432	91817	5069.44	100%	100%	91817
g15	10288	11960	12808	11839	585.51	100%	100%	11839
g16	26512	30760	33176	30615	1829.15	100%	100%	30615
g17	63976	71024	161608	92195	32845.84	100%	100%	92195
g18	74048	82024	95560	83586	5588.53	100%	100%	83586
g19	243360	262936	292600	264423	12521.55	100%	100%	264423
g20	-	-	-	-	-	0%	0%	-
g21	88040	90052	237656	101595	42137.78	100%	96%	101595
g22	-	-	-	-	-	0%	0%	-
g23	171800	199824	231864	199496	19517.02	100%	100%	199496
g24	14400	24736	29408	23728	4546.14	100%	100%	23728

Table 7.4 Comparison of PMODE with respect to CMODE on $\overline{f^r(x)}$ and success performance. The winner values are shown in bold.

Prob.	$\overline{f^r(x)}$		Success Performance		
	PMODE	CMODE	PMODE	CMODE	p-value
g01	1.7209E-13	**0.0000E+00**	166889	**121077**	1.1739E-11
g02	6.7092E-04	**2.0387E-08**	**179421**	189820	1.8520E-02
g03	**-2.7622E-15**	1.1665E-09	**53123**	75085	7.2272E-12
g04	**-3.6379E-12**	7.6398E-11	76745	**72748**	3.9811E-06
g05	-1.8189E-12	-1.8190E-12	**26497**	28873	3.1053E-39
g06	**-1.6370E-11**	3.3651E-11	37602	**35464**	1.3063E-03
g07	**-2.2851E-13**	7.9793E-11	**12394**	155968	1.0295E-23
g08	**4.1078E-17**	8.1964E-11	5970	5885	7.9285E-01
g09	-2.0463E-13	**-9.8198E-11**	**57842**	71122	5.1561E-15
g10	**-7.2759E-12**	6.2827E-11	**138412**	183255	2.8388E-27
g11	0.0000E+00	0.0000E+00	5655	6023	1.8332E-01
g12	0.0000E+00	0.0000E+00	4643	5009	3.5277E-01
g13	**-2.1954E-16**	4.1897E-11	29042	30689	9.2308E-02
g14	**1.4779E-14**	8.5159E-12	**91817**	107976	2.8797E-14
g15	**-1.1368E-13**	6.0822E-11	**11839**	12855	7.3365E-09
g16	**3.7747E-15**	6.5213E-11	30615	**29332**	1.8059E-03
g17	**-1.8189E-12**	1.8189E-12	**92195**	139746	1.7682E-07
g18	**2.2204E-16**	1.5561E-11	**83586**	105020	4.6431E-16
g19	3.9057E-09	**2.4644E-10**	264423	**251676**	3.2721E-05
g21	**5.2391E+00**	2.6195E+01	**101595**	128758	4.4012E-03
g23	**-3.4560E-13**	4.4772E-11	**199496**	244612	2.7069E-11
g24	**3.2862E-14**	4.6735E-12	23728	**21820**	4.6499E-02
Number of winners	**15**	4	**12**	6	-

Table 7.5 Comparison of PMODE and CMODE on feasible rate and success rate

Prob.	Feasible Rate		Success Rate	
	PMODE	CMODE	PMODE	CMODE
g01	100%	100%	100%	100%
g02	100%	100%	92%	**100%**
g03	100%	100%	100%	100%
g04	100%	100%	100%	100%
g05	100%	100%	100%	100%
g06	100%	100%	100%	100%
g07	100%	100%	100%	100%
g08	100%	100%	100%	100%
g09	100%	100%	100%	100%
g10	100%	100%	100%	100%
g11	100%	100%	100%	100%
g12	100%	100%	100%	100%
g13	100%	100%	100%	100%
g14	100%	100%	100%	100%
g15	100%	100%	100%	100%
g16	100%	100%	100%	100%
g17	100%	100%	100%	100%
g18	100%	100%	100%	100%
g19	100%	100%	100%	100%
g21	100%	100%	**96%**	80%
g22	0%	100%	0%	0%
g23	100%	0%	100%	100%
g24	100%	100%	100%	100%
Mean	**95.65%**	**95.65%**	**95.13%**	94.78

Table 7.6 Comparison of PMODE, CMODE and all EAs in CEC 2006 Competition on feasible rate and success rate

Algorithms	Feasible Rate	Success Rate
DE	95.65%	78.09%
DMS-PSO	**100%**	90.61%
ε_DE	**100%**	**95.65%**
GDE	92.00%	77.39%
jDE-2	95.65%	80.00%
MDE	95.65%	87.65%
MPDE	94.96%	87.65%
PCX	95.65%	94.09%
PESO+	95.48%	67.83%
SaDE	**100%**	87.13%
CMODE	95.65%	94.78
PMODE	95.65%	95.13%

Table 7.7 Comparison of PMODE, CMODE and all EAs in CEC 2006 Competition on success performance FEs divided by FEs of the best algorithm. Note: g20 (with a mark EX) is excluded in the competition. g22 (with a mark -) no values were available in the competition data.

FEs_{best} / EAs	g01	g02	g03	g04	g05	g06	g07	g08
	25115	96222	24861	15281	21306	5202	26578	918
DE	1.3304	1.4017	-	1.0461	5.0256	1.3731	3.5290	1.1830
DMS-PSO	1.3272	1.8201	1.0289	1.6625	1.3790	5.3126	**1.0000**	4.4928
ε_DE	2.3615	1.5571	3.5963	1.7156	4.5729	1.4189	2.7957	1.2407
GDE	1.6133	1.5543	143.8877	**1.0000**	9.0821	1.2501	4.6654	1.6002
jDE-2	2.0062	1.5163	-	2.6653	20.9724	5.6686	4.8064	3.5251
MDE	3.0011	**1.0000**	1.8096	2.7198	**1.0000**	**1.0000**	7.3069	**1.0000**
MPDE	1.7292	3.1694	**1.0000**	1.3666	10.1600	2.0327	2.1597	1.6498
PCX	2.1981	1.3292	1.4053	2.0279	4.4478	6.5015	4.4067	3.0784
PESO+	4.0427	4.2905	18.1268	5.2271	21.2267	10.8627	13.8191	6.6710
SaDE	**1.0000**	1.9107	12.0254	1.6430	3.4263	2.4118	1.0398	1.4412
CMODE	4.8209	1.9727	3.0201	4.7606	1.3551	6.8173	5.8683	6.4106
PMODE	6.6449	1.8646	2.1368	5.0222	1.2436	7.2283	4.6619	6.5032

FEs_{best} / EAs	g09	g10	g11	g12	g13	g14	g15	g16
	16152	25520	3000	1308	21732	25220	10458	8730
DE	1.5976	4.6715	4.4600	3.9021	1.5976	2.7052	5.5429	1.3278
DMS-PSO	1.8237	**1.0000**	4.8750	4.1356	1.8237	**1.0000**	2.7634	6.1260
ε_DE	1.4315	4.1236	5.4733	3.1529	1.4315	4.4980	8.0528	1.4875
GDE	1.8716	3.2368	2.8200	2.4075	1.8716	9.1247	7.1605	1.5148
jDE-2	3.4001	5.7269	17.9760	4.8593	3.4001	3.8797	23.0812	3.6306
MDE	**1.0000**	6.4326	**1.0000**	**1.0000**	**1.0000**	11.5639	**1.0000**	**1.0000**
MPDE	1.3029	1.9055	7.7854	3.2401	1.3029	1.6937	19.1408	1.4963
PCX	2.8806	3.4886	12.8960	6.8502	2.8806	2.3488	4.4880	3.4817
PESO+	6.0391	110.8383	150.0333	6.1835	6.0391	-	43.0388	5.6174
SaDE	1.3278	1.7307	8.3703	1.9694	1.3278	1.7843	2.5818	1.7123
CMODE	4.4032	7.1808	2.0076	3.8295	1.4121	4.2813	1.2292	3.3599
PMODE	3.5811	5.4236	1.8850	3.5496	1.3363	3.6406	1.1320	3.5068

FEs_{best} / EAs	g17	g18	g19	g21	g22	g23	g24	g20
	26364	28261	21830	38217	-	129550	1794	EX
DE	50.3891	2.8151	8.1186	4.2571	-	-	1.6856	EX
DMS-PSO	-	1.1741	**1.0000**	3.6722	-	1.6251	10.8004	EX
ε_DE	3.7498	2.0931	16.3239	3.5362	-	1.5497	1.6455	EX
GDE	81.4890	16.9874	10.5489	15.1615	-	8.2081	1.7051	EX
jDE-2	426.0602	3.6963	9.1548	3.3103	-	2.7592	5.6834	EX
MDE	**1.0000**	3.6617	-	2.9455	-	2.7821	**1.0000**	EX
MPDE	27.7422	1.5585	5.4180	5.4703	-	1.6261	2.4204	EX
PCX	5.1627	2.4779	5.9403	**1.0000**	-	1.2900	6.4916	EX
PESO+	-	8.2431	-	-	-	-	11.1371	EX
SaDE	474.1314	**1.0000**	2.3896	4.2958	-	**1.0000**	2.5775	EX
CMODE	5.3006	3.7160	11.5289	3.3691	-	1.8881	12.1627	EX
PMODE	3.4970	2.9576	12.1128	2.6583	-	1.5399	13.2263	EX

Table 7.8 The ranking of PMODE, CMODE and all ten algorithms in CEC 2006 Competition on $\overline{f^r(x)}$, feasible rate, success rate, success performance and the final rank

Algorithms	$\overline{f^r(x)}$	Feasible Rate	Success Rate	Success Performance	Final Rank
DE	9	4	10	6	8
DMS-PSO	4	**1**	5	**3**	**2**
ε_DE	**2**	**1**	**1**	4	**1**
GDE	12	12	11	10	11
jDE-2	10	4	9	11	10
MDE	7	4	6	**1**	4
MPDE	5	11	7	5	8
PCX	**3**	4	4	7	4
PESO+	11	10	12	12	11
SaDE	9	**1**	8	**1**	6
CMODE	6	4	**3**	9	7
PMODE	**1**	4	**2**	8	**3**

Table 7.9 Total ranks of CMODE, PMODE, HECO-DE, HECO-PDE, DeCODE and seven EAs in CEC2018 competition

Algorithm/Dimension	10D	30D	50D	100D	Total
CAL_LSAHDE(2017)	418	398	428	435	1679
LSHADE44+IDE(2017)	299	365	385	353	1402
LSAHDE44(2017)	319	313	308	310	1250
UDE(2017)	330	344	345	390	1409
MA_ES(2018)	266	240	243	246	995
IUDE(2018)	193	226	224	292	935
LSAHDE_IEpsilon(2018)	199	246	292	333	1070
DeCODE	237	276	277	296	1086
CMODE	443	618	628	631	2320
PMODE	425	610	630	626	2291
HECO-DE	173	164	177	183	697
HECO-PDE	155	138	152	177	622

Table 7.10 Ranks of HECO-PDE and Other EAs based on mean values on the 28 functions of 10 dimensions

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	10	10	12	6	7	12	11	12	7	1	12	7	8	10	11	12	1	8	12	12	9	10	10	12	10	1	235
LSHADE44+IDE(2017)	1	1	8	7	1	12	6	1	1	2	1	3	1	9	5	8	6	10	4	4	2	7	8	8	8	6	11	10	151
LSAHDE44(2017)	1	1	9	5	1	11	5	1	10	2	2	10	1	8	9	9	7	9	2	1	4	9	7	7	9	7	9	11	167
UDE(2017)	1	1	7	8	11	7	4	1	8	2	10	1	11	6	6	7	8	8	9	10	8	11	6	4	7	9	8	7	186
MA_ES(2018)	1	1	1	9	1	5	2	1	1	2	5	12	7	10	12	1	12	1	12	9	9	8	10	6	1	8	2	9	158
IUDE(2018)	1	1	5	3	1	1	8	1	1	2	6	7	1	3	4	1	5	7	4	7	3	10	1	4	1	4	7	6	105
LSAHDE_IEpsilon(2018)	1	1	6	4	1	8	3	1	1	2	3	5	1	2	7	1	4	2	3	6	5	1	3	9	1	5	1	12	99
DeCODE	1	1	1	6	1	1	1	11	8	1	9	4	10	1	1	6	3	11	10	5	1	1	5	1	6	3	12	4	125
CMODE	1	1	11	11	1	10	12	1	12	10	12	6	7	11	11	11	9	6	11	12	11	1	11	11	11	11	6	8	236
PMODE	1	1	12	12	1	9	10	1	1	10	11	11	7	12	10	12	10	5	4	11	10	1	12	12	12	10	5	5	218
HECO-DE	1	1	1	1	1	1	11	1	1	2	8	9	1	5	2	1	1	4	4	3	7	1	4	3	1	1	3	3	82
HECO-PDE	1	1	1	1	1	1	9	1	1	2	4	8	1	3	2	1	1	3	4	2	6	1	2	2	1	2	4	2	68

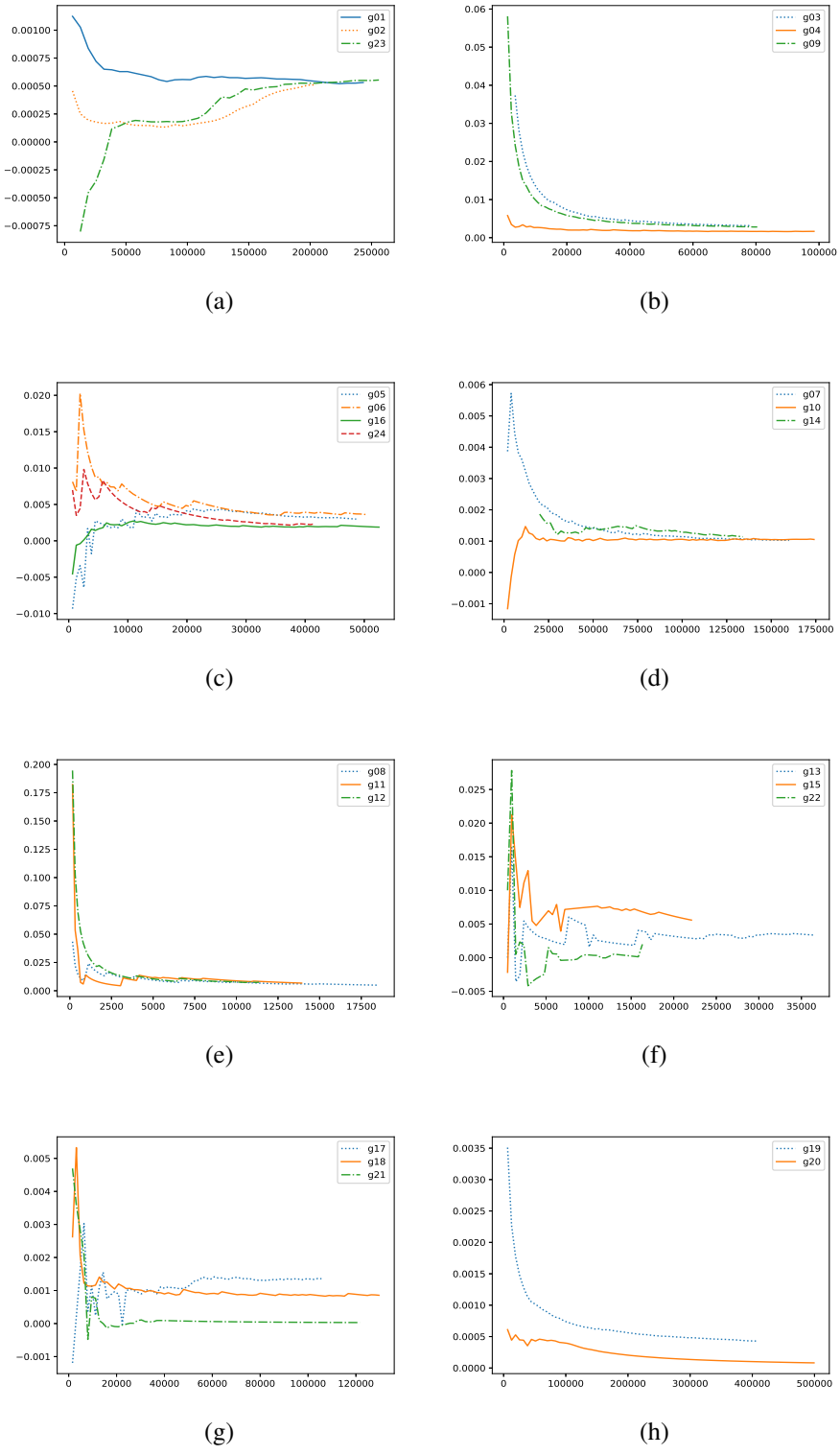


Fig. 7.10 Convergent speed graphs for g01-g24

Table 7.11 Ranks of HECO-PDE and Other EAs based on median solution on the 28 functions of 10 dimensions

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	10	10	1	8	5	12	12	1	4	3	1	8	8	10	10	11	1	8	4	1	9	12	10	10	11	1	183
LSHADE44+IDE(2017)	1	1	9	5	1	10	6	1	1	2	2	4	1	10	7	9	6	10	4	4	5	1	10	9	9	8	10	2	148
LSAHDE44(2017)	1	1	8	7	1	9	7	1	1	2	4	12	1	9	10	8	7	8	2	1	1	1	8	8	8	6	9	11	152
UDE(2017)	1	1	6	8	1	7	4	1	10	2	10	1	1	1	5	7	9	9	11	10	1	1	1	5	7	9	8	7	144
MA_ES(2018)	1	1	1	9	1	1	2	1	1	2	2	5	1	3	12	1	8	1	10	9	6	1	3	7	1	7	1	10	108
IUDE(2018)	1	1	1	1	1	1	8	1	1	2	1	8	1	3	5	1	1	6	4	7	7	1	4	5	1	1	7	7	88
LSAHDE_IEpsilon(2018)	1	1	7	4	1	1	3	1	1	2	6	6	1	3	4	1	5	2	3	6	11	1	6	3	1	5	2	12	100
DeCODE	1	1	1	6	1	1	1	1	10	2	9	1	1	1	1	6	4	12	12	5	1	1	1	1	6	4	12	9	112
CMODE	1	1	11	12	1	11	11	1	1	12	12	7	1	11	11	11	11	5	4	12	8	1	11	10	11	11	6	2	207
PMODE	1	1	12	11	1	12	12	1	1	2	11	9	1	12	9	12	12	7	4	11	10	1	12	11	12	12	5	2	207
HECO-DE	1	1	1	1	1	1	9	1	1	2	8	11	1	6	2	1	2	4	4	3	9	1	7	4	1	3	3	2	91
HECO-PDE	1	1	1	1	1	1	10	1	1	2	7	10	1	6	2	1	2	3	4	2	12	1	5	2	1	2	4	2	87

