DIFFERENTIAL EVOLUTION-BASED METHODS FOR NUMERICAL OPTIMIZATION

QIU XIN

NATIONAL UNIVERSITY OF SINGAPORE

2016

DIFFERENTIAL EVOLUTION-BASED METHODS FOR NUMERICAL OPTIMIZATION

QIU XIN

(B.Eng, Nanjing University)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

NUS GRADUATE SCHOOL FOR

INTEGRATIVE SCIENCES AND ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2016

Declaration

I hereby declare that the thesis is my original work

and it has been written by me in its entirely. I have

duly acknowledged all the sources of information

which have been used in the thesis.

This thesis has also not been submitted for any

degree in any university previously.

Qiu Xin

Qiu Xin

4 August, 2016

Acknowledgments

The accomplishment of this thesis had to be the ensemble of many causes. First and foremost, I wish to express my great thanks to my Ph.D. supervisors, Professor Xu Jian-Xin and Associate Professor Tan Kay Chen, for their professional guidance during my wonderful journey in computational intelligence research. Prof. Xu not only taught me how to be an excellent researcher but also showed me how to be a respectable person. His kindness and integrity will encourage me throughout my life. Prof. Tan provided countless sincere helps during my PhD study. It is him that makes me grow from a curious boy into a mature researcher. I will remember his insightful words and earnest teachings forever. I also would like to thank Professor David Hsu and Associate Professor Dipti Srinivasan for their kind suggestions as my Thesis Advisory Committee.

My great thanks also goes to my seniors as well as other lab buddies who accompanied me in the past four years: Sen Bong for teaching me a lot of invaluable knowledges and skills, Hu Jun and Yu Qiang for giving me guidance on future path, Arrchana for bringing me infinite happiness and warmth, Willson for comforting and encouraging me during my toughest time, Lim Pin for patiently helping me to improve my English writing, Zhang Chong and Sim Kuan for listening to my funny stories, Weinan for bringing me lots of joy, Ruoxu, Raj, Berrak and Sivam for providing instant helps whenever I need. I would also like to express my gratitude to the lab officers, HengWei and Sara, for their consistent assistance in Control and Simulation lab.

Last but not least, I would like to express my deep seated appreciation to my families for their selfless love and care. Thanks, mom. I love you.

Contents

Summary

Differential Evolution (DE) is arguably one of the most powerful metaheuristics for solving numerical optimization problems. Although considerable research has been devoted to the development and improvement of DE, there still exist several open issues in DE community. This thesis focuses on designing new DE operators and algorithms to overcome the limitations of existing approaches in solving single-objective, multi-objective and minimax optimization problems.

First, a multiple exponential recombination strategy is proposed to fill the research gap in DE crossover operator development. Most of the existing DE variants simply utilized traditional binomial recombination or exponential recombination, which have intrinsic limitations in handling dependent subsets of variables. By exchanging multiple segments among individuals simultaneously, the new strategy is able to inherit all the main advantages of existing crossover operators while possessing a stronger ability in handling dependent variables. The properties of multiple exponential recombination is examined both theoretically and empirically. Experimental results reveal that the proposed operator is robust in solving problems with unknown variable interrelations.

Second, a novel multi-objective DE algorithm is designed for circumventing the convergence slowdown, parametric sensitivity and lack of flexibility issues while extending DE to multi-objective optimization. Two new mutation operators and one parameter adaptation mechanism are proposed based on the involvement of individuals from different generations. The new cross-generation mechanisms enable the algorithm to achieve good convergence-diversity trade-

off and robust self-adaptation of control parameters. Implementations in two famous multi-objective optimization frameworks indicate the superiority of the proposed method over other state-of-the-art multi-objective optimization evolutionary algorithms.

Finally, a new minimax DE algorithm is developed to efficiently address the fundamental issues in current minimax optimization area, e.g., restriction on problem properties, low optimization efficiency and high computational cost. A Bottom-Boosting Scheme is proposed to substantially reduce the computational expense in identifying promising solutions without loss of reliability. Furthermore, a Partial-Regeneration Strategy together with a new mutation operator is devised to enhance the exploration efficiency. Integration of all the new mechanisms contributes to an algorithm that can handle problems with various properties. Empirical comparisons with famous minimax evolutionary algorithms demonstrate the outstanding ability of the proposed algorithm in solving minimax optimization problems. Successful applications in robust optimal design of iterative learning control and robust stabilization of uncertain time-delay systems further validate the effectiveness of the new approach.

List of Tables

List of Figures

List of Algorithms

List of Abbreviations

DE Differential Evolution

EA Evolutionary Algorithm

MO Multi-objective Optimization

POS Pareto Optimal Set

POF Pareto Optimal Front

MOEA Multi-Objective Evolutionary Algorithms

LS-EXP Linearly Scalable Exponential

GDE Generalized Differential Evolution

PBDE Pareto-Based Differential Evolution

NSDE Nondominated Sorting Differential Evolution

DEMO Differential Evolution for Multi-objective Optimization

NSGA-II Nondominated Sorting Genetic Algorithm-II

MOEA/D Multi-Objective Evolutionary Algorithm/ Decomposition

SO Single-objective Optimization

DMEA Direction-based Multi-objective Evolutionary Algorithm

FADE Fuzzy Adaptive Differential Evolution

GA Genetic Algorithm

ES Evolution Strategies

EP Evolutionary Programming

PSO Particle Swarm Optimization

ASGA Aging Sampled Genetic Algorithm

SOP Single-objective Optimization Problem

EMO Evolutionary Multi-objective Optimization

NCG Neighborhood-based Cross-Generation

PCG Population-based Cross-Generation

CGA Cross-Generation Adaptation

ACGDE Adaptive Cross-Generation Differential Evolution

MOP Multi-objective Optimization Problem

IGD Inverted Generational Distance

HYP Hypervolume

MMDE Minimax Differential Evolution

ACPSO Alternating Coevolutionary Particle Swarm Optimization

ACGA Alternating Coevolutionary Genetic Algorithm

PCGA Parallel Coevolutionary Genetic Algorithm

BRCGA Best Remaining Coevolutionary Genetic Algorithm

RBCGA Rank-Based Coevolutionary Genetic Algorithm

SSCGA Stackelberg Strategy Coevolutionary Genetic Algorithm

MSE Mean Square Error

FE Fitness Evaluation

ILC Iterative Learning Control

MIMO Multi-Input-Multi-Output

LMI Linear Matrix Inequality

Chapter 1

Introduction

Numerical optimization aims at finding the best input values with regard to some criteria for the given objective function(s). It is widely involved in science, engineering, economics and industry [1]. In the past century, many deterministic optimization techniques were developed, e.g., linear programming, integer programming and quadratic programming. The most significant limitation of these techniques is the additional requirements for objective functions, e.g., linearity, convexity, differentiability and continuity. However, in real world problems, it is very common to encounter mathematically intractable objective functions, where traditional deterministic methods cannot be applied. Under these circumstances, stochastic searching techniques like metaheuristics are more suitable and efficient.

As one of the most famous metaheuristics, Evolutionary Algorithm (EA) has proved its capability of solving complex optimization problems in many practical applications [2–8]. EA is a population-based optimization algorithm inspired by natural evolution process. Each individual in the population repre-

sents a single solution, and these individuals will evolve themselves by performing crossover, mutation and survival selection iteratively. The most significant advantage of EAs is that they ideally do not make any assumption about the underlying landscape of the optimization problems, thereby leading to exceptional ability in solving mathematically intractable problems.

Among numerous EAs, Differential Evolution (DE) [9] is arguably one of the most powerful approaches for solving numerical optimization problems. DE has many inherent advantages [10], e.g., it is simple and straightforward to implement and exhibits good performance on a wide variety of problems; the number of control parameters in DE is small and the space complexity is low. Because of these superiorities, DE has gained more and more attention in recent years. Although considerable research has been devoted to the development and improvement of DE-based methods, there still exist several open issues and research gaps in DE community: the intrinsic limitations of existing DE crossover operators in handling problems with dependent variables, the difficulties in applying DE for multi-objective optimization (MO), and the lack of efficient algorithms in tackling minimax optimization problems. This thesis aims at addressing all the afore-mentioned issues by proposing new operators and algorithms.

1.1 Basic Concepts in Differential Evolution

1.1.1 Initialization

The first step of DE is the initialization of the population of N and D over the search space, where N denotes the population size and D denotes the variable dimensions. We symbolize each individual by $X_{i,g} = (x^1_{i,g}, x^2_{i,g}, \ldots, x^D_{i,g})$, where $i = 1, 2, \ldots, N, g = 0, 1, \ldots, G_{\max}$ and $G_{\max}$ denotes the maximum number of generations. Furthermore, let us define the lower search bound as $X_{\min} = (x^1_{\min}, x^2_{\min}, \ldots, x^D_{\min})$ and the upper search bound as $X_{\max} = (x^1_{\max}, x^2_{\max}, \ldots, x^D_{\max})$. Finally, the initial value of the ith individual is generated as:

$$x^j_{i,0} = x^j_{\min} + rand(0,1) \cdot (x^j_{\max} - x^j_{\min}), j = 1, 2, 3, \ldots, D \tag{1.1}$$

where $rand(0,1)$ is a uniformly distributed random number lying within the range (0,1).

1.1.2 Mutation

In this step, each individual will generate a new individual, called the mutant vector $V_{i,g}$. The most frequently used mutation strategies are listed below:

"DE/rand/1"

$$V_{i,g} = X_{r_1,g} + F \cdot (X_{r_2,g} - X_{r_3,g}) \tag{1.2}$$

"DE/best/1"

$$V_{i,g} = X_{\text{best},g} + F \cdot (X_{r_1,g} - X_{r_2,g}) \tag{1.3}$$

"DE/rand/2"

$$V_{i,g} = X_{r_1,g} + F \cdot (X_{r_2,g} - X_{r_3,g}) + F \cdot (X_{r_4,g} - X_{r_5,g}) \tag{1.4}$$

"DE/best/2"

$$V_{i,g} = X_{best,g} + F \cdot (X_{r_1,g} - X_{r_2,g}) + F \cdot (X_{r_3,g} - X_{r_4,g}) \tag{1.5}$$

"DE/current-to-best/2"

$$V_{i,g} = X_{i,g} + F \cdot (X_{best,g} - X_{i,g}) + F \cdot (X_{r_1,g} - X_{r_2,g}) + F \cdot (X_{r_3,g} - X_{r_4,g}) \tag{1.6}$$

"DE/current-to-rand/2"

$$V_{i,g} = X_{i,g} + F \cdot (X_{r_1,g} - X_{i,g}) + F \cdot (X_{r_2,g} - X_{r_3,g}) + F \cdot (X_{r_4,g} - X_{r_5,g}) \tag{1.7}$$

where $X_{\mathrm{best},g}$ denotes the individual with the best fitness value in current generation. The indices r_1, r_2, r_3, r_4, r_5 are randomly selected integers from $\{1, 2, \ldots, N\}$ that are distinct from i and mutually different. $F \in [0, 1]$ is a real parameter, called the scaling factor.

1.1.3 Crossover

After mutation, crossover operation is employed to generate the trial vectors $U_{i,g}$. During crossover, mutant vectors are recombined with the original members of the current population, called target vectors, to form trial vectors. Two basic crossover schemes of DE are exponential recombination and binomial recombination. Binomial recombination is mostly used in DE literature [11].

Binomial recombination is performed on each variable and it could be outlined as below:

$$u_{i,g}^{j} = \begin{cases} v_{i,g}^{j}, \text{ if } rand(0,1) \leq Cr \text{ or } j = j_{\mathrm{rand}} \\ x_{i,g}^{j}, \text{ otherwise} \end{cases} \tag{1.8}$$

where $j_{\mathrm{rand}} \in \{1, 2, 3, \ldots, D\}$ is a randomly selected index to ensure that the trial vector could get at least one component from the mutant vector. $rand(0, 1)$

is a uniform random number on the interval $[0, 1]$ and independently generated for each i and each j. Cr is called the crossover probability.

In exponential recombination, first two integers n and L are generated separately, where n is the starting point of crossover operation in the involved vectors and L is the number of variables that get exchanged during the crossover operation. More specifically, n is randomly chosen between the interval $[1, D]$, where D denotes the problem dimension. The pseudo-code to obtain L is shown below:

$L = 0$;

WHILE $((rand(0, 1) \leq Cr)$ AND $(L \leq D))$

DO$\{L = L + 1\}$;

where Cr is the crossover probability. $rand(0, 1)$ is a uniform random number generator on the inverval $[0, 1]$. If $L \geq 1$, then the trial vectors are generated as:

$$u_{i,g}^{j} = \begin{cases} v_{i,g}^{j}, \text{ for } j = n, n+1, n+2, \ldots, n+L-1 \\ x_{i,g}^{j}, \text{ for all other } j \in [1, D] \end{cases} \tag{1.9}$$

Otherwise, trial vectors will be identical to target vectors ($L = 0$).

1.1.4 Selection

Selection is the last step to generate the population of next generation. The process of selection is to determine whether the target vector or the trial vector survives to the next generation according to their fitness value. The selection operation in DE is described below:

$$X_{i,g+1} = \begin{cases} U_{i,g}, \text{ if } f(U_{i,g}) \leq f(X_{i,g}) \\ X_{i,g}, \text{ otherwise} \end{cases} \tag{1.10}$$

where $f(X)$ is the fitness function to be minimized.

After selection, the algorithm will proceed to next generation and continuously perform mutation, crossover and selection until the termination criterion is satisfied (e.g., the maximum generation is reached).

1.2 Multi-objective Optimization Problem

Problems in real world usually have multiple objectives instead of one single objective. A Multi-Objective Optimization Problem (MOP) can be defined as

$$\max / \min F(X) = (f_1(X), f_2(X), \ldots, f_m(X)) \tag{1.11}$$

subject to:

$$x \in \Omega$$

where f_i is the i-th objective function; m is the total number of objective functions; X refers to the decision variables and Ω represents the decision space. The MOP consists of m objective functions and it maps the decision space Ω into an m-dimensional objective space $\mathbb{R}^m$, i.e., $F : \Omega \to \mathbb{R}^m$.

Since the objectives of a MOP are often conflicting with each other, the optimal solution for one objective generally does not produce optimal results for the other objectives. The concepts of Pareto dominance are therefore defined for convenience of solution comparisons:
Assuming that all the objective functions are minimization problems, then solution X_1 is said to dominate X_2 if and only if

$$f_i(X_1) \leq f_i(X_2) \text{ for } i = 1, 2, \ldots, m \text{ and } \exists f_i(X_1) < f_i(X_2) \text{ for at least one } i.$$

Solution X_1 and X_2 are incomparable if and only if

$$\exists i, j \in \{1, 2, \ldots, m\} \text{ such that} f_i(X_1) < f_i(X_2) \text{ and } f_j(X_1) > f_j(X_2).$$

Based on the above definitions, a solution is said to be Pareto optimal if there is no other solutions that can dominate it. All the Pareto optimal solutions in desicion space jointly form the Pareto Optimal Set (POS), and their corresponding objective vectors (each element represents the value of one objective function) define the Pareto Optimal Front (POF) in objective space. Thus, instead of finding one optimal solution, the aim of MO solvers is to generate a set of solutions that can approximate the best trade-offs among multiple objective functions. More specifically, the obtained solutions should distribute their corresponding objective vectors not only close to the POF but also evenly along the POF. These two aspects are often referred to as convergence and diversity, respectively.

1.3 Minimax Optimization Problem

Minimax optimization, which is actively involved in numerous robust design problems [12–19], aims at pursuing the solutions with best worst-case performances. In general, a minimax optimization problem is represented as

$$\min_{X \in \mathbb{X}} \max_{S \in \mathbb{S}} f(X, S) \tag{1.12}$$

where $f(X, S)$ is the objective function , X is a solution selected from solution space $\mathbb{X}$, and S is a scenario selected from scenario space $\mathbb{S}$. For each X, its cost is measured by maximizing the objective function over scenario space $\mathbb{S}$, that is, to search for the worst-case cost of X. Subsequently, the problem aims

at minimizing the worst-case cost over solution space $\mathbb{X}$. An optimal solution for this problem is the one that provides the best worst-case performance.

The problem is defined as symmetrical if

$$\min_{X\in\mathbb{X}} \max_{S\in\mathbb{S}} f(X,S) = \max_{S\in\mathbb{S}} \min_{X\in\mathbb{X}} f(X,S) \tag{1.13}$$

This statement is a sufficient and necessary condition [20] for the existence of $X^* \in \mathbb{X}$ and $S^* \in \mathbb{S}$ such that

$$f(X^*,S) \leq f(X^*,S^*) \leq f(X,S^*) \tag{1.14}$$

for $\forall X \in \mathbb{X}$ and $\forall S \in \mathbb{S}$. According to [21], this implies that there exists a scenario $S^* \in \mathbb{S}$ such that

$$\min_{X\in\mathbb{X}} \max_{S\in\mathbb{S}} f(X,S) = \min_{X\in\mathbb{X}} f(X,S^*) \tag{1.15}$$

This symmetrical condition will substantially simplify the original problem because now it is possible to reach the global optima by optimizing over solution space $\mathbb{X}$ and scenario space $\mathbb{S}$ separately. For most existing algorithms [22–27], this symmetrical condition is necessary because they evaluate the fitness of each solution based on the same scenario set.

However, most minimax optimization problems do not satisfy condition (1.13), and this type of problems are defined as asymmetrical. For asymmetrical problems, no single scenario is eligible for evaluating all the solutions, thereby leading to essential difficulty for most optimization approaches.

1.4 Scope of the Thesis

This thesis focuses on solving the open problems related to DE and numerical optimization as listed below:

1. As one of the basic algorithmic components of DE, the crossover operation has not been sufficiently examined in existing works. Most of the main DE variants solely employed traditional binomial recombination, and few works utilized traditional exponential recombination. However, these two conventional crossover operators have intrinsic limitations in handling dependent subsets of variables, thereby leading to difficulties for existing DE algorithms in solving non-separable problems. A more robust crossover strategy that can efficaciously handle different types of variable interrelations is desirable.

2. Although many researchers have made efforts to extend DE for MO, there are still several open issues that tend to be neglected: first, DE is very sensitive to the setup of control parameters, and this issue will become more challenging when multiple objective functions are optimized simultaneously; second, while most existing algorithms emphasize on the maintenance of population diversity, a satisfactory convergence speed towards POF cannot be guaranteed; third, most existing MO-DEs are inconvenient to implement among different MO frameworks (more details will be discussed in section 4.1). Since varied types of MO frameworks may be needed depending on the nature of problems, it is helpful to design MO-DE in a more flexible manner.

3. Although considerable research has been devoted by EA community to the resolution of minimax optimization problems, there do exist several fundamental drawbacks for existing minimax optimization EAs: first, many of the existing methods can only work properly under a symmetrical

condition (see section 1.3 for detailed definition). However, generally this condition does not hold, and problems not satisfying this condition are known to be extraordinarily difficult to solve [28] (detailed analysis of the reason is provided in subsection 5.2.1); second, since existing algorithms spend most of the computational budget on exploration in scenario space, the optimization over solution space is insufficient. A bias towards reliability at the cost of quality is therefore inevitable; third, the evaluation criteria for solutions in existing approaches are underdeveloped. Either expensive computational cost or omission of excellent solutions may result from current evaluation mechanisms.

1.5 Contributions

The contributions of this thesis are summarized as follows:

1. A multiple exponential recombination is proposed to inherit all the main advantages of existing crossover operators while possessing a stronger ability in managing dependent variables. Multiple segments of the involved solutions will be exchanged during the proposed operator. The properties of the new scheme are examined both theoretically and empirically. Experimental results demonstrate the robustness of the proposed operator in solving problems with unknown variable interrelations.

2. A multi-objective DE variant is developed through integration of two novel mutation operators and a new parameter adaptation mechanism. The main innovation of the proposed algorithm is the simultaneous use of

individuals across generations from an objective-based perspective. Good convergence-diversity trade-off and satisfactory exploration-exploitation balance are achieved via the hybrid cross-generation mutation operation. Furthermore, the cross-generation adaptation mechanism enables individuals to self-adapt their associated parameters not only optimization-stage-wise but also objective-space-wise. Empirical results indicate the statistical superiority of the proposed algorithm over several state-of-the-art evolutionary algorithms in handling multi-objective optimization problems.

3. A Minimax DE algorithm is proposed to overcome the limitations of existing minimax optimization approaches. First, a novel Bottom-Boosting Scheme enables the algorithm to identify the promising solutions in a reliable yet efficient manner. After that, a Partial-Regeneration Strategy together with a new mutation operator contribute to an in-depth exploration over solution space. Finally, a proper integration of these newly proposed mechanisms leads to an algorithmic structure that can appropriately handle both symmetrical and asymmetrical problems. Empirical comparison with 7 famous methods demonstrates the statistical superiority of the proposed algorithm. Successful applications in two open problems of robust design further validate the effectiveness of the new approach.

1.6 Organization

The organization of the remaining thesis is as follows:

Chapter 2 provides a through literature review covering several related re-

search topics. Current research gaps are concluded based on the review of existing works.

Chapter 3 presents a new crossover operator for DE, namely, multiple exponential recombination. The limitations of traditional DE crossover operators are discussed followed by the technical details and underlying rationales of the proposed strategy. The properties of the new operator is analyzed via both theoretical mathematics and empirical studies. The optimization performance of multiple exponential recombination is then evaluated through comparison experiments.

Chapter 4 proposes a novel DE algorithm for solving MOPs. Detailed explanations about the basic structures and underlying rationales of each algorithmic component are provided. The new algorithm is implemented in different MO frameworks, and the effectiveness of the proposed mechanisms are validated via experimental comparisons with several state-of-the-art multi-objective evolutionary algorithms (MOEAs) on a wide variety of benchmark problems. A parametric sensitivity study is also conducted empirically.

Chapter 5 describes a new algorithm based on DE for minimax optimization. The fundamental drawbacks of existing methods are analyzed before discussing the algorithmic details and underlying rationales of the proposed approach. Optimization performance and parametric sensitivity of the new algorithm is studied via experiments. Furthermore, two open problems in robust control area are formulated into minimax optimization problems, and the proposed algorithm is applied to solve them.

Chapter 6 concludes this thesis, and highlights future research directions.

Chapter 2

Literature Review

This chapter summarizes the existing research works that are related to the focus of this thesis. Current research gaps are also discussed based on the literature review.

2.1 Crossover Operators for Differential Evolution

The behavior of the DE algorithm is deeply influenced by selection of mutation scheme, crossover strategy and control parameters. In order to improve the performance of traditional DE, a good volume of enhanced DE variants were proposed in the past 20 years [11, 29–41]. Among these studies, most of them are related with mutation operators [42–47] and control parameters [44, 48, 49]. In comparison, crossover operator attracted much less attention and only few works focused on examining the crossover in DE.

The major role of crossover is to enhance the potential diversity of the population by recombining the mutant vector with one existing solution. Two different crossover strategies were proposed in the original work of Storn and

Price [9, 50], namely, the exponential recombination and the binomial recombination. In the later development of new DE variants, the binomial version was much more frequently employed as the crossover scheme [10, 51, 52]. Mezura-Montes $et\ al.$ [53] compared the performances of several DE variants with different mutation and crossover strategies. Based on the results on various types of problems, they concluded that binomial recombination was much better than exponential recombination due to the fact that not all the recombinations of mutant vector and current parent can be sampled in the exponential variant. In Josef's empirical study [54], the use of exponential recombination only increased efficiency in comparison with binomial recombination for small part of the tested problems. The superiority of binomial recombination over exponential one was also claimed by Jeyakumar $et\ al.$ in their two comparative studies [55, 56]. Although different behaviors for the two crossover schemes were observed in the above works, no further investigation was conducted to examine the cause of these differences. Chuan $et\ al.$ [52] discovered that the consecutive crossover was more reliable in solving some non-separable problems but did not provide detailed explanations for this phenomenon. Zaharie [51] analyzed the relationship between parameter Cr and mutation probability for both crossover strategies from a mathematical perspective. It was suggested after implementing empirical tests [51, 57] that the behavior of the algorithm during crossover depended more on the mutation probability.

Besides the comparative studies on existing crossover strategies, a small number of improved crossover operations were proposed in the recent years. Islam $et\ al.$ [38] introduced the p-Best crossover operation, in which a greedy

parent selection strategy was incorporated with binomial recombination. Guo $et\ al.$ [58] utilized eigenvectors of covariance matrix of individual solutions in binomial recombination. Zhao $et\ al.$ [59] fixed the length of the crossover in exponential recombination according to the dimensionality of the problems and named it as the linearly scalable exponential crossover operator (LS-EXP). It is notable that actually these new strategies are intrinsically identical with the original binomial recombination or exponential recombination in terms of selection strategy for exchanging positions, only the involved solutions or the crossover length are changed. Development of an essentially new crossover operator for DE is still a research niche. Chapter 3 further examines the limitations of existing crossover strategies, and proposes a new crossover operator to overcome these limitations.

2.2 Differential Evolution Algorithms for Multi-objective Optimization

Being a powerful heuristic for numerical optimization, DE has been extended into MO by many researchers. Chang et al. [60] constitute the first systematical attempt to extend DE in MO. In their paper, DE/rand/1/bin is utilized with a Pareto optimal set, which is an external archive to store the non-dominated solutions during the search process. Fitness sharing is also incorporated into their approach in order to maintain the diversity of the whole population. Babu and Jehan [61] proposed the Differential Evolution for Multi-Objective Optimization approach. One objective function is incorporated as an additional constraint

in their algorithm, and an aggregation function is utilized. Li and Zhang [62] developed a MO Differential Evolution based on decomposition for continuous MO problems with variable linkages. Kukkonen and Lampinen [63] [64] modified basic DE algorithm for MO problems, and named the framework Generalized Differential Evolution (GDE). The survival selection between the parent and the offspring is based on Pareto Dominance and the constraints of the problems are also handled with Pareto Dominance in the constraint space. In their later work [65] [66], two new versions of GDE were proposed to achieve a better diversity of the final population. Other MO-DEs where Pareto Dominance works as the criterion for survival selection include [67–70]. Another branch of the Pareto-based algorithms encompass approaches that introduce a Pareto ranking procedure into DE. Representative methods include Pareto-Based Differential Evolution (PBDE) algorithm [71], Nondominated Sorting Differential Evolution (NSDE) [72] and Differential Evolution for Multi-Objective Optimization (DEMO) [73]. In these algorithms, a $(\mu + \lambda)$-selection is implemented after a set of trial vectors have been generated from the current population. One of the more recent works that gave an outstanding performance would be the MOEA/D-DE and NSGA-II-DE proposed in [74]. In that paper, a neighborhood selection mechanism is utilized to enhance the diversity maintenance ability of MOEA/D-DE, whilst the NSGA-II-DE with tournament selection is an improved version of NSDE [72].

Based on the above review, most existing works on MO-DE focus on combining the traditional DE mutation strategies with diversity-based survival selection procedures, which are commonly used in other MOEAs. However, the

natures of single-objective optimization (SO) and MO are inherently different. SO aims at reaching the global optimal solution of one particular objective function. MO involves simultaneous optimization of multiple conflicting objectives and targets at obtaining a solution set with not only high quality but also good diversity. During the design of traditional DE recombination operators, only the SO scenarios were considered and the special requirements of MO were not taken into account. Therefore, it is inefficient to directly apply the traditional SO-DE operators for MO. Development of new mutation schemes that are oriented exclusively toward MO is still a research gap. In Chapter 4, two novel mutation strategies are specially designed to enhance both diversity and convergence in MO.

2.3 Direction-guided Evolutionary Algorithms

Steering the optimization process in an effective and efficient way is desirable in the design of an evolutionary algorithm. Among the numerous search schemes, direction-guided search has proved to be quite promising [75]. In SO, the idea of "direction" is incorporated into DE via various ways. Fan and Lampinen improved TDE [76] with a weighted directed mutation strategy [77]. Feoktistov $et\ al.$ [78] [79] proposed to use fitness values to determine a "good" direction, and similar mutation strategy was introduced in [80]. Kaelo and Ali [81] employed the attraction-repulsion concept to boost the mutation operator of canonical DE. In [82], direction information is utilized in a classification-based self-adaptive DE. More recently, Cai and Wang [42] designed two new operators for DE to better exploit the neighborhood and direction information of the population, re-

spectively.