Table 7.12 Ranks of HECO-PDE and Other EAs based on mean values on the 28 functions of 30 dimensions

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	10	10	10	6	8	1	10	10	4	10	10	10	11	12	10	11	1	3	10	10	10	12	10	10	10	1	222
LSHADE44+IDE(2017)	1	1	9	6	1	11	4	10	1	9	3	7	7	9	7	8	7	9	4	5	8	7	9	9	8	7	11	9	187
LSAHDE44(2017)	1	1	8	4	1	10	5	2	1	1	2	5	6	8	8	9	6	6	2	1	7	8	7	8	9	6	9	10	151
UDE(2017)	1	1	5	9	7	4	2	9	7	8	8	8	8	6	6	7	8	7	9	9	4	6	5	6	6	8	8	6	178
MA_ES(2018)	1	1	1	8	1	3	1	2	1	1	1	9	1	7	12	1	9	2	10	10	9	1	6	7	1	9	1	4	120
IUDE(2018)	1	1	6	5	1	7	7	2	7	1	5	2	5	1	5	4	3	5	7	8	6	5	4	3	5	2	5	7	120
LSAHDE_IEpsilon(2018)	1	1	7	1	1	9	3	2	1	1	9	6	9	5	3	6	5	1	3	2	3	9	2	5	7	5	2	12	121
DeCODE	1	1	1	7	9	5	6	8	9	1	12	1	1	2	4	5	4	12	8	4	5	4	8	4	4	1	12	5	144
CMODE	12	12	12	11	12	8	11	12	12	12	10	12	11	11	10	10	12	10	11	11	12	12	11	10	11	11	7	11	307
PMODE	11	11	11	12	11	12	12	11	11	11	11	11	12	12	9	11	11	8	12	12	11	11	12	11	12	12	6	8	305
HECO-DE	1	1	1	2	8	1	9	2	1	1	7	4	1	4	1	1	1	4	4	6	2	3	3	2	1	4	3	3	81
HECO-PDE	1	1	1	2	1	1	10	2	1	1	6	3	1	3	1	1	2	3	4	7	1	2	1	1	1	3	4	2	67

Table 7.13 Ranks of HECO-PDE and Other EAs based on median solution on the 28 functions of 30 dimensions

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	10	10	1	6	8	1	1	1	4	8	10	7	9	12	10	9	1	3	3	10	6	12	12	10	9	1	176
LSHADE44+IDE(2017)	1	1	9	2	1	10	6	10	2	10	3	2	1	10	7	8	8	11	4	5	9	7	9	9	8	5	11	9	178
LSAHDE44(2017)	1	1	8	5	1	9	5	2	2	2	2	7	8	9	8	9	7	5	2	1	8	8	8	7	9	8	10	10	162
UDE(2017)	1	1	5	9	1	4	2	9	8	9	8	9	7	6	6	5	9	7	7	9	3	6	1	6	6	9	7	6	166
MA_ES(2018)	1	1	1	8	1	3	1	2	2	2	1	10	1	8	12	1	6	1	10	10	10	1	7	8	1	6	1	4	120
IUDE(2018)	1	1	6	6	1	8	4	2	8	2	6	4	1	1	3	4	3	6	7	8	3	1	1	1	5	1	5	7	106
LSAHDE_IEpsilon(2018)	1	1	7	1	1	7	3	2	2	2	9	5	9	3	5	7	5	2	3	2	7	9	4	5	7	2	2	12	125
DeCODE	1	1	1	6	1	5	7	2	8	2	12	1	1	1	3	5	4	12	9	4	3	1	10	4	4	7	12	5	132
CMODE	12	11	12	11	12	11	11	12	12	11	11	12	11	11	11	10	12	10	11	11	12	12	11	10	11	11	8	11	311
PMODE	11	12	11	12	11	12	12	11	11	12	10	11	12	12	10	11	11	8	12	12	11	11	12	11	10	12	6	8	305
HECO-DE	1	1	1	2	1	1	9	2	2	2	7	6	1	4	1	1	2	4	4	6	2	5	5	3	1	4	3	2	83
HECO-PDE	1	1	1	2	1	1	10	2	2	2	5	3	1	4	1	1	1	3	4	7	1	1	3	2	1	3	4	3	71

Table 7.14 Ranks of HECO-PDE and Other EAs based on mean values on the 28 functions of 50 dimensions

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	10	10	9	4	8	10	10	8	4	7	10	10	10	10	10	8	1	12	6	9	10	10	10	10	7	1	234
LSHADE44+IDE(2017)	9	1	9	5	1	10	5	9	7	10	1	4	7	9	7	8	8	7	4	4	9	5	8	9	9	8	10	9	192
LSAHDE44(2017)	1	1	8	3	1	9	4	1	3	1	2	9	6	8	8	9	7	6	3	1	10	6	6	8	8	7	6	8	150
UDE(2017)	1	1	5	9	10	3	2	8	1	9	5	6	9	6	6	6	9	11	9	7	3	8	5	6	6	9	11	5	176
MA_ES(2018)	1	1	1	8	1	1	1	2	8	1	3	10	8	7	9	1	6	1	10	9	8	4	7	7	1	5	1	4	126
IUDE(2018)	1	1	6	7	1	11	7	4	1	1	7	3	5	1	3	5	3	5	8	8	4	3	4	3	5	1	5	7	120
LSAHDE_IEpsilon(2018)	1	1	7	4	8	8	3	7	6	7	8	8	4	4	5	7	5	2	7	2	5	10	2	5	7	6	2	10	151
DeCODE	1	1	1	6	1	2	6	3	9	6	12	5	1	2	4	4	4	12	2	3	7	2	9	4	4	4	12	6	133
CMODE	12	12	11	11	11	12	12	12	11	12	11	11	12	12	12	11	12	9	12	10	11	11	11	11	11	11	9	11	314
PMODE	11	11	12	12	12	7	11	11	12	11	10	12	11	11	11	12	11	10	11	11	12	12	12	12	12	12	8	12	312
HECO-DE	1	1	1	1	1	6	9	6	3	5	9	1	3	5	1	1	2	3	4	5	1	7	3	2	1	3	3	2	90
HECO-PDE	1	1	1	1	1	5	10	5	3	4	6	2	2	3	1	1	1	4	4	6	2	1	1	1	1	2	4	3	77

Table 7.15 Ranks of HECO-PDE and Other EAs based on median solution on the 28 functions of 50 dimensions

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	10	10	1	6	8	10	7	8	4	7	10	6	12	10	10	9	1	3	7	9	6	10	10	10	7	1	194
LSHADE44+IDE(2017)	1	1	9	2	1	9	5	9	9	10	3	3	8	10	7	9	8	8	5	5	9	7	9	9	9	8	11	9	193
LSAHDE44(2017)	1	1	8	5	1	8	4	1	3	1	1	10	7	9	9	8	7	6	4	1	10	8	8	8	8	7	6	8	158
UDE(2017)	1	1	5	9	10	5	2	8	1	9	5	5	9	7	6	6	9	7	10	8	1	6	2	7	6	9	10	5	169
MA_ES(2018)	1	1	1	8	1	3	1	2	8	1	2	9	1	8	8	1	5	1	9	10	8	5	7	6	1	4	1	4	117
IUDE(2018)	1	1	6	7	1	10	7	3	1	1	6	4	1	3	1	5	1	5	2	9	4	4	2	1	4	2	5	7	104
LSAHDE_IEpsilon(2018)	1	1	7	1	1	7	3	7	3	7	8	8	6	1	5	7	6	2	8	2	5	10	5	5	7	6	2	10	141
DeCODE	1	1	1	6	1	4	6	6	10	6	12	5	1	3	4	4	4	12	3	4	6	3	10	4	4	5	12	6	144
CMODE	12	12	12	11	11	12	12	12	11	11	11	12	12	11	10	11	11	10	12	11	11	11	12	11	11	11	8	12	314
PMODE	11	11	11	12	12	11	11	11	12	12	10	11	11	12	11	12	12	11	11	12	12	12	11	12	12	12	9	11	318
HECO-DE	1	1	1	2	1	2	9	4	3	4	9	2	1	5	2	1	3	4	5	6	3	2	4	3	1	3	3	2	87
HECO-PDE	1	1	1	2	1	1	10	5	3	4	7	1	1	2	2	1	2	3	5	7	2	1	1	2	1	1	4	3	75

Table 7.16 Ranks based on mean values on the 28 functions of 100 dimensions

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)	9	9	11	10	6	3	9	10	8	9	5	5	9	10	10	10	10	11	1	12	10	7	10	10	10	10	9	1	234
LSHADE44+IDE(2017)	10	10	8	4	4	11	4	6	2	7	1	1	6	9	5	8	8	7	3	4	5	6	6	7	9	8	10	7	176
LSAHDE44(2017)	1	1	7	3	3	10	5	1	1	1	4	10	7	8	8	9	6	6	2	3	9	5	8	6	8	6	6	9	153
UDE(2017)	8	8	5	9	10	4	2	9	10	8	6	2	10	5	7	5	9	8	9	8	4	10	2	8	5	9	11	5	196
MA_ES(2018)	1	1	1	7	9	1	1	2	9	4	2	8	4	7	9	1	5	1	10	9	8	2	7	4	1	5	3	4	126
IUDE(2018)	1	1	9	8	5	12	7	3	6	5	3	3	8	1	4	6	3	5	8	7	2	8	5	5	6	3	5	8	147
LSAHDE_IEpsilon(2018)	7	7	6	5	8	9	3	7	7	6	8	4	5	4	6	7	7	2	6	2	3	9	4	9	7	7	4	10	169
DeCODE	1	1	1	6	7	2	6	8	3	10	10	6	3	6	3	4	4	12	7	1	1	1	9	3	4	4	12	6	141
CMODE	12	12	10	12	12	8	12	11	12	12	11	11	12	11	12	12	11	10	12	11	12	12	11	11	12	12	8	11	315
PMODE	11	11	12	11	11	6	11	12	11	11	12	12	11	12	11	11	12	9	11	10	11	11	12	12	11	11	7	12	305
HECO-DE	1	1	1	1	1	7	8	4	3	2	9	7	1	3	1	1	1	3	3	5	7	3	3	2	3	2	2	2	87
HECO-PDE	1	1	1	2	1	5	10	5	3	3	7	9	2	2	1	1	2	4	3	6	6	4	1	1	2	1	1	3	88

Table 7.17 Ranks based on median solution on the 28 functions of 100 dimensions

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)	9	9	10	10	7	5	9	10	6	9	5	4	9	1	10	10	10	9	1	1	3	7	6	10	10	10	10	1	201
LSHADE44+IDE(2017)	10	10	9	1	3	9	5	6	2	8	4	2	6	10	6	8	7	6	3	5	7	5	7	8	9	7	6	8	177
LSAHDE44(2017)	1	1	8	4	4	8	6	1	1	1	2	10	7	9	9	9	5	7	2	4	8	4	9	9	8	4	7	9	157
UDE(2017)	8	8	5	9	10	4	2	9	8	10	6	1	10	6	8	5	9	8	9	9	5	10	3	6	5	8	8	5	194
MA_ES(2018)	1	1	1	7	9	2	1	2	10	5	1	6	4	8	4	1	4	1	10	10	6	3	8	4	1	5	1	4	120
IUDE(2018)	1	1	7	8	5	10	4	3	9	4	3	3	8	2	7	6	3	5	7	8	2	8	5	5	6	3	5	7	145
LSAHDE_IEpsilon(2018)	7	7	6	5	8	6	3	7	7	6	8	5	5	5	4	7	6	2	6	3	4	9	4	7	7	6	4	10	164
DeCODE	1	1	1	6	6	3	7	8	3	7	12	7	3	7	3	4	8	12	8	2	1	1	10	3	4	9	12	6	155
CMODE	12	12	11	12	11	11	12	11	12	12	10	11	11	11	11	12	11	10	11	11	12	12	11	11	11	12	11	11	316
PMODE	11	11	12	11	12	12	11	12	11	11	11	12	12	12	12	11	12	11	12	12	11	11	12	12	12	11	9	12	321
HECO-DE	1	1	1	3	1	7	8	4	3	2	9	9	2	4	1	1	1	3	3	6	10	2	2	2	3	2	3	2	96
HECO-PDE	1	1	1	1	1	1	10	5	3	3	7	8	1	3	1	1	2	4	3	7	9	6	1	1	2	1	2	3	89

Chapter 8

Conclusions and Future Work

8.1 Conclusions

In this thesis, we developed several different MOEAs for solving CSOPs with theoretical analysis and experimental studies.

In Chapter 2, we summarised exsiting multi-objective methods for constrained optimization into three schemes: schemes with two objectives, schemes with many objectives and schemes with other objective(s).

In Chapter 3, we introduced the problem definitions and evaluation criteria for the IEEE CEC2006 and IEEE CEC2017 competition on single-objective constrained real-parameter optimization. The methodology of comparing different algorithms are also summarized.

In Chapter 4 and 5, two different models for solving CSOPs were proposed. The first model is to construct other helper fitness functions from weighted sums of the normalised original objective and the normalised degree of constraint violation besides the standard model with two objectives. The new model is compared with the standard model used in CMODE. Experimental results show that CMODE with three fitness functions obtains remarkable better performance than that with the standard two fitness functions on most benchmark functions (12/13) from IEEE CEC2006 competition. However, as shown in Chapter 7 this model cannot maintain high performance in IEEE CEC2017/2018 benchmark functions. Thus, the other model converts a constrained optimization problem into a problem with helper and equivalent objectives (HECO). We theoretically analyse the computation time of HECO 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 objective HECO may shorten the time of crossing the "wide gap".

Then, in Chapter 6, a case study was conducted to validate our HECO framework. A new algorithm, called HECO-DE, is designed which employs helper and equivalent objectives

and reuses search operators from LSHADE44 [107]. Experimental results show that with the aid of helper and equivalent objectives, HECO-DE has outperformed LSHADE44, the winner of the CEC 2018 competition [133] and a latest decomposition-based MOEA, DeCODE [148]. This case study proves the efficiency of the helper and equivalent objective method for constrained optimization. Tchebycheff decomposition a multi-population version of HECO-DE, called HECO-DEm are proposed to enhance our results. HECO-DEm divides a population into three subpopulations. Different from HECO-DE, different weights are assigned to objective decomposition in each subproblems. This makes search directions in one subpopulation are different from another subpopulation. Unlike HECO-DE, HECO-DEm is suitable for parallel computation. It is expected that using multi-population can maintain population diversity and increase search directions, then it may improve the capacity of HECO-DE to handle complex fitness landscapes especially in a high dimensional search space. Experiment results show that its overall performance is the same as HECO-DE and much better than other eight algorithms under comparison. The multi-population version performs slightly better than HECO-DE in 50 and 100 dimensional functions. However, the Tchebycheff decomposition approach might not be as effective as weighted sum approach in HECO-DE according to the experiment study. These extensive experimental indicates that our HECO performs very well.

In Chapter 7, we presented two methods of studying valleys on a fitness landscape. The first method is based on the topological homeomorphism. It establishes a rigorous definition of a valley. A valley is regarded as a one-dimensional manifold. The second method takes a different viewpoint from statistics. It provides an algorithm of identifying the valley direction and location using principle component analysis. From an experimental observation, we find that given a valley landscape, the maximal variance direction in a population can be regarded as the valley direction. Based on this finding, a new search operator, called PCA-projection, is proposed, in which PCA is used to project points along the maximal variance direction. PCA-projection can be easily added into an existing MOEA through a mixed strategy. We design two MOEAs enhanced with PCA-projection, called HECO-PDE and PMODE, for evaluating the effectiveness of this new operator. Experimental results show that an EA enhanced with PCA-projection performs better than its corresponding opponent without this operator. The proposed PMODE not only has significantly improved the solution quality when compared with CMODE, an state-of-the art MOEA for CSOPs, but is also very competitive with the EAs in IEEE CEC 2006 Competition and is ranked third. Furthermore, HECO-PDE is ranked first on all dimensions when compared with the state-of-art single-objective EAs from the IEEE CEC 2017/2018 competition and another recent MOEA (DeCODE) for constrained optimization. This study also reveals that decomposition-based MOEAs, such as HECO-

PDE and HECO-DE, are competitive with best single-objective and multi-objective EAs in constrained optimization, but MOEAs based on non-dominance, such as PMODE and CMODE, may not perform so well on the IEEE CEC 2017/2018 benchmark functions.

8.2 Future Work

Further studies are required to understand and extend the methods proposed in the thesis.

In Chapters 4 and 5, we proposed two novel models for solving CSOPs. However, search operators are still from existing EAs, such as CMODE [151] and LSHADE44 [107] which are developed from DE. In the future, we may try different EAs, for example, PSO or CMA-ES with the HECO framework. PCA-projections in Chapter 7 could also be tested with other EAs. Further, we may develop our own search operators which are totally different from current EAs for solving CSOPs.

There are an extensive research of conducting into the three types of schemes explored in this Thesis. However, there are still several research issues that remain to be solved in each area. Some possible fields of future work for each area can be identified in the future work. We have explored multi-objective methods for constrained single-objective optimization problems is the area and analyzed in this thesis. Thus, one of the main challenges is the selection of the robust technique to be applied. Since they arose with the aim of avoiding the tuning of parameters in penalty-based schemes, this is a large drawback to its use. Thus, in our opinion, there should be an effort to apply these techniques using a common framework with the aim of better analyzing their performance. For instance, HECO framework in the thesis. A set of solvers might be picked and integrated with adaptive selection mechanisms as hyperheuristics. If successful, this might facilitate the solving of new problems where there is not much information on the fitness landscape. As we have shown, for some constrained optimization problems, some single- objective schemes are superior to multi-objective schemes. It would be very interesting to identify those properties that hamper optimization for multi-objective methods. In addition, exploring the properties of the best single-objective schemes to understand the differences might give some insight into possible areas to explore. For instance, many successful single-objective schemes incorporate the use of a local search. This area of research has also been explored in some multi-objective schemes, but the number of proposals is very scarce. In addition, in keeping with the idea of applying hyperheuristics, these might be used to combine single-objective and multi-objective methods.