For MO, only a limited number of works have investigated the role of "direction" in DE (subsection 4.2.2 provides a detailed definition of "direction" in MO, which may be different from those in existing literatures). Zhang and Sanderson [83] proposed JADE2 by obtaining direction information from the archived solutions and current population. Iorio and Li examined the directional convergence and directional spread in [84]. Liu $et\ al.$ ranked the population to provide the direction information in SRDE [85]. Li $et\ al.$ considered the search direction during the strategy adaptation in MODE/SN [86]. Besides DE, researchers have designed other evolutionary algorithms based on direction guidance for MO. Bui $et\ al.$ [75] proposed the direction-based multi-objective evolutionary algorithm (DMEA), in which a population evolves over time along some directions of improvement. Later, several new features were added into DMEA to make it more robust (named DMEA-II) [87], and a new niching method was designed for DMEA [88]. In [89] [90], local models based on direction information are utilized to guide the evolutionary algorithm. Li and Dario [91] proposed Evolutionary Multi-objective Simulated Annealing Algorithm (EMOSA) with adaptive and competitive search direction. Goh $et\ al.$ [92] investigated on different types of gradient information and successfully implemented the evolutionary gradient search in MO (MOEGS).

Although external archives are used in some direction-guided MOEAs (e.g., DMEA, DMEA-II and JADE2), little attention is paid to the application of information across generations, which may imply the efficient searching directions. Another open problem with current direction-guided MOEAs is the

insufficient analysis regarding objective space. Objective-based relationships among individuals are in fact critical in the estimation of converging directions (see Chapter 4 for definition). To fill this research niche, Chapter 4 develops a cross-generation mutation strategy, in which individuals from different generations are employed based on their relationships in objective space. Promising searching directions for specified solutions are then generated to guide the exploration.

2.4 Parameter Adaptation in Differential Evolution

The success of DE in solving a particular problem is very closely related to the fine-tuning of its control parameters. In the literature, a good volume of work has been undertaken to enhance the ultimate performance of DE via adaptive parameters.

One of the early works is Liu and Lampinen's fuzzy adaptive differential evolution (FADE) algorithm [93]. Fuzzy logic controllers are used to adapt the scaling factor F and crossover probability Cr by incorporating the relative function values and individuals of the successive generations into inputs. Qin $et\ al.$ [30] proposed the famous SaDE algorithm, in which both the mutation strategies and their associated control parameters are self-adapted according to their previous performance. Another representative self-adaptation scheme is the one that proposed by Brest $et\ al.$ [29] in their jDE. They encoded F and Cr into each individual and updated them according to respective uniform distributions.

Following the same line of thinking, Zhang $et\ al.$ [31] implemented Gaussian distribution and Cauchy distribution to help regenerate the control parameters in JADE algorithm. According to their empirical studies, JADE achieves overall better performance than jDE and SaDE. Inspired by JADE, Islam [38] designed a new parameter adaptation scheme based on the record of recent successful F and Cr. In their paper, the proposed mechanism was also integrated with JADE and jDE, and significant improvement was reported. Recently, Yu $et\ al.$ [48] introduced a two-level parameter adaptation approach that considers not only the overall state of the population but also the characteristics of different individuals. Comparisons with SaDE, jDE and JADE demonstrate the efficiency of tuning parameters in individual-level and population-level. Other adaptive DE algorithms that derived from above work include [94–97].

In contrast to SO, adapting the parameters in DE for solving multi-objective problems received less interest from researchers. Zaharie and Petcu [98] developed an adaptive Pareto DE for multiobjective optimization from their previous single-objective DE [99], in which the parameter adaptation is guided by the population diversity evolution. In [68], Abbass enhanced their PDE [67] algorithm with self-adaptive parameters for crossover and mutation. Zamuda $et\ al.$ [100] applied the self adaptation mechanism from evolution strategies to DEMO algorithm [73] [101]. Huang $et\ al.$ extended the original SaDE into multi-objective optimization [102], and proposed objective-wise learning strategies (OW-MOSaDE) [103] to further improve the algorithm.

According to the above survey, knowledge in objective space is neglected by most existing adaptation methods for MO-DE. The intrinsic difference be-

tween SO and MO is not revealed in the design of these schemes. The only exception is the OW-MOSaDE, in which the good parameters are recorded for each objective respectively. However, the algorithm is not fully developed because only one set of parameters is randomly selected for optimizing all the objectives. Moreover, the mechanism is only applied to crossover probability Cr, while another important control parameter, namely scaling factor F, is simply generated from a normal distribution. To fill this research gap, a new parameter adaptation mechanism is introduced in Chapter 4 to dynamically adapt the parameters from an objective-based perspective.

2.5 Evolutionary Algorithms for Minimax Optimization

When mathematically intractable properties are involved in the minimax optimization problems, EAs are promising replacements for traditional deterministic approaches. In recent years, a good number of algorithms have been proposed by EA community for handling minimax optimization problems.

Following the consideration that two decision spaces are explored simultaneously in minimax optimization, it is natural for researchers to apply co-evolutionary algorithms, in which two populations are maintained during the optimization process. Herrmann [27] proposed a two-space genetic algorithm (GA) for solving minimax optimization problems. Two populations, representing solutions and scenarios respectively, are evolved in parallel. An individual in one population is evaluated with respect to all the individuals in the other

population. More specifically, the cost of a solution X is given by

$$h(X) = \max_{S \in P_S} f(X, S) \tag{2.1}$$

where P_S represents the scenario population. The cost of a scenario S is calculated as

$$g(S) = \min_{X \in P_X} f(X, S) \tag{2.2}$$

where P_X represents the solution population. The algorithm then evolves the two populations concurrently using traditional GA operators. Similarly, Barbosa [26] proposed another coevolutionary GA using the above fitness evaluation mechanisms. The only modification is that the two populations will be optimized alternately, which means the algorithm will evolve one population for several generations while fixing the other population. In [22, 24, 25], three algorithms were introduced by replacing the GA operators in Herrmann's framework [27] with evolution strategies (ES), evolutionary programming (EP) and particle swarm optimization (PSO). The most significant limitation for these methods is that they will fail on asymmetrical problems [104], which is more common than symmetrical problems in real-world applications.

In order to better handle the asymmetrical situations, various types of fitness assignment methods were developed for evaluating scenarios in coevolutionary EAs. Hur $et\ al.$ [105] proposed the best remaining strategy wherein the fitness of a scenario is decided by the rank of corresponding solution that it performs best. Jensen [104] designed a rank-based evaluation method by assigning fitness based on the maximum rank a scenario can achieve against any solution. An evaluation scheme based on stackelberg strategy was suggested in [21]. Four new ranking-based methods were introduced in [106], namely,

worst case method, average greedy method, distance greedy method and ranking greedy method. All these approaches aim at properly addressing the asymmetrical problems by improving the scenario evaluation strategies. However, as we will analyze in section 5.2, due to the intrinsic limitations of coevolution-based frameworks, the existing solution evaluation mechanisms will lead to inevitable problems in face of asymmetrical conditions. Another issue with the coevolutionary algorithms is the prohibitively expensive evaluation for each solution and scenario. If the population size is N, then N times of objective function evaluations are required to determine the fitness of one single solution or scenario. This will severely impair the overall optimization performance given a limited computational budget.

Apart from coevolutionary approaches, a small number of non-coevolutionary EAs were also proposed for tackling minimax optimization problems. Laskari $et\ al.$ [107] modified traditional PSO algorithm by evaluating each solution over all the possible scenarios. This algorithm is only applicable to discrete problems and the computational cost is too high. Zhou and Zhang [108] proposed a surrogate-assisted EA for minimax optimization. A surrogate model based on Gaussian process is built to regress the fitness function for solution space. One drawback with this method is that a large number of hidden objective function evaluations (1×10^5 according to the original paper) are required to obtain the true fitness of one single solution. The difficulty for building reliable models may also increase dramatically when the problems become more complex. In [21], Cramer $et\ al.$ proposed an aging sampled GA (ASGA) that can handle both symmetrical and asymmetrical problems properly.

The fitness of a solution is based on the worst performance over all the scenarios it has been tested so far, and all these scenarios are sampled randomly. Except for fitness, the concept of age is introduced as a second criterion during the survival selection. Experimental results showed that ASGA could outperform most state-of-the-art EAs in solving different types of minimax optimization problems. However, since all the tested scenarios are simply sampled randomly, it may take great numbers of trials for one solution to reach its worst-case scenario, and poor solutions may replace good solutions because of "lucky" samplings. The overall optimization efficiency will then be diminished.

Based on the above literature review, there are several unsolved issues with the existing minimax optimization EAs. In order to circumvent these issues, Chapter 5 presents a new minimax DE algorithm that can overcome all the aforementioned limitations.

Chapter 3

Multiple Exponential Recombination for Differential Evolution

As the first part of main works in this thesis, this chapter focuses on enhancing the basic DE algorithms by proposing a new crossover operator. Both theoretical and empirical studies demonstrate that the proposed multiple exponential recombination successfully overcome the limitations of existing DE crossover strategies in handling dependent variables.

3.1 Introduction

DE has gained more attention in recent years, and many new variants of DE are emerging [11, 29–31, 37–41, 109–112]. Among the existing DE variants, most works focus on designing new mutation strategies and adapting control parameters intelligently. However, as one of the basic steps in traditional DE [9],

crossover operation has not been sufficiently examined in the main DE variants and most of the algorithms solely applied binomial recombination [11, 51]. In contrast, exponential recombination received less attention during the design of new DE algorithms.

According to some comparative studies [52, 53], binomial recombination is generally more robust and efficient than exponential recombination. To seek the reason, the relation between the mutation probability [51] and the value of control parameter Cr (crossover rate) is linear in binomial recombination while in exponential recombination it is nonlinear. Thus it may result in difficulties in choosing proper values for Cr during implementations of exponential recombination, especially in solving high-dimensional problems [51]. Moreover, binomial recombination is able to generate all the 2^D possible solutions during crossover (where D is the dimension of decision space of the problem) whereas exponential recombination can only cover part of the 2^D possibilities [53].

Intrinsically, the main difference between the two recombination operators lies in the distribution of the exchanged variables. Exponential recombination leads to a consecutive crossover so that a set of adjacent elements are exchanged. In binomial recombination, the exchanged variables are dispersed randomly after the non-consecutive crossover. Based on existing empirical investigations [51, 52], consecutive crossover shows promising results in solving non-separable problems, in which the variables are strongly dependent with each other. The global optima of a non-separable problem cannot be achieved by optimizing each variable separately, thus maintaining the dependent subsets of variables is very critical in solving this kind of problems [113].

As a consecutive crossover operator, exponential recombination tends to preserve strongly dependent components that are adjacent or physically proximate to each other. For binomial recombination, all the elements are presumed to be linked equally strongly and all the dependent variables stand the same chance to be split up during crossover. Consequently, the efficiency of the two crossover operators depend heavily on the interrelationships among variables. Considering the fact that the variable linkage information is not available in advance for most optimization problems [114], selection of the proper crossover strategy becomes a difficult issue for existing DE algorithms. In addition, even though the highly related variables are arranged adjacent to one another, the current exponential recombination is still inconvenient to use in face of high-dimensional problems and may also be ineffective regardless of the problem dimensionality due to its intrinsic disadvantages mentioned above. A more robust crossover strategy that can efficaciously handle different types of variable interrelations is highly desirable.

To circumvent the above issue, a new crossover operator named multiple exponential recombination is proposed in this chapter. The main innovation of the proposed operator is the semi-consecutive crossover strategy, in which multiple segments of the involved individuals will be combined to form a new solution. In this way, the new operator becomes more robust than the traditional crossover strategies in handling dependent variables. Moreover, the specially designed mechanisms enable the new operator to preserve all the main advantages of binomial recombination and exponential recombination: linear relationship between mutation probability and control parameter value, cover-

age of all possible offsprings, and strengths in preserving particular types of dependent subsets. The properties of multiple exponential recombination is examined both theoretically and empirically. Implementation in different DE variants demonstrates the robustness of the proposed operator in solving problems with unknown variable interrelations.

3.2 Multiple Exponential Recombination

3.2.1 Overview

The aim of the proposed new crossover strategy is to inherit all the main advantages of traditional crossover operators while providing a more robust performance in handling different types of variable interrelations.

As the most widely used crossover scheme in DE literatures, binomial recombination offers two main advantages when compared with exponential recombination: it is convenient to control the mutation probability in binomial recombination via adapting the value of parameter Cr; all possible 2^D recombinations of the mutant vector and target vector can be generated during binomial recombination, where D is the dimension of decision space. In order to preserve these two strengths, a linear relationship between the control parameter and the mutation probability has been created during the design of the new operator. In addition, the new operator is developed as a multi-point crossover operation [115] so that all the possible configurations can be covered during recombination.

Besides the above features, the new strategy is expected to overcome

the limitations of existing crossover operators in handling dependent variables. More specifically, under similar mutation probability [51], the adjacent or physically proximate variables are more likely to be disrupted in binomial recombination than in exponential recombination. Conversely, the chance to update together the variables that are distributed distantly from each other is much lower in exponential recombination than in binomial recombination.

Fig. 3.1 illustrates the limitations of the two traditional crossover operators in solving non-separable problems. Since the crossover operation is conducted in a circular manner in exponential recombination, we use the outer circle and inner circle to represent the target vector and mutant vector, respectively. Each rectangle represents a decision variable and the variables marked with red and blue represent two dependent subsets. Optimizing the elements within one subset simultaneously will lead to more efficient exploration. The two-headed arrows mean that the pointed variable in the target vector will be replaced by the corresponding variable in the mutant vector to form the trial vector. Assuming that both crossover operations are going to change 6 out of 12 variables, two common situations are depicted in Fig. 3.1 to demonstrate the limitations of each strategies. For binomial recombination, the crossover is performed on each variable independently so that the exchange points are more likely to be distributed uniformly. In this case, the dependent subset x^3 and x^4 will be disrupted since they are adjacent to each other. As a consecutive crossover operation, exponential recombination exchanges a segment of the involved solutions at each time. Thus, adjacent variables like x^3 and x^4 have a higher chance to be updated together. However, if the distance between two variables exceeds the

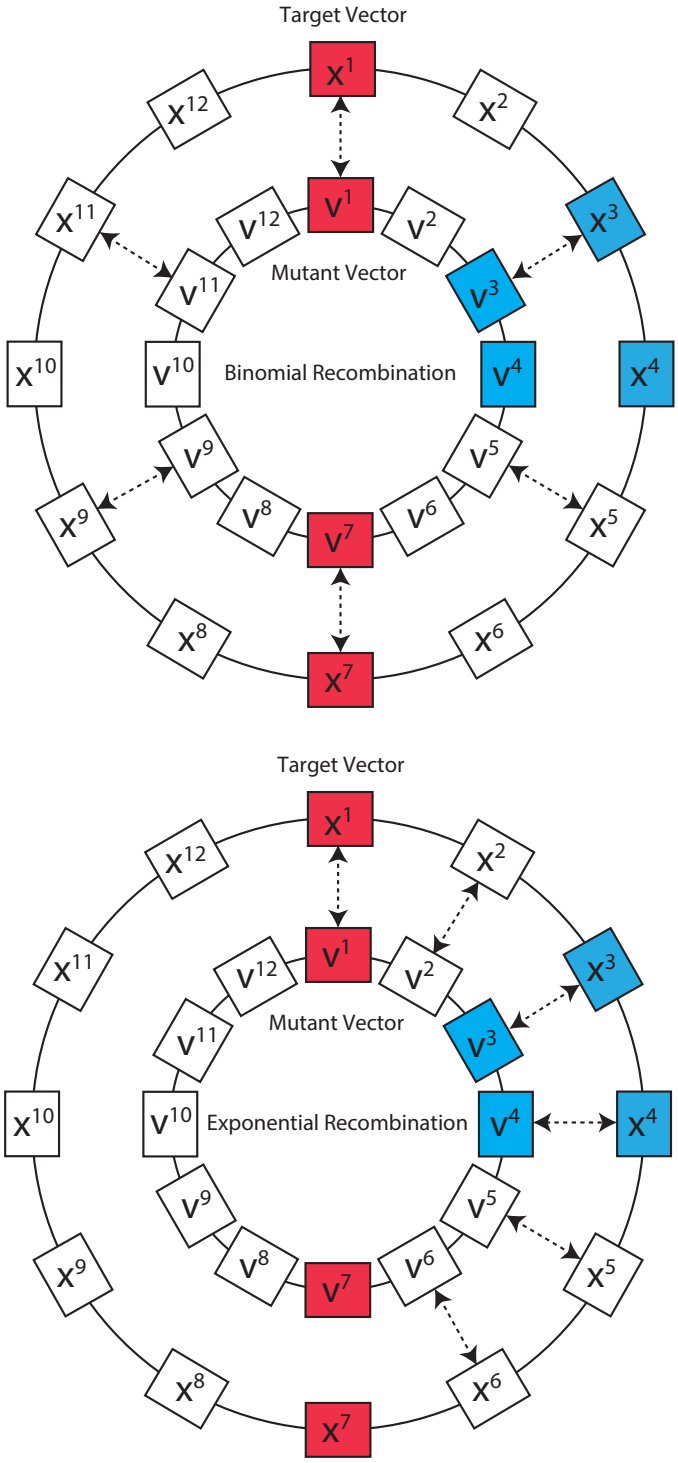

Figure 3.1: This figure illustrates the limitations of binomial recombination and exponential recombination in handling dependent variables via two examples. The elements with same color (red or blue) are strongly dependent on each other. All the involved solutions are visualized in a circular manner.

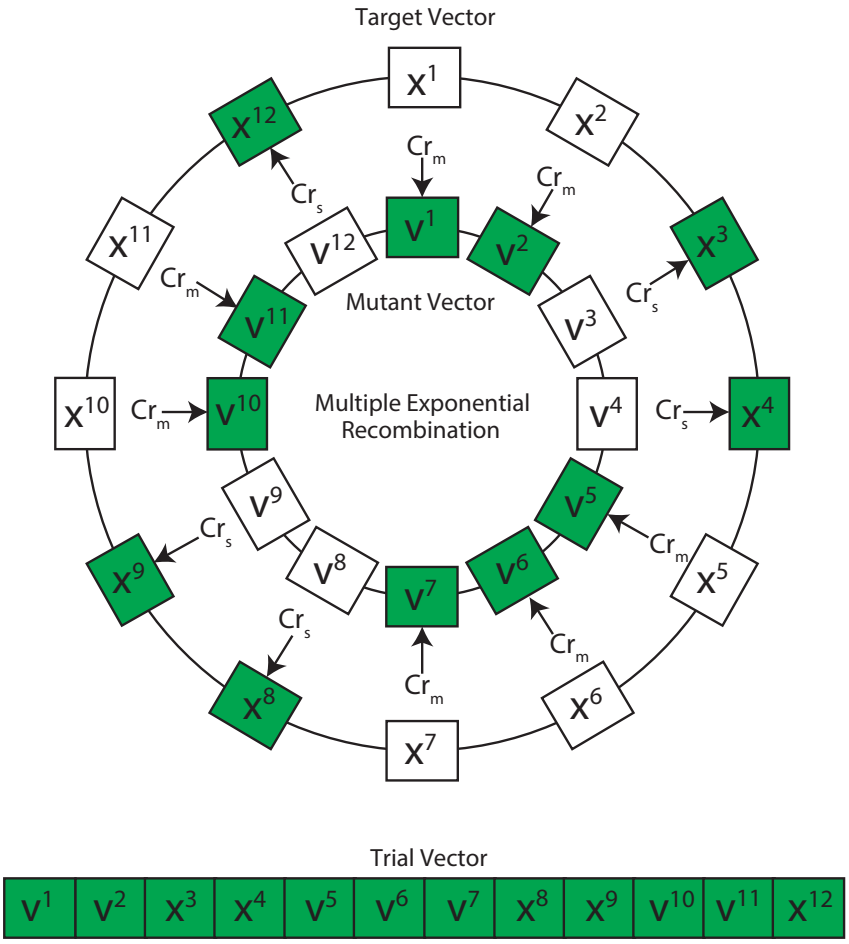

Figure 3.2: This figure depicts the detailed behaviors during a multiple exponential recombination. The elements in color are the ones selected to construct the trial vector.

length of this segment, it is impossible to update these variables concurrently in exponential recombination. As shown in Fig. 3.1, given the fact that the length of exchange segment is less than 7, under no circumstances can x^1 and x^7 be replaced by the corresponding elements in mutant vector at the same time. Only binomial recombination is able to update x^1 and x^7 simultaneously. In summary, solely applying a consecutive or non-consecutive crossover operator is not sufficient to properly handle different types of variable interrelations. To address this issue, a semi-consecutive crossover operator is developed in this chapter.

3.2.2 Algorithmic Structure

The main idea of the proposed strategy is to perform the basic operations of traditional exponential recombination multiple times. Segments from the mutant vector and the target vector will alternately constitute the trial vector, thereby leading to a semi-consecutive crossover. Fig. 3.2 and Algorithm 3.1 demonstrate the standard procedure of multiple exponential recombination.

First, a starting point n is randomly selected from $[1, D]$, where D is the dimension of the optimization problem. Starting from dimension n, a segment of the mutant vector will be extracted to construct the trial vector. The length of this segment is decided by a sequence of Bernoulli trials with success probability Cr_m. Similar to the operations in traditional exponential recombination, each successful event will increase the segment length by 1. This process will be terminated once an unsuccessful trial happened. Please note that the length can be 0 if the first Bernoulli trial fails. Subsequently, starting from the dimension where last unsuccessful event happened, another segment from the target vector will be selected to constitute the trial vector. The procedure to decide this segment length is the same as in last step, and the only difference is that the success probability for each trial is changed to Cr_s. The proposed operator will keep switching between the above two processes until all the components of the trial vector have been decided.

From the above description, it is obvious that the behavior of the operator depends heavily on the selection of Cr_m and Cr_s. However, the relationship between Cr_m (or Cr_s) and the segment length is nonlinear, and the effect of Cr_m and Cr_s on mutation probability is even more complicated. This issue

Algorithm 3.1 Procedure of Multiple Exponential Recombination

Require:

$X_{i,g} = (x^1_{i,g}, x^2_{i,g}, \ldots, x^D_{i,g})$: Target Vector
$V_{i,g} = (v^1_{i,g}, v^2_{i,g}, \ldots, v^D_{i,g})$: Mutant Vector
Cr: Parameter to control the mutation probability
T: Parameter to control the lengths of the exchanged segments

Ensure:

$U_{i,g} = (u^1_{i,g}, u^2_{i,g}, \ldots, u^D_{i,g})$: Trial vector

$E_m = T \cdot Cr$
$E_s = T \cdot (1 - Cr)$
$Cr_m = E_m/(E_m + 1)$
$Cr_s = E_s/(E_s + 1)$
Set n as an integer randomly generated from the interval $[1, D]$
$k = 1, Mutation\_Enable = 1$
while $k \leq D$ **do**
  if $Mutation\_Enable = 1$ **then**
    while $k \leq D$ **and** $rand(0, 1) \leq Cr_m$ **do**
      $d = \langle n \rangle_D$, where $\langle n \rangle_D$ equals to n if $n \leq D$ and equals to $n - D$ if $n > D$
      $u^d_{i,g} = v^d_{i,g}$
      $n = n + 1$
      $k = k + 1$
    end while
    $Mutation\_Enable = 0$
  else
    while $k \leq D$ **and** $rand(0, 1) \leq Cr_s$ **do**
      $d = \langle n \rangle_D$
      $u^d_{i,g} = x^d_{i,g}$
      $n = n + 1$
      $k = k + 1$
    end while
    $Mutation\_Enable = 1$
  end if
end while

will cause difficulty in controlling the behavior of the proposed operator during applications. To address this issue, another two control parameters are used instead, namely Cr and T. The users are only required to set Cr and T, and the value of Cr_m and Cr_s will then be automatically decided as follows:

$$E_m = T \cdot Cr \tag{3.1}$$

$$E_s = T \cdot (1 - Cr) \tag{3.2}$$

$$Cr_m = E_m/(E_m + 1) \tag{3.3}$$

$$Cr_s = E_s/(E_s + 1) \tag{3.4}$$

Briefly speaking, E_m and E_s are the approximate average lengths of the segments extracted from the mutant vector and target vector, respectively. Cr decides the mutation probability and T controls the length of exchanged segments. The underlying rationale of this design will be discussed from a mathematical perspective in the next subsection.

3.2.3 Theoretical Analysis

In this subsection, the properties of the proposed operator will be investigated theoretically. The purpose of this theoretical analysis is to prove the three main advantages of multiple exponential recombination from a mathematical perspective.

In both traditional binomial and exponential recombination, the value of parameter Cr strongly affects the mutation probability p_m [51], which represents the probability that a component of the trial vector is mutated (means this component is taken from the mutant vector). A higher p_m will lead to more

mutant elements in the trial vector, whereas a lower p_m tends to deliver more variables from the target vector to the trial vector. As a result, the convergence speed of the population and even the entire optimization performance are heavily influenced by the value of p_m. Creating a linear relationship between Cr and p_m will make it more convenient for the users to control the behavior of the algorithm. Among the existing crossover operations, this is only achieved by binomial recombination. In order to preserve this main advantage of binomial recombination, the control parameters in multiple exponential recombination are specially designed based on the following mathematical analysis.

Proposition 1. An approximate linear relationship is achieved between the mutation probability p_m and control parameter Cr in multiple exponential recombination.

Proof. In the proposed strategy, multiple segments from both mutant vector and target vector construct the trial vector. Thus, the mutation probability is actually decided by the average lengths of these segments. Following the definition of Cr_m, the length of one segment extracted from the mutant vector (denoted as L_m) is decided via the following process:

$L_m = 0$;

WHILE $((rand(0,1) \leq Cr_m)$ AND $(L_m \leq D))$

DO$\{L_m = L_m + 1\}$;

where $rand(0,1)$ is a uniform random number generator on the interval $[0,1]$. L_m is the number of variables contained by the segment and D is the dimension of the decision space.

In such a process, the probability for L_m to be h is given by:

$$P(L_m = h) = \begin{cases} Cr_m^h \cdot (1 - Cr_m), & \text{if } 0 \leq h < D \\ Cr_m^D \qquad\qquad\quad , & \text{if } h = D \end{cases} \tag{3.5}$$

The mathematical expectation of this length L_m is computed as follows:

$$\begin{aligned} E(L_m) &= \sum_{h=0}^{D-1} h \cdot Cr_m^h \cdot (1 - Cr_m) + D \cdot Cr_m^D \\ &= (1 - Cr_m) \cdot (Cr_m + 2Cr_m^2 + 3Cr_m^3 + \ldots + (D-1)Cr_m^{D-1}) + D \cdot Cr_m^D \\ &= Cr_m \cdot [(1 - Cr_m) \cdot (1 + 2Cr_m + 3Cr_m^2 + \ldots + (D-1)Cr_m^{D-2}) + D \cdot Cr_m^{D-1}] \\ &= Cr_m \cdot [(1 - Cr_m)(\frac{1 - Cr_m^{D-1}}{1 - Cr_m} + Cr_m \cdot \frac{1 - Cr_m^{D-2}}{1 - Cr_m} \\ &\quad + Cr_m^2 \cdot \frac{1 - Cr_m^{D-3}}{1 - Cr_m} + \ldots + Cr_m^{D-2} \cdot \frac{1 - Cr_m}{1 - Cr_m}) + D \cdot Cr_m^{D-1}] \\ &= Cr_m[1 + Cr_m + Cr_m^2 + Cr_m^3 + \ldots + Cr_m^{D-2} - (D-1)Cr_m^{D-1} + D \cdot Cr_m^{D-1}] \\ &= Cr_m \cdot \frac{1 - Cr_m^D}{1 - Cr_m} \end{aligned} \tag{3.6}$$

Analogously, given a Cr_s, the expectation of the length of one segment taken from the trial vector is calculated as below:

$$E(L_s) = Cr_s \cdot \frac{1 - Cr_s^D}{1 - Cr_s} \tag{3.7}$$

where L_s represents the number of components inside the segment.