Finally, it is important to note that in recent years, several advances have been made in the field of many-optimization. Since in some optimization problems a large number of constraints arise, the application of some of the latest advances in this field is very

promising. Considering the importance in constrained problems of producing some bias in the search so as to avoid the over-exploration of non-promising regions, the direct use of many-optimization is probably not helpful. However, some of the ideas explored in this area might be successfully adapted to constrained optimization.

The number of methods that consider diversity as an objective is more limited. In any case, several different schemes have been devised, and a large number of different optimization problems have been tackled. As was mentioned earlier, some of the schemes proposed have not been compared against some traditional diversity preservation techniques, such as fitness sharing or crowding. Thus, developing a comparison among the different proposals with some of the most recent published benchmark problems would be of great value. It is also important to note that some of the currently used schemes have limited their use to some specific areas. For instance, the multi-objective novelty-based approaches have only been used in the field of evolutionary robotics. Since they have obtained very promising results, it would be very interesting to test them, for example, in real-parameter optimization environments. In addition, some other diversity preservation techniques might inspire new innovations.

The current test problems are mainly from CEC2006 and CEC2017/2018 benchmark problems. However, CEC2006 benchmark functions have already been solved by various EAs. Additionally, CEC2017/2018 benchmark functions are still artificial real-parameter problems. Different from the first two, the brand new CEC2020 benchmark functions [134] are a set of 57 real-world optimization problems which have been difficult for solving because of their complex objective function with a substantial number of constraints. To validate effectiveness and strength of our proposed algorithms, EAs based on HECO and future algorithms will be tested on the new benchmark functions.

References

[1] Aguirre, A. H., Rionda, S. B., Coello Coello, C. A., Lizárraga, G. L., and Montes, E. M. (2004). Handling constraints using multiobjective optimization concepts. *International Journal for Numerical Methods in Engineering*, 59(15):1989–2017.

[2] Angantyr, A., Andersson, J., and Aidanpaa, J.-O. (2003). Constrained optimization based on a multiobjective evolutionary algorithm. In *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.*, volume 3, pages 1560–1567. IEEE.

[3] Antipov, D. and Buzdalova, A. (2017). Runtime analysis of random local search on jump function with reinforcement based selection of auxiliary objectives. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 2169–2176. IEEE.

[4] Aslam, Y. and Santhi, N. (2020). A comprehensive survey on optimization techniques in image processing. *Materials Today: Proceedings*, 24:1758–1765.

[5] Back, T., Hoffmeister, F., and Schwefel, H.-P. (1991). A survey of evolution strategies. In *Proceedings of the fourth international conference on genetic algorithms*, volume 2. Morgan Kaufmann Publishers San Mateo, CA.

[6] Barber, D. (2012). *Bayesian reasoning and machine learning*. Cambridge University Press.

[7] Barbosa, H. J., Bernardino, H. S., and Barreto, A. M. (2010). Using performance profiles to analyze the results of the 2006 CEC constrained optimization competition. In *Proceedings of 2010 IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE.

[8] Barbosa, H. J. and Lemonge, A. C. (2002). An adaptive penalty scheme in genetic algorithms for constrained optimization problems. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pages 287–294. Citeseer.

[9] Barkat Ullah, A. S., Sarker, R., and Cornforth, D. (2008). Search space reduction technique for constrained optimization with tiny feasible space. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 881–888.

[10] Ben Hadj-Alouane, A. and Bean, J. C. (1997). A genetic algorithm for the multiple-choice integer program. *Operations research*, 45(1):92–101.

[11] Boggs, P. T. and Tolle, J. W. (1995). Sequential quadratic programming. *Acta numerica*, 4(1):1–51.

[12] Brest, J., Zumer, V., and Maucec, M. S. (2006). Self-adaptive differential evolution algorithm in constrained real-parameter optimization. In *2006 IEEE international conference on evolutionary computation*, pages 215–222. IEEE.

[13] Cai, Z. and Wang, Y. (2006). A multiobjective optimization-based evolutionary algorithm for constrained optimization. *IEEE Transactions on Evolutionary Computation*, 10(6):658–675.

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

[15] Chen, T., He, J., Chen, G., and Yao., X. (2010). Choosing selection pressure for wide-gap problems. *Theoretical Computer Science*, 411(6):926–934.

[16] Chu, W., Gao, X., and Sorooshian, S. (2011). Fortify particle swam optimizer (pso) with principal components analysis: A case study in improving bound-handling for optimizing high-dimensional and complex problems. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 1644–1648. IEEE.

[17] Churchill, A. W., Husbands, P., and Philippides, A. (2013). Multi-objectivization of the tool selection problem on a budget of evaluations. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 600–614. Springer.

[18] Coello, C. A. C. (2000). Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry*, 41(2):113–127.

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

[20] Coello, C. A. C. and Montes, E. M. (2002). Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Advanced Engineering Informatics*, 16(3):193–203.

[21] Coello Coello, C. A. (2000). Constraint-handling using an evolutionary multiobjective optimization technique. *Civil Engineering Systems*, 17(4):319–346.

[22] Coello Coello, C. A. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245–1287.

[23] Crossley, W., Williams, E., Crossley, W., and Williams, E. (1997). A study of adaptive penalty functions for constrained genetic algorithm-based optimization. In *35th Aerospace Sciences Meeting and Exhibit*, page 83.

[24] Cui, Z., Li, F., and Zhang, W. (2018). Bat algorithm with principal component analysis. *International Journal of Machine Learning and Cybernetics*, pages 1–20.

[25] Das, S. and Suganthan, P. (2011). Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15(1):4–31.

[26] Datta, R. and Deb, K. (2016). Uniform adaptive scaling of equality and inequality constraints within hybrid evolutionary-cum-classical optimization. *Soft Computing*, 20(6):2367–2382.

[27] Davidor, Y. (1991). Epistasis variance: A viewpoint on ga-hardness. In *Foundations of genetic algorithms*, volume 1, pages 23–35. Elsevier.

[28] Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer methods in applied mechanics and engineering*, 186(2-4):311–338.

[29] Deb, K. and Datta, R. (2010). A fast and accurate solution of constrained optimization problems using a hybrid bi-objective and penalty function approach. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE.

[30] Deb, K. and Datta, R. (2013). A bi-objective constrained optimization algorithm using a hybrid evolutionary and penalty function approach. *Engineering Optimization*, 45(5):503–527.

[31] Deb, K., Lele, S., and Datta, R. (2007). A hybrid evolutionary multi-objective and SQP based procedure for constrained optimization. In Kang, L., Liu, Y., and Zeng, S., editors, *Advances in Computation and Intelligence*, pages 36–45. Springer.

[32] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197.

[33] Deb, K. and Sundar, J. (2006). Reference point based multi-objective optimization using evolutionary algorithms. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 635–642.

[34] Dong, N. and Wang, Y. (2014). An unbiased bi-objective optimization model and algorithm for constrained optimization. *International Journal of Pattern Recognition and Artificial Intelligence*, 28(08):1459008.

[35] Dorigo, M. (1992). Optimization, learning and natural algorithms. *Ph. D. Thesis, Politecnico di Milano*.

[36] Fan, Z., Fang, Y., Li, W., Yuan, Y., Wang, Z., and Bian, X. (2018). Lshade44 with an improved ε constraint-handling method for solving constrained single-objective optimization problems. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE.

[37] Friedrich, T., He, J., Hebbinghaus, N., Neumann, F., and Witt, C. (2009). Analyses of simple hybrid algorithms for the vertex cover problem. *Evolutionary Computation*, 17(1):3–19.

[38] g Luenberger, D. (2008). *Linear and nonlinear programming*. Springer Science+ Business Media, LLC.

[39] Gao, W.-F., Yen, G. G., and Liu, S.-Y. (2014). A dual-population differential evolution with coevolution for constrained optimization. *IEEE transactions on cybernetics*, 45(5):1108–1121.

[40] García, S., Molina, D., Lozano, M., and Herrera, F. (2009). A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the cec'2005 special session on real parameter optimization. *Journal of Heuristics*, 15(6):617–644.

[41] Garza-Fabre, M., Rodriguez-Tello, E., and Toscano-Pulido, G. (2015). Constraint-handling through multi-objective optimization: the hydrophobic-polar model for protein structure prediction. *Computers & Operations Research*, 53:128–153.

[42] Garza-Fabre, M., Toscano-Pulido, G., and Rodriguez-Tello, E. (2013). Handling constraints in the hp model for protein structure prediction by multiobjective optimization. In *2013 IEEE Congress on Evolutionary Computation*, pages 2728–2735. IEEE.

[43] Gong, W. and Cai, Z. (2008). A multiobjective differential evolution algorithm for constrained optimization. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 181–188. IEEE.

[44] Hamda, H. and Schoenauer, M. (2000). Adaptive techniques for evolutionary topological optimum design. In *Evolutionary design and manufacture*, pages 123–136. Springer.

[45] Hamida, S. B. and Schoenauer, M. (2002). Aschea: new results using adaptive segregational constraint handling. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 1, pages 884–889. IEEE.

[46] He, J., Chen, T., and Yao, X. (2015). On the easiest and hardest fitness functions. *IEEE Transactions on Evolutionary Computation*, 19(2):295–305.

[47] He, J. and Lin, G. (2016). Average convergence rate of evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 20(2):316–321.

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

[49] He, J. and Yao, X. (2003). Towards an analytic framework for analysing the computation time of evolutionary algorithms. *Artificial Intelligence*, 145(1-2):59–97.

[50] He, J. and Yao, X. (2017). Average drift analysis and population scalability. *IEEE Transactions on Evolutionary Computation*, 21(3):426–439.

[51] Hellwig, M. and Beyer, H.-G. (2018). A matrix adaptation evolution strategy for constrained real-parameter optimization. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE.

[52] Hoffmeister, F. and Sprave, J. (1996). Problem-independent handling of constraints by use of metric penalty functions.

[53] Holland, J. H. et al. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

[54] Homaifar, A., Qi, C. X., and Lai, S. H. (1994). Constrained optimization via genetic algorithms. *Simulation*, 62(4):242–253.

[55] Horn, J., Nafpliotis, N., and Goldberg, D. E. (1994). A niched pareto genetic algorithm for multiobjective optimization. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 82–87. Ieee.

[56] Huang, V. L., Qin, A. K., and Suganthan, P. N. (2006). Self-adaptive differential evolution algorithm for constrained real-parameter optimization. In *Proceedings of 2006 IEEE Congress on Evolutionary Computation*, pages 17–24. IEEE.

[57] Islam, S. M., Das, S., Ghosh, S., Roy, S., and Suganthan, P. N. (2011). 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)*, 42(2):482–500.

[58] Jensen, M. T. (2004). Helper-objectives: Using multi-objective evolutionary algorithms for single-objective optimisation. *Journal of Mathematical Modelling and Algorithms*, 3(4):323–347.

[59] Ji, B., Yuan, X., and Yuan, Y. (2017). Modified nsga-ii for solving continuous berth allocation problem: Using multiobjective constraint-handling strategy. *IEEE transactions on cybernetics*, 47(9):2885–2895.

[60] Ji, J.-Y., Yu, W.-J., Gong, Y.-J., and Zhang, J. (2018). Multiobjective optimization with ε-constrained method for solving real-parameter constrained optimization problems. *Information Sciences*, 467:15–34.

[61] Jiao, L., Li, L., Shang, R., Liu, F., and Stolkin, R. (2013). A novel selection evolutionary strategy for constrained optimization. *Information Sciences*, 239:122–141.

[62] Jiao, R., Zeng, S., Alkasassbeh, J. S., and Li, C. (2017). Dynamic multi-objective evolutionary algorithms for single-objective optimization. *Applied Soft Computing*, 61:793–805.

[63] Jiménez, F., Gómez-Skarmeta, A. F., and Sánchez, G. (2002). How evolutionary multiobjective optimization can be used for goals and priorities based optimization. In *Primer Congreso Espanol de Algoritmos Evolutivos y Bioinspirados (AEB'02)*, pages 460–465. Mérida, Espana, Universidad de Extremadura.

[64] Joines, J. A. and Houck, C. R. (1994). On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with ga's. In *Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence*, pages 579–584. IEEE.

[65] Jones, T. and Forrest, S. (1995). Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In Eshelman, L. J., editor, *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 184–192. Morgan Kaufmann.

[66] Kaelo, P. and Ali, M. (2006). A numerical study of some modified differential evolution algorithms. *European journal of operational research*, 169(3):1176–1184.

[67] Kazarlis, S. and Petridis, V. (1998). Varying fitness functions in genetic algorithms: Studying the rate of increase of the dynamic penalty terms. In *International conference on parallel problem solving from nature*, pages 211–220. Springer.

[68] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE.

[69] Kothari, V., Anuradha, J., Shah, S., and Mittal, P. (2011). A survey on particle swarm optimization in feature selection. In *International Conference on Computing and Communication Systems*, pages 192–201. Springer.

[70] Kukkonen, S. and Lampinen, J. (2006). Constrained real-parameter optimization with generalized differential evolution. In *2006 IEEE International Conference on Evolutionary Computation*, pages 207–214. IEEE.

[71] Laumanns, M., Thiele, L., Deb, K., and Zitzler, E. (2002). Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary computation*, 10(3):263–282.

[72] Le Riche, R., Knopf-Lenoir, C., and Haftka, R. T. (1995). A segregated genetic algorithm for constrained structural optimization. In *ICGA*, pages 558–565.

[73] Li, X., Zeng, S., Li, C., and Ma, J. (2017). Many-objective optimization with dynamic constraint handling for constrained optimization problems. *Soft Computing*, 21(24):7435–7445.

[74] Li, X. and Zhang, G. (2014). Biased multiobjective optimization for constrained single-objective evolutionary optimization. In *Proceeding of the 11th World Congress on Intelligent Control and Automation*, pages 891–896. IEEE.

[75] Li, Y.-l., Zhang, J., and Chen, W.-n. (2012). Differential evolution algorithm with pca-based crossover. In *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*, pages 1509–1510. ACM.

[76] Liang, J., Runarsson, T. P., Mezura-Montes, E., Clerc, M., Suganthan, P. N., Coello, C. C., and Deb, K. (2006). Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization. *Journal of Applied Mechanics*, 41(8):8–31.

[77] Liang, J. J. and Suganthan, P. N. (2006). Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism. In *2006 IEEE International Conference on Evolutionary Computation*, pages 9–16. IEEE.

[78] Maccarthy, B. L. and Liu, J. (1993). Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling. *The International Journal of Production Research*, 31(1):59–79.

[79] Malan, K. M. and Engelbrecht, A. P. (2013). A survey of techniques for characterising fitness landscapes and some possible ways forward. *Information Sciences*, 241:148–163.

[80] Mallipeddi, R. and Suganthan, P. N. (2010). Ensemble of constraint handling techniques. *IEEE Transactions on Evolutionary Computation*, 14(4):561–579.

[81] Mankiewicz, R. (2000). *The story of mathematics*. Cassell.

[82] Masuda, K. and Kurihara, K. (2012). A constrained global optimization method based on multi-objective particle swarm optimization. *Electronics and Communications in Japan*, 95(1):43–54.

[83] Merz, P. (2004). Advanced fitness landscape analysis and the performance of memetic algorithms. *Evolutionary Computation*, 12(3):303–325.

[84] Mezura-Montes, E. and Coello, C. A. C. (2005a). A simple multimembered evolution strategy to solve constrained optimization problems. *IEEE Transactions on Evolutionary computation*, 9(1):1–17.

[85] Mezura-Montes, E. and Coello, C. A. C. (2005b). Use of multiobjective optimization concepts to handle constraints in genetic algorithms. In *Evolutionary Multiobjective Optimization*, pages 229–254. Springer.

[86] Mezura-Montes, E. and Coello, C. A. C. (2008). Constrained optimization via multiobjective evolutionary algorithms. In Knowles, J., Corne, D., Deb, K., and Chair, D., editors, *Multiobjective Problem Solving from Nature*, pages 53–75. Springer Berlin Heidelberg.

[87] Mezura-Montes, E. and Coello Coello, C. A. (2011). Constraint-handling in nature-inspired numerical optimization: past, present and future. *Swarm and Evolutionary Computation*, 1(4):173–194.

[88] Mezura-Montes, E., Velázquez-Reyes, J., and Coello, C. C. (2006). Modified differential evolution for constrained optimization. In *Proceedings of 2006 IEEE Congress on Evolutionary Computation*, pages 25–32. IEEE.

[89] Michalewicz, Z. and Attia, N. (1994). Evolutionary optimization of constrained problems. In *Proceedings of the 3rd annual conference on evolutionary programming*, pages 98–108. World Scientific.

[90] Michalewicz, Z. and Schoenauer, M. (1996). Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary computation*, 4(1):1–32.

[91] Miettinen, K. (2012). *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media.

[92] Morales, A. K. and Quezada, C. V. (1998). A universal eclectic genetic algorithm for constrained optimization. In *Proceedings of the 6th European congress on intelligent techniques and soft computing*, volume 1, pages 518–522.

[93] Moré, J. J. and Sorensen, D. C. (1982). Newton's method. Technical report, Argonne National Lab., IL (USA).

[94] Moser, I., Gheorghita, M., and Aleti, A. (2017). Identifying features of fitness landscapes and relating them to problem difficulty. *Evolutionary computation*, 25(3):407–437.

[95] Munkres, J. (2000). *Topology*. Prentice Hall.

[96] Muñoz, M. A., Kirley, M., and Halgamuge, S. K. (2015). Exploratory landscape analysis of continuous space optimization problems using information content. *IEEE Transactions on Evolutionary Computation*, 19(1):74–87.

[97] Munoz-Zavala, A. E., Hernandez-Aguirre, A., Villa-Diharce, E. R., and Botello-Rionda, S. (2006). Peso+ for constrained optimization. In *Proceedings of 2006 IEEE Congress on Evolutionary Computation*, pages 231–238. IEEE.