Under the assumption that the dimension of the problem D is larger than 50 and both Cr_m and Cr_s are less than 0.9, we will have:

$$1 - Cr_m^D \approx 1 - Cr_s^D \approx 1 \tag{3.8}$$

Thus,

$$E(L_m) \approx \frac{Cr_m}{1 - Cr_m} \tag{3.9}$$

$$E(L_s) \approx \frac{Cr_s}{1 - Cr_s} \tag{3.10}$$

In the proposed strategy, E_m and E_s are the expected values of $E(L_m)$ and $E(L_s)$, respectively. By replacing $E(L_m)$ and $E(L_s)$ with E_m and E_s in (3.9) and (3.10), we will obtain:

$$E_m \approx \frac{Cr_m}{1 - Cr_m} \tag{3.11}$$

$$E_s \approx \frac{Cr_s}{1 - Cr_s} \tag{3.12}$$

Consequently, the required values of Cr_m and Cr_s can be computed using (3.3) and (3.4), respectively.

Considering the mutation probability p_m is defined as the probability that an element in the trial vector is inherited from the mutant vector, the value of p_m can be approximated using $E(L_m)$ and $E(L_s)$ as below:

$$p_m \approx \frac{E(L_m)}{E(L_m) + E(L_s)} \tag{3.13}$$

It is notable that the higher the dimension D is, the closer the true p_m and the above approximated value will be. Thus, the relationship between p_m and Cr is given by:

$$p_m \approx \frac{E_m}{E_m + E_s} = \frac{T \cdot Cr}{T \cdot Cr + T \cdot (1 - Cr)} = Cr \tag{3.14}$$

An approximate linear relationship is achieved between the mutation probability p_m and control parameter Cr.

Remark 1. This is the first advantage of the proposed operator. In order to fulfill the assumption that both Cr_m and Cr_s are less than 0.9 (0.9 included), we will have:

$$Cr_m = E_m/(E_m + 1) \leq 0.9 \Rightarrow E_m \leq 9 \tag{3.15}$$

$$Cr_s = E_s/(E_s + 1) \leq 0.9 \Rightarrow E_s \leq 9 \tag{3.16}$$

Thus, considering (3.1), (3.2) and the fact that the Cr value is commonly chosen from $[0.1, 0.9]$, we will have:

$$E_m = T \cdot Cr \leq T \cdot 0.9 \leq 9 \Rightarrow T \leq 10 \tag{3.17}$$

$$E_s = T \cdot (1 - Cr) \leq T \cdot 0.9 \leq 9 \Rightarrow T \leq 10 \tag{3.18}$$

Based on (3.1) and (3.2), a relatively higher T value will lead to a longer average length of the exchange segments. This is beneficial to increase the chance of preserving the physically proximate dependent variables. Taking all the above analysis into account, the value of T will be fixed as 10 for all the experiments in this chapter.

Proposition 2. All 2^D possible trial vectors can be generated during the crossover process of multiple exponential recombination.

Proof. To explain briefly, the number of the exchanged segments is unfixed and the lengths of these segments can vary between 0 to D, thereby leading to 2^D possibilities during the recombination process.

Remark 2. This is the second advantage of multiple exponential recombination. As a result, the explorative ability of the crossover operator will be enhanced.

The last and most important strength of the new operator is the robustness in handling different types of variable interrelations. Since preservation of dependent subsets is very critical in solving non-separable problems, one quantitative criterion to judge the ability of an operator in managing dependent variables is the distribution of probabilities to split up these subsets. Both the physical distance between the dependent variables in the vector representation and the values of other control parameters will affect the disruption probability. We have mathematically derived the probabilities to disrupt two dependent

variables with different physical distances while employing multiple exponential recombination. The derivation is based on modeling the crossover process as a discrete-time Markov Chain [116].

Proposition 3. A transition matrix A is used in the representation of the obtained probability:

$$A = \begin{bmatrix} \frac{Cr_m}{1-(1-Cr_m)(1-Cr_s)} & \frac{(1-Cr_s)Cr_m}{1-(1-Cr_m)(1-Cr_s)} \\ \frac{(1-Cr_m)Cr_s}{1-(1-Cr_m)(1-Cr_s)} & \frac{Cr_s}{1-(1-Cr_m)(1-Cr_s)} \end{bmatrix} \tag{3.19}$$

The probability to disrupt two variables during multiple exponential recombination is calculated as:

$$P_{\text{disruption}} = p_v \cdot q_x + p_x \cdot q_v \tag{3.20}$$

where p_v, p_x, q_v and q_x are given by:

$$\begin{bmatrix} p_v \\ p_x \end{bmatrix} = \frac{1}{n}(A^{D-1} + \ldots + A + A^0) \begin{bmatrix} \frac{Cr_m}{1-(1-Cr_m)(1-Cr_s)} \\ \frac{(1-Cr_m)Cr_s}{1-(1-Cr_m)(1-Cr_s)} \end{bmatrix} \tag{3.21}$$

$$\begin{bmatrix} q_v \\ q_x \end{bmatrix} = A^L \begin{bmatrix} p_v \\ p_x \end{bmatrix} \tag{3.22}$$

where D is the dimensionality of the optimization problem and L is the dimension difference between the two variables in the circular representation of the solutions (count from the variable that first undergoes the crossover operation).

Proof. During the proposed Multiple Exponential Recombination, segments from the mutant vector ($V_{i,g} = (v^1_{i,g}, v^2_{i,g}, \ldots, v^D_{i,g})$) and the target vector ($X_{i,g} = (x^1_{i,g}, x^2_{i,g}, \ldots, x^D_{i,g})$) will alternately constitute the trial vector ($U_{i,g} = (u^1_{i,g}, u^2_{i,g}, \ldots, u^D_{i,g})$). Here, D is the dimension of decision space. i is the index of the vector, and g is the current generation number. First, we will calculate the probability for $u^d_{i,g}$ to be $v^d_{i,g}$ given that we already knew $u^{d-1}_{i,g}$ is inherited

from $v_{i,g}^{d-1}$. According to the process of Multiple Exponential Recombination (Algorithm 3.1), this conditional probability is given by:

$$\begin{aligned} P(u_{i,g}^{d} = v_{i,g}^{d} | u_{i,g}^{d-1} = v_{i,g}^{d-1}) =& Cr_m + (1 - Cr_m) \cdot (1 - Cr_s) \cdot Cr_m \\ &+ (1 - Cr_m)^2 \cdot (1 - Cr_s)^2 \cdot Cr_m \\ &+ \ldots + (1 - Cr_m)^{\infty} \cdot (1 - Cr_s)^{\infty} \cdot Cr_m \\ &= \frac{Cr_m}{1 - (1 - Cr_m)(1 - Cr_s)} \end{aligned} \tag{3.23}$$

Similarly, we can have the conditional probabilities for the other three situations:

$$P(u_{i,g}^{d} = v_{i,g}^{d} | u_{i,g}^{d-1} = x_{i,g}^{d-1}) = \frac{(1 - Cr_m)Cr_s}{1 - (1 - Cr_m)(1 - Cr_s)} \tag{3.24}$$

$$P(u_{i,g}^{d} = x_{i,g}^{d} | u_{i,g}^{d-1} = x_{i,g}^{d-1}) = \frac{Cr_s}{1 - (1 - Cr_m)(1 - Cr_s)} \tag{3.25}$$

$$P(u_{i,g}^{d} = x_{i,g}^{d} | u_{i,g}^{d-1} = v_{i,g}^{d-1}) = \frac{(1 - Cr_s)Cr_m}{1 - (1 - Cr_m)(1 - Cr_s)} \tag{3.26}$$

From the above analysis, the probability for $u_{i,g}^{d}$ to be $v_{i,g}^{d}$ or $x_{i,g}^{d}$ depends only on the state of previous element $u_{i,g}^{d-1}$. Thus, the procedure of Multiple Exponential Recombination can be modeled as a discrete-time Markov Chain, and the transition matrix A is as below:

$$\begin{aligned} A &= \begin{bmatrix} P(u_{i,g}^{d} = v_{i,g}^{d} | u_{i,g}^{d-1} = v_{i,g}^{d-1}) & P(u_{i,g}^{d} = x_{i,g}^{d} | u_{i,g}^{d-1} = v_{i,g}^{d-1}) \\ P(u_{i,g}^{d} = v_{i,g}^{d} | u_{i,g}^{d-1} = x_{i,g}^{d-1}) & P(u_{i,g}^{d} = x_{i,g}^{d} | u_{i,g}^{d-1} = x_{i,g}^{d-1}) \end{bmatrix} \\ &= \begin{bmatrix} \frac{Cr_m}{1-(1-Cr_m)(1-Cr_s)} & \frac{(1-Cr_s)Cr_m}{1-(1-Cr_m)(1-Cr_s)} \\ \frac{(1-Cr_m)Cr_s}{1-(1-Cr_m)(1-Cr_s)} & \frac{Cr_s}{1-(1-Cr_m)(1-Cr_s)} \end{bmatrix} \end{aligned} \tag{3.27}$$

Second, we are going to calculate the probability for $u_{i,g}^d$ to be $v_{i,g}^d$ or $x_{i,g}^d$ without precondition. Without loss of generality, we assume that the dimension difference between the starting element $u_{i,g}^{start}$ (the first element in $U_{i,g}$ that undergoes the crossover operation) and $u_{i,g}^d$ is K, then we will have:

$$\begin{aligned}\begin{bmatrix} P(u_{i,g}^d = v_{i,g}^d | \langle d - start \rangle_D = K) \\ P(u_{i,g}^d = x_{i,g}^d | \langle d - start \rangle_D = K) \end{bmatrix} &= A^K \begin{bmatrix} P(u_{i,g}^{start} = v_{i,g}^{start}) \\ P(u_{i,g}^{start} = x_{i,g}^{start}) \end{bmatrix} \\ &= A^K \begin{bmatrix} \frac{Cr_m}{1-(1-Cr_m)(1-Cr_s)} \\ \frac{(1-Cr_m)Cr_s}{1-(1-Cr_m)(1-Cr_s)} \end{bmatrix} \end{aligned} \tag{3.28}$$

where $\langle d - start \rangle_D$ equals to $d - start$ if $d - start \leq D$ and equals to $d - start - D$ if $d - start > D$. Thus, in order to calculate the unconditional probability $P(u_{i,g}^d = v_{i,g}^d)$ and $P(u_{i,g}^d = x_{i,g}^d)$, we need to consider all the possible situations, in which K ranges from 0 to $D - 1$. Since each dimension has the same probability to become the starting position, we will have:

$$\begin{bmatrix} P(u_{i,g}^d = v_{i,g}^d) \\ P(u_{i,g}^d = x_{i,g}^d) \end{bmatrix} = \frac{1}{n}(A^{D-1} + A^{D-2} + \ldots + A + A^0) \begin{bmatrix} \frac{Cr_m}{1-(1-Cr_m)(1-Cr_s)} \\ \frac{(1-Cr_m)Cr_s}{1-(1-Cr_m)(1-Cr_s)} \end{bmatrix} \tag{3.29}$$

where $\frac{1}{n}$ is the probability for a particular dimension to become the starting position.

Third, we consider two elements $u_{i,g}^d$ and $u_{i,g}^{d2}$. If the dimension difference between them is $\langle d2 - d \rangle_D = L$, we will have:

$$\begin{bmatrix} P(u_{i,g}^{d2} = v_{i,g}^{d2} | \langle d2 - d \rangle_D = L) \\ P(u_{i,g}^{d2} = x_{i,g}^{d2} | \langle d2 - d \rangle_D = L) \end{bmatrix} = A^L \begin{bmatrix} P(u_{i,g}^d = v_{i,g}^d) \\ P(u_{i,g}^d = x_{i,g}^d) \end{bmatrix} \tag{3.30}$$

where $P(u_{i,g}^d = v_{i,g}^d)$ and $P(u_{i,g}^d = x_{i,g}^d)$ are given by (3.29).

Finally, the probability that $u_{i,g}^d$ and $u_{i,g}^{d2}$ are inherited from different vectors

(either $V_{i,g}$ or $X_{i,g}$) is calculated as follows:

$$P^L_{\text{disruption}} = P(u^d_{i,g} = x^d_{i,g}) \cdot P(u^{d2}_{i,g} = v^{d2}_{i,g} | \langle d2 - d \rangle_D = L) \\ + P(u^d_{i,g} = v^d_{i,g}) \cdot P(u^{d2}_{i,g} = x^{d2}_{i,g} | \langle d2 - d \rangle_D = L) \tag{3.31}$$

Since we do not make any assumption in the exact value of d or $d2$(we only assume the dimension difference between d and $d2$), the above $P^L_{\text{disruption}}$ actually represents the probability for Multiple Exponential Recombination to disrupt two variables that have the dimension difference L.

In the final results, we use symbols p_v, p_x, q_v and q_x to represent $P(u^d_{i,g} = v^d_{i,g})$, $P(u^d_{i,g} = x^d_{i,g})$, $P(u^{d2}_{i,g} = v^{d2}_{i,g} | \langle d2 - d \rangle_D = L)$ and $P(u^{d2}_{i,g} = x^{d2}_{i,g} | \langle d2 - d \rangle_D = L)$, respectively.

Remark 3. From the above derivation results, it is not straightforward to interpret the properties of this disruption probability. In order to provide a clearer demonstration, simulation experiments will be conducted in the next section to approximate the distribution of disruption probability using relative frequency. Based on the simulation result (Fig. 3.4), in comparison with traditional binomial recombination, the proposed operator has a similar probability to preserve the distant dependent subsets while possessing a lower probability to disrupt the physically proximate variables. This characteristic enables the proposed strategy to be more robust in managing variables with unknown interrelations. Details of the simulation experiments will be given in the next section.

3.3 Empirical Study

3.3.1 Properties of the proposed strategy

Two critical properties of the proposed strategy will be examined in this subsection via experiments. One is the relationship between mutation probability p_m and the control parameter Cr. An approximate linear relationship will provide convenience in controlling the behavior of algorithm. The other one is the probability to split up two dependent variables during crossover operations. Visualizing the disruption probability under different circumstances will give a clear picture about the robustness of the operator in handling dependent variables.

Based on the theoretical analysis in subsection 3.2.3, the relationship between p_m and Cr is approximately linear in multiple exponential recombination. In order to verify this conclusion empirically, the values of p_m under different Cr are approximated using the relative frequency (or empirical probability), which is defined as the number of times an event occurred normalized by the total number of trials. When the number of trials is large enough, the obtained relative frequency can accurately reveal the true probability. In this study, T is fixed as 10 for multiple exponential recombination and the value of Cr will change from 0 to 1 gradually at an interval of 0.01. For each value of Cr, 1×10^6 independent trials will be performed to simulate the process of the proposed operation and the traditional binomial recombination, respectively. The relative frequency of the event that a variable is mutated during the crossover is calculated for each Cr value, and we name this relative frequency as "Mutation frequency". Fig. 3.3 shows simulation results when the dimensionality of deci-

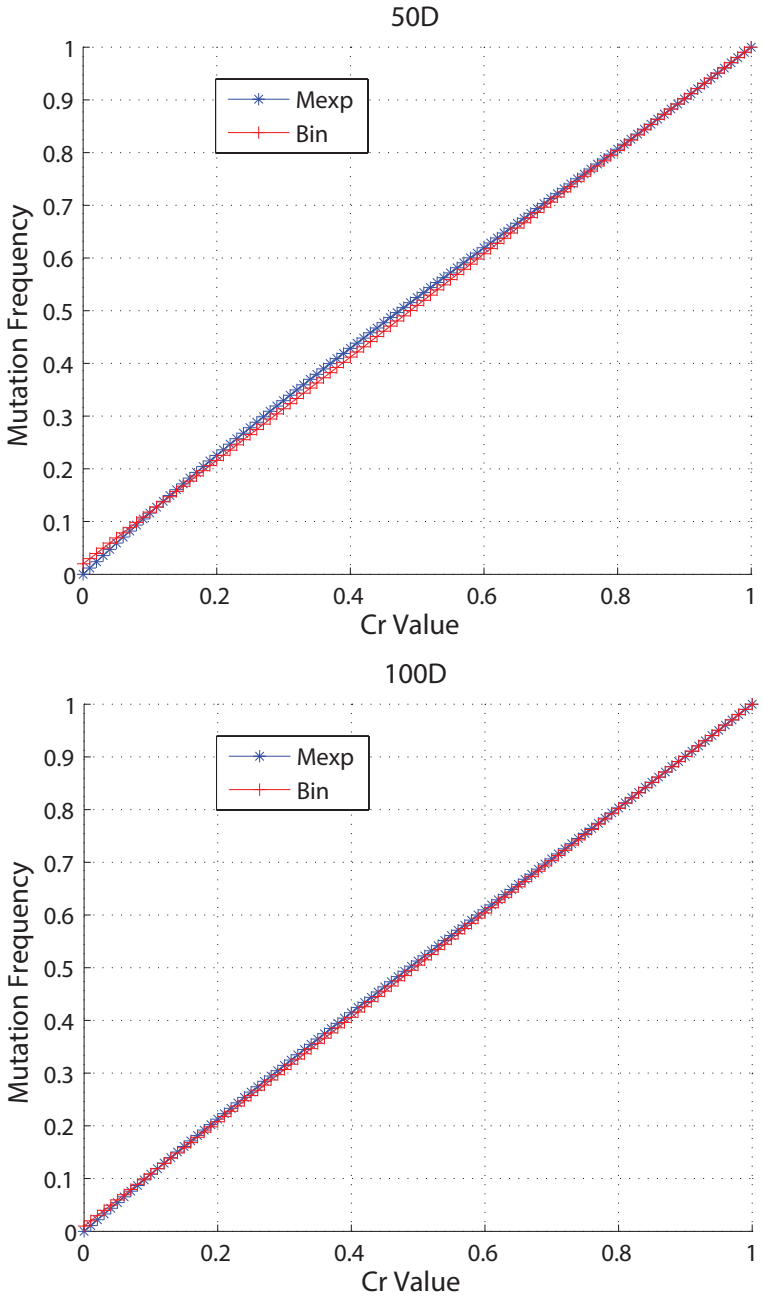

Figure 3.3: This figure plots the mutation frequency of multiple exponential recombination and binomial recombination under different Cr values. The upper plot is for the 50-dimensional problems and the lower one is for the 100-dimensional case.

sion space is 50 and 100. "Mexp" refers to multiple exponential recombination and "Bin" refers to binomial recombination. Consistent abbreviations are also used in all the following figures and tables.

From Fig. 3.3, it can be observed that both the crossover operators are able to produce an approximate linear relationship between the mutation frequency and Cr value. More specifically, the value of mutation frequency is very close to that of Cr in 50-dimensional problems and is nearly identical with that of Cr in 100-dimensional situations. Considering that the mutation frequency is a reliable approximation of p_m, this observation is in accordance with our previous discussions in subsection 3.2.3. A higher dimensionality will lead to a smaller discrepancy between p_m and Cr and an approximate linear relationship is achieved between them in the proposed strategy.

Next, behavior of multiple exponential recombination in handling dependent variables is investigated via simulations. To demonstrate the robustness of the proposed strategy, dependent variables with different distances are tested in the experiments. The probabilities to split up these dependent subsets is compared between multiple exponential recombination and binomial recombination. The dimension of the decision space is set as 50, T is set as 10 for multiple exponential recombination and Cr is fixed as 0.5 for both operators. Since the value of p_m is similar for the two strategies under same Cr, any difference in the performance is mainly due to the positions of the mutated components instead of the number of mutated variables. Without loss of generality, one variable is always placed in the first dimension and the other dependent variable will change its position from the second dimension till the last. The difference between their

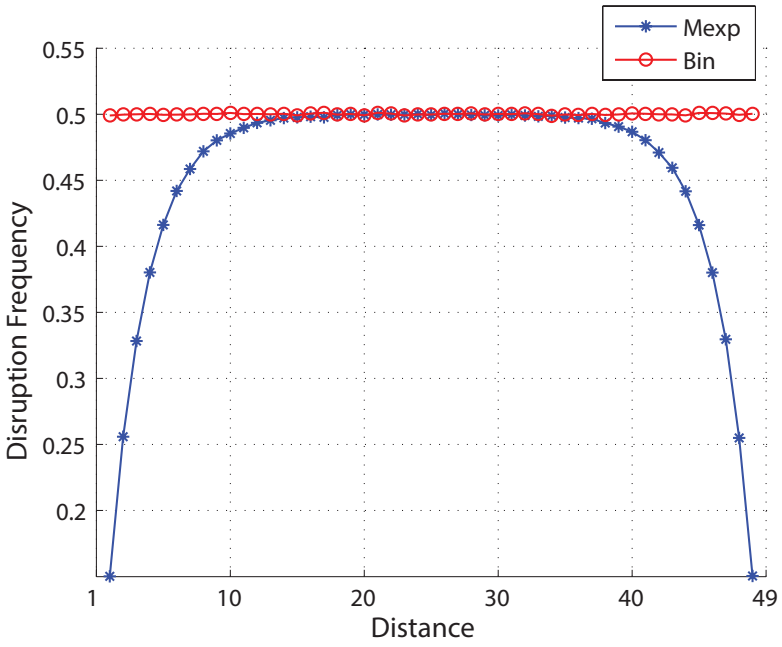

Figure 3.4: This figure shows the disruption frequency for multiple exponential recombination and binomial recombination under different distance settings. The dimensionality of the decision space is 50 and the Cr value is 0.5 for both strategies.

dimension indices is defined as the "distance" between them (e.g., a distance of "1" means the two variables are adjacent to each other). In the experiments, the distance between the two dependent variables will change from 1 to 49. 1×10^6 independent trials will be simulated for each setting. The relative frequency for the event that the two dependent variables are disrupted during crossover is computed and referred to as "disruption frequency". Fig. 3.4 presents the disruption frequency over different distances for both crossover operators.

For binomial recombination, the disruption frequency stays around 0.5 for all the distance settings. Similar disruption frequency is also observed in multiple exponential recombination for distances between 15 to 35. However, when the distance is less than 15 or larger than 35, the performances of the two strategies become vastly different. Multiple exponential recombination exhibits a gradually decreasing disruption frequency when the distance value becomes

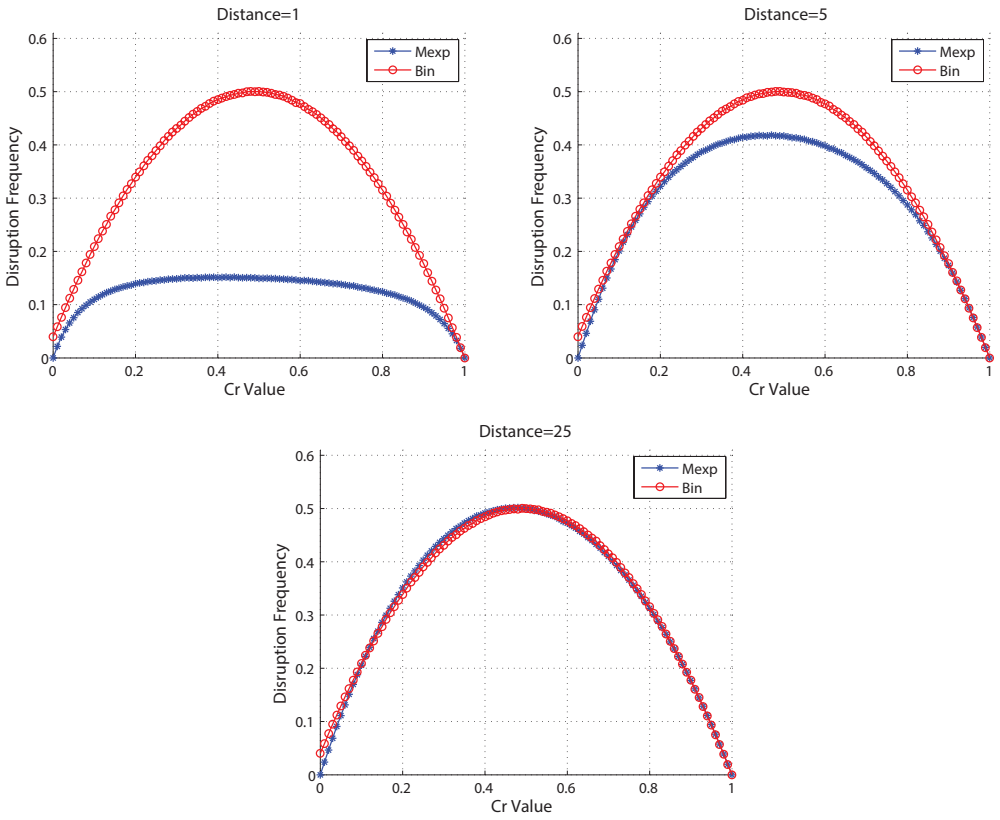

Figure 3.5: This figure compares the disruption frequency for multiple exponential recombination and binomial recombination under different Cr values and distance settings. The dimensionality of the decision space is 50.

closer to 1 or 49. Considering the crossover operation in multiple exponential recombination is conducted in a circular manner, actually a distance of 49 also means the two variables are adjacent to each other and a distance of 25 is the most distant situation. Following this consideration, it can be concluded that the closer the two variables are placed, the less probable the two variables will be split up by multiple exponential recombination. In the most extreme case, where the two dependent variables are adjacent to each other, the disruption frequency may drop to around 0.15. Even in the worst case, where the distance between the two dependent variables is 25, the disruption frequency of the proposed strategy will not exceed that of the traditional binomial recombination.

To examine further, we have changed the Cr value from 0 to 1 at an interval of 0.01, and tested each Cr value under different distance settings. Similarly, 1×10^6 independent trials will be conducted for each parametric setup on each crossover operator. Fig. 3.5 shows the obtained disruption frequencies after aforementioned simulations. Based on Fig. 3.5, the behaviors of the two operators are almost the same when the distance between the two dependent variables is 25. In contrast, under the "Distance=1" setting, multiple exponential recombination is able to provide a much lower disruption frequency under most Cr values. For the "Distance=5" case, this lower disruption frequency is observed when the Cr value falls between 0.2 to 0.8. These results validate the conclusion that the proposed strategy is better than binomial recombination in preserving physically proximate variables without deteriorating the performance in handling distant variables.

To summarize, the proposed multiple exponential recombination preserves the advantages of traditional binomial recombination in handling distant dependent variables whilst has a lower chance to disrupt the physically proximate dependent subsets. A more robust behavior is achieved by the proposed strategy in handling different types of variable interrelations.

3.3.2 Performance Evaluation

To evaluate the optimization performance of the new crossover operator, the proposed multiple exponential recombination will be implemented with 6 classical DE mutation strategies, namely "DE/rand/1", "DE/best/1", "DE/rand/2", "DE/best/2", "DE/current-to-best/2", "DE/current-to-rand/2". 14 representative

benchmark problems from the special session and competition on real parameter optimization held under the IEEE CEC 2005 [117] (f_1-f_{14}) are utilized to evaluate the performance of the algorithms. Both the 50-dimensional and 100-dimensional versions of these problems are tested. Numerous types of problems are covered including unimodal, multi-modal, shifted and rotated. All the problems are non-separable except for f_1 and f_9. The performance of the algorithms with multiple exponential recombination is compared with those variants with binomial recombination. For all algorithms, population size is fixed as 100, F is set as 0.5, Cr is set as 0.5, and the maximum number of function evaluations is set to 50×10^4. T is fixed as 10 for multiple exponential recombination. The performance evaluation of the algorithm is based on the best-of-run error, which corresponds to the absolute difference between the best-of-run fitness value and the actual global optimum. All of the simulations were done on an Intel (R) Core (TM) i7 machine with 16-GB RAM and 3.40-GHz speed. Table 3.1 and Table 3.2 show the mean and standard deviation of the best-of-run errors over 30 independent runs for each algorithm on 50-dimensional and 100-dimensional benchmarks, respectively. In order to judge whether the results of multiple exponential variants and binomial variants differ in a statistically significant way, a non-parametric statistical test called Wilcoxon rank-sum test [118] is conducted at the 5% significance level. The entries which are $significantly$ better than the counterparts are marked in boldface in Table 3.1 and Table 3.2.

From the experimental results, the variants with multiple exponential recombination exhibits superior overall performance in comparison with the binomial variants. When incorporated with "DE/rand/1", "DE/rand/2", "DE/current-

Table 3.1: Mean and Standard Deviation of the Best-of-run Errors for 30 Independent Runs Over 50-Dimensional Benchmark Set

f_i	DE/rand/1/bin		DE/rand/1/Mexp			DE/best/1/bin		DE/best/1/Mexp		
(50D)	Mean	Std	Mean	Std	+/=/-	Mean	Std	Mean	Std	+/=/-
f_1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	=	4.54E+00	2.48E+01	**2.97E-13**	**1.52E-13**	+
f_2	6.43E+04	1.05E+04	**5.38E+03**	**9.83E+02**	+	3.76E+00	1.92E+01	**1.28E-12**	**1.18E-12**	+
f_3	4.59E+08	5.59E+07	**2.94E+08**	**4.52E+07**	+	1.12E+07	4.85E+06	**3.76E+06**	**1.65E+06**	+
f_4	8.95E+04	1.14E+04	**1.66E+04**	**2.49E+03**	+	4.19E+03	4.28E+03	**1.95E+03**	**3.79E+03**	+
f_5	4.90E+03	1.08E+03	**3.43E+03**	**1.20E+03**	+	5.57E+03	1.45E+03	6.13E+03	1.42E+03	=
f_6	3.58E+01	4.52E-01	**2.73E+00**	**1.24E+00**	+	5.66E+00	1.24E+01	**6.64E-01**	**1.51E+00**	+
f_7	1.36E-02	5.17E-03	**4.48E-03**	**1.99E-03**	+	1.25E-02	1.38E-02	9.23E-03	1.77E-02	=
f_8	2.11E+01	3.83E-02	2.11E+01	3.84E-02	=	2.11E+01	3.36E-02	2.11E+01	4.72E-02	=
f_9	2.33E+02	1.10E+01	**1.82E+02**	**1.14E+01**	+	8.52E+01	2.42E+01	8.77E+01	2.76E+01	=
f_{10}	3.85E+02	1.26E+01	3.79E+02	1.32E+01	=	3.06E+02	1.10E+02	**1.89E+02**	**1.08E+02**	+
f_{11}	7.30E+01	1.07E+00	**7.24E+01**	**1.05E+00**	+	3.95E+01	1.94E+01	3.26E+01	1.17E+01	=
f_{12}	7.93E+05	5.49E-14	7.99E+05	5.49E-14	=	1.12E+05	8.09E+04	1.29E+05	1.16E+05	=
f_{13}	2.98E+01	1.41E+00	**2.23E+01**	**1.10E+00**	+	1.24E+01	5.66E+00	1.43E+01	8.22E+00	=
f_{14}	2.32E+01	1.44E-01	2.33E+01	1.66E-01	=	2.27E+01	2.89E-01	2.26E+01	2.72E-01	=
	Total Number of (+/=/-)				9/5/0	Total Number of (+/=/-)				6/8/0
f_i	DE/rand/2/bin		DE/rand/2/Mexp			DE/best/2/bin		DE/best/2/Mexp		
(50D)	Mean	Std	Mean	Std	+/=/-	Mean	Std	Mean	Std	+/=/-
f_1	4.90E-03	8.93E-04	**6.92E-04**	**1.17E-04**	+	0.00E+00	0.00E+00	0.00E+00	0.00E+00	=
f_2	1.03E+05	1.33E+04	**3.15E+04**	**2.48E+03**	+	2.93E+04	7.19E+03	**6.55E+02**	**3.56E+02**	+
f_3	6.96E+08	7.89E+07	**5.29E+08**	**7.22E+07**	+	1.52E+08	4.38E+07	**6.08E+07**	**2.04E+07**	+
f_4	1.30E+05	1.36E+04	**5.28E+04**	**6.35E+03**	+	4.97E+04	1.03E+04	**5.10E+03**	**2.90E+03**	+
f_5	1.59E+04	1.35E+03	**1.47E+04**	**1.40E+03**	+	5.28E+03	2.08E+03	**3.73E+03**	**1.59E+03**	+
f_6	7.42E+03	2.14E+03	**7.19E+02**	**1.60E+02**	+	3.82E+01	2.22E+01	**2.36E-07**	**9.10E-07**	+
f_7	1.49E+02	3.03E+01	**5.22E+01**	**9.97E+00**	+	2.31E-03	4.75E-03	**1.81E-03**	**4.38E-03**	+
f_8	2.11E+01	3.74E-02	2.11E+01	3.38E-02	=	2.11E+01	2.78E-02	2.11E+01	2.88E-02	=
f_9	2.84E+02	8.61E+00	**2.29E+02**	**1.01E+01**	+	2.38E+02	1.27E+01	**1.96E+02**	**1.38E+01**	+
f_{10}	4.33E+02	1.24E+01	4.34E+02	1.40E+01	=	3.88E+02	1.43E+01	3.82E+02	1.70E+01	=
f_{11}	7.28E+01	1.36E+00	7.28E+01	1.24E+00	=	7.27E+01	1.56E+00	**7.13E+01**	**2.10E+00**	+
f_{12}	1.74E+06	1.12E+05	**1.40E+06**	**1.57E+05**	+	9.91E+04	8.64E+04	8.81E+04	7.08E+04	=
f_{13}	5.87E+01	6.09E+00	**3.47E+01**	**1.60E+00**	+	2.94E+01	1.41E+00	**2.30E+01**	**1.39E+00**	+
f_{14}	2.33E+01	1.73E-01	2.33E+01	1.31E-01	=	2.31E+01	1.79E-01	2.32E+01	1.38E-01	=
	Total Number of (+/=/-)				10/4/0	Total Number of (+/=/-)				9/5/0
f_i	DE/c-t-b/2/bin		DE/c-t-b/2/Mexp			DE/c-t-r/2/bin		DE/c-t-r/2/Mexp		
(50D)	Mean	Std	Mean	Std	+/=/-	Mean	Std	Mean	Std	+/=/-
f_1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	=	0.00E+00	0.00E+00	0.00E+00	0.00E+00	=
f_2	2.44E+04	2.76E+03	**3.26E+02**	**7.32E+01**	+	5.03E+04	5.07E+03	**3.50E+03**	**4.29E+02**	+
f_3	1.84E+08	2.32E+07	**1.09E+08**	**2.35E+07**	+	3.30E+08	3.19E+07	**2.42E+08**	**3.03E+07**	+
f_4	3.90E+04	7.13E+03	**3.63E+03**	**6.76E+02**	+	7.17E+04	8.09E+03	**1.58E+04**	**2.10E+03**	+
f_5	3.76E+03	1.49E+03	3.79E+03	1.61E+03	=	1.01E+04	7.28E+02	**9.16E+03**	**9.47E+02**	+
f_6	2.78E+01	2.48E+00	**4.91E-06**	**1.86E-05**	+	3.79E+01	1.33E+00	**1.26E+01**	**3.08E+00**	+
f_7	1.92E-03	4.79E-03	**7.45E-05**	**2.18E-04**	+	1.18E-01	4.64E-02	**7.73E-02**	**4.21E-02**	+
f_8	2.11E+01	3.21E-02	2.11E+01	4.51E-02	=	2.11E+01	3.98E-02	2.11E+01	4.33E-02	=
f_9	2.54E+02	1.15E+01	**2.06E+02**	**1.00E+01**	+	2.56E+02	9.21E+00	**2.08E+02**	**1.16E+01**	+
f_{10}	3.88E+02	1.14E+01	**3.79E+02**	**1.26E+01**	+	3.93E+02	1.46E+01	3.92E+02	1.24E+01	=
f_{11}	7.21E+01	2.10E+00	7.19E+01	1.43E+00	=	7.25E+01	1.87E+00	7.18E+01	1.72E+00	=
f_{12}	3.60E+05	2.28E+05	4.70E+05	2.07E+05	=	1.30E+06	1.37E+05	**1.11E+06**	**1.04E+05**	+
f_{13}	2.90E+01	1.48E+00	**2.25E+01**	**8.71E-01**	+	3.04E+01	1.41E+00	**2.30E+01**	**1.10E+00**	+
f_{14}	2.32E+01	1.57E-01	2.31E+01	2.17E-01	=	2.32E+01	1.53E-01	2.31E+01	1.55E-01	=
	Total Number of (+/=/-)				8/6/0	Total Number of (+/=/-)				9/5/0

The symbols "+/=/-" indicate the statistical test results using Wilcoxon rank sum test at the 5% significance level. "+" and "-" mean that the performance of the variant with multiple exponential recombination is significantly superior or inferior than that of the variant with binomial recombination, respectively. "=" means the performance difference between the two variants is not statistically significant. "c-t-r" and "c-t-b" represent "current-to-rand" and "current-to-best", respectively.

Table 3.2: Mean and Standard Deviation of the Best-of-run Errors for 30 Independent Runs Over 100-Dimensional Benchmark Set

f_i	DE/rand/1/bin		DE/rand/1/Mexp			DE/best/1/bin		DE/best/1/Mexp		
(100D)	Mean	Std	Mean	Std	+/=/-	Mean	Std	Mean	Std	+/=/-
f_1	1.61E-05	4.09E-06	**5.28E-06**	**1.41E-06**	+	0.00E+00	0.00E+00	0.00E+00	0.00E+00	=
f_2	4.31E+05	3.02E+04	**2.46E+05**	**1.94E+04**	+	2.50E+04	1.20E+04	**2.78E+03**	**3.92E+03**	+
f_3	2.55E+09	2.98E+08	**2.37E+09**	**2.85E+08**	+	4.87E+07	1.48E+07	**3.70E+07**	**1.39E+07**	+
f_4	5.01E+05	3.65E+04	**3.32E+05**	**2.82E+04**	+	1.35E+05	3.62E+04	**7.53E+04**	**2.02E+04**	+
f_5	2.48E+04	1.75E+03	2.39E+04	1.55E+03	=	1.62E+04	2.90E+03	1.61E+04	2.88E+03	=
f_6	3.86E+02	7.63E+01	**1.13E+02**	**7.78E+00**	+	1.09E+02	4.81E+01	**1.19E+00**	**1.86E+00**	+
f_7	1.08E+01	1.82E+00	**4.85E+00**	**1.11E+00**	+	4.76E-03	6.66E-03	6.89E-03	8.40E-03	=
f_8	2.13E+01	2.11E-02	2.13E+01	3.54E-02	=	2.13E+01	2.48E-02	2.13E+01	2.70E-02	=
f_9	7.01E+02	1.62E+01	**6.61E+02**	**1.78E+01**	+	2.80E+02	5.34E+01	2.86E+02	5.05E+01	=
f_{10}	9.69E+02	2.29E+01	9.65E+02	2.65E+01	=	8.21E+02	2.82E+02	**6.85E+02**	**2.94E+02**	+
f_{11}	1.62E+02	2.15E+00	1.62E+02	1.70E+00	=	1.60E+02	2.67E+00	**8.59E+01**	**2.39E+01**	+
f_{12}	9.37E+06	6.29E+05	**8.93E+06**	**6.18E+05**	+	5.94E+05	3.19E+05	5.62E+05	2.80E+05	=
f_{13}	9.74E+01	4.65E+00	**8.24E+01**	**2.74E+00**	+	8.54E+01	3.17E+01	8.99E+01	2.10E+01	=
f_{14}	4.79E+01	1.47E-01	4.79E+01	1.65E-01	=	4.75E+01	3.49E-01	4.74E+01	3.19E-01	=
	Total Number of (+/=/-)				9/5/0	Total Number of (+/=/-)				6/8/0
f_i	DE/rand/2/bin		DE/rand/2/Mexp			DE/best/2/bin		DE/best/2/Mexp		
(100D)	Mean	Std	Mean	Std	+/=/-	Mean	Std	Mean	Std	+/=/-
f_1	6.57E+03	4.64E+02	**4.39E+03**	**2.81E+02**	+	0.00E+00	0.00E+00	0.00E+00	0.00E+00	=
f_2	5.25E+05	3.69E+04	**3.63E+05**	**2.09E+04**	+	2.70E+05	2.66E+04	**1.35E+05**	**1.98E+04**	+
f_3	3.79E+09	3.56E+08	**3.55E+09**	**3.40E+08**	+	7.51E+08	1.65E+08	6.63E+08	1.73E+08	=
f_4	6.19E+05	6.01E+04	**4.53E+05**	**4.20E+04**	+	3.37E+05	3.65E+04	**2.17E+05**	**3.64E+04**	+
f_5	4.11E+04	1.89E+03	4.07E+04	2.15E+03	=	1.72E+04	2.20E+03	1.75E+04	1.78E+03	=
f_6	1.14E+09	1.36E+08	**6.93E+08**	**9.37E+07**	+	1.17E+02	3.23E+01	**8.11E+01**	**1.51E+01**	+
f_7	1.75E+04	9.76E+02	**1.49E+04**	**8.68E+02**	+	6.32E-01	9.61E-02	**5.45E-01**	**1.14E-01**	+
f_8	2.13E+01	2.20E-02	2.13E+01	2.39E-02	=	2.13E+01	2.51E-02	2.13E+01	2.86E-02	=
f_9	8.45E+02	2.15E+01	**7.94E+02**	**1.72E+01**	+	7.17E+02	2.91E+01	**6.76E+02**	**2.43E+01**	+
f_{10}	1.16E+03	2.57E+01	1.16E+03	2.67E+01	=	9.78E+02	4.05E+01	9.74E+02	3.70E+01	=
f_{11}	1.62E+02	2.12E+00	1.61E+02	1.76E+00	=	1.61E+02	1.95E+00	1.62E+02	1.38E+00	=
f_{12}	1.45E+07	6.02E+05	**1.33E+07**	**7.53E+05**	+	5.68E+05	2.22E+05	5.82E+05	2.56E+05	=
f_{13}	4.36E+05	6.65E+04	**2.69E+05**	**3.77E+04**	+	8.01E+01	2.83E+00	**7.32E+01**	**2.40E+00**	+
f_{14}	4.80E+01	1.80E-01	4.79E+01	1.57E-01	=	4.79E+01	1.95E-01	4.79E+01	1.83E-01	=
	Total Number of (+/=/-)				9/5/0	Total Number of (+/=/-)				6/8/0
f_i	DE/c-t-b/2/bin		DE/c-t-b/2/Mexp			DE/c-t-r/2/bin		DE/c-t-r/2/Mexp		
(100D)	Mean	Std	Mean	Std	+/=/-	Mean	Std	Mean	Std	+/=/-
f_1	1.13E-10	2.33E-11	**4.28E-11**	**1.23E-11**	+	4.47E-04	8.26E-05	**2.91E-04**	**4.56E-05**	+
f_2	2.41E+05	2.26E+04	**1.08E+05**	**1.13E+04**	+	3.44E+05	1.84E+04	**1.99E+05**	**1.04E+04**	+
f_3	1.09E+09	1.50E+08	**9.39E+08**	**1.41E+08**	+	1.89E+09	1.83E+08	**1.76E+09**	**1.58E+08**	+
f_4	2.99E+05	3.70E+04	**1.78E+05**	**1.71E+04**	+	4.17E+05	3.77E+04	**2.79E+05**	**1.72E+04**	+
f_5	2.01E+04	1.09E+03	**1.93E+04**	**1.52E+03**	+	3.17E+04	1.19E+03	**3.09E+04**	**1.69E+03**	+
f_6	9.12E+01	3.13E-01	**7.48E+01**	**1.07E+00**	+	9.42E+02	1.18E+02	**3.92E+02**	**6.12E+01**	+
f_7	7.44E-01	3.09E-02	**6.60E-01**	**4.49E-02**	+	4.60E+01	9.81E+00	**2.96E+01**	**5.18E+00**	+
f_8	2.13E+01	2.85E-02	2.13E+01	2.32E-02	=	2.13E+01	2.24E-02	2.13E+01	1.96E-02	=
f_9	7.37E+02	2.11E+01	**6.99E+02**	**2.23E+01**	+	7.62E+02	1.32E+01	**7.19E+02**	**1.84E+01**	+
f_{10}	9.67E+02	1.79E+01	**9.56E+02**	**2.45E+01**	+	1.01E+03	1.42E+01	1.01E+03	2.64E+01	=
f_{11}	1.62E+02	1.62E+00	1.61E+02	1.87E+00	=	1.62E+02	1.57E+00	1.61E+02	2.19E+00	=
f_{12}	5.75E+06	1.66E+06	5.84E+06	1.46E+06	=	1.26E+07	6.95E+05	**1.17E+07**	**7.92E+05**	+
f_{13}	7.95E+01	2.13E+00	**7.28E+01**	**1.60E+00**	+	1.08E+02	5.86E+00	**8.79E+01**	**3.09E+00**	+
f_{14}	4.78E+01	2.01E-01	4.78E+01	2.55E-01	=	4.78E+01	1.97E-01	4.79E+01	1.56E-01	=
	Total Number of (+/=/-)				10/4/0	Total Number of (+/=/-)				10/4/0

The symbols "+/=/-" indicate the statistical test results using Wilcoxon rank sum test at the 5% significance level. "+" and "-" mean that the performance of the variant with multiple exponential recombination is significantly superior or inferior than that of the variant with binomial recombination, respectively. "=" means the performance difference between the two variants is not statistically significant. "c-t-r" and "c-t-b" represent "current-to-rand" and "current-to-best", respectively.

to-best/2" and "DE/current-to-rand/2", the proposed crossover operator is able to $significantly$ improve the performance of the algorithm in most benchmark problems under both 50 dimensions and 100 dimensions. For "DE/best/2", the "Mexp" version $significantly$ outperforms the "bin" version in 9 out of 14 50-dimensional problems and 6 out of 14 100-dimensional problems. For "DE/best/1", the proposed strategy also provides $significantly$ better performance on 6 out 14 benchmarks in both 50-dimensional and 100-dimensional cases. As discussed before, the mutation probabilities are almost the same in these two crossover operations under identical Cr values. Thus, the different performances of these two operators only result from their different behaviors in selecting the exchange positions.

It is notable that for all the mutation strategies, in no benchmarks can binomial recombination show a $significantly$ better performance than the proposed operator. To seek the reason, the specially designed mechanism enables multiple exponential recombination to be more robust than binomial recombination in tackling different types of dependent subsets. Compared to binomial recombination, multiple exponential recombination not only handle the distant variables similarly but also possess a higher chance to preserve the physically proximate dependent subsets. Among all tested problems, only f_1 and f_9 are separable problems. A very interesting phenomenon is that even for these two separable problems, multiple exponential recombination is able to $significantly$ outperform traditional binomial recombination in most situations. The explanation is that some favorable variable configurations may form during the optimization process and multiple exponential recombination is more likely to preserve these

structures when the involved variables are physically proximate to each other.

To give a clearer picture of the performance difference between the two crossover operators, Fig. 3.6 plots the change of best-of-run errors over generations for each algorithms on 6 problems. It can be observed that all the "Mexp" variants converge towards the global optimum faster than the "bin" counterparts during the entire optimization process. A more effective search is achieved by multiple exponential recombination in solving these problems. The different convergence speeds of each mutation strategy are beyond the scope of our discussion.

3.3.3 Implementation in SaDE

In this subsection, the proposed multiple exponential recombination is implemented in SaDE [30], which is a very powerful state-of-the-art DE variant. The original SaDE will self-adapt the control parameter values and mutation strategies by learning from the successful experiences in generating promising solutions. However, the crossover operator in the original SaDE is fixed as binomial recombination due to its popularity in other DE literatures [30]. During the experiments, multiple exponential recombination will replace binomial recombination in crossover operation and all the other mechanisms of the original SaDE will be reserved. The performance of the modified SaDE will be compared with that of the original SaDE.

The purpose of this implementation is to further investigate the behaviors of the proposed operator when incorporated with self-adaptive DE variant. Both the 50-dimensional and 100-dimensional versions of f_1-f_{14} from CEC-05

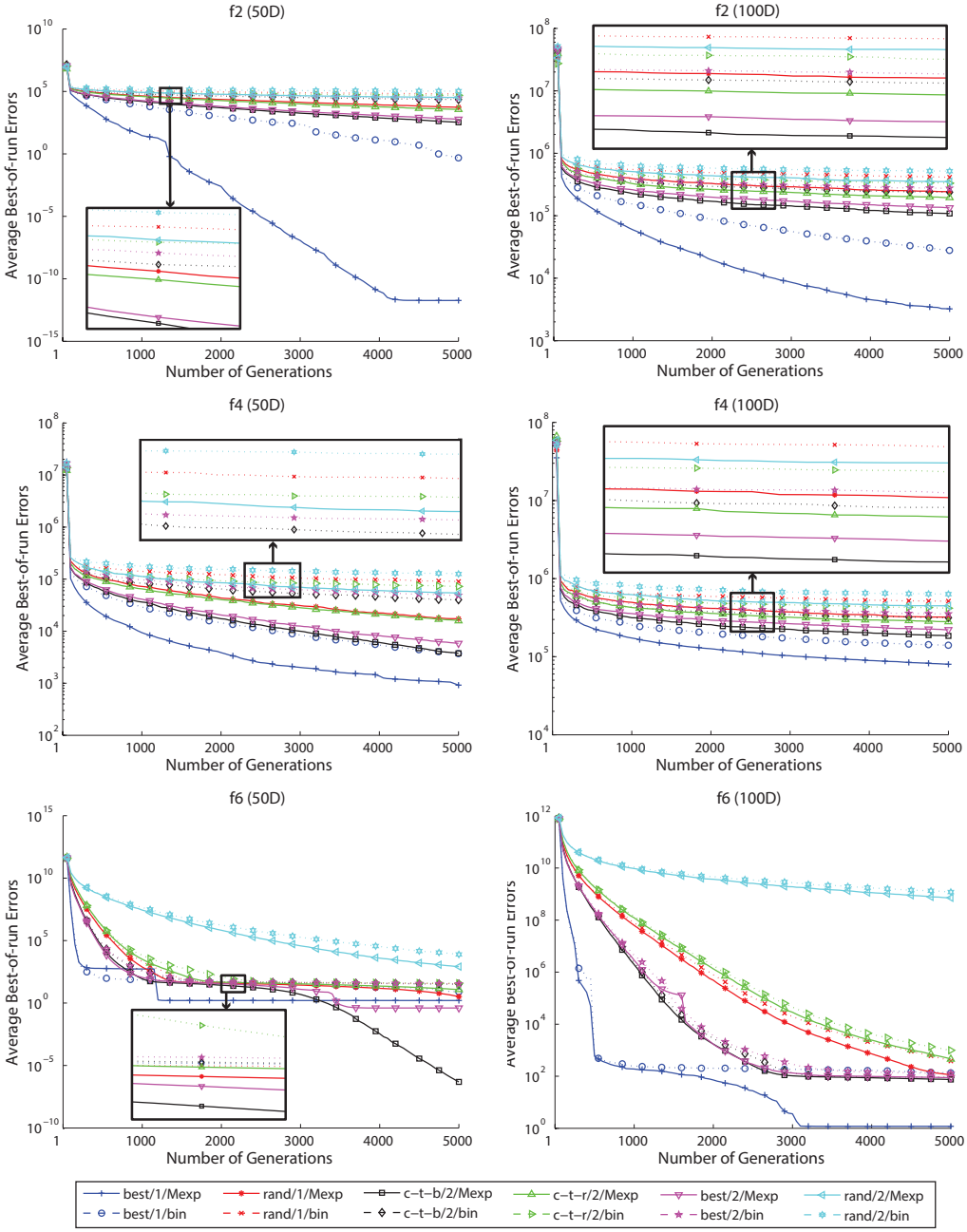

Figure 3.6: This figure plots the average best-of-run errors of 30 independent runs over generations. "c-t-r" and "c-t-b" represent "current-to-rand" and "current-to-best", respectively.

benchmark suite will be tested. For both algorithms, the population size is fixed as 100 and the maximum number of fitness evaluation is set to 50×10^4 for 50-dimensional problems and 100×10^4 for 100-dimensional problems. T is fixed as 10 for multiple binomial recombination. The Cr value used in the two crossover operators will be adapted via the parameter adaptation mechanism in SaDE. For all the remaining parameters in SaDE, the recommended parametric setup in the original literature [30] is utilized. Table 3.3 presents the mean and standard deviation of the best-of-run errors over 30 independent runs for each algorithm on each problem. Wilcoxon rank-sum test is conducted at the 5% significance level for each pair of results. The entries that $significantly$ outperform the counterparts are marked in boldface.

From Table 3.3, the modified SaDE variant with multiple exponential recombination provides $significantly$ better performance than the original version with binomial recombination in 11 out of 28 benchmarks. In all the remaining 17 problems, the performance difference between the two variants is not statistically significant. This observation indicates that replacing binomial recombination with multiple exponential recombination has enhanced the robustness of SaDE in solving problems with different types of variable interrelations.

Subsequently, the behaviors of the parameter adaptation mechanism in tuning Cr values are studied. In SaDE, the adaptation of Cr value is based on a normal distribution with mean value CRm (please note this is the control parameter in the original SaDE, it is different from our control parameter Cr_m) and standard deviation 0.1. Each mutation strategy in the candidate pool will

Table 3.3: Mean and Standard Deviation of the Errors (30 runs)

Problems	Original SaDE	SaDE/Mexp
(Dimensions)	Mean ± Std	Mean± Std
f_1 (50)	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
f_2 (50)	3.20E-10 ± 4.39E-10	**4.36E-16 ± 7.91E-16**
f_3 (50)	8.87E+04 ± 3.47E+04	8.97E+04 ± 4.32E+04
f_4 (50)	7.48E+02 ± 7.22E+02	**3.13E+01 ± 4.15E+01**
f_5 (50)	3.57E+03 ± 5.87E+02	3.52E+03 ± 6.40E+02
f_6 (50)	6.05E+00 ± 1.55E+01	**1.99E+00 ± 2.03E+00**
f_7 (50)	5.66E-03 ± 1.04E-02	**3.44E-03 ± 8.16E-03**
f_8 (50)	2.11E+01 ± 4.94E-02	2.11E+01 ± 3.60E-02
f_9 (50)	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
f_{10} (50)	9.39E+01 ± 1.38E+01	9.28E+01 ± 1.37E+01
f_{11} (50)	3.43E+01 ± 9.58E+00	3.27E+01 ± 9.26E+00
f_{12} (50)	9.60E+03 ± 6.05E+03	**7.17E+03 ± 7.25E+03**
f_{13} (50)	9.34E+00 ± 6.13E-01	**8.46E+00 ± 5.77E-01**
f_{14} (50)	2.24E+01 ± 1.91E-01	2.24E+01 ± 2.38E-01
f_1 (100)	1.68E-30 ± 9.22E-30	1.68E-30 ± 9.22E-30
f_2 (100)	3.28E-03 ± 3.21E-03	**6.59E-05 ± 9.53E-05**
f_3 (100)	9.43E+05 ± 2.48E+05	9.10E+05 ± 3.00E+05
f_4 (100)	3.50E+04 ± 6.25E+03	**1.98E+04 ± 4.94E+03**
f_5 (100)	9.14E+03 ± 1.29E+03	9.48E+03 ± 1.31E+03
f_6 (100)	4.88E+01 ± 3.25E+01	**2.52E+00 ± 1.95E+00**
f_7 (100)	5.25E-03 ± 7.33E-03	4.60E-03 ± 6.65E-03
f_8 (100)	2.13E+01 ± 2.17E-02	2.13E+01 ± 2.50E-02
f_9 (100)	1.33E+00 ± 1.09E+00	1.53E+00 ± 1.22E+00
f_{10} (100)	3.14E+02 ± 3.79E+01	3.12E+02 ± 3.97E+01
f_{11} (100)	8.86E+01 ± 5.96E+00	8.86E+01 ± 7.10E+00
f_{12} (100)	4.20E+04 ± 1.63E+04	**3.67E+04 ± 2.56E+04**
f_{13} (100)	2.94E+01 ± 1.39E+00	**2.58E+01 ± 1.81E+00**
f_{14} (100)	4.67E+01 ± 2.86E-01	4.67E+01 ± 2.87E-01

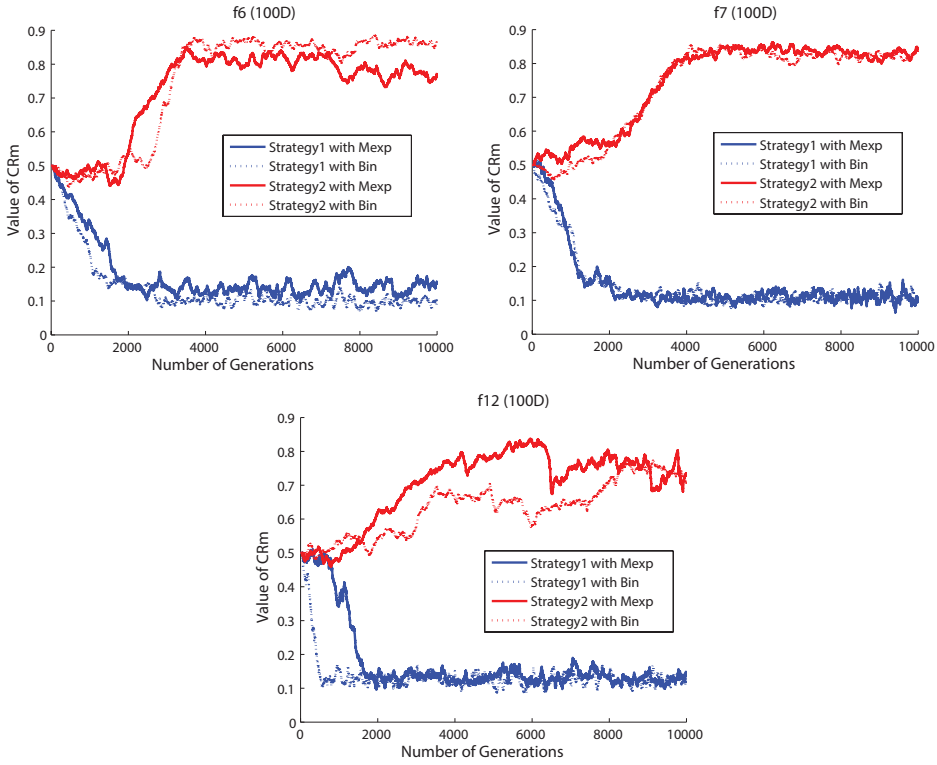

Figure 3.7: This figure shows the adaptation behaviors of the CRm values in SaDE. Strategy 1 and Strategy 2 are the first two default candidate mutations strategies in original SaDE, namely, DE/rand/1 and DE/current-to-best/2.

have one associated CRm and the value of this CRm is adapted according to previous Cr values that have generated promising trial vectors. Fig. 3.7 plots the change of CRm values during optimization process for the first two mutation strategies in both modified and original SaDE on 100-dimensional f_6, f_7 and f_{12}. According to Fig. 3.7, changes of CRm values in both operators show similar pattern. For strategy 1, both the operator tends to preserve higher CRm values whereas in strategy 2, lower CRm values are preferred by both crossover operations. Considering that the Cr values are generated based on CRm, the distribution of actually used Cr values will be similar in both variants . Thus, the performance difference between the two variants are not caused by utilizing different Cr values. All the improvements in the "Mexp" variants are mainly due to the new mechanisms in selecting crossover positions. Another observation is

that either the value above 0.8 or below 0.2 is preferred by the Cr adaptation mechanisms in SaDE. Based on the simulation results in Fig. 3.5, when the Cr value is close to 0 or 1, the disruption frequency distribution will become similar for both crossover operators. As a result, the performance difference between the two SaDE variants may not be $significant$ in some benchmark problems.