[98] Munteanu, C. and Lazarescu, V. (1999). Improving mutation capabilities in a real-coded genetic algorithm. In *Workshops on Applications of Evolutionary Computation*, pages 138–149. Springer.

[99] Murugan, P., Kannan, S., and Baskar, S. (2009). Application of nsga-ii algorithm to single-objective transmission constrained generation expansion planning. *IEEE Transactions on Power Systems*, 24(4):1790–1797.

[100] Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *The computer journal*, 7(4):308–313.

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

[102] Neumann, F. and Wegener, I. (2006). Minimum spanning trees made easier via multi-objective optimization. *Natural Computing*, 5(3):305–319.

[103] Ong, B. T. and Fukushima, M. (2015). Automatically terminated particle swarm optimization with principal component analysis. *International Journal of Information Technology & Decision Making*, 14(01):171–194.

[104] Parmee, I. and Purchase, G. (1994). The development of a directed genetic search technique for heavily constrained design spaces. In *Adaptive computing in engineering design and control*, volume 94, pages 97–102.

[105] Peng, C., Liu, H.-L., and Gu, F. (2018). A novel constraint-handling technique based on dynamic weights for constrained optimization problems. *Soft Computing*, 22(12):3919–3935.

[106] Pereyra, M., Schniter, P., Chouzenoux, E., Pesquet, J.-C., Tourneret, J.-Y., Hero, A. O., and McLaughlin, S. (2015). A survey of stochastic simulation and optimization methods in signal processing. *IEEE Journal of Selected Topics in Signal Processing*, 10(2):224–241.

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

[108] Price, K., Storn, R. M., and Lampinen, J. A. (2006). *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media.

[109] Qin, A. K. and Suganthan, P. N. (2005). Self-adaptive differential evolution algorithm for numerical optimization. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 2, pages 1785–1791. IEEE.

[110] Rasheed, K. (1998). An adaptive penalty approach for constrained genetic-algorithm optimization. In *Proceedings of the third annual genetic programming conference*, pages 584–590. Citeseer.

[111] Ray, T., Kang, T., and Chye, S. K. (2000). An evolutionary algorithm for constrained optimization. In *Proceedings of 2000 Genetic and Evolutionary Computation Conference*, pages 771–777, San Francisco. Morgan Kaufmann.

[112] Ray, T. and Liew, K.-M. (2003). Society and civilization: An optimization algorithm based on the simulation of social behavior. *IEEE Transactions on Evolutionary Computation*, 7(4):386–396.

[113] Ray, T., Singh, H., Isaacs, A., and Smith, W. (2009). Infeasibility driven evolutionary algorithm for constrained optimization. In Mezura-Montes, E., editor, *Constraint-Handling in Evolutionary Optimization*, volume 198, pages 145–165. Springer Berlin Heidelberg.

[114] Reeves, C. R. (1999). Landscapes, operators and heuristic search. *Annals of Operations Research*, 86:473–490.

[115] Reeves, C. R. (2014). Fitness landscapes. In *Search methodologies*, pages 681–705. Springer.

[116] Reeves, C. R. and Eremeev, A. V. (2004). Statistical analysis of local search landscapes. *Journal of the Operational Research Society*, 55(7):687–693.

[117] Reidys, C. M. and Stadler, P. F. (2002). Combinatorial landscapes. *SIAM review*, 44(1):3–54.

[118] Reynoso-Meza, G., Blasco, X., Sanchis, J., and Martínez, M. (2010). Multiobjective optimization algorithm for solving constrained single objective problems. In *IEEE congress on evolutionary computation*, pages 1–7. IEEE.

[119] Runarsson, T. P. and Yao, X. (2000). Stochastic ranking for constrained evolutionary optimization. *Evolutionary Computation, IEEE Transactions on*, 4(3):284–294.

[120] Runarsson, T. P. and Yao, X. (2005). Search biases in constrained evolutionary optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 35(2):233–243.

[121] Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the first international conference on genetic algorithms and their applications, 1985*. Lawrence Erlbaum Associates. Inc., Publishers.

[122] Schoenauer, M. and Xanthakis, S. (1993). Constrained ga optimization. In *ICGA*, pages 573–580.

[123] Schwefel, H.-P. (1981). *Numerical optimization of computer models*. John Wiley & Sons, Inc.

[124] Segura, C., Coello, C. A. C., Miranda, G., and León, C. (2013). Using multi-objective evolutionary algorithms for single-objective optimization. *4OR*, 11(3):201–228.

[125] Segura, C., Coello, C. A. C., Miranda, G., and León, C. (2016). Using multi-objective evolutionary algorithms for single-objective constrained and unconstrained optimization. *Annals of Operations Research*, 240(1):217–250.

[126] Singh, H. K., Ray, T., and Sarker, R. (2013). Optimum oil production planning using infeasibility driven evolutionary algorithm. *Evolutionary Computation*, 21(1):65–82.

[127] Singh, H. K., Ray, T., and Smith, W. (2010). Performance of infeasibility empowered memetic algorithm for cec 2010 constrained optimization problems. In *IEEE congress on evolutionary computation*, pages 1–8. IEEE.

[128] Sinha, A., Srinivasan, A., and Deb, K. (2006). A population-based, parent centric procedure for constrained real-parameter optimization. In *Proceedings of 2006 IEEE Congress on Evolutionary Computation*, pages 239–245. IEEE.

[129] Stadler, P. F. (1995). Towards a theory of landscapes. In *Complex systems and binary networks*, pages 78–163. Springer.

[130] Stadler, P. F. and Wagner, G. P. (1997). Algebraic theory of recombination spaces. *Evolutionary computation*, 5(3):241–275.

[131] Storn, R. and Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359.

[132] Suganthan, P. N. (2017). The CEC 2017 competition on constrained real-parameter optimization.

[133] Suganthan, P. N. (2018). The CEC 2018 competition on constrained real-parameter optimization.

[134] Suganthan, P. N. (2020). The CEC 2020 competition on real-world single objective constrained optimization.

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

[136] Surry, P. D., Radcliffe, N. J., and Boyd, I. D. (1995). A multi-objective approach to constrained optimisation of gas supply networks: The comoga method. In *AISB Workshop on Evolutionary Computing*, pages 166–180. Springer.

[137] Takahama, T. and Sakai, S. (2006). Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites. In *IEEE Congress on Evolutionary Computation*, pages 1–8.

[138] Tanabe, R. and Fukunaga, A. S. (2014). Improving the search performance of shade using linear population size reduction. In *2014 IEEE congress on evolutionary computation (CEC)*, pages 1658–1665. IEEE.

[139] Tasgetiren, M. F. and Suganthan, P. N. (2006). A multi-populated differential evolution algorithm for solving constrained optimization problem. In *Proceedings of 2006 IEEE Congress on Evolutionary Computation*, pages 33–40. IEEE.

[140] Trived, A., Srinivasan, D., and Biswas, N. (2018). An improved unified differential evolution algorithm for constrained optimization problems. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–10. IEEE.

[141] Trivedi, A., Sanyal, K., Verma, P., and Srinivasan, D. (2017a). A unified differential evolution algorithm for constrained optimization problems. In *Evolutionary Computation (CEC), 2017 IEEE Congress on*, pages 1231–1238. IEEE.

[142] Trivedi, A., Srinivasan, D., Sanyal, K., and Ghosh, A. (2017b). A survey of multiobjective evolutionary algorithms based on decomposition. *IEEE Transactions on Evolutionary Computation*, 21(3):440–462.

[143] Tvrdık, J. (2006). Competitive differential evolution. In *MENDEL*, pages 7–12.

[144] Tvrdík, J. (2009). Adaptation in differential evolution: A numerical comparison. *Applied Soft Computing*, 9(3):1149–1155.

[145] Tvrdík, J. and Poláková, R. (2017). A simple framework for constrained problems with application of l-shade44 and ide. In *Evolutionary Computation (CEC), 2017 IEEE Congress on*, pages 1436–1443. IEEE.

[146] Venkatraman, S. and Yen, G. G. (2005). A generic framework for constrained optimization using genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 9(4):424–435.

[147] Venter, G. and Haftka, R. (2010). Constrained particle swarm optimization using a bi-objective formulation. *Structural and Multidisciplinary Optimization*, 40(1-6):65.

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

[149] Wang, Y. (2015). Incorporating social media in public relations: A synthesis of social media-related public relations research. *Public Relations Journal*, 9(3):2.

[150] Wang, Y. and Cai, Z. (2011). A dynamic hybrid framework for constrained evolutionary optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(1):203–217.

[151] Wang, Y. and Cai, Z. (2012). Combining multiobjective optimization with differential evolution to solve constrained optimization problems. *IEEE Transactions on Evolutionary Computation*, 16(1):117–134.

[152] Wang, Y., Cai, Z., Guo, G., and Zhou, Y. (2007a). Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 37(3):560–575.

[153] Wang, Y., Cai, Z., Zhou, Y., and Zeng, W. (2008). An adaptive tradeoff model for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 12(1):80–92.

[154] Wang, Y., Liu, H., Cai, Z., and Zhou, Y. (2007b). An orthogonal design based constrained evolutionary optimization algorithm. *Engineering Optimization*, 39(6):715–736.

[155] Wang, Y., Xiang, J., and Cai, Z. (2012). A regularity model-based multiobjective estimation of distribution algorithm with reducing redundant cluster operator. *Applied Soft Computing*, 12(11):3526–3538.

[156] Watanabe, S. and Sakakibara, K. (2005). Multi-objective approaches in a single-objective optimization environment. In *2005 IEEE Congress on Evolutionary Computation*, volume 2, pages 1714–1721. IEEE.

[157] Weinberger, E. (1990). Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological cybernetics*, 63(5):325–336.

[158] Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82.

[159] Wright, S. (1932). The roles of mutation, inbreeding, crossbreeding, and selection in evolution. In *Proceedings of the Sixth International Congress on Genetics*, volume 1, pages 356—-366.

[160] Wu, B., Yu, X., and Liu, L. (2001). Fuzzy penalty function approach for constrained function optimization with evolutionary algorithms. In *Proceedings of the 8th international conference on neural information processing*, pages 299–304. Fudan University Press Shanghai, China.

[161] Wu, G., Mallipeddi, R., and Suganthan, P. (2017). Problem definitions and evaluation criteria for the cec 2017 competition on constrained real-parameter optimization. *National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, Technical Report*.

[162] Xu, G., Zhao, X., and Li, R. (2018). Cooperative co-evolution with principal component analysis for large scale optimization. In *International Conference on Bio-Inspired Computing: Theories and Applications*, pages 426–434. Springer.

[163] Xu, T. and He, J. (2015). Multi-objective differential evolution with helper functions for constrained optimization. *arXiv preprint arXiv:1509.09060*.

[164] Xu, T. and He, J. (2019). A multi-population helper and equivalent objective differential evolution algorithm. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2237–2244. IEEE.

[165] Xu, T., He, J., and Shang, C. (2019). Helper and equivalent objectives: An efficient approach to constrained optimisation. *arXiv preprint arXiv:1903.04886*.

[166] Xu, T., He, J., and Shang, C. (2020). Helper and equivalent objectives: Efficient approach for constrained optimization. *IEEE transactions on cybernetics*.

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

[168] Zaharie, D. (2009). Influence of crossover on the behavior of differential evolution algorithms. *Applied soft computing*, 9(3):1126–1138.

[169] Zamuda, A. (2017). Adaptive constraint handling and success history differential evolution for cec 2017 constrained real-parameter optimization. In *Evolutionary Computation (CEC), 2017 IEEE Congress on*, pages 2443–2450. IEEE.

[170] Zeng, S., Jiao, R., Li, C., Li, X., and Alkasassbeh, J. S. (2017). A general framework of dynamic constrained multiobjective evolutionary algorithms for constrained optimization. *IEEE transactions on Cybernetics*, 47(9):2678–2688.

[171] Zhang, J. and Sanderson, A. C. (2009). Jade: adaptive differential evolution with optional external archive. *IEEE Transactions on evolutionary computation*, 13(5):945–958.

[172] Zhang, Q. and Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731.

[173] Zhang, Q., Zhou, A., and Jin, Y. (2008). RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm. *IEEE Transactions on Evolutionary Computation*, 12(1):41–63.

[174] Zhao, X., Lin, W., and Zhang, Q. (2014). Enhanced particle swarm optimization based on principal component analysis and line search. *Applied Mathematics and Computation*, 229:440–456.

[175] Zhou, Y., Li, Y., He, J., and Kang, L. (2003). Multi-objective and MGG evolutionary algorithm for constrained optimisation. In *Proceedings of 2003 IEEE Congress on Evolutionary Computation*, pages 1–5, Canberra, Australia. IEEE Press.

[176] Zielinski, K. and Laur, R. (2006). Constrained single-objective optimization using particle swarm optimization. In *2006 IEEE International Conference on Evolutionary Computation*, pages 443–450. IEEE.

[177] Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4):257–271.

Appendix A

Detailed Results

Table A.1 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 A.2 Function Values of HECO-DE Achieved for 30D ($FES_{\max} = 20000 \times D$)

problem	C01	C02	C03	C04	C05	C06	C07
Best	0.00000e+00	0.00000e+00	0.00000e+00	1.35728e+01	0.00000e+00	0.00000e+00	-2.97725e+03
Median	2.44177e-29	2.44177e-29	4.10208e-29	1.35728e+01	0.00000e+00	0.00000e+00	-1.77011e+03
c	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 2
$\overline{v}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	6.07711e-04
mean	2.78897e-29	3.05062e-29	3.93642e-29	1.35728e+01	1.59465e-01	0.00000e+00	-2.10204e+03
Worst	1.06656e-28	9.93965e-29	9.95937e-29	1.35728e+01	3.98662e+00	0.00000e+00	-2.76259e+03
std	2.55048e-29	2.53801e-29	2.96881e-29	2.63476e-15	7.81216e-01	0.00000e+00	8.69986e+02
SR	100	100	100	100	100	100	16
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	8.80450e-04
Problem	C8	C9	C10	C11	C12	C13	C14
Best	-2.83981e-04	-2.66551e-03	-1.02842e-04	-9.24361e-01	3.98327e+00	0.00000e+00	1.40852e+00
Median	-2.83981e-04	-2.66551e-03	-1.02842e-04	6.09128e+00	3.98463e+00	0.00000e+00	1.40852e+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	8.12995e-22	0.00000e+00	0.00000e+00	0.00000e+00
mean	-2.83981e-04	-2.66551e-03	-1.02842e-04	-3.07691e+02	3.98460e+00	0.00000e+00	1.41033e+00
Worst	-2.83981e-04	-2.66551e-03	-1.02841e-04	-1.88170e+03	3.98646e+00	0.00000e+00	1.42415e+00
std	9.41607e-11	3.55271e-16	1.02224e-10	5.66173e+02	8.84065e-04	0.00000e+00	4.03595e-03
SR	100	100	100	44	100	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	3.11118e+01	0.00000e+00	0.00000e+00	0.00000e+00
Problem	C15	C16	C17	C18	C19	C20	C21
Best	2.35612e+00	0.00000e+00	3.08555e-02	4.22186e+01	0.00000e+00	2.20283e+00	3.98290e+00
Median	2.35612e+00	0.00000e+00	1.47580e+00	5.34725e+01	0.00000e+00	2.59641e+00	3.98486e+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	1.45000e+01	0.00000e+00	2.13749e+04	0.00000e+00	0.00000e+00
mean	2.35612e+00	0.00000e+00	5.95048e-01	5.07349e+01	0.00000e+00	2.60824e+00	3.98469e+00
Worst	2.35612e+00	0.00000e+00	6.86588e-01	5.48137e+01	0.00000e+00	3.02403e+00	3.98602e+00
std	1.32633e-15	0.00000e+00	4.74854e-01	4.57596e+00	0.00000e+00	2.17753e-01	9.06801e-04
SR	100	100	0	100	0	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	1.49000e+01	0.00000e+00	2.13749e+04	0.00000e+00	0.00000e+00
Problem	C22	C23	C24	C25	C26	C27	C28
Best	3.50703e-12	1.40854e+00	2.35612e+00	0.00000e+00	6.66654e-01	2.08761e+02	0.00000e+00
Median	1.05155e-06	1.40862e+00	2.35612e+00	0.00000e+00	8.81989e-01	2.08769e+02	1.78375e-06
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	2.13749e+04
mean	8.00599e-05	1.40889e+00	2.35612e+00	0.00000e+00	8.64329e-01	2.27503e+02	3.33268e-01
Worst	1.81267e-03	1.41193e+00	2.35612e+00	0.00000e+00	9.61474e-01	2.51358e+02	3.90569e+00
std	3.54137e-04	8.42014e-04	1.00417e-07	0.00000e+00	8.88294e-02	2.11389e+01	9.60838e-01
SR	100	100	100	100	0	100	0
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	1.55000e+01	0.00000e+00	2.13761e+04