3.4 Conclusion

This chapter has presented a new DE crossover operator, in which multiple segments will be exchanged among involved vectors. The specially designed mechanisms enable the proposed strategy to inherit all the main advantages of existing crossover operators while providing a more robust performance in handling variable interrelations. The properties of the new operator is investigated both theoretically and empirically. Implementation in 6 classical DE algorithms and 1 adaptive DE variant demonstrates the superiority of the proposed strategy in solving problems with unknown variable interrelations.

Chapter 4

Adaptive Cross-Generation Differential Evolution Operators for Multi-objective Optimization

In previous chapter, the target problems are restricted to the general single-objective optimization problems due to the nature of traditional DE algorithms. However, many optimization problems involve multiple objectives that are conflicting to each other. In order to circumvent the limitation of DE in solving MOPs, this chapter proposes a novel multi-objective DE algorithm, which will make use of the information across generations to improve the overall performance. Superiority of the proposed mechanisms are validated via experiments.

4.1 Introduction

DE, proposed by Storn and Price [9], is arguably one of the most efficient evolutionary algorithms for numerical optimization. Due to its inherent superiority

over many EAs [10], DE generally produces good performance in a wide variety of single-objective optimization problems (SOPs) [119–123]. This success triggered the popularity of extending DE to multi-objective optimization (MO) in recent years.

Intrinsically, the main task of any evolutionary multi-objective optimization (EMO) algorithm is to achieve best trade-offs among multiple objectives. In practice, EMO algorithms operate by evolving a set of solutions to approximate the Pareto Optimal Front (POF), which is defined by the ideally optimal trade-offs [124–126]. How to distribute these solutions along the POF more evenly, which means a better diversity of the population, and more closely, which means a faster convergence speed of the algorithm, then becomes the core problem during the design of EMO algorithms. When extending DE to MO, researchers focus more on maintaining diversity and developing survival selection criterion [127]. Although impressive progress has been reported, there are still several issues that tend to be neglected: first, the performance of DE is sensitive to the setup of its control parameters [10,29–31]. It emerges as a thorny issue when multiple objectives are optimized simultaneously [68, 99, 128]; second, while much effort has been paid to maintain the diversity of the population, a satisfactory convergence speed cannot be guaranteed [129] [130]; third, most existing MO-DEs are inconvenient to implement among different MO frameworks since they have already fixed the whole evolutionary process [60, 61, 71, 72] or are tightly combined with one category of MO frameworks (e.g., Pareto dominance-based or decomposition-based) [62, 66, 67, 70, 131–133]. During applications, varied types of MO frameworks are needed depending on the nature of the prob-

lem [7, 8, 74, 128]. It is desirable to design MO-DE in a more flexible way.

To address the above issues, this chapter proposes a novel multi-objective Differential Evolution algorithm utilizing information across generations. First, two new mutation strategies are developed to enhance both convergence speed and diversity maintenance. One is the neighborhood-based cross-generation (NCG) mutation, which tries to estimate the ideal searching directions that can guide the optimization. In general thought, defining an efficient direction is not a trivial task, especially when the evolution process involves non-linear or black-box functions [75]. Considering the essence of evolutionary algorithms, solutions are evolved better and better throughout the whole optimization. Thus, the differences between consecutive generations are analyzed in NCG mutation to generate promising searching directions. Another is the population-based cross-generation (PCG) mutation, in which populations from two generations are involved to facilitate a more explorative search. The two new mutation operators are employed in a hybrid manner in order to strike a delicate balance between exploitation and exploration. Subsequently, a new scheme called cross-generation adaptation (CGA) mechanism is introduced to tune the parameters automatically. For DE, the optimal parametric setting varies over different optimization problems. Even for a single problem, the proper parameters may change over generations. In MO, this issue becomes more challenging because of the possible distinct requirements by multiple objectives. The proposed CGA mechanism self-adapts the parameters not only in a dynamic way but also from the perspective of objective space. In each generation, the optimal parameters for a limited region in objective space are gauged via integrating the parame-

ter values of consecutive generations. Based on the aforementioned algorithmic components, an adaptive cross-generation differential evolution (ACGDE) algorithm for multi-objective optimization is developed. There is no dedicated survival selection criterion in ACGDE, thereby leading to sufficient flexibility in implementation. To implement ACGDE into a MO framework, the main step is to replace the original reproduction operator (e.g., mutation and crossover operators) with ACGDE. Regeneration of more promising offspring is the principal role of ACGDE. To the best of the author's knowledge, there is no previous study that focuses on examining the usability of information across generations. The experimental results validate the effectiveness of each algorithmic component. The implementation into two well-known MO frameworks (i.e., NSGA-II [134] and MOEA/D [135]) demonstrates that ACGDE is very powerful and robust in handling multi-objective optimization problems (MOPs).

4.2 Proposed Algorithm

This section outlines the proposed ACGDE and discusses the details of each algorithmic component.

4.2.1 Overview of the Algorithm

This chapter presents an adaptive Differential Evolution algorithm for multi-objective optimization. Two novel mutation strategies and one new parameter adaptation mechanism are developed. The proposed ACGDE differs from the classical DE algorithms in the following aspects: first, information across different generations will be utilized to enhance both convergence and diversity;

second, the relationships among individuals in objective space are considered during selection of parents; third, adaptation of parameters operates from an objective-based perspective.

Algorithm 4.1 shows the general structure of ACGDE. Detailed explanations for the newly introduced mechanisms, parameters and concepts will be provided in the following subsections.

4.2.2 Cross-Generation Mutation

Converging Direction and Searching Direction

To better explain the development of our idea, two kinds of directions are defined, namely, converging direction and searching direction. Similar to convergence direction in [75], converging direction is defined as the direction that approaches the POF given a solution in the objective space, whereas searching direction is referred to the moving direction of an individual in the decision space. It is intuitive that the most efficient searching direction of an individual is the mapping of its converging direction. Thus, it would be beneficial if the correct converging directions could be learned during the optimization process. Although the exact converging directions are unavailable, it is possible to estimate the converging directions using the information across generations. Considering the nature of evolutionary algorithms, the solutions are getting better and better while the number of evolving generations increases. In MO, such selection pressure will move the population towards the true POF. The movement of solutions across generations may, therefore, provide some hints for estimating the converging directions. As shown in Fig. 4.1, the converging direction varies

Algorithm 4.1 Procedure of ACGDE

Set the control parameters of ACGDE: population size N, Neighborhood size $T = 5\% \cdot N$, θ_F, θ_{Cr}, $F_{\max}$, $F_{\min}$, $Cr_{\max}$ and $Cr_{\min}$.
Set the generation number $g = 0$ and randomly initialize the population of N individuals $P_g = \{X_{1,g}, X_{2,g}, ..., X_{N,g}\}$ together with their associated parameters $\{F_{1,g}, F_{2,g}, ..., F_{N,g}\}$ and $\{Cr_{1,g}, Cr_{2,g}, ..., Cr_{N,g}\}$. The generated parameter values should be within $[F_{\min}, F_{\max}]$ and $[Cr_{\min}, Cr_{\max}]$, respectively.
for $g = 1$ to $G_{\max}$ **do**
  if $g = 1$ **then**
    $P_g = P_0$
  end if
  Calculate the sub-rank for each individual in P_g.
  for $i = 1$ to N **do**
    Determine the neighborhood of current generation and previous generation for individual $X_{i,g}$ in terms of sub-rank.
    Perform CGA mechanism to produce $F_{\text{spring},i,g}$ and $Cr_{\text{spring},i,g}$.
    if $rand[0, 1.0] \leq 0.5$ **then**
      Utilize NCG mutation with $F_{\text{spring},i,g}$ to generate the mutant vector $V_{i,g}$.
    else
      Utilize PCG mutation with $F_{\text{spring},i,g}$ to generate the mutant vector $V_{i,g}$.
    end if
    Perform binomial recombination with $Cr_{\text{spring},i,g}$ on $X_{i,g}$ and $V_{i,g}$ to generate the offspring $U_{i,g}$.
    Survival selection by the MO framework that ACGDE is implemented to.
  end for
  Generation number $g = g + 1$
  Update Current Population P_g
end for

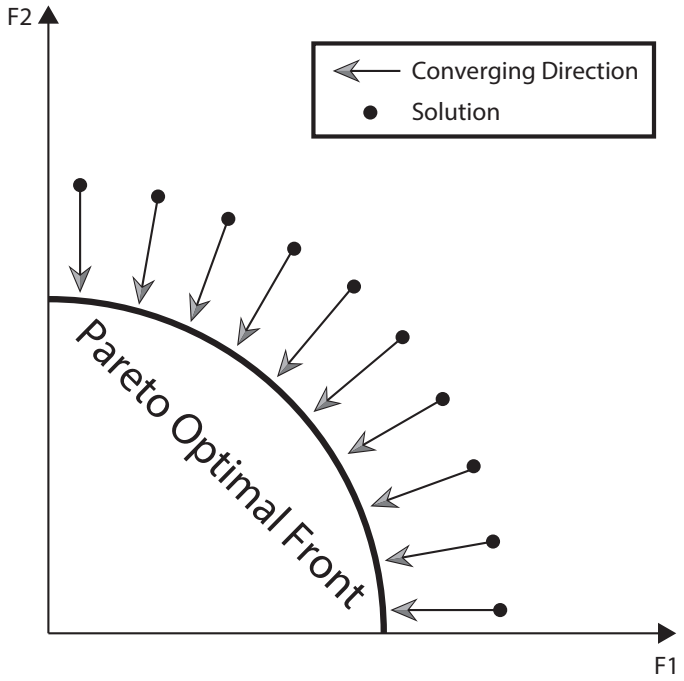

Figure 4.1: Converging directions of different solutions in the objective space. The curve denotes the Pareto Optimal Front of the 2-objective minimization problem. Each dot represents a solution and the arrows denote their converging directions.

for the solutions from different regions in objective space. It is more reasonable to separately estimate converging directions within different regions. Defining neighborhood for each individual is an effective way to extract information over a certain limited area in objective space. In the following subsection, a simple yet efficient way to define the neighborhood will be discussed.

Neighborhood Based on Sub-rank

In MO, a common way to define the neighborhood of a solution in objective space is to measure the Euclidean distance between two solutions based on their objective values. The major weakness of this method is that the calculated distance may be biased by the different ranges of objective functions [126, 136, 137]. Under these circumstances, normalization is a necessary preprocessing step [138, 139]. Nevertheless, it cannot be guaranteed that the true

range of each objective is known before optimization [136]. Normalization becomes difficult without knowledge of true ranges. To circumvent this issue, a new term, called "sub-rank", is introduced. Similar to the ranking in [140], sub-rank is a vector comprising the ranks of an individual in each separate objective among current population. For instance, in a 2-objective problem, if the sub-rank of one individual is $(1, 1)$, it means this individual has the best fitness value for both objectives. In the proposed algorithm, sub-rank would replace fitness value to measure the distance between two individuals in objective space. In this way, normalization could be skipped and the bias caused by various ranges is eliminated. In the proposed algorithm, firstly the Euclidean distance is calculated using sub-rank between any two individuals. Next, the neighborhood of each individual will be decided based on the calculated distance and the pre-defined neighborhood size T. The T nearest solutions are marked as the neighborhood of this individual.

Neighborhood-based Cross-Generation Mutation

Classical DE mutation operators only make use of the information within the current generation to generate the mutant vector [9, 141, 142]. However, based on the above discussions, the information across generations may reveal the trend of how solutions moved in the search space, and in turn help to guide searching directions. In order to estimate the converging directions for different regions in objective space and utilize the corresponding searching directions to guide the evolutionary process, a novel neighborhood-based cross-generation (NCG) mutation strategy is proposed. The mutant vector is generated based on

the difference between two individuals from consecutive generations.

In the proposed strategy, each individual will generate one mutant vector, and the individual is called the main parent of this mutant vector. One interesting feature of the proposed mutation strategy is that the main parent is not involved in the mutation process. Instead, the mutant vector is generated from the two neighborhood pools of the main parent. One neighborhood pool is formed by T individuals selected from the population of current generation, another consists of T individuals picked from the population of previous generation. More specifically, given a main parent, first the Euclidean distance from it to all the other individuals at the current generation is computed using sub-rank, then the T nearest individuals are marked to become the neighborhood of current generation for this main parent. Similarly, the distance between this main parent and all the individuals of previous generation would be calculated based on sub-rank, and the T nearest individuals are recorded as the main parent's neighborhood of previous generation. With these two neighborhood pools, the new mutation operation can be conducted following the formula below:

$$V_{i,g} = X_{rn_1,g} + F \cdot (X_{rn_1,g} - X_{rn_2,g-1}) \tag{4.1}$$

where $V_{i,g}$ denotes the new generated individual, called the mutant vector, and i is the index of the main parent, g is the number of current generation. $X_{rn_1,g}$ is an individual randomly selected from the main parent's neighborhood of current generation, where index rn_1 is a randomly selected integer from $\{I_{i,g,1}, I_{i,g,2}, I_{i,g,3}, \ldots, I_{i,g,T}\}$, which records the indices of neighborhood, and T is the pre-defined neighborhood size. $X_{rn_2,g-1}$ is an individual randomly selected from the main parent's neighborhood of previous generation, where index

rn_2 is a randomly selected integer from the recorded neighborhood indices set $\{I_{i,g-1,1}, I_{i,g-1,2}, I_{i,g-1,3}, \ldots, I_{i,g-1,T}\}$.

The underlying rationale of the designed strategy is that the movement of the solutions during evolutionary process may assist to guide the subsequent search direction. Following the proposed formula, $X_{rn_1,g}$ stays in the current generation while $X_{rn_2,g-1}$ comes from the previous generation, so taking into account that both are selected from the neighborhood pools of the same main parent, the difference between them may reveal the moving trend of the solutions near the main parent in objective space. Since the essence of evolutionary algorithm is to evolve solutions, the better solutions would have a higher chance to survive to next generation and the weaker individuals will be discarded. The moving directions of the solutions may be close to the true converging directions. Adding the difference between $X_{rn_1,g}$ and $X_{rn_2,g-1}$ as a perturbation to $X_{rn_1,g}$ is equivalent to making the current solution keep exploring based on the searching direction corresponding to the estimated converging direction. It is notable that proper selection of the neighborhood size T is critical to the overall performance of the algorithm. One extreme case is to set T as 1, then the algorithm may get stuck because of the overlap of $X_{rn_1,g}$ and $X_{rn_2,g-1}$. Conversely, if the neighborhood size is too large, it will become difficult to estimate the converging direction near $X_{i,g}$ since $X_{rn_1,g}$ and $X_{rn_2,g-1}$ can be selected from distant regions in objective space. Based on our parametric sensitivity tests (see subsection 4.3.4), it is recommended to choose neighborhood size T as 5% of population size. Fig. 4.2 illustrates the procedure of the proposed NCG mutation and the relationship among involved individuals. In objective space, the

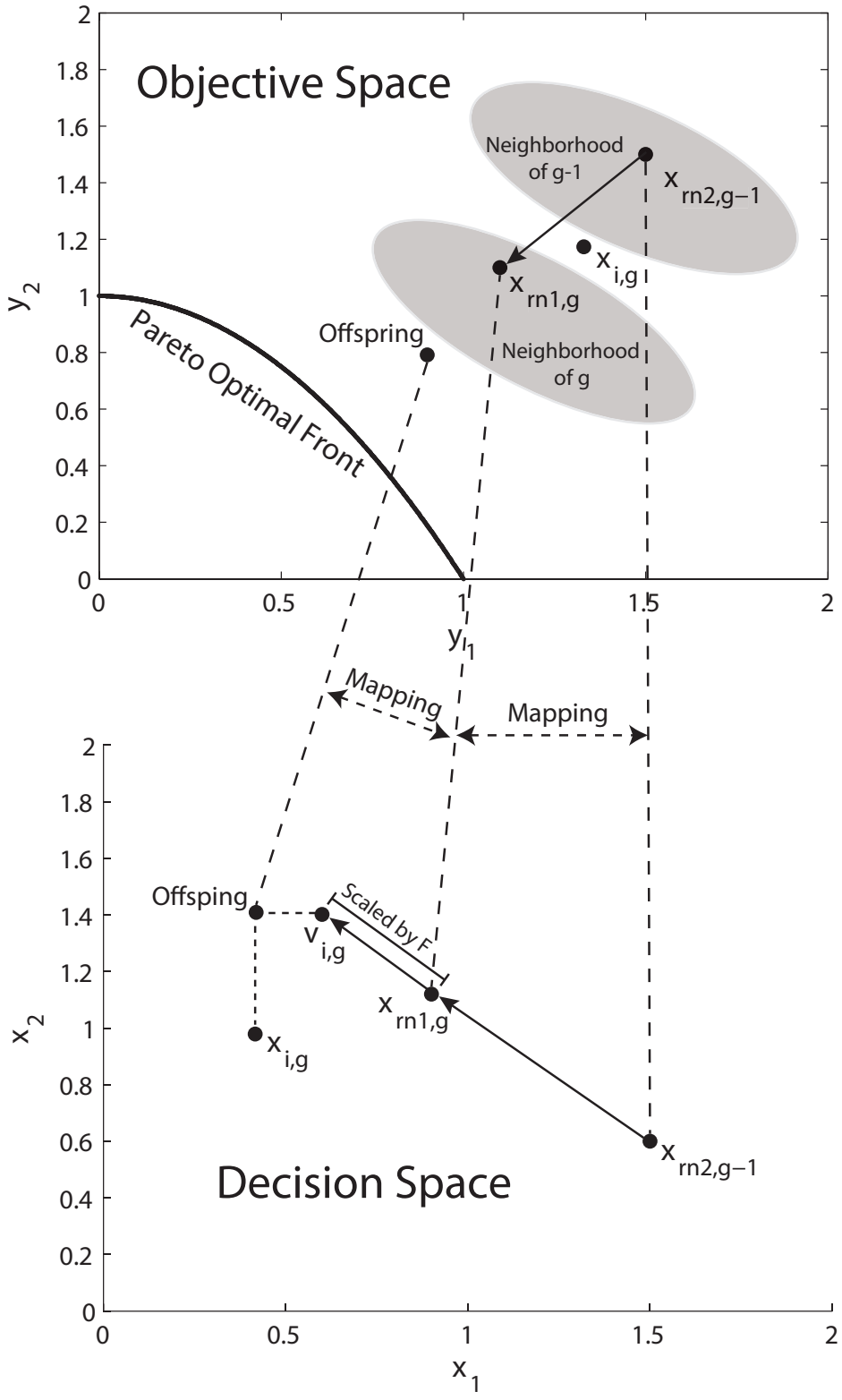

Figure 4.2: Procedure of NCG mutation and the relationship among involved individuals. Both the objective functions are minimized. The Pareto Optimal Front, variable values and objective values are arbitrarily generated for convenience of illustration.

vector from $X_{rn_2,g-1}$ to $X_{rn_1,g}$ is the estimated converging direction. In decision space, the difference of the two individuals decides the corresponding searching direction. Subsequently, $X_{rn_1,g}$ will keep exploring following the searching direction to generate the mutant vector $V_{i,g}$. The searching step is scaled by the parameter F. Offspring is produced via recombination of main parent $X_{i,g}$ and mutant vector $V_{i,g}$. The effectiveness of the obtained searching direction is validated through a contrast experiment in subsection 4.3.3. Algorithm 4.2 presents the pseudo-codes for implementing procedures of NCG mutation.

Algorithm 4.2 Procedure of NCG Mutation

Require:

P_g: Population at generation *g*
P_{g-1}: Population at generation *g-1*
$X_{i,g}$: Main parent
F: Scaling factor, in ACGDE, it is generated by CGA mechanism
T: Neighborhood size

Ensure:

$V_{i,g}$: Mutant vector

1: Calculate Euclidean Distance from $X_{i,g}$ to all the other individuals in P_g using sub-rank
2: Based on the calculated distances, the T nearest individuals in P_g are recorded as $X_{i,g}$'s neighborhood of current generation $P_{neighbor,i,g}$
3: Calculate Euclidean Distance from $X_{i,g}$ to all the individuals in P_{g-1} using sub-rank
4: Based on the calculated distances, the T nearest individuals in P_{g-1} are recorded as $X_{i,g}$'s neighborhood of previous generation $P_{neighbor,i,g-1}$
5: Randomly pick one individual $X_{rn_1,g}$ from $P_{neighbor,i,g}$
6: Randomly pick one individual $X_{rn_2,g-1}$ from $P_{neighbor,i,g-1}$
7: Generate mutant vector $V_{i,g} = X_{rn_1,g} + F \cdot (X_{rn_1,g} - X_{rn_2,g-1})$

Population-based Variant

One common issue with neighborhood-based mutation strategies is that the searching step may become excessively small because of the gradually converging population [11,37,74,135]. Since the individuals involved in NCG mutation are selected from neighborhoods of the same main parent, the explorative ability

of the algorithm is impaired. In order to compensate for the above-mentioned weakness of NCG mutation, another new mutation strategy is proposed, namely, population-based cross-generation (PCG) mutation. The NCG mutation and PCG mutation will be utilized simultaneously so that a reasonable trade-off between exploitation and exploration can be achieved. The details of the PCG mutation operation are shown as follows:

$$V_{i,g} = X_{i,g} + F \cdot (X_{rp_1,g} - X_{rp_2,g-1}) \tag{4.2}$$

where $V_{i,g}$ denotes the new generated mutant vector, and i is the index of the main parent, g is the number of current generation. $X_{i,g}$ is the main parent. $X_{rp_1,g}$ is an individual randomly selected from the whole population in current generation, where index rp_1 is a randomly selected integer from $\{1, 2, 3, \ldots, N\}$ and N is the population size. $X_{rp_2,g-1}$ is an individual randomly selected from the whole population of the previous generation, where index rp_2 is a randomly selected integer from $\{1, 2, 3, \ldots, N\}$.

Unlike the NCG mutation, the difference vector in PCG mutation is computed with two individuals randomly selected from the whole population, which is similar to the traditional DE mutation strategy. However, because the two individuals come from two different generations, the scale of their difference will vary larger than the original DE mutation strategy. In this way, the explorative ability could be further enhanced. Another special modification of the population-based variant is that the difference vector would be added to the main parent directly. By this means, the searching will continue centering on the main parent instead of performing a purely stochastic exploration. This could lead to a more efficient optimization process. Algorithm 4.3 describes the procedures

of PCG mutation.

Algorithm 4.3 Procedure of PCG Mutation

Require:

P_g: Population at generation *g*
P_{g-1}: Population at generation *g-1*
$X_{i,g}$: Main parent
F: Scaling factor, in ACGDE, it is generated by CGA mechanism

Ensure:

$V_{i,g}$: Mutant vector

1: Randomly pick one individual $X_{rp_1,g}$ from P_g
2: Randomly pick one individual $X_{rp_2,g-1}$ from P_{g-1}
3: Generate mutant vector $V_{i,g} = X_{i,g} + F \cdot (X_{rp_1,g} - X_{rp_2,g-1})$

In ACGDE, the above two strategies are employed in a half-half manner, which means they have the same probability (50%) to be utilized during each mutation operation. Parametric sensitivity of this ratio would be analyzed via experiments in subsection 4.3.5. Simulation results show that the performance of the algorithm is better than, or at least comparable to, the versions with only one of them in most testing problems. A final note about the new mutation strategy is that each individual in the current generation would be a main parent of a mutant vector, and after the above mutation operation, each mutant vector still needs to go through a binomial recombination (crossover operation) with the main parent to generate the final offspring.

Enhancing Both the Diversity and Convergence

When designing a multi-objective optimization algorithm, it is difficult to hasten the convergence speed without loss of diversity as if they were conflicting to each other. Nonetheless, the specially designed mechanism enables the proposed approach to enhance convergence along with diversity. In the NCG mutation, the convergence is sped up by guiding the searching direction with

information across generations. Meanwhile, due to the neighborhood-based selection mode, the final offspring will stay around the main parent in objective space so that the diversity of the population could be consistently maintained. Analogously, the parent-centric search in PCG mutation helps preserve the diversity, whilst the more explorative mutation operator contributes to a higher converging speed to true POF.

4.2.3 Cross-Generation Adaptation Mechanism

Parametric Sensitivity in Multi-objective Optimization

Sensitivity to the parametric setting is a critical issue during the extension of DE into MO [68, 99, 128]. The performance of original DE may vary tremendously with different selections of parameters [10, 29–31]. This problem becomes more challenging in solving MOPs. Multiple objectives may have disparate requirements for the parametric setup, and even for a certain objective, the optimal setup of parameters may vary over different searching stages, e.g., a large searching step is needed at the start of exploration whereas a relatively smaller searching scope is preferred near the end of the search. Hence, it is desirable to have an adaptation mechanism with the following features: first, the scheme is able to estimate the appropriate parameters for a specific region in objective space; second, there is a selection pressure to reserve more proper parameters and discard poor parameters; third, as the current fitted parameters may not conform to the requirement of the next searching stage, complete convergence of parameters should be avoided. In order to achieve the above targets, a novel adaptation mechanism that makes use of the information across genera-

tions is proposed in this chapter.

Proposed Adaptation Mechanism

In the proposed Cross-Generation Adaptation (CGA) mechanism, the two control parameters in DE, namely scaling factor F and crossover probability Cr, are applied at the individual level. Each individual in the population is associated with its own F and Cr. The neighborhood pools utilized in the NCG mutation also contributes to the reproduction of parameters for offspring.

During initialization, for each individual X_i, the associated F_i and Cr_i are randomly generated from the pre-defined interval $[F_{\min}, F_{\max}]$ and $[Cr_{\min}, Cr_{\max}]$. After that, the parameter values for any newly generated individual will be determined by the following formula:

$$F_{\text{spring},i,g} = F_{\text{neighbor},i,g} + \theta_F \cdot Gaussian(0,1) \tag{4.3}$$

$$Cr_{\text{spring},i,g} = Cr_{\text{neighbor},i,g} + \theta_{Cr} \cdot Gaussian(0,1) \tag{4.4}$$

where $F_{\text{spring},i,g}$ and $Cr_{\text{spring},i,g}$ denote the parameter values assigned to the newly generated offspring, and i is the index of the main parent shared in the mutation operation, g is the current generation number. $Gaussian(0,1)$ is a random number sampled from a Gaussian Distribution accordingly with mean 0 and standard deviation 1. θ_F and θ_{Cr} are constants for scaling the outputs of Gaussian Distribution. $F_{\text{neighbor},i,g}$ and $Cr_{\text{neighbor},i,g}$ are produced in the following manner:

$$F_{\text{neighbor},i,g} = mean_A(NF_{i,g} \cup NF_{i,g-1}) \tag{4.5}$$

$$Cr_{\text{neighbor},i,g} = mean_A(NCr_{i,g} \cup NCr_{i,g-1}) \tag{4.6}$$

with $NF_{i,g}$ denoting the set of scaling factors associated with all the individuals in main parent's neighborhood of current generation, and $NF_{i,g-1}$ comprising all the scaling factors for main parent's neighborhood of previous generation. Same rule is also applied for generating the Cr sets $NCr_{i,g}$ and $NCr_{i,g-1}$. $mean_A$ stands for the simple arithmetic mean. If the calculated $F_{\text{spring},i,g}$ or $Cr_{\text{spring},i,g}$ falls outside the interval $[F_{\min}, F_{\max}]$ or $[Cr_{\min}, Cr_{\max}]$, its value will be set as the nearest bound.

Following the above adaptation operation, the offspring will then be generated via mutation and crossover utilizing the obtained $F_{\text{spring},i,g}$ and $Cr_{\text{spring},i,g}$. In other words, the offspring itself is produced only after the generation of its associated parameters. Once an individual is generated, its associated parameters will not be altered until the individual fails to survive to the next generation. Algorithm 4.4 shows the procedure of CGA mechanism.

Explanations of CGA mechanism

By encoding the control parameters at the individual level, CGA mechanism is able to evaluate multiple sets of parameters simultaneously and preserve the favorable control values for each solution. The computational complexity does not increase compared to the fixed parametric scheme in original DE [29]. The fundamental principle behind CGA mechanism is that suitable parameter values tend to generate better individuals, and favorable individuals help to propagate their associated parameters. Based on the above essence, CGA mechanism tries to estimate the most proper parametric setting for the current interested region in objective space. In SO, the target of optimization is to obtain the optimal

Algorithm 4.4 Procedure of CGA Mechanism

Require:

P_g: Population at generation *g*
P_{g-1}: Population at generation *g-1*
$X_{i,g}$: Main parent
T: Neighborhood size
θ_F, θ_{Cr}: constants for scaling the outputs of Gaussian Distribution
$F_{\min}$, $F_{\max}$, $Cr_{\min}$, $Cr_{\max}$: Boundaries for the new generated parameters

Ensure:

$F_{\text{spring},i,g}$: Newly generated scaling factor
$Cr_{\text{spring},i,g}$: Newly generated crossover probability

1: Calculate Euclidean Distance from $X_{i,g}$ to all the other individuals in P_g using sub-rank
2: Based on the calculated distances, the T nearest individuals in P_g are recorded as $X_{i,g}$'s neighborhood of current generation $P_{neighbor,i,g}$
3: Calculate Euclidean Distance from $X_{i,g}$ to all the individuals in P_{g-1} using sub-rank
4: Based on the calculated distances, the T nearest individuals in P_{g-1} are recorded as $X_{i,g}$'s neighborhood of previous generation $P_{neighbor,i,g-1}$
5: Create parameter set $NF_{i,g}$ by including all the associated F in $P_{neighbor,i,g}$
6: Create parameter set $NF_{i,g-1}$ by including all the associated F in $P_{neighbor,i,g-1}$
7: Create parameter set $NCr_{i,g}$ by including all the associated Cr in $P_{neighbor,i,g}$
8: Create parameter set $NCr_{i,g-1}$ by including all the associated Cr in $P_{neighbor,i,g-1}$
9: Calculate arithmetic mean of two parameter sets: $F_{\text{neighbor},i,g} = mean_A(NF_{i,g} \cup NF_{i,g-1})$
10: Calculate arithmetic mean of two parameter sets: $Cr_{\text{neighbor},i,g} = mean_A(NCr_{i,g} \cup NCr_{i,g-1})$
11: Generate new scaling factor $F_{\text{spring},i,g} = F_{\text{neighbor},i,g} + \theta_F \cdot Gaussian(0,1)$, where $Gaussian(0,1)$ is a random number generator based on a Gaussian Distribution with mean 0 and standard deviation 1
12: **if** $F_{\text{spring},i,g} > F_{\max}$ **then**
13: $\quad F_{\text{spring},i,g} = F_{\max}$
14: **else**
15: $\quad$ **if** $F_{\text{spring},i,g} < F_{\min}$ **then**
16: $\quad\quad F_{\text{spring},i,g} = F_{\min}$
17: $\quad$ **end if**
18: **end if**
19: Generate new crossover probability $Cr_{\text{spring},i,g} = Cr_{\text{neighbor},i,g} + \theta_{Cr} \cdot Gaussian(0,1)$
20: **if** $Cr_{\text{spring},i,g} > Cr_{\max}$ **then**
21: $\quad Cr_{\text{spring},i,g} = Cr_{\max}$
22: **else**
23: $\quad$ **if** $Cr_{\text{spring},i,g} < Cr_{\min}$ **then**
24: $\quad\quad Cr_{\text{spring},i,g} = Cr_{\min}$
25: $\quad$ **end if**
26: **end if**

solution for one objective, thus, the adaptation of parameters is performed without considering the individuals' location in objective space. However, when solving MOPs, multiple objectives may have disparate requirements for parametric setup and, hence, adjustment of parameters needs to be conducted in an objecitve-space-based manner. Relationship in terms of the position in objective space plays an important role in CGA mechanism. Two neighborhood pools, which are identical to those in NCG mutation, are defined for each individual. Based on the neighborhood pools of current generation and previous generation, parameter sets $NF_{i,g}$, $NF_{i,g-1}$, $NCr_{i,g}$ and $NCr_{i,g-1}$ are generated. As stated before, good parameters are more likely to produce individuals that are able to survive. The associated parameters of the neighborhood members, therefore, reflect the required parametric setting for current region. Calculating the mean of their parameters is a reasonable way to approximate the optimal setup and moderate the effect of possible outliers.

Similar to NCG mutation, selection of the neighborhood size T is to achieve a balance between robustness and accuracy. If the T is too large, then the calculated mean is incapable of revealing the requirement of current region. In case that the T is too small, the estimated results will depend heavily on the parameters of a single or few individuals, which are unstable because of the applied perturbation operation. In this scenario, employment of the information across consecutive generations provides an advantage. With the neighborhood size T, actually $2T$ individuals are involved in the mean calculation. The information collected is twice what can be extracted from a single generation. The estimation becomes more accurate, and the influence of outliers is further

reduced. In the main algorithm, the neighborhood pools are shared by CGA mechanism and NCG mutation with the recommended T as 5% of the population size.

Subsequently, a perturbation randomly sampled from a scaled Gaussian Distribution is added to the computed mean $F_{\text{neighbor},i,g}$ and $Cr_{\text{neighbor},i,g}$ to finalize the parameter values of offspring. The purpose of introducing perturbations is to keep exploring new parameters throughout the evolutionary process. Premature convergence of parameters should be avoided because even for a single region in objective space, the optimal parameters may not stay consistent in different optimization stages. Owing to the scaled Gaussian Distribution, exploration is centered on the estimated optimal values, whilst having a low probability to produce a dramatic variation. Values of $F_{\text{neighbor},i,g}$ and $Cr_{\text{neighbor},i,g}$ remain relatively stable unless most neighborhood members are updated simultaneously by individuals with new parameters. It indicates that the previous parametric setup is not suitable for the current searching stage, thereby the CGA mechanism proceeds to adapt the parameters. It is notable that the newly generated $F_{\text{spring},i,g}$ and $Cr_{\text{spring},i,g}$ will be employed in the mutation and crossover operators instead of the associated control parameters of parents. If the corresponding offspring succeeds to survive, then the $F_{\text{spring},i,g}$ and $Cr_{\text{spring},i,g}$ become its associated parameters.

4.2.4 The Contributions of ACGDE Algorithm

The main task of ACGDE is actually to regenerate better individuals for the implemented MO framework (e.g. NSGA-II [134] and MOEA/D [135]). The

mutation and crossover operation in original MO framework will be replaced by components of ACGDE. Survival selection is performed with the existing mechanisms, e.g., non-dominated-sorting in NSGA-II or scalar function in MOEA/D. As a result, ACGDE is flexible in responding to numerous optimization requirements. Depending on the properties of current problem, ACGDE is convenient to implement into different MO frameworks. Experimental results in subsection 4.3 shows a significant improvement in NSGA-II and MOEA/D with ACGDE.

Compared with conventional DE operators, ACGDE introduces sub-rank calculations and neighborhood assignment. The sub-rank computations require sorting the population according to each objective function value in ascending order of magnitude. The complexity of this procedure is $O(MN\log N)$, where M is the number of objectives and N is the population size. Thereafter, for each solution we calculate the distance from that solution to all the other individuals in terms of sub-rank and determine the neighborhood members according to the computed distances. In a proper implementation, this has $O(MN^2)$ computational complexity. Thus, the overall complexity of ACGDE is $O(MN^2)$, which is identical with that of original NSGA-II [134].

4.3 Empirical Study

This section evaluates the overall performance of ACGDE and effectiveness of each algorithmic component through experiments.

4.3.1 Comparison with State-of-the-art MOEAs

To evaluate the optimization performance of the new algorithm, the proposed ACGDE is implemented into nondominated sorting genetic algorithm II (NSGA-II) [134], which is a well-known powerful multi-objective optimization algorithm. The offspring in NSGA-II will be generated via ACGDE, and the survival selection part of the original framework is preserved including nondominated sorting and density estimation. The performance of ACGDE-NSGA-II is compared with seven state-of-the-art MOEAs:

1) NSGA-II(SBX) [134] with $p_c = 1.0,\ \eta_c = 20,\ \eta_m = 20,\ p_m = 1/N$;

2) NSGA-II-DE [74] with $F = 0.5,\ p_c = 1.0,\ \eta_m = 20,\ p_m = 1/N,\ \delta = 0.9,\ T = 20,\ n_r = 2$;

3) MOEA/D(SBX) [135] with $p_c = 1.0,\ \eta_c = 20,\ \eta_m = 20,\ p_m = 1/N$;

4) MOEA/D-DE [74] with $F = 0.5,\ p_c = 1.0,\ \eta_m = 20,\ p_m = 1/N,\ \delta = 0.9,\ T = 20,\ n_r = 2$;

5) CCPSO [143] with $p_c = 1.0,\ p_m = 1/N$, sub-population size: 4 for two-objective problems and 15 for three-objective problems, turbulence operator with probability of 0.1, inertia weight as 0.4, dynamic sharing for niche radius;

6) MOEGS [92] with $\kappa = 10$, No. of trial solutions as 50, Hypervolume performance indicator for fitness assignment approach;

7) SPEA2 [144] with $p_c = 0.8,\ p_m = 1/N, n_{bit} = 15$.

Recommended parametric settings in the original literatures are utilized for the above MOEAs. Following the standard experimental settings [74, 129, 134], for all the testing algorithms, the population size is fixed as 100 for 2-objective

benchmarks, 300 for 3-objective benchmarks, and the maximum number of function evaluations is set to 5×10^4 for 2-objective problems, 15×10^4 for 3-objective problems. The control parameters of ACGDE are set as follows: $T = 5$ for 2-objective problems, $T = 15$ for 3-objective problems. $\theta_F = 0.4,\ \theta_{Cr} = 0.2,\ F_{\max} = 0.9,\ F_{\min} = 0.1,\ Cr_{\max} = 0.5,\ Cr_{\min} = 0.2$. The selection of T is according to our parametric sensitivity tests (see subsection 4.3.4), and the selection of other parameters are based on some empirical studies about parametric setup in DE [10, 29, 51].

In total, 19 frequently-used MO benchmark problems were tested to evaluate the performances of the algorithms, among which UF1 to UF10 are unconstrained problems from CEC-09 Special Session and Competition [145] and WFG1 to WFG9 are from WFG test suites [146]. Regarding the number of objective functions, UF1-UF7 and WFG1-WFG9 are 2-objective problems, and UF8-UF10 are 3-objective problems. Numerous types of problems are covered in terms of separability, modality, bias and shape of Pareto Optimal Front, and all of them are minimization problems. Inverted Generational Distance (IGD) [147] and Hypervolume (HYP) [148] are selected as the performance metrics to quantitatively compare the performance of algorithms in terms of both convergence and diversity [149]. The values of hypervolume are calculated by means of Monte Carlo simulation as in [150]. All of the simulations were done on an Intel (R) Core (TM) i7 machine with 16-GB RAM and 3.40-GHz speed. Microsoft (R) Visual Studio (R) 2012 Express is used to develop the coding and run the experiments. Table 4.1, Table 4.2, Table 4.3 and Table 4.4 show the mean and standard deviation of the IGD values and HYP values

Table 4.1: Mean and Standard Deviation of the IGD Values for UF test suite (30 runs)

Problems	UF1	UF2	UF3	UF4	UF5
	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)
ACGDE-NSGA-II	0.0528† (0.0148)	**0.0205 (0.0027)**	**0.0947 (0.0139)**	**0.0410 (0.0003)**	**0.2870 (0.0932)**
NSGA-II-DE	0.0603† (0.0162)	0.0429†‡ (0.0047)	0.1515†‡ (0.0271)	0.0723†‡ (0.0078)	0.8494†‡ (0.1698)
NSGA-II(SBX)	0.1230†‡ (0.0318)	0.0481†‡ (0.0125)	0.2179†‡ (0.0666)	0.0533†‡ (0.0018)	0.3257 (0.0943)
MOEA/D-DE	**0.0475 (0.0372)**	0.0426†‡ (0.0316)	0.1513†‡ (0.0688)	0.0866†‡ (0.0104)	0.7643†‡ (0.1307)
MOEA/D(SBX)	0.1568†‡ (0.0652)	0.0640†‡ (0.0310)	0.3064†‡ (0.0300)	0.0560†‡ (0.0034)	0.4318†‡ (0.0812)
CCPSO	0.0483† (0.0108)	0.0481†‡ (0.0079)	0.3024†‡ (0.0390)	0.0639†‡ (0.0068)	0.4535†‡ (0.0734)
MOEGS	0.2207†‡ (0.1025)	0.0484†‡ (0.0091)	0.1318†‡ (0.0370)	0.1277†‡ (0.0105)	0.8727†‡ (0.5201)
SPEA2	0.1341†‡ (0.0407)	0.0626†‡ (0.0071)	0.3025†‡ (0.0410)	0.0682†‡ (0.0031)	0.4741†‡ (0.0877)
Problems	UF6	UF7	UF8 (3-Obj)	UF9 (3-Obj)	UF10 (3-Obj)
	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)
ACGDE-NSGA-II	**0.1576 (0.0849)**	**0.0262 (0.0071)**	0.1383†‡ (0.0420)	0.1776†‡ (0.1091)	0.6440†‡ (0.2715)
NSGA-II-DE	0.4181†‡ (0.0819)	0.0389† (0.0422)	0.1520†‡ (0.0300)	0.1938†‡ (0.0646)	2.4308†‡ (0.1848)
NSGA-II(SBX)	0.2302 (0.0680)	0.2359†‡ (0.1447)	0.2194†‡ (0.0098)	0.1635†‡ (0.0491)	0.3236† (0.0703)
MOEA/D-DE	0.4386†‡ (0.2206)	0.1018‡ (0.1648)	**0.0911 (0.0124)**	**0.1065 (0.0452)**	0.5826†‡ (0.0716)
MOEA/D(SBX)	0.4374†‡ (0.1509)	0.3536†‡ (0.1552)	0.148†‡ (0.0358)	0.134† (0.0624)	**0.2937 (0.1304)**
CCPSO	0.4701†‡ (0.0356)	0.0961†‡ (0.0381)	0.257†‡ (0.0528)	0.2873†‡ (0.0501)	0.4859†‡ (0.0298)
MOEGS	0.6323†‡ (0.4519)	0.0833†‡ (0.1027)	0.1585†‡ (0.0319)	0.199†‡ (0.0632)	6.9530†‡ (1.4222)
SPEA2	0.4609†‡ (0.1292)	0.1693†‡ (0.1285)	0.2781†‡ (0.0195)	0.3689†‡ (0.0535)	0.8201†‡ (0.2349)

† and ‡ indicate that the difference between the marked entry and the best entry is statistically significant using Wilcoxon rank sum test and unpaired two-sample student's t-test, respectively (both are at the 5% significance level).

Table 4.2: Mean and Standard Deviation of the IGD Values for WFG test suite (30 runs)

Problems	WFG1	WFG2	WFG3	WFG4	WFG5
	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)
ACGDE-NSGA-II	**0.8084 (0.0316)**	**0.0139 (0.0008)**	**0.0201 (0.0009)**	0.0196†‡ (0.0014)	**0.0682 (0.0015)**
NSGA-II-DE	1.2181†‡ (0.0052)	0.0461†‡ (0.0253)	0.0344†‡ (0.0017)	0.0927†‡ (0.0037)	0.0754†‡ (0.0017)
NSGA-II(SBX)	1.0790†‡ (0.0813)	0.1604†‡ (0.0277)	0.0211†‡ (0.0016)	0.0189†‡ (0.0011)	0.0705†‡ (0.0005)
MOEA/D-DE	1.1634†‡ (0.0138)	0.1666†‡ (0.0880)	0.0204 (0.0018)	0.0811†‡ (0.0081)	0.0692†‡ (0.0003)
MOEA/D(SBX)	1.0483†‡ (0.0458)	0.1871†‡ (0.0643)	0.0203† (0.0059)	**0.0167 (0.0015)**	0.0691†‡ (0.0006)
CCPSO	0.9758†‡ (0.0359)	0.5447†‡ (0.1463)	0.1826†‡ (0.0521)	0.0614†‡ (0.0328)	0.0899†‡ (0.0134)
MOEGS	1.1818†‡ (0.0183)	0.5816†‡ (0.1444)	0.2399†‡ (0.0558)	0.1842†‡ (0.0342)	0.1698†‡ (0.1090)
SPEA2	1.0859†‡ (0.0185)	0.2381†‡ (0.0281)	0.1036†‡ (0.0305)	0.0363†‡ (0.0052)	0.0734†‡ (0.0012)

Problems	WFG6	WFG7	WFG8	WFG9
	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)
ACGDE-NSGA-II	0.1175†‡ (0.0205)	**0.0166 (0.0007)**	**0.1089 (0.0036)**	0.1259†‡ (0.0004)
NSGA-II-DE	0.1079†‡ (0.0349)	0.0306†‡ (0.0019)	0.1413†‡ (0.0101)	0.1106†‡ (0.0380)
NSGA-II(SBX)	**0.0640 (0.0068)**	0.0170 (0.0010)	0.1371†‡ (0.0065)	0.0844‡ (0.0529)
MOEA/D-DE	0.1072†‡ (0.0319)	0.0190†‡ (0.0011)	0.1271†‡ (0.0128)	**0.0597 (0.0287)**
MOEA/D(SBX)	0.0820†‡ (0.0237)	0.0205 (0.0111)	0.1270†‡ (0.0097)	0.0606‡ (0.0381)
CCPSO	0.1228†‡ (0.0489)	0.2122†‡ (0.0730)	0.2203†‡ (0.0489)	0.1519†‡ (0.0534)
MOEGS	0.1335†‡ (0.0373)	0.2250†‡ (0.1229)	0.2502†‡ (0.0337)	0.2660†‡ (0.1514)
SPEA2	0.0867†‡ (0.0162)	0.0507†‡ (0.0184)	0.1700†‡ (0.0107)	0.1070† (0.0386)

† and ‡ indicate that the difference between the marked entry and the best entry is statistically significant using Wilcoxon rank sum test and unpaired two-sample student's t-test, respectively (both are at the 5% significance level).

Table 4.3: Mean and Standard Deviation of the HYP Values for UF test suite (30 runs)

Problems	UF1	UF2	UF3	UF4	UF5
	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)
ACGDE-NSGA-II	**0.9479 (0.0115)**	**0.9052 (0.0103)**	0.9287†‡ (0.0179)	**0.7106** (**0.0017**)	**0.9560 (0.0278)**
NSGA-II-DE	0.9431†‡ (0.0100)	0.8848†‡ (0.0128)	**0.9347 (0.0044)**	0.6801†‡ (0.0086)	0.8709†‡ (0.0227)
NSGA-II(SBX)	0.9037†‡ (0.0199)	0.8750†‡ (0.0169)	0.7716†‡ (0.0316)	0.7021†‡ (0.0017)	0.8979†‡ (0.0299)
MOEA/D-DE	0.9309†‡ (0.0279)	0.8727†‡ (0.0391)	0.8276†‡ (0.0735)	0.6657†‡ (0.0102)	0.8402†‡ (0.0308)
MOEA/D(SBX)	0.8687†‡ (0.0429)	0.8556†‡ (0.0344)	0.7370†‡ (0.0160)	0.6927†‡ (0.0048)	0.8681†‡ (0.0223)
CCPSO	0.9441†‡ (0.0126)	0.8536†‡ (0.0147)	0.7799†‡ (0.0333)	0.6970†‡ (0.0041)	0.8931†‡ (0.0211)
MOEGS	0.8714†‡ (0.0502)	0.8824†‡ (0.0198)	0.8776†‡ (0.0463)	0.6314†‡ (0.0109)	0.8445†‡ (0.0688)
SPEA2	0.9035†‡ (0.0190)	0.8729†‡ (0.0164)	0.7496†‡ (0.0337)	0.6869†‡ (0.0038)	0.8758†‡ (0.0323)

Problems	UF6	UF7	UF8 (3-Obj)	UF9 (3-Obj)	UF10 (3-Obj)
	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)
ACGDE-NSGA-II	**0.9756 (0.0205)**	**0.9497** (**0.0037**)	0.9985†‡ (0.0024)	**0.9986 (0.0015)**	**0.9989 (0.0015)**
NSGA-II-DE	0.9473†‡ (0.0159)	0.9389†‡ (0.0303)	0.9968†‡ (0.0013)	0.9955†‡ (0.0017)	0.9347†‡ (0.0070)
NSGA-II(SBX)	0.9490†‡ (0.0182)	0.8326†‡ (0.0836)	0.9908†‡ (0.0005)	0.9545†‡ (0.0160)	0.9940†‡ (0.0099)
MOEA/D-DE	0.9346†‡ (0.0255)	0.9001†‡ (0.0801)	**0.9986 (0.0005)**	0.9970†‡ (0.0010)	0.9650†‡ (0.0044)
MOEA/D(SBX)	0.9241†‡ (0.0198)	0.7722†‡ (0.0734)	0.9961‡ (0.0037)	0.9638†‡ (0.0207)	0.9870†‡ (0.0144)
CCPSO	0.9314†‡ (0.0164)	0.9303†‡ (0.0330)	0.9372†‡ (0.0265)	0.9372†‡ (0.0157)	0.9632†‡ (0.0045)
MOEGS	0.9266†‡ (0.0545)	0.9311†‡ (0.0536)	0.9935†‡ (0.0032)	0.9911†‡ (0.0082)	0.8477†‡ (0.0396)
SPEA2	0.9193†‡ (0.0199)	0.8663†‡ (0.0764)	0.9923†‡ (0.0026)	0.9641†‡ (0.0279)	0.9762†‡ (0.0107)

† and ‡ indicate that the difference between the marked entry and the best entry is statistically significant using Wilcoxon rank sum test and unpaired two-sample student's t-test, respectively (both are at the 5% significance level).

Table 4.4: Mean and Standard Deviation of the HYP Values for WFG test suite (30 runs)

Problems	WFG1	WFG2	WFG3	WFG4	WFG5
	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)
ACGDE-NSGA-II	**0.4690 (0.0105)**	**0.6161 (0.0013)**	**0.5841 (0.0017)**	0.3466†‡ (0.0016)	**0.3651 (0.0019)**
NSGA-II-DE	0.3301†‡ (0.0067)	0.5969†‡ (0.0050)	0.5741†‡ (0.0022)	0.3030†‡ (0.0034)	0.3582†‡ (0.0021)
NSGA-II(SBX)	0.3201†‡ (0.0312)	0.5999†‡ (0.0040)	0.5824†‡ (0.0018)	0.3462†‡ (0.0020)	0.3645†‡ (0.0014)
MOEA/D-DE	0.3430†‡ (0.0102)	0.5923†‡ (0.0142)	0.5822†‡ (0.0021)	0.3170†‡ (0.0039)	0.3641†‡ (0.0016)
MOEA/D(SBX)	0.3297†‡ (0.0210)	0.5950†‡ (0.0129)	0.5824†‡ (0.0033)	**0.3483 (0.0014)**	0.3643†‡ (0.0015)
CCPSO	0.4009†‡ (0.0134)	0.4962†‡ (0.0258)	0.5068†‡ (0.0176)	0.2928†‡ (0.0209)	0.3239†‡ (0.0150)
MOEGS	0.3479†‡ (0.0058)	0.4199†‡ (0.0279)	0.4717†‡ (0.0257)	0.2581†‡ (0.0191)	0.3115†‡ (0.0462)
SPEA2	0.3774†‡ (0.0062)	0.5478†‡ (0.0119)	0.5384†‡ (0.0157)	0.3324†‡ (0.0048)	0.3608†‡ (0.0024)

Problems	WFG6	WFG7	WFG8	WFG9
	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)
ACGDE-NSGA-II	0.2852†‡ (0.0112)	**0.3942 (0.0013)**	**0.4082 (0.0015)**	0.4444†‡ (0.0018)
NSGA-II-DE	0.2904†‡ (0.0181)	0.3852†‡ (0.0017)	0.3872†‡ (0.0053)	0.4490†‡ (0.0164)
NSGA-II(SBX)	**0.3127 (0.0039)**	0.3937†‡ (0.0016)	0.3970†‡ (0.0020)	0.4621‡ (0.0231)
MOEA/D-DE	0.2918†‡ (0.0178)	0.3931†‡ (0.0016)	0.4014†‡ (0.0052)	**0.4741 (0.0120)**
MOEA/D(SBX)	0.3088†‡ (0.0063)	0.3918†‡ (0.0045)	0.4040†‡ (0.0032)	0.4710 (0.0159)
CCPSO	0.2825†‡ (0.0094)	0.2859†‡ (0.0342)	0.3093†‡ (0.0219)	0.4223†‡ (0.0256)
MOEGS	0.2867†‡ (0.0110)	0.2865†‡ (0.0571)	0.3487†‡ (0.0180)	0.3782†‡ (0.0599)
SPEA2	0.3006†‡ (0.0079)	0.3720†‡ (0.0098)	0.3859†‡ (0.0038)	0.4515†‡ (0.0164)

† and ‡ indicate that the difference between the marked entry and the best entry is statistically significant using Wilcoxon rank sum test and unpaired two-sample student's t-test, respectively (both are at the 5% significance level).

for 30 independent runs of each algorithm on each benchmark. The best entries in terms of mean value are marked in boldface. In order to judge whether the results of the best performing algorithm differ from the results of competitors in a statistically significant way, Wilcoxon rank-sum test [118] and unpaired two-sample student's t-test are conducted at the 5% significance level. The entries which are *significantly* different from the best entries are indicated by symbol † and ‡.