Table A.3 Function Values of HECO-DE Achieved for 50D ($FES_{\max} = 20000 \times D$)

problem	C01	C02	C03	C04	C05	C06	C07
Best	1.10354e-28	1.36473e-28	2.24431e-28	1.35728e+01	0.00000e+00	0.00000e+00	-4.00718e+03
Median	2.29078e-28	2.44756e-28	5.41553e-28	1.35728e+01	9.12120e-30	2.00943e+02	-2.44889e+03
c	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 2
$\overline{v}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	2.16020e-04
mean	3.37842e-28	2.92104e-28	5.58198e-28	1.35728e+01	1.64818e-29	1.39232e+02	-3.08941e+03
Worst	1.53740e-27	6.90845e-28	9.95937e-28	1.35728e+01	8.92399e-29	2.89572e+02	-3.25560e+03
std	2.82620e-28	1.37442e-28	2.27814e-28	2.89693e-14	2.52948e-29	1.27084e+02	7.42860e+02
SR	100	100	100	100	100	52	24
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	1.39134e-03	3.26548e-04
Problem	C8	C9	C10	C11	C12	C13	C14
Best	-1.34463e-04	-2.03709e-03	-4.82662e-05	-2.52965e+01	3.98156e+00	0.00000e+00	1.09995e+00
Median	-1.33775e-04	-2.03709e-03	-4.82592e-05	-2.90737e+03	3.98261e+00	5.32402e-27	1.11954e+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	1.08397e+02	0.00000e+00	0.00000e+00	0.00000e+00
mean	-1.30457e-04	-2.03709e-03	-4.82539e-05	-2.32090e+03	3.98257e+00	6.15871e+00	1.12293e+00
Worst	-9.48622e-05	-2.03709e-03	-4.81877e-05	-2.55864e+03	3.98319e+00	7.52752e+01	1.15587e+00
std	8.54223e-06	6.15348e-16	1.64358e-08	9.08096e+02	3.98495e-04	2.03125e+01	1.75211e-02
SR	100	100	100	0	100	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	1.29300e+02	0.00000e+00	0.00000e+00	0.00000e+00
Problem	C15	C16	C17	C18	C19	C20	C21
Best	2.35612e+00	0.00000e+00	1.77100e-01	4.42174e+01	0.00000e+00	4.89241e+00	3.98145e+00
Median	2.35612e+00	0.00000e+00	6.41481e-01	4.61632e+01	0.00000e+00	5.59980e+00	3.98235e+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	0.00000e+00	7.55210e-01	4.64731e+01	0.00000e+00	5.60828e+00	3.98242e+00
Worst	2.35612e+00	0.00000e+00	1.03384e+00	4.92174e+01	0.00000e+00	6.17004e+00	3.98453e+00
std	1.69686e-15	0.00000e+00	3.71225e-01	1.08924e+00	0.00000e+00	3.45360e-01	6.31632e-04
SR	100	100	0	100	0	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	2.53000e+01	0.00000e+00	3.61162e+04	0.00000e+00	0.00000e+00
Problem	C22	C23	C24	C25	C26	C27	C28
Best	1.37618e+01	1.09997e+00	2.35612e+00	0.00000e+00	8.96182e-01	2.47643e+02	2.56358e-04
Median	1.56511e+01	1.10006e+00	2.35612e+00	0.00000e+00	9.99371e-01	2.47664e+02	6.60701e-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	2.55000e+01	0.00000e+00	3.61166e+04
mean	9.59266e+01	1.10114e+00	2.35612e+00	1.58932e-15	9.97925e-01	2.55605e+02	1.22326e+00
Worst	1.95227e+03	1.12130e+00	2.35612e+00	3.97330e-14	1.04168e+00	2.64241e+02	8.39748e+00
std	3.79187e+02	4.18541e-03	4.99851e-08	7.78605e-15	3.46115e-02	8.27783e+00	2.01178e+00
SR	96	100	100	100	0	100	0
$\overline{vio}$	5.00660e+00	0.00000e+00	0.00000e+00	0.00000e+00	2.55000e+01	0.00000e+00	3.61202e+04

Table A.4 Function Values of HECO-DE Achieved for 100D ($FES_{\max} = 20000 \times D$)

problem	C01	C02	C03	C04	C05	C06	C07
Best	1.39645e-20	2.82470e-19	1.70210e-19	1.35728e+01	6.25206e-16	7.81280e+02	-5.86436e+03
Median	6.45277e-18	8.86775e-18	2.26735e-18	1.35728e+01	5.34121e-14	8.81984e+02	-5.08220e+03
c	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 3	0 0 0
$\overline{v}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	2.91733e-03	9.30853e-05
mean	1.73680e-17	1.70709e-17	3.89365e-18	1.35728e+01	1.16002e-12	8.59412e+02	-4.71743e+03
Worst	9.62873e-17	8.54605e-17	1.94437e-17	1.35728e+01	2.35813e-11	9.39791e+02	-4.93686e+03
std	2.45766e-17	2.23586e-17	4.69314e-18	3.52814e-06	4.59692e-12	1.19393e+02	6.87463e+02
SR	100	100	100	100	100	4	44
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	2.77383e-03	1.56297e-04
Problem	C8	C9	C10	C11	C12	C13	C14
Best	-3.45438e-05	0.00000e+00	-1.71209e-05	-6.63258e+03	1.02834e+01	3.32054e+01	7.84202e-01
Median	2.13381e-04	0.00000e+00	-1.68397e-05	-7.37344e+03	3.18082e+01	3.32106e+01	8.16159e-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	1.00349e+02	0.00000e+00	0.00000e+00	0.00000e+00
mean	2.11219e-04	0.00000e+00	-1.68393e-05	-7.25820e+03	2.96135e+01	3.36611e+01	8.17791e-01
Worst	5.05202e-04	0.00000e+00	-1.64859e-05	-7.98338e+03	3.35380e+01	3.71947e+01	8.81631e-01
std	1.43678e-04	0.00000e+00	1.40929e-07	4.61216e+02	6.66521e+00	1.06502e+00	2.51557e-02
SR	100	100	100	0	100	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	1.23998e+02	0.00000e+00	0.00000e+00	0.00000e+00
Problem	C15	C16	C17	C18	C19	C20	C21
Best	2.35612e+00	0.00000e+00	2.87759e-01	4.62791e+01	0.00000e+00	1.31030e+01	1.24660e+01
Median	2.35612e+00	0.00000e+00	7.33502e-01	5.20791e+01	0.00000e+00	1.46739e+01	3.17885e+01
c	0 0 0	0 0 0	1 0 0	0 1 0	1 0 0	0 0 0	0 0 0
$\overline{v}$	0.00000e+00	0.00000e+00	5.05000e+01	4.11325e-03	7.29695e+04	0.00000e+00	0.00000e+00
mean	2.35612e+00	0.00000e+00	7.41230e-01	8.71777e+01	0.00000e+00	1.45065e+01	3.12120e+01
Worst	2.35612e+00	0.00000e+00	1.04448e+00	4.65393e+01	0.00000e+00	1.52835e+01	3.30128e+01
std	1.39022e-15	0.00000e+00	2.38862e-01	4.24470e+01	0.00000e+00	5.76377e-01	3.84882e+00
SR	100	100	0	32	0	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	5.03400e+01	9.33013e+00	7.29695e+04	0.00000e+00	0.00000e+00
Problem	C22	C23	C24	C25	C26	C27	C28
Best	7.94816e+01	7.84233e-01	2.35612e+00	1.57066e+00	9.78423e-01	2.84774e+02	3.80708e+01
Median	2.14603e+03	7.84294e-01	2.35612e+00	6.28305e+00	1.05080e+00	3.18841e+02	4.59190e+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	7.30555e+04
mean	1.75393e+04	7.88594e-01	2.35612e+00	5.27774e+00	1.05947e+00	3.05235e+02	4.40258e+01
Worst	4.61479e+03	8.12544e-01	2.35612e+00	1.88494e+01	1.09882e+00	3.18900e+02	6.41817e+01
std	3.23366e+04	8.85911e-03	2.00166e-08	4.28180e+00	3.31383e-02	1.66844e+01	1.09266e+01
SR	52	100	100	100	0	100	0
$\overline{vio}$	3.20548e+01	0.00000e+00	0.00000e+00	0.00000e+00	5.05000e+01	0.00000e+00	7.30540e+04

Table A.5 Function values of HECO-DEtch 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	-5.83545e+02
Median	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	-1.56209e+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	0.00000e+00	0.00000e+00	0.00000e+00	-2.86443e+02
Worst	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	-1.56636e+02
std	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	1.58630e+02
SR	1.0	1.0	1.0	1.0	1.0	1.0	0.64
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	6.11581e-04
Problem	C8	C9	C10	C11	C12	C13	C14
Best	-1.34840e-03	-4.97525e-03	-5.09647e-04	-1.00675e-01	3.98790e+00	0.00000e+00	2.37633e+00
Median	-1.34840e-03	-4.97525e-03	-5.09647e-04	-2.16947e+00	3.98791e+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	1.14980e-07	0.00000e+00	0.00000e+00	0.00000e+00
mean	-1.34840e-03	-4.97525e-03	-5.09647e-04	-3.12366e+00	3.98811e+00	0.00000e+00	2.38973e+00
Worst	-1.34840e-03	-4.97525e-03	-5.09647e-04	-2.19040e+01	3.98901e+00	0.00000e+00	2.64782e+00
std	0.00000e+00	0.00000e+00	0.00000e+00	4.08583e+00	3.53821e-04	0.00000e+00	5.31318e-02
SR	1.0	1.0	1.0	0.04	1.0	1.0	1.0
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	3.77175e-02	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	6.16717e-02	3.98790e+00
Median	2.35612e+00	0.00000e+00	1.08553e-02	5.04203e+01	0.00000e+00	2.17529e-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	2.85878e+00	0.00000e+00	1.08142e-02	4.20874e+01	0.00000e+00	2.10051e-01	3.98809e+00
Worst	8.63931e+00	0.00000e+00	1.03418e-02	4.71896e+01	0.00000e+00	4.39838e-01	3.98924e+00
std	1.45466e+00	0.00000e+00	1.39288e-04	1.60693e+01	0.00000e+00	9.15871e-02	3.56503e-04
SR	1.0	1.0	0.0	0.84	0.0	1.0	1.0
$\overline{vio}$	0.00000e+00	0.00000e+00	4.58000e+00	5.65968e-06	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	1.14425e-76	1.08553e-02	3.45960e+01	7.70404e-17
Median	6.17530e-30	2.37633e+00	2.35612e+00	1.04739e-63	1.08553e-02	3.65978e+01	5.00778e-12
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	7.50355e-30	2.37633e+00	2.35612e+00	1.88479e-01	2.23145e-02	3.62779e+01	3.89176e-09
Worst	1.31888e-29	2.37633e+00	2.35613e+00	1.57066e+00	2.27185e-01	3.65997e+01	4.66031e-08
std	1.83291e-30	4.16406e-07	1.89626e-06	5.10403e-01	4.40828e-02	7.33678e-01	1.00152e-08
SR	1.0	1.0	1.0	1.0	0.0	1.0	0.0
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	4.66000e+00	0.00000e+00	6.63359e+03

Table A.6 Function values of HECO-DEtch achieved for $30D$ ($FES_{\max} = 20000 \times D$) on IEEE CEC2017 benchmarks

problem	C01	C02	C03	C04	C05	C06	C07
Best	1.02552e-29	0.00000e+00	6.31089e-30	0.00000e+00	0.00000e+00	0.00000e+00	-1.11604e+03
Median	4.40406e-29	3.54987e-29	1.04130e-28	1.35728e+01	0.00000e+00	0.00000e+00	-6.55567e+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	4.63998e-29	4.02166e-29	2.09094e+01	1.38401e+01	1.59465e-01	0.00000e+00	-6.71969e+02
Worst	1.18576e-28	1.00148e-28	7.46827e+01	3.38284e+01	3.98662e+00	0.00000e+00	-2.08488e+02
std	2.49005e-29	2.78473e-29	3.35296e+01	4.86919e+00	7.81216e-01	0.00000e+00	2.08023e+02
SR	1.0	1.0	1.0	1.0	1.0	1.0	1.0
$\overline{vio}$	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	-4.36655e+01	3.98253e+00	0.00000e+00	1.40852e+00
Median	-2.83981e-04	-2.66551e-03	-1.02842e-04	-6.59455e+02	3.98253e+00	0.00000e+00	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	3.58931e+01	0.00000e+00	0.00000e+00	0.00000e+00
mean	-2.83981e-04	-2.66551e-03	-1.02842e-04	-6.08398e+02	3.98253e+00	1.83975e-27	1.40852e+00
Worst	-2.83981e-04	-2.66551e-03	-1.02842e-04	-1.09775e+03	3.98254e+00	9.74717e-27	1.40852e+00
std	1.27221e-14	2.51215e-16	2.17034e-14	3.48382e+02	5.04377e-06	3.07577e-27	9.65830e-16
SR	1.0	1.0	1.0	0.0	1.0	1.0	1.0
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	3.84761e+01	0.00000e+00	0.00000e+00	0.00000e+00
Problem	C15	C16	C17	C18	C19	C20	C21
Best	2.35612e+00	0.00000e+00	3.08555e-02	4.32174e+01	0.00000e+00	7.10332e-01	3.98253e+00
Median	2.35612e+00	1.57066e+00	3.06980e-02	5.34725e+01	0.00000e+00	1.20626e+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	1.50783e+00	2.05774e-01	5.15377e+01	0.00000e+00	1.23421e+00	3.98253e+00
Worst	2.35612e+00	1.57066e+00	9.80099e-01	4.32472e+01	0.00000e+00	1.72825e+00	3.98254e+00
std	1.14089e-15	3.07785e-01	3.15697e-01	3.35522e+00	0.00000e+00	2.13322e-01	3.89641e-06
SR	1.0	1.0	0.0	0.76	0.0	1.0	1.0
$\overline{vio}$	0.00000e+00	0.00000e+00	1.50200e+01	1.29011e-04	2.13749e+04	0.00000e+00	0.00000e+00
Problem	C22	C23	C24	C25	C26	C27	C28
Best	7.08035e-09	1.40852e+00	2.35612e+00	1.57066e+00	3.02961e-02	3.57765e+01	4.85465e-01
Median	5.06148e-04	1.40853e+00	2.35612e+00	6.28305e+00	7.33787e-01	3.57792e+01	-2.25721e-02
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	2.13852e+04
mean	3.80126e+00	1.41394e+00	2.35612e+00	4.96358e+00	6.59117e-01	3.57326e+01	2.40995e+00
Worst	8.94695e+01	1.49544e+00	2.35612e+00	6.28305e+00	9.48340e-01	3.51349e+01	4.72726e+00
std	1.75049e+01	1.82287e-02	2.42989e-07	2.11586e+00	2.69646e-01	1.39494e-01	2.32024e+00
SR	1.0	1.0	1.0	1.0	0.0	0.8	0.0
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	1.55000e+01	8.11154e-05	2.13846e+04

Table A.7 Function Values of HECO-DEtch Achieved for $50D$ ($FES_{\max} = 20000 \times D$) on IEEE CEC2017 benchmarks

problem	C01	C02	C03	C04	C05	C06	C07
Best	2.33811e-28	2.22471e-28	7.46384e+01	1.35728e+01	1.62025e-28	0.00000e+00	-1.56358e+03
Median	4.33590e-28	4.41516e-28	7.46827e+01	1.35728e+01	6.36019e-28	4.16449e+02	-1.09218e+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	4.86975e-28	5.11279e-28	8.00090e+01	1.46248e+01	4.78395e-01	3.50447e+02	-1.08671e+03
Worst	1.43257e-27	1.22501e-27	1.49360e+02	3.28334e+01	3.98662e+00	5.27481e+02	-4.56629e+02
std	2.47574e-28	2.70971e-28	1.70695e+01	3.79380e+00	1.29550e+00	1.56952e+02	2.37029e+02
SR	1.0	1.0	1.0	1.0	1.0	1.0	1.0
$\overline{vio}$	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	-4.36655e+01	3.98253e+00	0.00000e+00	1.40852e+00
Median	-2.83981e-04	-2.66551e-03	-1.02842e-04	-6.59455e+02	3.98253e+00	0.00000e+00	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	3.58931e+01	0.00000e+00	0.00000e+00	0.00000e+00
mean	-2.83981e-04	-2.66551e-03	-1.02842e-04	-6.08398e+02	3.98253e+00	1.83975e-27	1.40852e+00
Worst	-2.83981e-04	-2.66551e-03	-1.02842e-04	-1.09775e+03	3.98254e+00	9.74717e-27	1.40852e+00
std	1.27221e-14	2.51215e-16	2.17034e-14	3.48382e+02	5.04377e-06	3.07577e-27	9.65830e-16
SR	1.0	1.0	1.0	0.0	1.0	1.0	1.0
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	3.84761e+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	1.04125e-01	4.61632e+01	0.00000e+00	1.81371e+00	3.98145e+00
Median	5.49772e+00	1.57066e+00	8.67242e-01	5.91157e+01	0.00000e+00	2.45603e+00	3.98145e+00
c	0 0 0	0 0 0	1 0 0	0 0 1	1 0 0	0 0 0	0 0 0
$\overline{v}$	0.00000e+00	0.00000e+00	2.55000e+01	5.82542e-04	3.61162e+04	0.00000e+00	0.00000e+00
mean	4.61807e+00	3.26712e+00	7.74806e-01	5.85986e+01	0.00000e+00	2.47483e+00	4.66337e+00
Worst	5.49772e+00	6.28305e+00	1.01154e+00	8.88318e+01	0.00000e+00	3.38270e+00	1.79425e+01
std	1.41057e+00	2.26195e+00	2.56555e-01	2.66451e+01	0.00000e+00	3.16030e-01	2.77716e+00
SR	1.0	1.0	0.0	0.28	0.0	1.0	1.0
$\overline{vio}$	0.00000e+00	0.00000e+00	2.54600e+01	2.53973e+00	3.61162e+04	0.00000e+00	0.00000e+00
Problem	C22	C23	C24	C25	C26	C27	C28
Best	1.25799e+01	1.09995e+00	2.35612e+00	6.28305e+00	6.26466e-01	3.57398e+01	5.96235e+00
Median	2.10684e+01	1.10007e+00	2.35612e+00	6.28305e+00	9.55149e-01	3.57407e+01	1.51406e+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	2.55000e+01	0.00000e+00	3.61453e+04
mean	4.73971e+01	1.12484e+00	3.23577e+00	7.53968e+00	8.91679e-01	3.56785e+01	9.99076e+00
Worst	1.84712e+02	1.24170e+00	5.49772e+00	2.51326e+01	9.64372e-01	3.48836e+01	2.08246e+01
std	4.49649e+01	3.46416e-02	1.41057e+00	3.97384e+00	1.01931e-01	1.81549e-01	5.83995e+00
SR	1.0	1.0	1.0	1.0	0.0	0.76	0.0
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	2.55000e+01	9.02305e-05	3.61466e+04