From the comparative results, it clearly elucidates that ACGDE-NSGA-II is powerful in tackling multi-objective problems when compared to its competitors: For 2-objective problems, it achieves the best results in 12 out of 16 benchmarks in terms of both IGD and HYP. For 3-objective problems, ACGDE-NSGA-II performs best in 2 out of 3 benchmarks in terms of HYP, whereas in terms of IGD, the proposed algorithm is outperformed by MOEA/D variants. The explanation is that NSGA-II is a dominance-based MOEA, the efficiency of non-dominated-sorting will be deteriorated when the number of objectives increases. The ability of ACGDE in solving 3-objective problems is further evaluated by implementation into MOEA/D in next subsection. To validate the effectiveness of ACGDE, it is reasonable to make a comparison between ACGDE-NSGA-II and original NSGA-II-DE. Separate statistical tests are performed between the two algorithms over all the benchmarks (partially shown in Table 4.1-Table 4.4). In terms of HYP, ACGDE-NSGA-II provides *significantly* better results in 16 out of 19 problems. In terms of IGD, ACGDE-NSGA-II *significantly* outperforms original NSGA-II-DE in 17 out of 19 problems, and the latter cannot give the best performance in any of the benchmarks. The only

disparity between these two algorithms lies in the DE operator, from which we can draw the conclusion that the success of ACGDE-NSGA-II benefit from the newly developed cross-generation mutation operators and the parameter adaptation mechanism.

4.3.2 Implementation in MOEA/D

In this subsection, ACGDE is implemented into MOEA/D [135] by replacing the original mutation and crossover operators in MOEA/D-DE [74]. Besides the 19 benchmarks used in subsection 4.3.1, 3-objective versions of WFG1-WFG9 are included as well. The experimental and parametric setups are identical with those in subsection 4.3.1. Table 4.5 presents the mean and standard deviation of the IGD values for 30 independent runs of both algorithms on each benchmark. The best entries in terms of mean value are marked in boldface.

According to the results, ACGDE-MOEA/D is able to provide better performance than original MOEA/D-DE in 26 out of 28 benchmark problems. Considering the statistical tests, ACGDE *significantly* improves the performance of MOEA/D in 11 out of 12 3-objective problems, and 12 out of 16 2-objective problems, respectively. To summarize, ACGDE is good at solving not only 2-objective problems but also those with 3 objectives. Moreover, successful implementation in both NSGA-II and MOEA/D demonstrate the robustness of ACGDE during combination with different MO frameworks.

Table 4.5: Mean and Standard Deviation of the IGD Values (30 runs)

Problems	ACGDE -MOEA/D	MOEA/D -DE	Problems	ACGDE -MOEA/D	MOEA/D -DE
	Mean(Std)	Mean(Std)		Mean(Std)	Mean(Std)
UF1	**0.0448**	0.0475	WFG5	**0.0685**	0.0692†‡
(2-obj)	**(0.0155)**	(0.0372)	(2-obj)	**(0.0006)**	(0.0003)
UF2	**0.0195**	0.0426†‡	WFG6	**0.0998**	0.1072†
(2-obj)	**(0.0039)**	(0.0316)	(2-obj)	**(0.0347)**	(0.0319)
UF3	**0.1306**	0.1513	WFG7	**0.0149**	0.019†‡
(2-obj)	**(0.0398)**	(0.0688)	(2-obj)	**(0.0001)**	(0.0011)
UF4	**0.0436**	0.0866†‡	WFG8	**0.1104**	0.1271†‡
(2-obj)	**(0.0014)**	(0.0104)	(2-obj)	**(0.0081)**	(0.0128)
UF5	**0.4654**	0.7643†‡	WFG9	0.1062†‡	**0.0597**
(2-obj)	**(0.0974)**	(0.1307)	(2-obj)	(0.0404)	**(0.0287)**
UF6	**0.2893**	0.4386†	WFG1	**1.0470**	1.4836†‡
(2-obj)	**(0.1694)**	(0.2206)	(3-obj)	**(0.0427)**	(0.0064)
UF7	**0.0521**	0.1018	WFG2	**0.4011**	0.4045†
(2-obj)	**(0.0882)**	(0.1648)	(3-obj)	**(0.0142)**	(0.0331)
UF8	0.1045†‡	**0.0911**	WFG3	**0.0484**	0.0674†‡
(3-obj)	(0.0112)	**(0.0124)**	(3-obj)	**(0.0010)**	(0.0076)
UF9	**0.0760**	0.1065†‡	WFG4	**0.3481**	0.3797†‡
(3-obj)	**(0.0401)**	(0.0452)	(3-obj)	**(0.0023)**	(0.009)
UF10	**0.2481**	0.5826†‡	WFG5	**0.1934**	0.1952†‡
(3-obj)	**(0.0660)**	(0.0716)	(3-obj)	**(0.0004)**	(0.0011)
WFG1	**0.9680**	1.1634†‡	WFG6	**0.2227**	0.2611†‡
(2-obj)	**(0.0128)**	(0.0138)	(3-obj)	**(0.0201)**	(0.0218)
WFG2	**0.1297**	0.1666†	WFG7	**0.1614**	0.1735†‡
(2-obj)	**(0.0674)**	(0.088)	(3-obj)	**(0.0004)**	(0.0062)
WFG3	**0.0159**	0.0204†‡	WFG8	**0.2457**	0.3392†‡
(2-obj)	**(0.0008)**	(0.0018)	(3-obj)	**(0.0188)**	(0.0340)
WFG4	**0.0226**	0.0811†‡	WFG9	**0.2136**	0.2233†
(2-obj)	**(0.0025)**	(0.0081)	(3-obj)	**(0.0348)**	(0.022)

† and ‡ indicate that the difference between ACGDE-MOEA/D and original MOEA/D-DE is statistically significant using Wilcoxon rank sum test and unpaired two-sample student's t-test, respectively (both are at the 5% significance level).

4.3.3 Effectiveness of Searching Directions in NCG Mutation

In the proposed NCG mutation, the exploration is guided by the searching direction corresponding to the estimated converging direction. For further investigation, one inevitable question to address is: Will these computed searching directions really help the optimization process? Or maybe they are only randomly generated perturbations? In order to clearly demonstrate the meaningfulness of calculated searching directions, a reverse version of NCG mutation will be employed for a contrast experiment.

Original NCG mutation:

$$V_{i,g} = X_{rn_1,g} + F \cdot (X_{rn_1,g} - X_{rn_2,g-1}) \tag{4.7}$$

Reverse version of NCG mutation:

$$V_{i,g} = X_{rn_1,g} + F \cdot (X_{rn_2,g-1} - X_{rn_1,g}) \tag{4.8}$$

As can be seen, all the individuals and parameters involved in the reverse version are identical to those in the original version. The only difference comes from the exchange of $X_{rn_1,g}$ and $X_{rn_2,g-1}$. Thus, the influences of all the other factors are eliminated, and the variation in performance, if any, only results from their different searching directions. With this reverse version of NCG mutation, it will be much easier to judge the effectiveness of searching directions in original NCG mutation: first, if the obtained searching direction is no different from a stochastic search, then the performance of the two mutation strategies should be almost the same; second, if the utilized searching directions in original version are meaningful and favorable, then an obvious improvement in performance should be observed compared with that of reverse version.

In this contrast experiment, the PCG mutation strategy and the CGA mechanism are removed from the original ACGDE. The simplified ACGDE only with NCG mutation or reverse NCG mutation are implemented into NSGA-II. For both algorithms, the scaling factor F is set as 0.5 and the crossover rate Cr is set as 0.35 based on the parametric setup in subsection 4.3.1. WFG1 to WFG8 from WFG test suites are utilized as benchmark problems. Other experimental and parametric setups are identical with those in subsection 4.3.1. IGD is employed as the indicator to quantitatively evaluate the performance of each algorithm. Fig. 4.3, Fig. 4.4, Fig. 4.5 and Fig. 4.6 present the final solution sets obtained by both algorithms for 30 independent runs and the average IGD values of 30 runs over generations.

From Fig. 4.3, Fig. 4.4, Fig. 4.5 and Fig. 4.6, the performance of the original NCG mutation is significantly better than the reverse version in 6 out of 8 benchmark problems. For WFG1, the difference between the results of the two algorithms is fairly significant. The final solution sets obtained by the reverse version is very poor in both diversity and convergence. It reveals that the current searching directions are completely erroneous, and they have hampered the whole optimization process of the algorithm. In other words, the opposite searching directions, that utilized in the original NCG mutation, are close to the correct searching directions. Actually, the performance provided by the original NCG mutation in WFG1 is the best when compared with those of other state-of-the-art MO algorithms in subsection 4.3.1. For WFG2, WFG3, WFG5, WFG7 and WFG8, the original NCG mutation outperforms the reverse version in terms of convergence enhancement and diversity maintenance. Since the so-

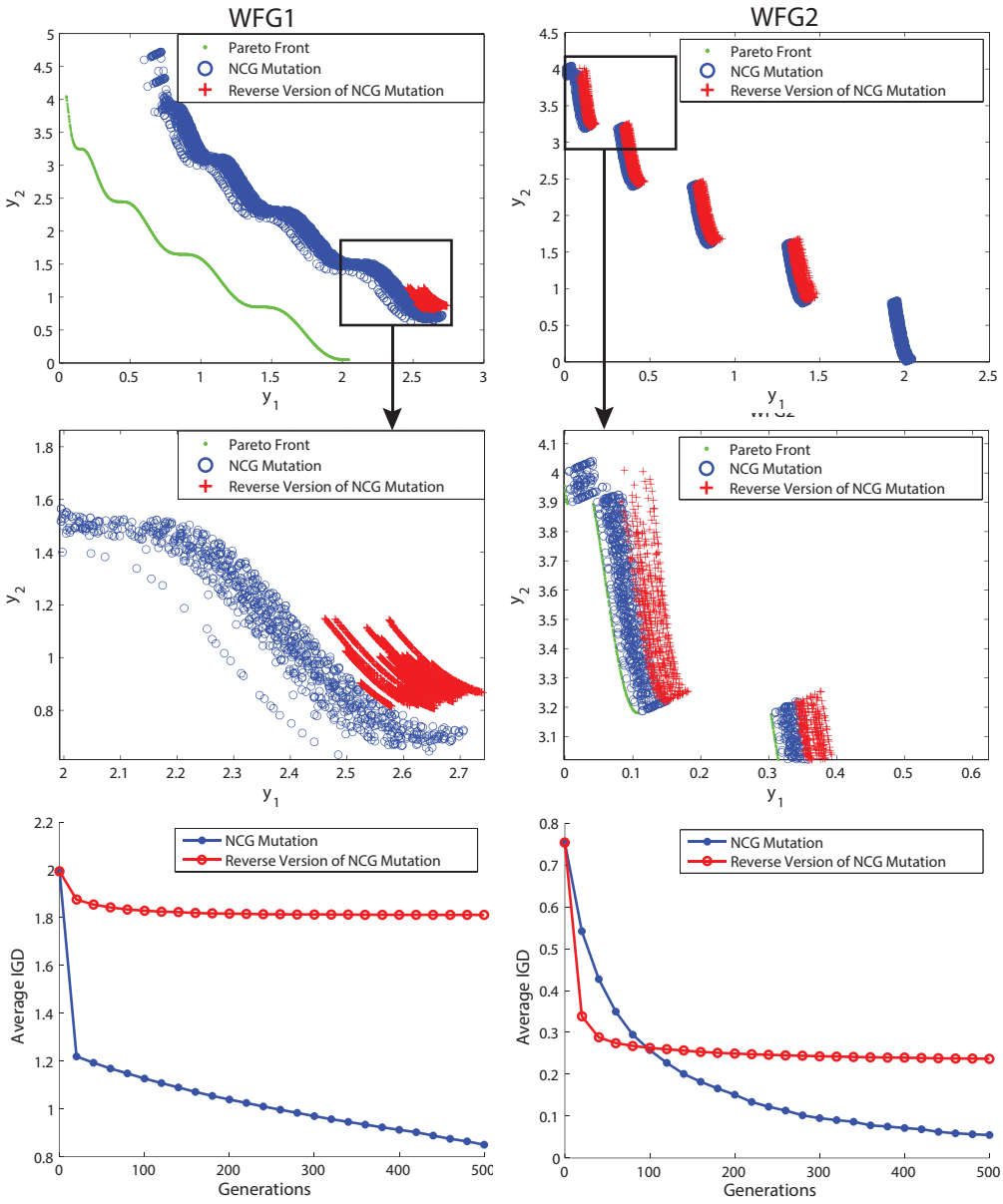

Figure 4.3: This figure compares the performance of the original NCG mutation with that of reverse NCG mutation on WFG1 and WFG2. For each benchmark, the final solution sets obtained by both algorithms over 30 independent runs are plotted. The average IGD values of 30 runs over generations are shown as well.

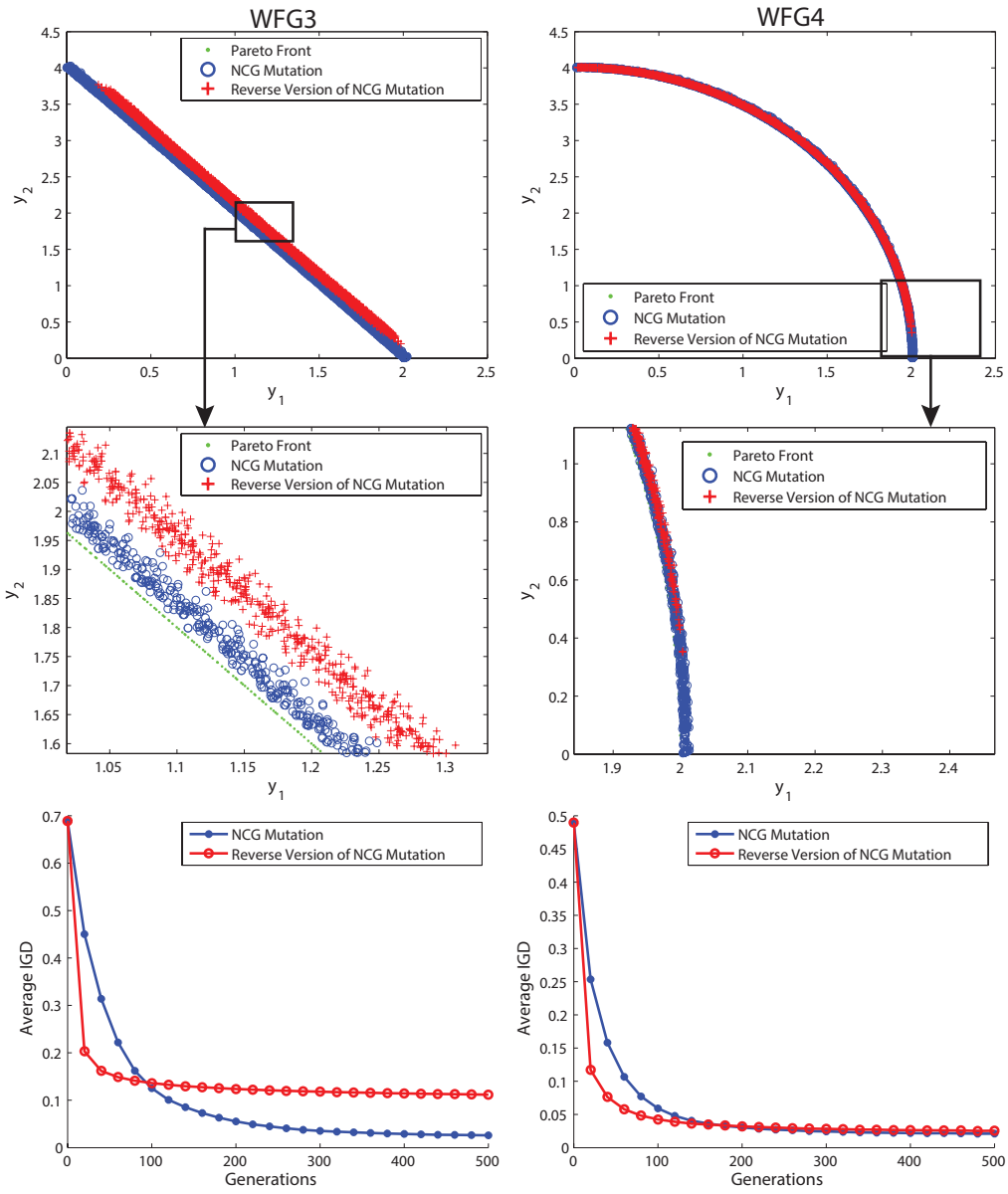

Figure 4.4: This figure compares the performance of the original NCG mutation with that of reverse NCG mutation on WFG3 and WFG4. For each benchmark, the final solution sets obtained by both algorithms over 30 independent runs are plotted. The average IGD values of 30 runs over generations are shown as well.

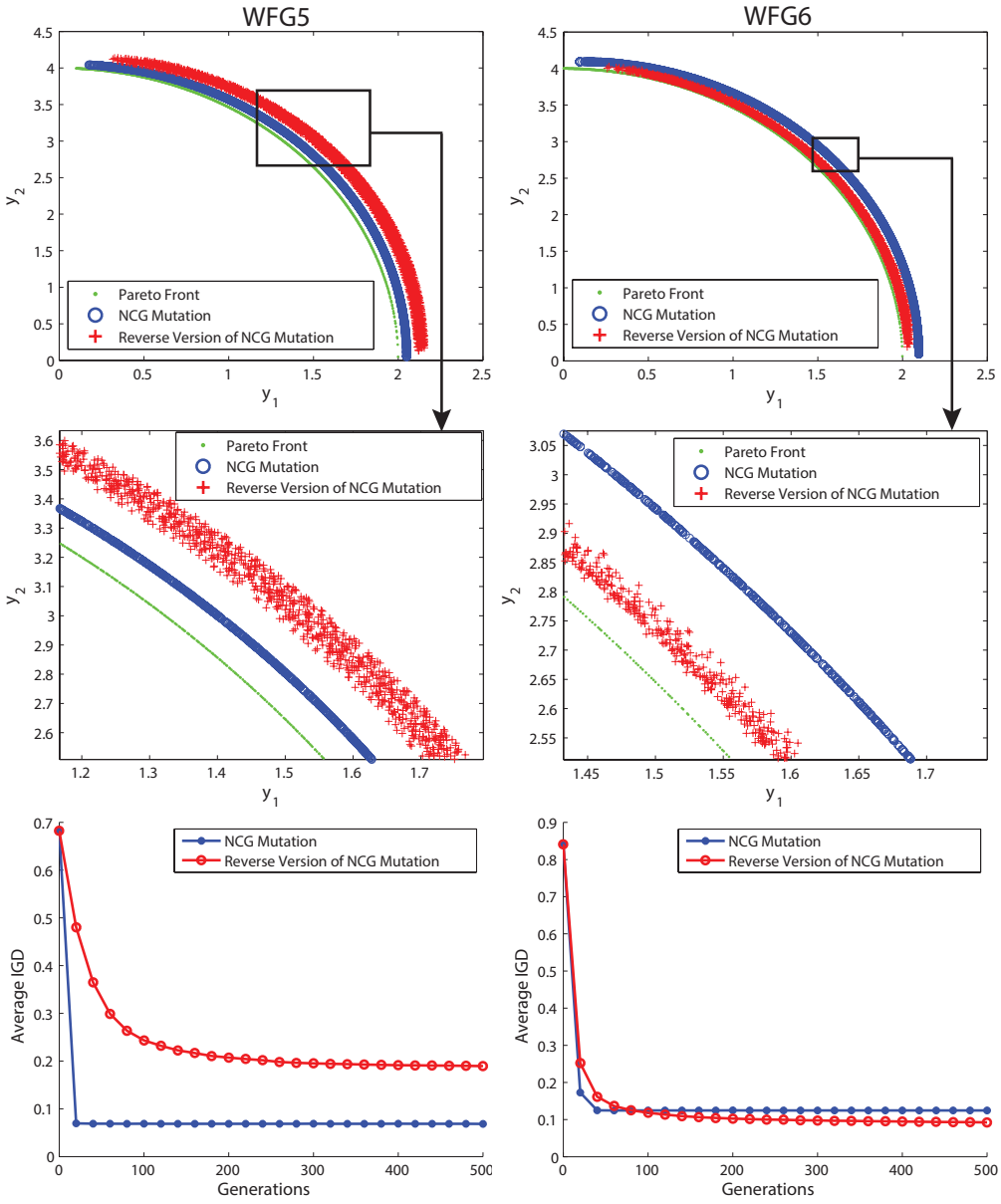

Figure 4.5: This figure compares the performance of the original NCG mutation with that of reverse NCG mutation on WFG5 and WFG6. For each benchmark, the final solution sets obtained by both algorithms over 30 independent runs are plotted. The average IGD values of 30 runs over generations are shown as well.

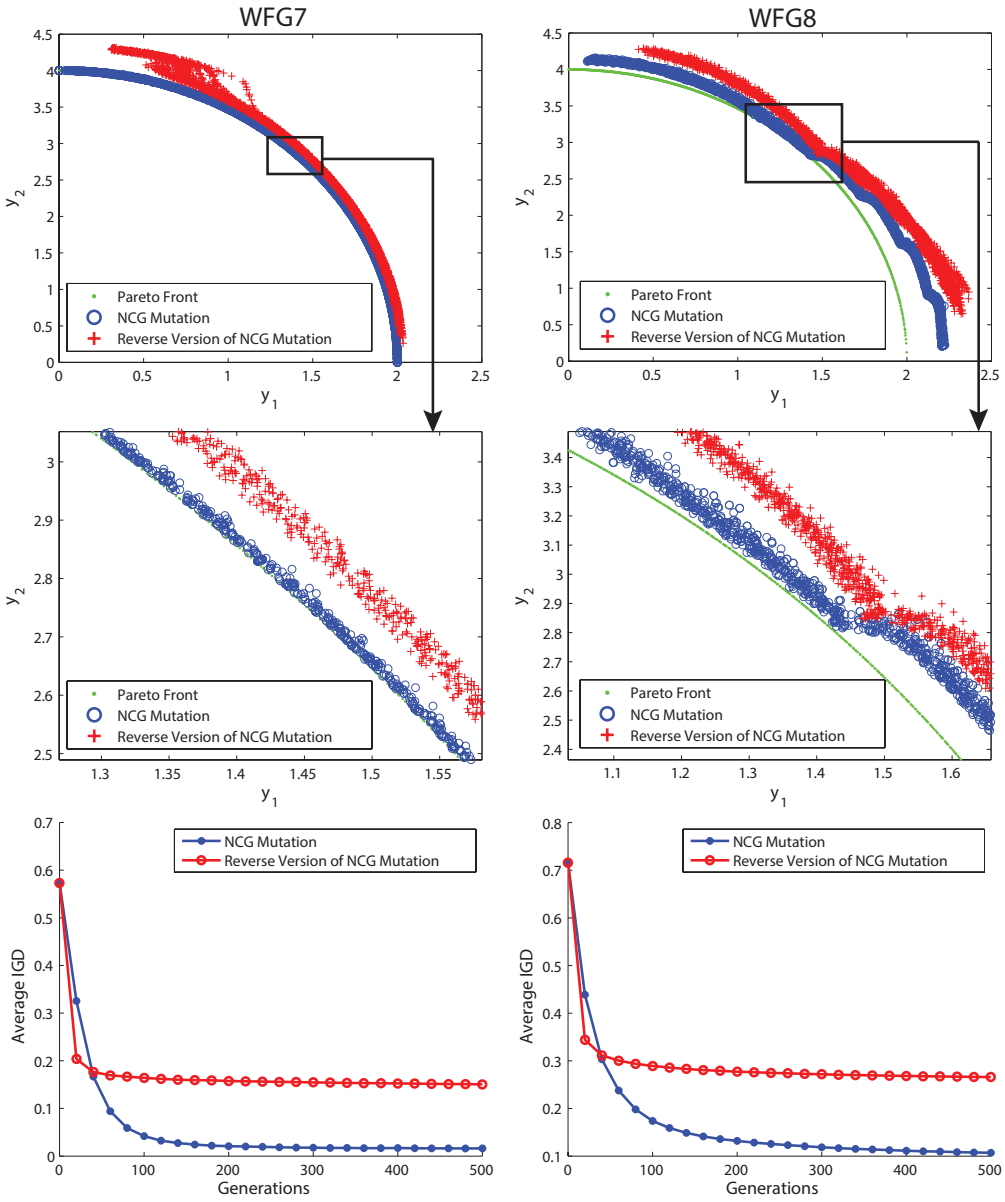

Figure 4.6: This figure compares the performance of the original NCG mutation with that of reverse NCG mutation on WFG7 and WFG8. For each benchmark, the final solution sets obtained by both algorithms over 30 independent runs are plotted. The average IGD values of 30 runs over generations are shown as well.

lutions plotted are from 30 independent runs, the consistent improvement is able to prove the effectiveness of the searching directions in NCG mutation. For WFG4, the two tested algorithms gave similar results. However, as shown in the zoomed plot, the performance of the original NCG mutation is slightly better at the aspect of coverage of true Pareto Front. For WFG6, it can be observed that the reverse version performs better in convergence while the original version achieves greater diversity. If we look into the enlarged plot and the IGD values over generations, it is obvious that the original NCG mutation is trapped into a local optimum. This is a common issue for direction-based or gradient-based approaches [42, 151–153]. Nevertheless, considering the significant improvement in most benchmark problems, the proposed NCG mutation is still promising for solving MO problems without complex local optima. Furthermore, the final version of ACGDE will include PCG mutation and CGA mechanism to enhance the explorative ability of the algorithm so that the optimization process is less likely to get stuck in local optimum.

4.3.4 Sensitivity Study for Neighborhood Size T

The selection of the control parameter T plays an important role in the overall performance of ACGDE. Both the neighborhood sizes of NCG mutation and CGA mechanism are decided by the value of T, which is pre-defined by users and kept fixed throughout the optimization process. To examine further, a sensitivity test is conducted in this subsection to investigate the effect of neighborhood size T. ACGDE is implemented into NSGA-II, and the value of T is selected as 1%, 5%, 10%, 15% and 20% of population size, respectively. The

population size N is fixed as 100. Other experimental settings are identical with those in subsection 4.3.1. All the 9 problems from WFG benchmark suite are utilized for evaluation. Table 4.6 presents the mean and standard deviation of the IGD values for 30 independent runs of each variant on each benchmark. The best entries in terms of mean value are marked in boldface.

Table 4.6: Mean and Standard Deviation of the IGD Values (30 runs)

Neighborhood	$1\% \cdot N$	$5\% \cdot N$	$10\% \cdot N$	$15\% \cdot N$	$20\% \cdot N$
Size T	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)
WFG1	1.0925	**0.8084**	0.8393	0.8340	0.8461
	(0.0163)	**(0.0322)**	(0.0289)	(0.0298)	(0.0238)
WFG2	0.0358	**0.0139**	0.0192	0.0341	0.0287
	(0.0375)	**(0.0008)**	(0.0273)	(0.0517)	(0.0456)
WFG3	0.0268	**0.0201**	0.0210	0.0206	0.0207
	(0.0026)	**(0.0009)**	(0.0008)	(0.0010)	(0.0009)
WFG4	0.0408	**0.0196**	0.0211	0.0219	0.0219
	(0.0070)	**(0.0014)**	(0.0016)	(0.0020)	(0.0018)
WFG5	0.0687	**0.0682**	0.0686	0.0684	0.0688
	(0.0004)	**(0.0016)**	(0.0006)	(0.0020)	(0.0006)
WFG6	0.1248	**0.1175**	0.1200	0.1220	0.1221
	(0.0001)	**(0.0209)**	(0.0181)	(0.0148)	(0.0142)
WFG7	0.0199	**0.0166**	0.0172	0.0173	0.0171
	(0.0013)	**(0.0007)**	(0.0009)	(0.0013)	(0.0009)
WFG8	0.1412	**0.1089**	0.1113	0.1101	0.1110
	(0.0062)	**(0.0037)**	(0.0043)	(0.0049)	(0.0037)
WFG9	0.1259	0.1259	0.1259	0.1259	0.1260
	(0.0003)	(0.0004)	(0.0003)	(0.0002)	(0.0003)

From Table 4.6, it can be observed that setting T as 5% of the population size will lead to best results in 8 out of 9 benchmark problems. The results on WFG9 are almost the same for each variant because all of them have been trapped into local optimum. This observation is in accordance with the theoretical analysis in subsection 4.2.2 and subsection 4.2.3. If the size of neighborhood pools is too small, the searching process will have a high probability to get stuck due to the overlap of $X_{rn_1,g}$ and $X_{rn_2,g-1}$ in (4.1). The estimated

control parameters of CGA mechanism will also become too oscillatory because of the small size of parameter pools. If the neighborhood size T is over large, the estimated searching directions in NCG mutation and the estimated control parameters in CGA mechanism are unable to reflect the true requirements of a particular region in objective space. The efficiency of the whole algorithm will thus be deteriorated. Experimental results in Table 4.6 show that a reasonable trade-off is achieved by choosing T as 5% of the population size.