Table A.8 Function Values of HECO-DEtch Achieved for $100D$ ($FES_{\max} = 20000 \times D$) on IEEE CEC2017 benchmarks

problem	C01	C02	C03	C04	C05	C06	C07
Best	7.87020e-24	2.19825e-23	1.39153e+02	1.59192e+01	2.00681e-25	8.31414e+02	-3.21361e+03
Median	5.92127e-22	2.81110e-22	2.34650e+02	5.07426e+01	1.58380e-18	1.07772e+03	-2.51495e+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	1.12470e-21	1.74755e-21	2.64652e+02	5.58766e+01	4.17532e-17	1.05746e+03	-2.54099e+03
Worst	5.78312e-21	1.42572e-20	5.87215e+02	1.06460e+02	8.26620e-16	1.29497e+03	-2.04866e+03
std	1.46967e-21	3.12464e-21	9.46691e+01	2.02339e+01	1.61154e-16	1.43582e+02	2.75687e+02
SR	1.0	1.0	1.0	1.0	1.0	1.0	1.0
$\overline{vio}$	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	4.66174e-02	0.00000e+00	-1.70239e-05	-4.17294e+03	3.98064e+00	3.37623e+01	7.84202e-01
Median	8.91640e-02	0.00000e+00	-1.68884e-05	-4.74436e+03	3.16133e+01	2.57830e+02	7.84202e-01
c	0 0 1	0 0 0	0 0 0	1 0 0	0 0 0	0 0 0	0 0 0
$\overline{v}$	3.92488e-03	0.00000e+00	0.00000e+00	1.82917e+02	0.00000e+00	0.00000e+00	0.00000e+00
mean	1.05340e-01	6.62849e-03	-1.68609e-05	-4.77995e+03	2.80143e+01	2.83757e+02	7.85257e-01
Worst	2.29738e-01	1.65712e-01	-1.64890e-05	-5.21684e+03	3.96545e+01	7.92822e+02	8.10576e-01
std	4.63973e-02	3.24729e-02	1.47418e-07	2.51454e+02	1.03992e+01	2.43915e+02	5.16821e-03
SR	0.0	1.0	1.0	0.0	1.0	1.0	1.0
$\overline{vio}$	5.83750e-03	0.00000e+00	0.00000e+00	1.67675e+02	0.00000e+00	0.00000e+00	0.00000e+00
Problem	C15	C16	C17	C18	C19	C20	C21
Best	8.63931e+00	6.28305e+00	8.77979e-01	6.18991e+01	0.00000e+00	5.29753e+00	3.98065e+00
Median	1.17809e+01	6.28305e+00	1.03052e+00	1.51742e+02	0.00000e+00	5.95815e+00	3.16157e+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	2.12265e+01	7.29695e+04	0.00000e+00	0.00000e+00
mean	1.05243e+01	6.28305e+00	1.02651e+00	9.12670e+01	0.00000e+00	6.07722e+00	2.52872e+01
Worst	1.17809e+01	6.28305e+00	1.06315e+00	8.36330e+01	0.00000e+00	7.07637e+00	3.19797e+01
std	1.53906e+00	6.63198e-07	3.88977e-02	2.88933e+01	0.00000e+00	4.35800e-01	9.57386e+00
SR	1.0	1.0	0.0	0.0	0.0	1.0	1.0
$\overline{vio}$	0.00000e+00	0.00000e+00	5.05000e+01	1.62008e+01	7.29695e+04	0.00000e+00	0.00000e+00
Problem	C22	C23	C24	C25	C26	C27	C28
Best	8.07539e+01	7.84210e-01	5.49772e+00	1.25662e+01	9.37221e-01	3.56722e+01	1.50539e+01
Median	4.04097e+02	7.84269e-01	8.63931e+00	2.51326e+01	1.00161e+00	3.56745e+01	2.03093e+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	7.30675e+04
mean	5.49638e+02	7.98405e-01	7.50834e+00	2.68291e+01	9.89270e-01	3.50493e+01	3.60545e+01
Worst	1.93901e+03	9.10384e-01	8.63931e+00	5.65485e+01	1.01292e+00	2.98540e+01	6.64827e+01
std	4.99989e+02	2.61596e-02	1.50796e+00	1.04660e+01	2.29410e-02	1.35451e+00	1.13389e+01
SR	1.0	1.0	1.0	1.0	0.0	0.64	0.0
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	5.05000e+01	1.39297e-03	7.30687e+04

Table A.9 Function Values of PMODE Achieved for 10D ($FES_{\max} = 20000 \times D$)

problem	C01	C02	C03	C04	C05	C06	C07
best	2.14611e-24	9.11866e-25	9.41726e+03	3.88033e+01	9.91336e-22	7.39402e+01	-1.60857e+02
median	5.68959e-24	4.44967e-24	1.71079e+04	8.32015e+01	3.52520e-20	1.78727e+02	-7.46900e+01
c	0 0 0	0 0 0	0 1 0	0 0 0	0 0 0	2 3 0	2 0 0
$\overline{v}$	0.00000e+00	0.00000e+00	1.16861e-02	0.00000e+00	0.00000e+00	5.24831e-01	7.57086e+01
Mean	8.63981e-24	6.48690e-24	5.94558e+04	8.13511e+01	6.08489e-20	2.16101e+02	-5.61896e+01
Worst	3.43159e-23	1.80413e-23	4.46449e+04	1.02793e+02	2.59659e-19	1.35417e+03	-9.13489e+01
std	7.75626e-24	4.92035e-24	8.34239e+04	1.39084e+01	6.96775e-20	2.51834e+02	5.98942e+01
SR	100	100	4	100	100	16	4
$\overline{vio}$	0.00000e+00	0.00000e+00	1.65910e-02	0.00000e+00	0.00000e+00	5.21616e-01	7.85737e+01
problem	C08	C09	C10	C11	C12	C13	C14
best	-1.34840e-03	-4.97525e-03	-5.09646e-04	-1.62228e+02	3.98790e+00	5.15504e-21	2.39559e+00
median	-1.34840e-03	-4.97525e-03	-5.09645e-04	-3.70137e+02	3.98863e+00	6.58037e-20	3.39761e+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	2.40353e+02	0.00000e+00	0.00000e+00	4.58844e-06
Mean	-1.34840e-03	-4.97525e-03	-5.09645e-04	-3.55967e+02	4.83886e+00	1.59463e-01	3.45367e+00
Worst	-1.34840e-03	-4.97525e-03	-5.09637e-04	-5.43096e+02	1.46065e+01	3.98658e+00	3.67062e+00
std	3.25752e-10	0.00000e+00	2.05524e-09	9.43249e+01	2.87977e+00	7.81207e-01	3.43063e-01
SR	100	100	100	0	100	100	48
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	2.37912e+02	0.00000e+00	0.00000e+00	5.32261e-05
problem	C15	C16	C17	C18	C19	C20	C21
best	5.49772e+00	5.18363e+01	9.97584e-01	1.00000e+01	0.00000e+00	1.47930e+00	3.98790e+00
median	1.17811e+01	5.65481e+01	9.14289e-01	3.62209e+01	0.00000e+00	1.81463e+00	3.98881e+00
c	0 0 1	0 0 1	1 1 0	0 1 0	1 0 0	0 0 0	0 0 0
$\overline{v}$	5.51058e-05	1.46131e-04	5.50540e+00	1.29979e-01	6.63359e+03	0.00000e+00	0.00000e+00
Mean	1.21578e+01	6.15751e+01	9.80767e-01	4.10083e+01	0.00000e+00	1.81955e+00	5.46111e+00
Worst	8.63678e+00	5.65464e+01	1.00244e+00	5.17217e+01	0.00000e+00	2.17692e+00	2.27853e+01
std	3.90569e+00	8.31210e+00	5.62635e-02	1.44737e+01	0.00000e+00	1.76447e-01	4.99452e+00
SR	40	12	0	4	0	100	100
$\overline{vio}$	3.24071e-04	1.99125e-04	5.50824e+00	2.93346e+00	6.63359e+03	0.00000e+00	0.00000e+00
problem	C22	C23	C24	C25	C26	C27	C28
best	4.75166e-21	2.40531e+00	5.49772e+00	5.65486e+01	1.51776e-01	6.28153e+01	0.00000e+00
median	1.04790e-19	3.81500e+00	1.49229e+01	4.39817e+01	1.00954e+00	1.37228e+01	0.00000e+00
c	0 0 0	0 0 0	0 0 1	0 0 1	1 0 1	0 2 0	1 0 0
$\overline{v}$	0.00000e+00	0.00000e+00	2.11608e-04	1.46534e-04	5.50411e+00	2.76359e-01	6.63359e+03
Mean	2.88192e-19	3.29617e+00	1.37914e+01	6.26432e+01	9.08549e-01	2.33382e+01	1.62221e+00
Worst	2.27974e-18	3.31445e+00	1.49199e+01	7.53947e+01	1.00543e+00	2.82342e+00	6.37173e+00
std	5.20942e-19	4.03648e-01	3.65075e+00	1.19573e+01	2.12119e-01	2.20528e+01	2.92278e+00
SR	100	52	20	4	0	0	0
$\overline{vio}$	0.00000e+00	6.82870e-05	4.23089e-04	2.31590e-04	5.46639e+00	2.48248e-01	6.63622e+03

Table A.10 Function Values of PMODE Achieved for 30D ($FES_{\max} = 20000 \times D$)

problem	C01	C02	C03	C04	C05	C06	C07
best	8.45246e+00	1.31645e+00	1.21533e+05	1.62387e+02	2.37034e+01	8.68151e+02	-7.78047e+01
median	6.22106e+01	6.14986e+01	4.46899e+05	3.66171e+02	1.22896e+02	2.23162e+03	-1.15271e+02
c	0 0 0	0 0 0	0 1 0	0 0 0	0 0 0	4 1 0	2 0 0
$\overline{v}$	0.00000e+00	0.00000e+00	1.56053e-02	0.00000e+00	0.00000e+00	3.04630e+00	1.40266e+03
Mean	1.01581e+02	8.79346e+01	5.24967e+05	3.25711e+02	1.76987e+02	2.75522e+03	-6.32051e+01
Worst	3.13217e+02	5.79893e+02	2.19709e+05	3.91920e+02	8.55390e+02	1.96412e+03	-1.85538e+02
std	8.75557e+01	1.15326e+02	4.36345e+05	7.78934e+01	1.76071e+02	1.34597e+03	8.25769e+01
SR	100	100	8	100	100	0	0
$\overline{vio}$	0.00000e+00	0.00000e+00	2.97523e-02	0.00000e+00	0.00000e+00	3.07096e+00	1.37837e+03
problem	C08	C09	C10	C11	C12	C13	C14
best	4.30883e+00	-2.66551e-03	2.93934e+00	-1.95483e+03	2.58898e+02	7.91928e+05	1.16005e+01
median	8.11294e+00	1.04331e+00	1.12307e+00	-2.39334e+03	8.71198e+02	4.03949e+06	1.73876e+01
c	2 0 0	0 0 0	2 0 0	1 0 0	1 0 0	1 0 0	2 0 0
$\overline{v}$	2.27078e+02	0.00000e+00	9.07387e+04	6.74605e+02	3.93445e+02	2.83340e+02	2.44359e+03
Mean	9.09124e+00	1.22155e+00	5.79186e+00	-1.58495e+03	9.71512e+02	1.01176e+07	1.69531e+01
Worst	1.59694e+01	5.12011e+00	8.62571e+00	-1.61106e+03	3.21002e+03	6.20270e+07	1.95298e+01
std	3.15098e+00	1.49632e+00	3.96077e+00	5.24220e+02	5.76453e+02	1.34472e+07	1.88152e+00
SR	0	60	0	0	0	0	0
$\overline{vio}$	2.63526e+02	4.79756e-02	1.04036e+05	8.29760e+02	4.46996e+02	3.57934e+02	2.53791e+03
problem	C15	C16	C17	C18	C19	C20	C21
best	1.17809e+01	2.01062e+02	1.02982e+00	2.55074e+02	4.15257e+01	8.22588e+00	2.89044e+02
median	1.49226e+01	2.08916e+02	1.15328e+00	8.36250e+02	4.02094e+01	9.25366e+00	7.53484e+02
c	0 0 0	0 0 1	2 0 0	1 0 0	1 0 0	0 0 0	1 0 0
$\overline{v}$	9.70150e-06	7.20124e-05	2.62061e+02	4.75943e+05	2.14626e+04	0.00000e+00	3.14773e+02
Mean	1.81898e+01	2.11052e+02	1.17571e+00	9.55141e+02	5.60410e+01	9.23266e+00	7.69190e+02
Worst	2.12045e+01	2.01059e+02	1.45716e+00	2.05000e+03	8.07226e+01	1.00181e+01	1.32907e+03
std	2.73591e+00	1.23599e+01	1.00234e-01	4.97848e+02	1.26095e+01	4.38197e-01	2.77606e+02
SR	48	16	0	0	0	100	0
$\overline{vio}$	1.84991e-04	1.85297e-04	3.06938e+02	9.84241e+05	2.14638e+04	0.00000e+00	3.45572e+02
problem	C22	C23	C24	C25	C26	C27	C28
best	2.62086e+06	1.90652e+01	1.49225e+01	1.90066e+02	1.21225e+00	9.75533e+02	6.22618e+01
median	2.21334e+07	2.03930e+01	1.80597e+01	2.01061e+02	1.59385e+00	5.32941e+03	7.70427e+01
c	2 0 0	2 0 0	0 0 1	0 0 1	2 0 0	1 0 0	1 0 0
$\overline{v}$	6.24786e+02	1.24452e+04	3.08672e-03	1.07590e-04	1.14320e+03	4.62537e+05	2.14813e+04
Mean	4.79527e+07	2.04905e+01	1.89402e+01	2.36413e+02	1.65233e+00	3.53610e+03	9.64144e+01
Worst	3.09761e+08	2.13998e+01	2.61751e+01	3.56724e+02	2.35945e+00	7.53113e+03	1.69217e+02
std	6.33025e+07	7.08163e-01	2.51595e+00	3.83776e+01	3.21242e-01	1.96683e+03	2.58908e+01
SR	0	0	36	28	0	0	0
$\overline{vio}$	7.67227e+02	1.37688e+04	1.47415e+02	2.75679e+02	1.26016e+03	9.43760e+05	2.14813e+04

Table A.11 Function Values of PMODE Achieved for 50D ($FES_{\max} = 20000 \times D$)

problem	C01	C02	C03	C04	C05	C06	C07
best	2.28064e+03	1.98396e+03	1.93920e+05	3.55349e+02	5.97421e+03	1.75834e+03	-5.36955e+02
median	4.52311e+03	3.66519e+03	8.18781e+06	4.77077e+02	1.92933e+04	8.24012e+03	-1.18883e+02
c	0 0 0	0 0 0	0 1 0	0 0 0	0 0 0	3 1 0	2 0 0
$\bar{v}$	0.00000e+00	0.00000e+00	2.11433e-02	0.00000e+00	0.00000e+00	3.67257e+00	2.89321e+03
Mean	4.57826e+03	3.86686e+03	2.56010e+06	5.28980e+02	2.07483e+04	5.21220e+03	-1.52570e+02
Worst	6.58848e+03	5.80477e+03	1.80087e+06	6.71820e+02	6.05995e+04	4.84112e+03	-2.33206e+02
std	9.65902e+02	9.24684e+02	3.45818e+06	1.19169e+02	1.16951e+04	2.34983e+03	1.35106e+02
SR	100	100	8	100	100	16	0
$\overline{vio}$	0.00000e+00	0.00000e+00	4.21099e-02	0.00000e+00	0.00000e+00	3.45598e+00	2.82329e+03
problem	C08	C09	C10	C11	C12	C13	C14
best	1.02717e+01	5.74202e+00	1.74030e+01	-1.47463e+03	3.75936e+03	4.56714e+07	1.83443e+01
median	1.47237e+01	6.01563e+00	2.35321e+01	-1.04457e+03	6.65909e+03	1.81715e+08	1.96039e+01
c	2 0 0	1 0 0	2 0 0	1 0 0	1 0 0	2 0 0	2 0 0
$\bar{v}$	1.36576e+03	1.25530e+02	2.42791e+06	4.02216e+03	3.25068e+03	2.07518e+03	1.03922e+04
Mean	1.67054e+01	7.26445e+00	1.77968e+01	-1.28194e+03	7.21543e+03	1.91988e+08	1.96436e+01
Worst	2.12763e+01	9.49550e+00	1.47180e+01	-9.95573e+02	1.61630e+04	6.34024e+08	2.09103e+01
std	3.86802e+00	1.35384e+00	5.56564e+00	3.36372e+02	2.58950e+03	1.28596e+08	6.79060e-01
SR	0	0	0	0	0	0	0
$\overline{vio}$	1.34029e+03	2.09972e+02	2.50252e+06	4.14513e+03	3.50114e+03	2.07567e+03	1.00248e+04
problem	C15	C16	C17	C18	C19	C20	C21
best	1.80641e+01	4.08407e+02	1.81562e+00	4.67375e+03	1.25353e+02	1.62035e+01	3.46898e+03
median	2.23511e+01	4.03140e+02	2.69752e+00	1.04115e+04	1.35750e+02	1.75255e+01	7.01829e+03
c	2 0 0	1 1 0	2 0 0	2 0 0	1 0 0	0 0 0	1 0 0
$\bar{v}$	7.95359e+02	5.63050e+02	3.32053e+03	5.38874e+07	3.63004e+04	0.00000e+00	3.40038e+03
Mean	2.41366e+01	4.36228e+02	2.61806e+00	8.17305e+03	1.28788e+02	1.74999e+01	6.90454e+03
Worst	2.81105e+01	6.32413e+02	3.56693e+00	9.97025e+03	1.40672e+02	1.87708e+01	1.08995e+04
std	5.05616e+00	6.03201e+01	3.83810e-01	2.02373e+03	1.62059e+01	5.63727e-01	1.86784e+03
SR	8	4	0	0	0	100	0
$\overline{vio}$	9.94883e+02	8.85680e+02	3.16162e+03	5.42517e+07	3.62998e+04	0.00000e+00	3.34437e+03
problem	C22	C23	C24	C25	C26	C27	C28
best	6.58082e+08	2.09123e+01	3.70265e+01	5.58189e+02	4.31932e+00	1.50354e+04	1.72173e+02
median	2.09933e+09	2.14984e+01	4.47071e+01	7.44848e+02	6.25467e+00	2.78702e+04	2.02544e+02
c	2 0 0	2 0 0	2 0 0	2 0 0	2 0 0	2 0 0	1 0 0
$\bar{v}$	7.26657e+03	4.93000e+04	9.13401e+03	7.52269e+03	1.04348e+04	4.85357e+07	3.63275e+04
Mean	2.02531e+09	2.13180e+01	4.55004e+01	7.72193e+02	6.51825e+00	2.64735e+04	2.12540e+02
Worst	4.14934e+09	2.14182e+01	7.05414e+01	9.87128e+02	9.74742e+00	5.25566e+04	3.08671e+02
std	7.91057e+08	2.26072e-01	9.58023e+00	1.16580e+02	1.46486e+00	7.94598e+03	3.13167e+01
SR	0	0	0	0	0	0	0
$\overline{vio}$	6.99710e+03	4.94889e+04	9.41546e+03	7.98474e+03	1.09620e+04	5.85456e+07	3.63287e+04

Table A.12 Function Values of PMODE Achieved for 100D ($FES_{\max} = 20000 \times D$)

problem	C01	C02	C03	C04	C05	C06	C07
best	2.19725e+04	3.08252e+04	8.81317e+05	8.80889e+02	1.74982e+05	4.28539e+03	-5.36330e+01
median	3.48956e+04	4.21954e+04	1.64864e+07	1.05770e+03	3.53987e+05	1.18699e+04	-3.22051e+02
c	0 0 0	0 0 0	0 1 0	0 0 0	0 0 0	5 0 0	2 0 0
$\overline{v}$	0.00000e+00	0.00000e+00	4.56214e-02	0.00000e+00	0.00000e+00	6.78371e+00	6.86213e+03
Mean	3.48826e+04	4.22447e+04	4.43083e+06	1.07373e+03	3.44449e+05	1.10830e+04	-1.49642e+02
Worst	4.50649e+04	5.53684e+04	1.44167e+07	1.40462e+03	5.35239e+05	5.70569e+03	-9.51640e+01
std	6.46300e+03	6.58780e+03	3.88766e+06	1.34842e+02	8.99856e+04	4.10730e+03	1.92704e+02
SR	100	100	8	100	100	8	0
$\overline{vio}$	0.00000e+00	0.00000e+00	6.03512e-02	0.00000e+00	0.00000e+00	6.29700e+00	6.90888e+03
problem	C08	C09	C10	C11	C12	C13	C14
best	2.07891e+01	1.02028e+01	3.48310e+01	-1.42141e+03	3.13441e+04	1.56586e+09	2.07419e+01
median	2.76848e+01	8.33249e+00	3.71183e+01	-9.50141e+02	3.75117e+04	3.12784e+09	2.12499e+01
c	2 0 0	1 0 0	2 0 0	1 0 0	1 0 0	1 0 0	2 0 0
$\overline{v}$	7.53044e+03	6.19357e+03	4.52030e+07	1.53424e+04	1.84512e+04	1.19932e+04	5.14897e+04
Mean	2.57686e+01	9.37371e+00	3.81451e+01	-1.44764e+03	3.79194e+04	3.21154e+09	2.12221e+01
Worst	3.55284e+01	9.85473e+00	3.92237e+01	-1.87066e+03	4.90522e+04	5.25677e+09	2.12777e+01
std	4.91663e+00	8.36433e-01	5.42529e+00	2.32202e+02	4.38064e+03	9.73355e+08	1.97632e-01
SR	0	0	0	0	0	0	0
$\overline{vio}$	7.58180e+03	6.67623e+03	4.67914e+07	1.58347e+04	1.86740e+04	1.22397e+04	5.20788e+04
problem	C15	C16	C17	C18	C19	C20	C21
best	3.77923e+01	1.29718e+03	6.87595e+00	3.08885e+04	2.94193e+02	3.74545e+01	2.74753e+04
median	3.78513e+01	1.49338e+03	1.01553e+01	3.87156e+04	3.56124e+02	3.90835e+01	3.62471e+04
c	2 0 0	2 0 0	2 0 0	2 0 0	1 0 0	0 0 0	1 0 0
$\overline{v}$	1.52068e+04	1.30303e+04	1.81611e+04	1.06212e+09	7.33963e+04	0.00000e+00	1.78372e+04
Mean	4.32079e+01	1.50611e+03	9.95507e+00	4.00109e+04	3.50146e+02	3.90733e+01	3.58695e+04
Worst	5.12716e+01	1.65646e+03	1.20672e+01	4.21590e+04	4.24377e+02	4.02646e+01	4.50014e+04
std	3.66450e+00	1.00916e+02	1.27062e+00	3.97574e+03	2.95260e+01	7.19816e-01	4.72648e+03
SR	0	0	0	0	0	100	0
$\overline{vio}$	1.49949e+04	1.33886e+04	1.77606e+04	1.03625e+09	7.33968e+04	0.00000e+00	1.76425e+04
problem	C22	C23	C24	C25	C26	C27	C28
best	1.61392e+10	2.14640e+01	6.86652e+01	2.11668e+03	2.04175e+01	9.99239e+04	3.98592e+02
median	3.19578e+10	2.15640e+01	7.90501e+01	2.58110e+03	2.85646e+01	1.39773e+05	5.50789e+02
c	2 0 0	2 0 0	2 0 0	2 0 0	2 0 0	2 0 0	1 0 0
$\overline{v}$	3.76342e+04	1.88806e+05	5.61628e+04	4.98537e+04	5.49798e+04	1.06865e+09	7.34298e+04
Mean	3.42906e+10	2.14957e+01	7.90906e+01	2.52591e+03	2.92427e+01	1.26691e+05	4.94682e+02
Worst	6.23980e+10	2.15599e+01	8.25268e+01	3.05519e+03	3.75458e+01	1.25816e+05	6.46487e+02
std	1.15359e+10	1.46944e-01	4.83272e+00	2.06318e+02	4.91052e+00	1.78593e+04	5.21990e+01
SR	0	0	0	0	0	0	0
$\overline{vio}$	3.82006e+04	1.85570e+05	5.70819e+04	4.89022e+04	5.63358e+04	1.09278e+09	7.34291e+04

Table A.13 Function Values of HECO-PDE Achieved for 10D ($FES_{\max} = 20000 \times D$)

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	-2.58748e+02
median	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	-1.17310e+03
c	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 2
$\overline{v}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	1.91686e-03
Mean	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	-7.46594e+02
Worst	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	-4.48664e+02
std	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	7.86557e+02
SR	100	100	100	100	100	100	4
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	2.31995e-03
problem	C08	C09	C10	C11	C12	C13	C14
best	-1.34840e-03	-4.97525e-03	-5.09647e-04	-1.68819e-01	3.98790e+00	0.00000e+00	2.37633e+00
median	-1.34840e-03	-4.97525e-03	-5.09647e-04	-1.68792e-01	3.98883e+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	-1.68092e-01	3.98918e+00	0.00000e+00	2.37633e+00
Worst	-1.34840e-03	-4.97525e-03	-5.09647e-04	-1.58782e-01	3.99096e+00	0.00000e+00	2.37633e+00
std	3.48093e-16	0.00000e+00	3.31451e-15	2.15091e-03	1.08240e-03	0.00000e+00	0.00000e+00
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	C15	C16	C17	C18	C19	C20	C21
best	2.35612e+00	0.00000e+00	1.08553e-02	1.00000e+01	0.00000e+00	1.04005e-01	3.98796e+00
median	2.35612e+00	0.00000e+00	1.08553e-02	5.04203e+01	0.00000e+00	3.68418e-01	3.98991e+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	0.00000e+00	1.08553e-02	3.91026e+01	0.00000e+00	3.58433e-01	3.98967e+00
Worst	2.35612e+00	0.00000e+00	1.08553e-02	5.04203e+01	0.00000e+00	5.20013e-01	3.99246e+00
std	1.11995e-15	0.00000e+00	1.08780e-16	1.81487e+01	0.00000e+00	1.07653e-01	1.26111e-03
SR	100	100	0	100	0	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	4.50000e+00	0.00000e+00	6.63359e+03	0.00000e+00	0.00000e+00
problem	C22	C23	C24	C25	C26	C27	C28
best	3.48642e-27	2.37633e+00	2.35612e+00	0.00000e+00	1.08553e-02	9.05515e+01	0.00000e+00
median	3.48642e-27	2.37633e+00	2.35612e+00	0.00000e+00	1.08553e-02	9.57215e+01	0.00000e+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	4.50000e+00	0.00000e+00	6.63359e+03
Mean	3.49026e-27	2.37633e+00	2.35612e+00	0.00000e+00	7.37529e-02	9.40671e+01	0.00000e+00
Worst	3.51844e-27	2.37633e+00	2.35612e+00	0.00000e+00	7.10694e-01	9.57217e+01	0.00000e+00
std	1.04062e-29	5.21413e-07	5.18534e-08	0.00000e+00	1.75700e-01	2.41167e+00	0.00000e+00
SR	100	100	100	100	0	100	0
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	4.50000e+00	0.00000e+00	6.63359e+03

Table A.14 Function Values of HECO-PDE Achieved for 30D ($FES_{\max} = 20000 \times D$)

problem	C01	C02	C03	C04	C05	C06	C07
best	0.00000e+00	0.00000e+00	0.00000e+00	1.35728e+01	0.00000e+00	0.00000e+00	-2.45791e+03
median	9.33075e-30	1.17959e-29	3.49071e-29	1.35728e+01	0.00000e+00	0.00000e+00	-2.68829e+03
c	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 2
$\overline{v}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	7.93332e-04
Mean	1.59799e-29	1.96421e-29	3.95614e-29	1.35728e+01	0.00000e+00	0.00000e+00	-1.94723e+03
Worst	7.51020e-29	6.70162e-29	1.34895e-28	1.35728e+01	0.00000e+00	0.00000e+00	-2.51299e+03
std	2.11761e-29	1.97404e-29	3.35121e-29	1.28144e-14	0.00000e+00	0.00000e+00	9.19644e+02
SR	100	100	100	100	100	100	0
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	1.05592e-03
problem	C08	C09	C10	C11	C12	C13	C14
best	-2.83981e-04	-2.66551e-03	-1.02842e-04	-9.24713e-01	3.98253e+00	0.00000e+00	1.40852e+00
median	-2.83981e-04	-2.66551e-03	-1.02842e-04	-8.77334e-01	3.98257e+00	0.00000e+00	1.40852e+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	-2.83981e-04	-2.66551e-03	-1.02842e-04	-1.87293e+02	3.98269e+00	0.00000e+00	1.40852e+00
Worst	-2.83981e-04	-2.66551e-03	-1.02840e-04	-1.25684e+03	3.98341e+00	0.00000e+00	1.40852e+00
std	5.00769e-11	3.79326e-16	2.41501e-10	3.71804e+02	2.14922e-04	0.00000e+00	7.00761e-16
SR	100	100	100	64	100	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	1.26005e+01	0.00000e+00	0.00000e+00	0.00000e+00
problem	C15	C16	C17	C18	C19	C20	C21
best	2.35612e+00	0.00000e+00	3.08555e-02	3.00000e+01	0.00000e+00	2.50134e+00	3.98253e+00
median	2.35612e+00	0.00000e+00	1.44943e+00	5.34725e+01	0.00000e+00	2.84742e+00	3.98262e+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	1.45000e+01	0.00000e+00	2.13749e+04	0.00000e+00	0.00000e+00
Mean	2.35612e+00	0.00000e+00	6.51135e-01	4.85514e+01	0.00000e+00	2.87611e+00	3.98269e+00
Worst	2.35612e+00	0.00000e+00	9.51999e-01	5.82173e+01	0.00000e+00	3.19426e+00	3.98327e+00
std	1.13742e-15	0.00000e+00	4.82474e-01	8.31285e+00	0.00000e+00	1.88661e-01	1.94653e-04
SR	100	100	0	100	0	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	1.49400e+01	0.00000e+00	2.13749e+04	0.00000e+00	0.00000e+00
problem	C22	C23	C24	C25	C26	C27	C28
best	3.59077e-14	1.40852e+00	2.35612e+00	0.00000e+00	6.11945e-01	2.08760e+02	0.00000e+00
median	7.91069e-10	1.40852e+00	2.35612e+00	0.00000e+00	8.53762e-01	2.51342e+02	3.33028e-06
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	2.13749e+04
Mean	4.13459e-08	1.40853e+00	2.35612e+00	0.00000e+00	8.41942e-01	2.36013e+02	4.23646e-06
Worst	7.52530e-07	1.40858e+00	2.35612e+00	0.00000e+00	1.01187e+00	2.51347e+02	1.77740e-05
std	1.48613e-07	1.46320e-05	8.05949e-08	0.00000e+00	8.41683e-02	2.04397e+01	3.62229e-06
SR	100	100	100	100	0	100	0
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	1.55000e+01	0.00000e+00	2.13749e+04

Table A.15 Function Values of HECO-PDE Achieved for 50D ($FES_{\max} = 20000 \times D$)

problem	C01	C02	C03	C04	C05	C06	C07
best	1.04672e-28	4.57539e-29	1.81832e-28	1.35728e+01	0.00000e+00	0.00000e+00	-3.78512e+03
median	2.08259e-28	2.17442e-28	3.30138e-28	1.35728e+01	9.12120e-30	1.41843e+02	-8.31534e+02
c	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 2
$\overline{v}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	5.66334e-04
Mean	2.51258e-28	2.17115e-28	4.12290e-28	1.35728e+01	4.52846e-29	1.16974e+02	-2.79070e+03
Worst	5.16815e-28	4.33787e-28	8.70114e-28	1.35728e+01	4.25245e-28	1.66179e+02	-3.52267e+03
std	1.21265e-28	8.04570e-29	1.98135e-28	2.10181e-15	8.97841e-29	9.83836e+01	9.79449e+02
SR	100	100	100	100	100	76	12
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	2.42773e-04	8.24621e-04
problem	C08	C09	C10	C11	C12	C13	C14
best	-1.34466e-04	-2.03709e-03	-4.82659e-05	-3.11531e+02	3.98145e+00	0.00000e+00	1.09995e+00
median	-1.33456e-04	-2.03709e-03	-4.82628e-05	-1.62884e+03	3.98150e+00	1.33101e-27	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	3.29899e+01	0.00000e+00	0.00000e+00	0.00000e+00
Mean	-1.31402e-04	-2.03709e-03	-4.82606e-05	-1.54196e+03	4.22857e+00	1.59465e-01	1.10073e+00
Worst	-1.12487e-04	-2.03709e-03	-4.82487e-05	-2.67003e+03	7.06997e+00	3.98662e+00	1.11946e+00
std	4.80757e-06	1.40141e-15	4.96661e-09	6.80331e+02	8.37725e-01	7.81216e-01	3.82241e-03
SR	100	100	100	0	100	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	3.55113e+01	0.00000e+00	0.00000e+00	0.00000e+00
problem	C15	C16	C17	C18	C19	C20	C21
best	2.35612e+00	0.00000e+00	6.04826e-02	4.42174e+01	0.00000e+00	5.15997e+00	3.98145e+00
median	2.35612e+00	0.00000e+00	5.45231e-01	4.61632e+01	0.00000e+00	6.24310e+00	3.98149e+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	0.00000e+00	6.17102e-01	4.76107e+01	0.00000e+00	6.10145e+00	4.10501e+00
Worst	2.35612e+00	0.00000e+00	9.20881e-01	6.22173e+01	0.00000e+00	6.92786e+00	7.06871e+00
std	1.55368e-15	0.00000e+00	3.32818e-01	3.65297e+00	0.00000e+00	4.34170e-01	6.04962e-01
SR	100	100	0	100	0	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	2.52200e+01	0.00000e+00	3.61162e+04	0.00000e+00	0.00000e+00
problem	C22	C23	C24	C25	C26	C27	C28
best	5.13492e-01	1.09995e+00	2.35612e+00	0.00000e+00	8.01254e-01	2.47635e+02	1.65653e-05
median	1.40918e+01	1.09996e+00	2.35612e+00	0.00000e+00	9.36281e-01	2.64205e+02	3.28074e+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	3.61248e+04
Mean	1.34654e+01	1.09997e+00	2.35612e+00	3.02025e-14	9.31995e-01	2.58242e+02	2.76288e+00
Worst	1.60992e+01	1.10009e+00	2.35612e+00	7.55062e-13	1.01406e+00	2.64215e+02	7.12744e+00
std	2.82924e+00	3.13374e-05	1.72250e-08	1.47961e-13	5.29458e-02	7.95311e+00	2.53458e+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	3.61255e+04

Table A.16 Function Values of HECO-PDE Achieved for 100D ($FES_{\max} = 20000 \times D$)

problem	C01	C02	C03	C04	C05	C06	C07
best	4.74860e-24	2.08968e-23	1.09781e-23	1.35728e+01	1.24693e-16	5.85856e+02	-6.04093e+03
median	2.96855e-22	7.65701e-22	2.63175e-22	1.35728e+01	5.60537e-15	9.10968e+02	-5.12059e+03
c	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 2
$\overline{v}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	3.43542e-04
Mean	7.92774e-22	6.12831e-21	1.04964e-21	1.36666e+01	1.13652e-13	8.17227e+02	-4.19316e+03
Worst	4.80812e-21	4.10046e-20	7.98385e-21	1.59192e+01	1.63788e-12	8.24776e+02	-5.64890e+03
std	1.10948e-21	1.11876e-20	1.84224e-21	4.59810e-01	3.33214e-13	1.19466e+02	1.20840e+03
SR	100	100	100	100	100	64	20
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	1.07363e-03	4.02313e-04
problem	C08	C09	C10	C11	C12	C13	C14
best	9.89426e-05	0.00000e+00	-1.69783e-05	-6.28804e+03	1.88621e+01	3.32042e+01	7.84202e-01
median	4.62686e-04	0.00000e+00	-1.66940e-05	-6.98033e+03	3.16345e+01	3.32046e+01	7.86865e-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	8.12562e+01	0.00000e+00	0.00000e+00	0.00000e+00
Mean	4.54336e-04	0.00000e+00	-1.67005e-05	-6.83288e+03	3.11306e+01	3.36826e+01	7.96313e-01
Worst	9.65755e-04	0.00000e+00	-1.62289e-05	-6.23668e+03	3.18122e+01	3.71880e+01	8.38008e-01
std	1.64588e-04	0.00000e+00	2.14329e-07	3.27748e+02	2.50488e+00	1.29435e+00	1.52042e-02
SR	100	100	100	0	100	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	9.43117e+01	0.00000e+00	0.00000e+00	0.00000e+00
problem	C15	C16	C17	C18	C19	C20	C21
best	2.35612e+00	0.00000e+00	2.59387e-01	4.62791e+01	0.00000e+00	1.47876e+01	1.20688e+01
median	2.35612e+00	0.00000e+00	8.79248e-01	1.30626e+02	0.00000e+00	1.58000e+01	3.15911e+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	2.11765e+01	7.29695e+04	0.00000e+00	0.00000e+00
Mean	2.35612e+00	0.00000e+00	8.01474e-01	8.51306e+01	1.68587e-09	1.58580e+01	3.03467e+01
Worst	2.35612e+00	0.00000e+00	1.01669e+00	4.48688e+01	4.21468e-08	1.72009e+01	3.21128e+01
std	1.37309e-15	0.00000e+00	2.12703e-01	4.01418e+01	8.25906e-09	6.38805e-01	4.49296e+00
SR	100	100	0	32	0	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	5.05000e+01	1.18429e+01	7.29695e+04	0.00000e+00	0.00000e+00
problem	C22	C23	C24	C25	C26	C27	C28
best	7.82296e+01	7.84209e-01	2.35612e+00	1.57066e+00	1.00167e+00	2.84751e+02	2.38846e+01
median	2.12380e+04	7.84225e-01	2.35612e+00	1.57066e+00	1.04543e+00	2.84829e+02	5.38774e+01
c	1 0 0	0 0 0	0 0 0	0 0 0	1 0 0	0 0 0	1 0 0
$\overline{v}$	1.54193e+01	0.00000e+00	0.00000e+00	0.00000e+00	5.05000e+01	0.00000e+00	7.30594e+04
Mean	3.23001e+04	7.84396e-01	2.35612e+00	5.27774e+00	1.05214e+00	2.99755e+02	5.15351e+01
Worst	8.20487e+02	7.87071e-01	2.35612e+00	3.29866e+01	1.09745e+00	3.18890e+02	9.74830e+01
std	8.81380e+04	5.65603e-04	3.39107e-09	6.63285e+00	2.54681e-02	1.68980e+01	1.51348e+01
SR	48	100	100	100	0	100	0
$\overline{vio}$	3.65205e+01	0.00000e+00	0.00000e+00	0.00000e+00	5.05000e+01	0.00000e+00	7.30575e+04

Table A.17 Function Values of HECO-DEm Achieved for 10D ($FES_{\max} = 20000 \times D$)

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	-1.81458e+03
Median	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	-2.82915e+03
c	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 2
$\overline{v}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	6.15401e-03
mean	0.00000e+00	5.04871e-31	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	-3.34819e+03
Worst	0.00000e+00	1.26218e-29	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	-1.17961e+03
std	0.00000e+00	2.47335e-30	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	2.05910e+03
SR	100	100	100	100	100	100	4
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	1.04202e-02
Problem	C8	C9	C10	C11	C12	C13	C14
Best	-1.34840e-03	-4.97525e-03	-5.09647e-04	-8.40903e+00	3.98830e+00	0.00000e+00	2.37633e+00
Median	-1.34840e-03	-4.97525e-03	-5.09647e-04	-3.17864e+02	3.98906e+00	0.00000e+00	2.37633e+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	1.97405e+02	0.00000e+00	0.00000e+00	0.00000e+00
mean	-1.34840e-03	-4.97522e-03	-5.09646e-04	-2.77575e+02	3.98911e+00	0.00000e+00	2.37741e+00
Worst	-1.34840e-03	-4.97471e-03	-5.09632e-04	-7.37789e+02	3.98995e+00	0.00000e+00	2.39926e+00
std	5.30054e-10	1.05983e-07	3.19858e-09	2.35591e+02	4.81542e-04	0.00000e+00	4.49574e-03
SR	100	100	100	0	100	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	2.98226e+02	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	1.08446e-01	3.98823e+00
Median	2.35612e+00	0.00000e+00	1.08553e-02	4.14306e+01	0.00000e+00	2.09644e-01	3.98901e+00
c	0 0 0	0 0 0	1 0 0	0 0 1	1 0 0	0 0 0	0 0 0
$\overline{v}$	0.00000e+00	0.00000e+00	4.50000e+00	4.51116e-04	6.63359e+03	0.00000e+00	0.00000e+00
mean	2.35613e+00	0.00000e+00	1.08553e-02	4.70807e+01	3.21236e-08	2.15090e-01	3.98911e+00
Worst	2.35615e+00	0.00000e+00	1.08553e-02	1.27908e+02	6.59566e-07	3.08327e-01	3.98993e+00
std	5.85373e-06	0.00000e+00	2.19061e-11	1.90103e+01	1.29748e-07	4.94361e-02	3.93591e-04
SR	100	100	0	4	0	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	4.50000e+00	6.64445e-01	6.63359e+03	0.00000e+00	0.00000e+00
Problem	C22	C23	C24	C25	C26	C27	C28
Best	3.48642e-27	2.37645e+00	2.35612e+00	0.00000e+00	1.08665e-02	6.30390e+01	2.27431e-06
Median	3.48642e-27	2.37760e+00	2.35613e+00	0.00000e+00	1.18308e-02	9.57559e+01	2.06865e-04
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	3.68203e-27	2.37770e+00	2.35613e+00	0.00000e+00	2.23054e-01	9.27724e+01	7.63192e-04
Worst	5.83550e-27	2.38249e+00	2.35613e+00	0.00000e+00	2.39788e-01	9.15890e+01	8.02173e-03
std	6.35166e-28	1.15371e-03	2.98231e-06	0.00000e+00	4.18028e-01	7.66527e+00	1.63246e-03
SR	100	100	100	100	0	96	0
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	4.66008e+00	3.16360e-07	6.63359e+03

Table A.18 Function Values of HECO-DEm Achieved for 30D ($FES_{\max} = 20000 \times D$)

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	-3.60727e+03
Median	1.67633e-29	1.73549e-29	2.52435e-29	0.00000e+00	0.00000e+00	0.00000e+00	-5.42620e+03
c	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 2
$\overline{v}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	1.68435e-03
mean	1.98315e-29	2.07663e-29	3.61535e-29	0.00000e+00	0.00000e+00	0.00000e+00	-4.54869e+03
Worst	4.96613e-29	6.61287e-29	1.38051e-28	0.00000e+00	0.00000e+00	0.00000e+00	-5.69223e+02
std	1.27343e-29	1.52493e-29	3.49185e-29	0.00000e+00	0.00000e+00	0.00000e+00	2.70823e+03
SR	100	100	100	100	100	100	4
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	2.29572e-03
Problem	C8	C9	C10	C11	C12	C13	C14
Best	-2.83981e-04	-2.66551e-03	-1.02842e-04	-1.57668e+03	3.98295e+00	0.00000e+00	1.40852e+00
Median	-2.83945e-04	-2.66551e-03	-1.02840e-04	-2.15506e+03	3.98312e+00	0.00000e+00	1.40864e+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	3.33707e+02	0.00000e+00	0.00000e+00	0.00000e+00
mean	-2.80345e-04	-2.66551e-03	-1.02834e-04	-2.06818e+03	3.98316e+00	0.00000e+00	1.42079e+00
Worst	-2.65088e-04	-2.66551e-03	-1.02734e-04	-2.51607e+03	3.98347e+00	0.00000e+00	1.48320e+00
std	6.42147e-06	6.41459e-16	2.12285e-08	3.30259e+02	1.24240e-04	0.00000e+00	1.97418e-02
SR	100	100	100	0	100	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	3.67494e+02	0.00000e+00	0.00000e+00	0.00000e+00
Problem	C15	C16	C17	C18	C19	C20	C21
Best	2.35612e+00	0.00000e+00	3.08555e-02	4.22186e+01	0.00000e+00	1.62688e+00	3.98301e+00
Median	2.35612e+00	0.00000e+00	3.37703e-01	5.04466e+01	0.00000e+00	1.99928e+00	3.98315e+00
c	0 0 0	0 0 0	1 0 0	0 0 1	1 0 0	0 0 0	0 0 0
$\overline{v}$	0.00000e+00	0.00000e+00	1.45000e+01	9.91424e-05	2.13749e+04	0.00000e+00	0.00000e+00
mean	2.35612e+00	0.00000e+00	5.25700e-01	5.03142e+01	2.22417e-06	1.98773e+00	3.98315e+00
Worst	2.35612e+00	0.00000e+00	2.72938e-01	2.78808e+01	2.61852e-05	2.25914e+00	3.98334e+00
std	1.23390e-15	0.00000e+00	5.65156e-01	1.51319e+01	5.29215e-06	1.46058e-01	8.94603e-05
SR	100	100	0	40	0	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	1.45400e+01	1.83117e+00	2.13749e+04	0.00000e+00	0.00000e+00
Problem	C22	C23	C24	C25	C26	C27	C28
Best	3.43758e-10	1.40866e+00	2.35612e+00	0.00000e+00	4.03349e-01	2.08792e+02	7.40860e-05
Median	3.12399e-08	1.40882e+00	2.35612e+00	0.00000e+00	5.31595e-01	2.51410e+02	-9.20630e-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	1.55000e+01	0.00000e+00	2.13759e+04
mean	8.26259e-06	1.40887e+00	2.35612e+00	0.00000e+00	5.48569e-01	2.40770e+02	1.57359e+00
Worst	1.83184e-04	1.40944e+00	2.35613e+00	0.00000e+00	9.65321e-01	2.51100e+02	6.32577e+00
std	3.57607e-05	1.87882e-04	7.92192e-07	0.00000e+00	2.58519e-01	1.80539e+01	2.30750e+00
SR	100	100	100	100	0	96	0
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	1.54200e+01	1.87457e-06	2.13803e+04

Table A.19 Function Values of HECO-DEm Achieved for 50D ($FES_{\max} = 20000 \times D$)

problem	C01	C02	C03	C04	C05	C06	C07
Best	1.54617e-28	1.74646e-28	2.29953e-28	0.00000e+00	0.00000e+00	0.00000e+00	-8.30066e+03
Median	3.13178e-28	2.90017e-28	4.37029e-28	0.00000e+00	0.00000e+00	0.00000e+00	-5.97795e+03
c	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 2
$\overline{v}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	5.73577e-04
mean	3.57986e-28	3.00664e-28	5.07364e-28	0.00000e+00	3.29675e-29	0.00000e+00	-7.09549e+03
Worst	1.15283e-27	4.65428e-28	1.08192e-27	0.00000e+00	2.43129e-28	0.00000e+00	-6.43523e+03
std	2.08734e-28	7.55438e-29	2.15736e-28	0.00000e+00	6.45782e-29	0.00000e+00	1.38069e+03
SR	100	100	100	100	100	100	16
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	1.05240e-03
Problem	C8	C9	C10	C11	C12	C13	C14
Best	-1.30636e-04	-2.03709e-03	-4.82623e-05	-3.38834e+03	3.98180e+00	0.00000e+00	1.09995e+00
Median	-7.40684e-05	-2.03709e-03	-4.81854e-05	-4.60226e+03	3.98376e+00	0.00000e+00	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	7.04581e+02	0.00000e+00	0.00000e+00	0.00000e+00
mean	7.59856e-04	-2.03709e-03	-4.81397e-05	-4.43649e+03	3.98361e+00	2.31226e-27	1.10205e+00
Worst	1.56964e-02	-2.03709e-03	-4.78305e-05	-5.05406e+03	3.98529e+00	1.68660e-26	1.14546e+00
std	3.07632e-03	3.18067e-11	1.21443e-07	3.81468e+02	8.68746e-04	3.74247e-27	8.96278e-03
SR	96	100	100	0	100	100	100
$\overline{vio}$	1.43881e-06	0.00000e+00	0.00000e+00	8.50673e+02	0.00000e+00	0.00000e+00	0.00000e+00
Problem	C15	C16	C17	C18	C19	C20	C21
Best	2.35612e+00	0.00000e+00	5.08555e-02	4.61632e+01	0.00000e+00	4.26739e+00	3.98172e+00
Median	2.35612e+00	0.00000e+00	1.28612e+00	5.28819e+01	0.00000e+00	4.66553e+00	3.98337e+00
c	0 0 0	0 0 0	1 0 0	0 1 0	1 0 0	0 0 0	0 0 0
$\overline{v}$	0.00000e+00	0.00000e+00	2.45000e+01	1.75961e-01	3.61162e+04	0.00000e+00	0.00000e+00
mean	2.35612e+00	0.00000e+00	7.63255e-01	7.64020e+01	7.99304e-06	4.63586e+00	3.98328e+00
Worst	2.35612e+00	0.00000e+00	1.02051e+00	7.12913e+01	1.13945e-04	4.89938e+00	3.98514e+00
std	1.15463e-15	0.00000e+00	4.99949e-01	3.08343e+01	2.44104e-05	1.62296e-01	1.04513e-03
SR	100	100	0	16	0	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	2.49000e+01	9.24021e+00	3.61162e+04	0.00000e+00	0.00000e+00
Problem	C22	C23	C24	C25	C26	C27	C28
Best	1.33283e+01	1.10002e+00	2.35612e+00	0.00000e+00	4.68759e-01	2.47726e+02	1.51589e+00
Median	1.50613e+01	1.10008e+00	2.35612e+00	0.00000e+00	7.93231e-01	2.47780e+02	9.12297e+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	3.61294e+04
mean	1.50437e+01	1.10009e+00	2.35612e+00	4.79535e-15	7.79536e-01	2.53585e+02	4.88724e+00
Worst	1.70559e+01	1.10026e+00	2.35612e+00	1.19884e-13	1.03293e+00	2.43784e+02	3.78243e+00
std	8.63486e-01	5.30948e-05	2.65096e-07	2.34923e-14	1.54321e-01	8.14018e+00	2.75696e+00
SR	100	100	100	100	0	92	0
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	2.55000e+01	1.20599e-05	3.61294e+04

Table A.20 Function Values of HECO-DEm Achieved for 100D ($FES_{\max} = 20000 \times D$)

problem	C01	C02	C03	C04	C05	C06	C07
Best	2.52934e-20	9.99851e-20	1.29343e-19	0.00000e+00	8.47526e-12	0.00000e+00	-1.12952e+04
Median	3.21140e-18	1.59257e-18	7.70459e-19	1.35728e+01	2.85091e-10	3.01216e-09	-1.06108e+04
c	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 2
$\overline{v}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	3.24552e-04
mean	5.25055e-18	3.34629e-18	1.36734e-18	1.30506e+01	1.62991e-09	2.79943e-04	-9.05882e+03
Worst	2.75162e-17	1.50461e-17	5.11185e-18	1.37466e+01	1.47056e-08	3.39886e-03	-8.59931e+03
std	6.07200e-18	3.62041e-18	1.46077e-18	2.66437e+00	3.31263e-09	7.83654e-04	1.77548e+03
SR	100	100	100	100	100	100	20
$\overline{vio}$	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	5.77794e-04
Problem	C8	C9	C10	C11	C12	C13	C14
Best	6.92863e-04	0.00000e+00	-1.66825e-05	-8.29270e+03	3.98089e+00	3.32095e+01	7.84202e-01
Median	1.84414e-02	0.00000e+00	-1.62462e-05	-9.49008e+03	3.98133e+00	3.37832e+01	7.87225e-01
c	0 0 1	0 0 0	0 0 0	1 0 0	0 0 0	0 0 0	0 0 0
$\overline{v}$	8.21285e-05	0.00000e+00	0.00000e+00	1.09098e+03	0.00000e+00	0.00000e+00	0.00000e+00
mean	9.90362e-02	4.14392e-08	-1.61547e-05	-9.51644e+03	9.15114e+00	3.36573e+01	7.90260e-01
Worst	5.96787e-01	1.03598e-06	-1.54195e-05	-1.02215e+04	3.23185e+01	3.44038e+01	8.05247e-01
std	1.52133e-01	2.03010e-07	3.73374e-07	3.73248e+02	1.01662e+01	3.35805e-01	7.30032e-03
SR	48	100	100	0	100	100	100
$\overline{vio}$	3.43250e-02	0.00000e+00	0.00000e+00	1.65473e+03	0.00000e+00	0.00000e+00	0.00000e+00
Problem	C15	C16	C17	C18	C19	C20	C21
Best	2.35612e+00	0.00000e+00	1.00856e-01	5.20825e+01	0.00000e+00	1.18717e+01	3.98079e+00
Median	2.35612e+00	0.00000e+00	1.00081e+00	9.04730e+01	0.00000e+00	1.28181e+01	3.98137e+00
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	2.12370e+01	7.29695e+04	0.00000e+00	0.00000e+00
mean	2.35612e+00	0.00000e+00	8.99323e-01	8.92581e+01	1.45123e-05	1.28040e+01	1.21426e+01
Worst	2.35612e+00	0.00000e+00	1.06281e+00	8.53313e+01	1.25188e-04	1.35483e+01	3.31102e+01
std	1.17495e-15	0.00000e+00	3.75765e-01	2.62678e+01	3.79977e-05	4.34238e-01	1.25873e+01
SR	100	100	0	8	0	100	100
$\overline{vio}$	0.00000e+00	0.00000e+00	5.02200e+01	1.78802e+01	7.29695e+04	0.00000e+00	0.00000e+00
Problem	C22	C23	C24	C25	C26	C27	C28
Best	7.97619e+01	7.84305e-01	2.35612e+00	1.57066e+00	1.00792e+00	2.85076e+02	6.01060e+01
Median	9.16889e+01	7.84394e-01	2.35612e+00	1.57066e+00	1.01577e+00	3.19137e+02	7.28285e+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	7.30857e+04
mean	2.21629e+03	7.84461e-01	2.35612e+00	4.27243e+00	1.02233e+00	3.02092e+02	6.55626e+01
Worst	5.12128e+04	7.85472e-01	2.35612e+00	1.88494e+01	1.09906e+00	3.01627e+02	9.30328e+01
std	1.00027e+04	2.30716e-04	1.01355e-07	4.25033e+00	2.26688e-02	1.67693e+01	1.17741e+01
SR	92	100	100	100	0	92	0
$\overline{vio}$	9.17850e+00	0.00000e+00	0.00000e+00	0.00000e+00	5.05000e+01	3.59885e-05	7.30831e+04