4.3.5 Effectiveness of Combining NCG mutation with PCG mutation

NCG mutation is good at estimating the efficient searching direction but limited by its diminishing explorative ability. PCG mutation possesses a powerful explorative capability but is weak in exploitation. To strike a delicate balance between exploration and exploitation, NCG mutation and PCG mutation are employed concurrently in ACGDE algorithm. A rate is pre-defined to decide the probability of utilizing NCG mutation scheme in each mutation operation, otherwise, PCG mutation is performed. In the current version of ACGDE, this rate is set as 50% so that both mutation strategies have the equal chance to be used. In order to investigate the influence of this rate and to validate the effectiveness of the combination, simulations have been run with different probabilities to utilize NCG mutation. The experimental setup and parametric setting is the same as those in subsection 4.3.3. 6 representative benchmark problems are selected from WFG test suites and CEC-09, and each problem has been tested for 30 independent runs. Fig. 4.7 compares the performance of different variants in

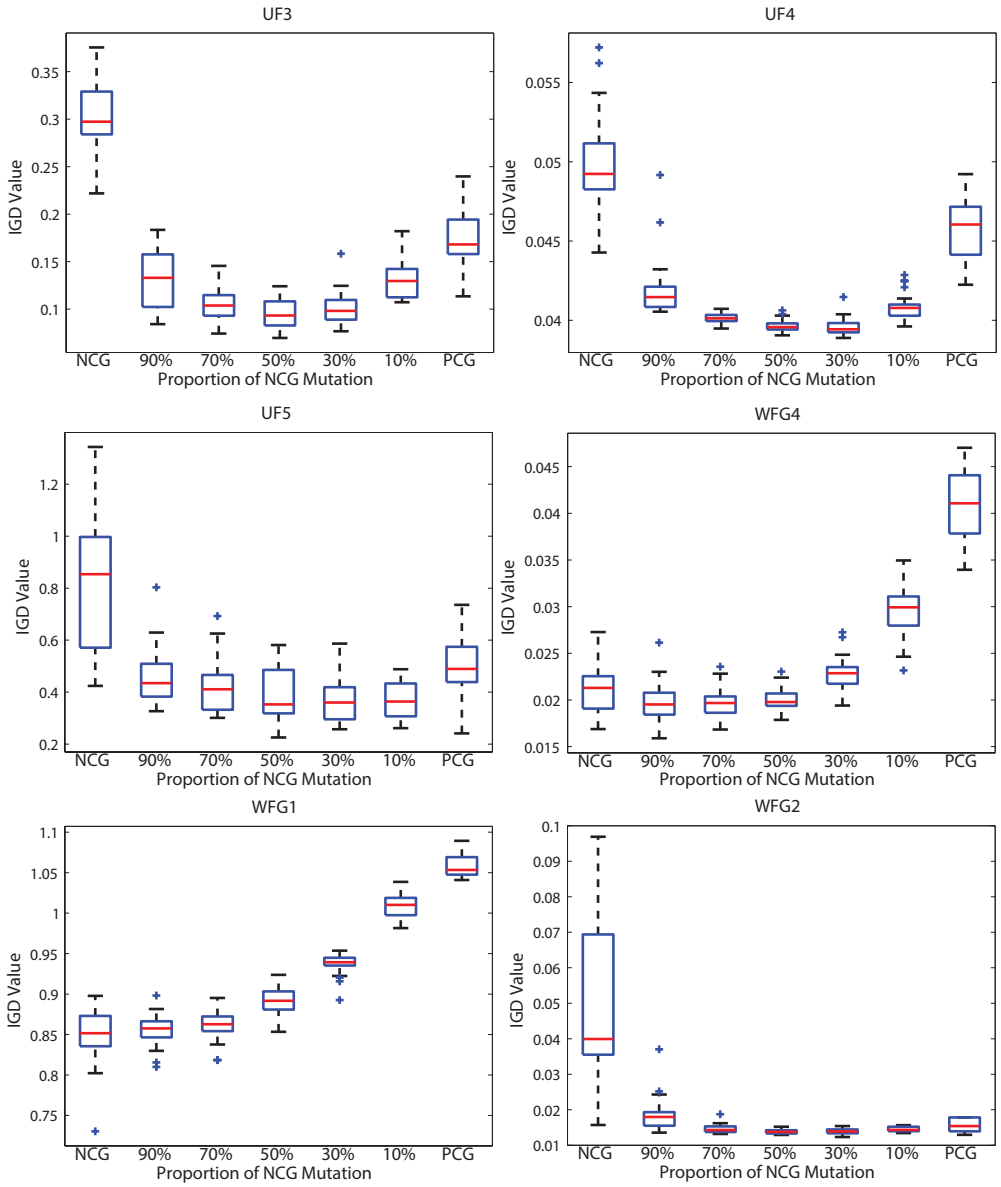

Figure 4.7: Boxplots of the IGD values over 30 independent runs. The simulations are performed on UF3, UF4, UF5, WFG1, WFG2 and WFG4 separately. In each plot, the leftmost box represents the variant only using NCG mutation, and the rightmost box denotes the variant only utilizing PCG mutation. The percentages under each box indicate the probability to employ NCG mutation in this variant.

terms of IGD. Each box represents the distribution of IGD values obtained by the corresponding variant in 30 independent simulation runs.

According to Fig. 4.7, the hybrid variants provide better results for 5 out of 6 benchmarks when compared with pure NCG variant and pure PCG variant. For WFG1, the searching directions estimated by NCG mutation are rather efficient, and there is no strong need for exploration. Thus, the performance of the algorithm declines with the decreased proportion of NCG mutation. For other problems, improvement over the variants with only one mutation strategy demonstrates the advantage of the combination. Among the several hybrid variants, the 50% version exhibits a relatively more robust performance. By contrast, the outcomes of 70% and 30% versions are more problem-dependent. For other experiments in this chapter, the 50% variant will always be applied.

4.3.6 Effectiveness of CGA mechanism

This subsection aims at observing the behavior of parameter adaptation and verifying the effectiveness of the proposed CGA mechanism. Compared to crossover probability Cr, the overall performance of DE depends more heavily on the selection of scaling factor F [154] [155]. As shown in the mutation formula, F has a big role in controlling the searching step of the algorithm. In order to investigate how CGA mechanism adapts F during the optimization process, the fluctuation of F values over generations are plotted for all the individuals in Fig. 4.8. To better visualize the distribution of F values in objective space, before plotting, the individuals have already been sorted according to their fitness in the first objective (better individuals are assigned with smaller

indices). Since the two objectives in the tested benchmarks are conflicting with each other, larger index means this individual focus more on optimizing the second objective. As a result, the order of the individuals in the plot actually follows their positions in objective space. In the simulation, ACGDE as stated in Algorithm 4.1 is implemented in NSGA-II. The experimental and parametric setups are identical with those in subsection 4.3.1. Fig. 4.8 visualizes the distribution of F values for the whole evolutionary process on three representative benchmark problems.

Based on Fig. 4.8, distinct patterns are observed in the distribution of F values while optimizing the three problems. For WFG3, F values below 0.5 are preferred in the early searching stage. From generation 300 onward, F values above 0.5 are accepted more frequently. The difference between the distributions in WFG5 and WFG8 is fairly obvious. Throughout the whole evolution, lower F values always hold a higher chance to survive in WFG8, whereas larger F values appear more in WFG5. From the perspective of objective space, in WFG5, the individuals which optimize the two objectives with similar weights (index around 50) are more likely to employ small F values in some searching stages (shown as the blue areas in the middle of the plot) although the whole population tend to reserve large F values. For WFG3 and WFG8, no evident pattern with regards to objective space emerged. Actually, in most existing benchmark problems, the requirements of parametric setup are roughly the same for its multiple objectives. From the testing results, it is explicit that CGA mechanism is intelligently adapting the parameters according to the current optimization task instead of a purely stochastic regeneration.

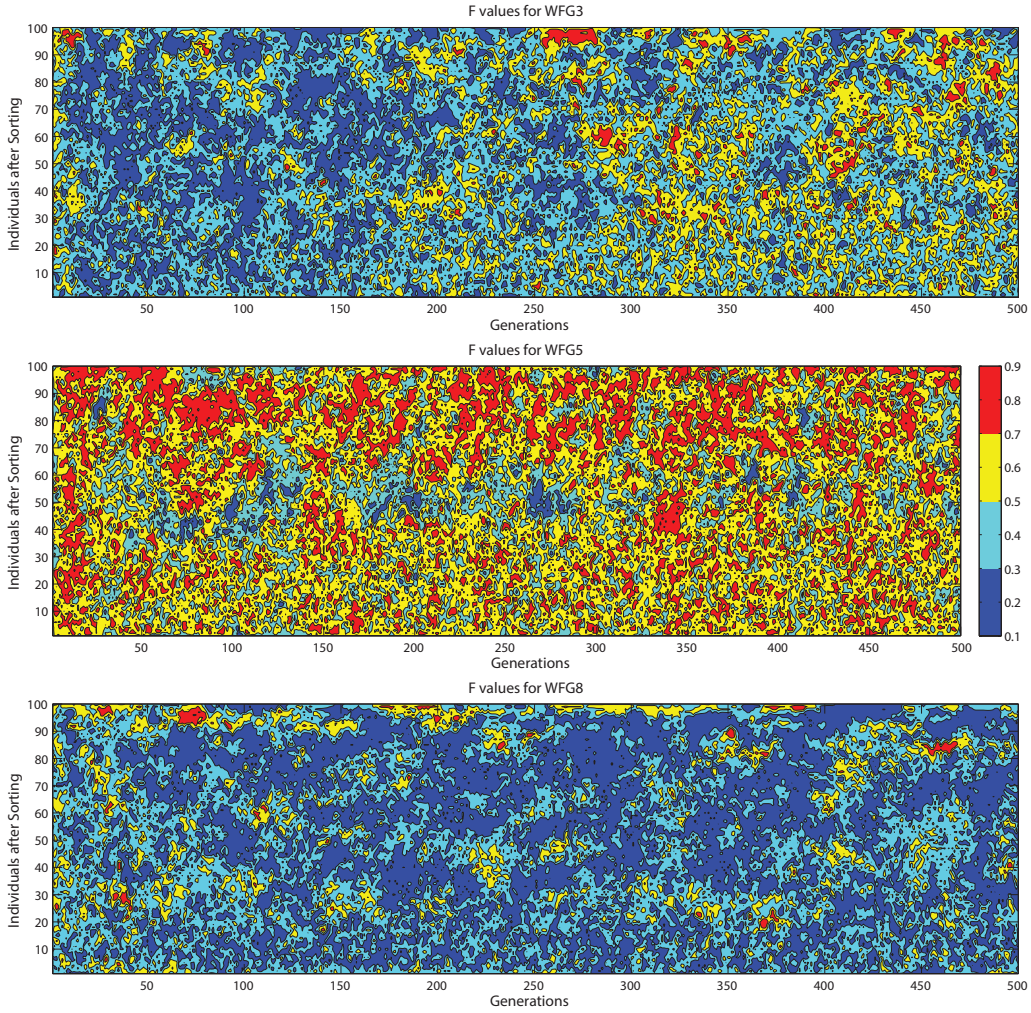

Figure 4.8: This figure shows the associated F values for each individual in every generation. Vertical axis indicates the index of each individual, and horizontal axis represents the number of current generation. The color of each dot reflects the F value of its corresponding individual in the following way: red: $F \in [0.7\ 0.9]$; yellow: $F \in [0.5\ 0.7]$; light blue: $F \in [0.3\ 0.5]$; dark blue: $F \in [0.1\ 0.3]$. A Gaussian smoothing operator is utilized in the post process of these contour plots.

Table 4.7: Mean and Standard Deviation of the IGD Values (30 runs)

Problems	with CGA	without CGA
	Mean ± Std	Mean± Std
UF1	**0.0528 ± 0.0148**	0.0729 ± 0.0410
UF2	**0.0205 ± 0.0027**	0.0264 ± 0.0069
UF3	**0.0947 ± 0.0139**	0.0953 ± 0.0158
UF4	0.0410 ± 0.0003	**0.0397 ± 0.0004**
UF5	**0.2870 ± 0.0932**	0.3852 ± 0.1044
UF6	**0.1576 ± 0.0849**	0.1918 ± 0.0797
UF7	**0.0262 ± 0.0071**	0.0467 ± 0.0465
UF8	**0.1383 ± 0.0420**	0.1965 ± 0.0813
UF9	**0.1776 ± 0.1091**	0.2914 ± 0.1385
UF10	**0.6440 ± 0.2715**	1.3504 ± 0.6490
WFG1	**0.8084 ± 0.0316**	0.8917 ± 0.0154
WFG2	**0.0139 ± 0.0008**	0.0387 ± 0.0558
WFG3	**0.0201 ± 0.0009**	0.0206 ± 0.0010
WFG4	**0.0196 ± 0.0014**	0.0200 ± 0.0011
WFG5	**0.0682 ± 0.0015**	0.0686 ± 0.0003
WFG6	**0.1175 ± 0.0205**	0.1200 ± 0.0179
WFG7	0.0166 ± 0.0007	**0.0164 ± 0.0009**
WFG8	**0.1089 ± 0.0036**	0.1094 ± 0.0044
WFG9	**0.1259 ± 0.0004**	0.1261 ± 0.0003

Next, a contrast experiment is performed to validate the effectiveness of CGA mechanism. CGA mechanism is removed from ACGDE, instead, fixed parameters are utilized as in original DE. The performance of the simplified version is compared with that of original ACGDE. Both algorithms are implemented in NSGA-II. In the simplified version, F is fixed as 0.5 and Cr is fixed as 0.35 based on the parameter intervals in CGA. Other experimental and parametric setups are identical with those in subsection 4.3.1. UF1 to UF10 from CEC-09 and WFG1 to WFG9 from WFG test suites are tested. Table 4.7 shows the mean and standard deviation of the IGD values for 30 independent runs on each benchmark. The best entries in terms of mean value are marked in boldface. According to Table 4.7, original ACGDE outperforms the simplified version without CGA mechanism in 17 out of 19 benchmark problems, from

which we can conclude that the introduction of CGA mechanism has substantially improved the performance of the algorithm.

4.4 Conclusion

This chapter has presented a new DE variant for multi-objective optimization, which employs information across generations to help guide the searching directions. Two variants of cross-generation mutation operators have been proposed to enhance both the convergence and diversity in the evolution. Furthermore, a cross-generation adaptation mechanism is introduced to tune the parameters dynamically from a perspective of objective space. Experimental results demonstrate that the proposed algorithmic components are effective and the new algorithm is able to significantly enhance the performance of NSGA-II and MOEA/D.

Chapter 5

Minimax Differential Evolution for Robust Design

In chapter 3 and chapter 4, all the involved objective functions are optimized under the assumption that one solution only has one corresponding objective value. However, in many real-world applications, the objective value is decided by both solution and the associated scenario. Even for the same solution, the objective value may vary dramatically under different scenarios. Taking the scenario factor into consideration, this chapter further extends the scope of this thesis by studying the minimax optimization problems, in which solutions with best worst-case performances are pursued. A new minimax DE algorithm is designed to circumvent the limitations of existing minimax optimization algorithms. Experimental comparisons over benchmark problems and implementations in two robust design applications demonstrate the effectiveness of the new approach.

5.1 Introduction

Minimax optimization problems, in which the solutions with best worst-case performances are preferred, can be utilized to model a wide variety of robust design tasks [12–19]. Unlike traditional optimization problems, two decision spaces are introduced simultaneously in minimax optimization, namely, solution space $\mathbb{X}$ and scenario space $\mathbb{S}$ (more detailed introduction is provided in subsection 1.3). For each solution in $\mathbb{X}$, its fitness depends on the corresponding worst-case scenario in $\mathbb{S}$. Thus, solving a minimax optimization problem involves explorations and communications among two interrelated spaces. This makes minimax optimization problems inherently challenging and difficult to address [27].

In the past 40 years, considerable research has been devoted to resolution of minimax optimization problems. Traditional approaches include branch-and-bound algorithms [156–158], mathematical programming [16, 159–164] and approximation methods [165]. The most significant limitation of these conventional approaches is that additional restrictions will be imposed on the objective functions [21], e.g., finite possibilities in scenario space [166], differentiability [167] and linearity [159].

Recently, a good volume of EAs have been developed to solve minimax optimization problems [22–26, 28, 106–108] where traditional methods are not applicable. The main advantage of EAs is that ideally no prior assumption is made about the underlying objective landscape. This generality allows EAs to handle those mathematically intractable problems, which may involve nondifferentiable, nonlinear, nonconvex or even more complex properties. Although

promising performances were observed in some works [21, 24, 104, 105], there do exist several fundamental drawbacks for existing minimax optimization EAs. First, many of the existing methods can only work properly under a symmetrical condition [22–27, 104–106], which means the original minimax optimization problem is equivalent to a symmetrical maximin problem (see subsection 1.3 for detailed definition). However, this condition does not hold generally, and problems not satisfying this condition are known to be extraordinarily difficult to solve [28]. Second, since most of the computational budget is assigned to the search of worst-case scenario for each solution, the overall optimization efficiency is deteriorated. While too much effort is spent on exploration in scenario space [21–26, 106], the optimization over solution space may be insufficient. This will lead to a bias towards reliability at the cost of quality. Third, while most existing algorithms focus on elaborating the evaluation mechanisms for scenarios, the evaluation criteria for solutions are underdeveloped. Either expensive computational cost or omission of excellent solutions may result from current evaluation approaches [21–27, 104–106].

To circumvent the above issues, a novel Minimax Differential Evolution (MMDE) algorithm is proposed by completely redesigning the whole algorithmic structure of conventional DE [9], which proves to be one of the most powerful methods for solving traditional optimization problems [10, 32, 34, 36, 40, 44, 45, 58, 109, 111, 112, 168]. In MMDE, a new Bottom-Boosting Scheme is developed to identify the promising solutions accurately while skipping considerable numbers of unnecessary objective function evaluations. Moreover, a Partial-Regeneration Strategy and a new DE mutation operator are introduced to

enhance both the reliability and efficiency of the solution optimization. Finally, with a proper combination of these algorithmic components, MMDE is able to overcome the limitations of existing algorithms in solving asymmetrical problems. Superior performance of MMDE is observed in empirical comparisons with other famous minimax optimization EAs. In addition, two open problems in robust design area are formulated into minimax optimization tasks, and both of them are successfully solved by applying the proposed MMDE algorithm.

5.2 Minimax Differential Evolution

5.2.1 Overview

During the development of MMDE, three main limitations of existing minimax optimization EAs are considered and addressed.

The first issue is the incapability of most approaches, especially coevolutionary EAs, in solving asymmetrical problems. Take the following two-plane function [104] as an example

$$f(x,s) = \min\{3 - \frac{2}{10}x + \frac{3}{10}s, 3 + \frac{2}{10}x - \frac{1}{10}s\} \tag{5.1}$$

Our target is to solve the following minimax optimization problem

$$\min_{x\in[0,10]} \max_{s\in[0,10]} f(x,s) \tag{5.2}$$

Fig. 5.1 shows the surface plot for (5.1). In coevolutionary EAs, the solution population and scenario population are optimized interactively. No matter what evaluation strategy is employed, the purpose for scenario optimization is always to find the worst-case scenarios for the individuals in solution population. Analogously, the target of solution optimization is always to obtain the

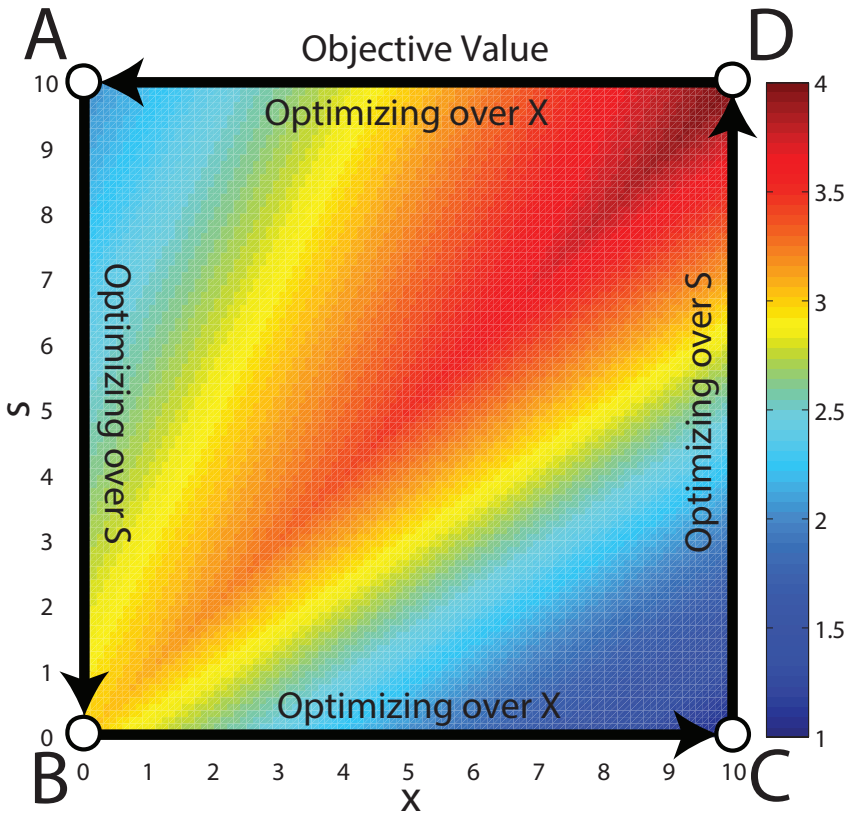

Figure 5.1: This figure plots the surface of objective function described by (5.1). The color level represents the objective value. The four arrows demonstrate the optimization cycle that will occur during existing coevolutionary algorithms.

best-performing solutions under the scenarios maintained in scenario population. According to Fig. 5.1, this objective function has 4 corners that may lead to optimization convergence, namely, A(0,10), B(0,0), C(10,0) and D(10,10). Suppose all the individuals in solution population and scenario population are randomly dispersed during the initialization, then in the solution optimization phase, all the individuals in solution population will have an evolutionary pressure to move to either 0 or 10. This will drive the solution-scenario combinations to distribute along the edge AB or CD. Similarly, in the scenario optimization process, the solution-scenario combinations tend to move to the diagonal BD, which can represent the worst-case situations for all the solutions. Based on the above analysis, the attractive areas overlap at point B and point D. However,

the solutions located in B and D are still under evolutionary pressure to move towards C and A, respectively. After that, the scenarios located in C and A will be optimized towards D and B, respectively. It can be observed that actually there is an endless optimization cycle among A, B, C and D. The existence of this cycle is the fundamental cause for the failures of coevolutionary EAs in solving asymmetrical problems. It is notable that although some coevolutionary algorithms were claimed to successfully handle asymmetrical problems by modifying the evaluation schemes for scenarios [104–106], none of these approaches are able to avoid the above issue. The reason is that in all coevolutionary EAs, the essence of the optimization over solution space is searching for the best-performing solutions under certain scenarios rather than finding the solutions with best worst-case performance. Even though the algorithm successfully converges to the global optima, which is point B(0,0), the optimization over solution space will move the solutions away from 0, since solution $X = 10$ gives a lower cost than $X = 0$ under scenario $S = 0$. Our theoretical analysis was verified by the empirical results in [21], which show that all the coevolutionary EAs even perform worse than a purely random search in solving asymmetrical problems. To overcome this issue, MMDE introduced a totally new algorithmic structure to avoid any optimization cycle. The solution optimization process in MMDE will only focus on moving the solutions towards the regions that provide better worst-case performance.

The second issue is the excessive search for worst-case scenarios, which may lead to imbalance between solution reliability and solution quality. In existing approaches, all the solutions are treated equally and same efforts are spent on

exploring their corresponding worst-case scenarios. However, a great number of fitness evaluations may be wasted in finding the worst-case scenarios for poor solutions, thereby degrading the overall optimization efficiency. In MMDE, the exploration of worst-case scenarios will be performed in a more flexible manner. A Bottom-Boosting Scheme is proposed to skip large quantities of unnecessary scenario evaluations.

The third issue with existing algorithms is the difficulty in selecting promising solutions both inexpensively and correctly. A greedy evaluation strategy is able to avoid the miss of good solutions, but the computational cost is extremely high. A low-cost evaluation strategy can substantially reduce the numbers of objective function evaluation while it is at the risk of propagating poor solutions. In order to trigger a delicate trade-off between efficiency and reliability, a Partial-Regeneration Strategy and a new DE mutation operator are developed in MMDE. Together with Bottom-Boosting Scheme, the combination of these new mechanisms enables the algorithm to efficiently identify the truly promising solutions and continue exploiting based on this information.

5.2.2 Algorithmic Description

General Structure

Algorithm 5.1 and Fig. 5.2 show the general structure of MMDE. The first step is the parametric setup and initialization of population, in which each individual is represented as a pair of solution and scenario. Subsequently, at the start of each generation, a min heap [169] will be constructed based on the current population. Each individual will form one node in the min heap, and the key of

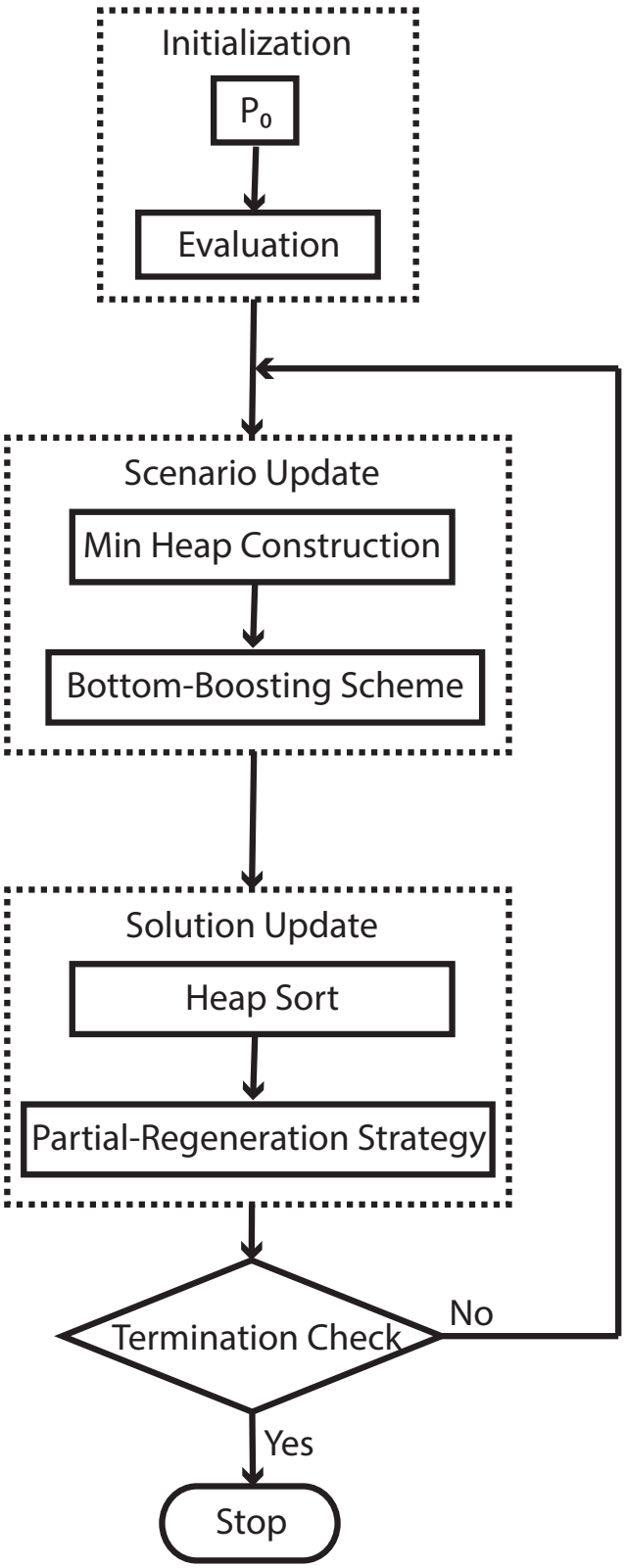

Figure 5.2: This flowchart presents the procedure of Minimax Differential Evolution algorithm.

Algorithm 5.1 Procedure of Minimax Differential Evolution

1: Parametric setup for MMDE: population size N, solution dimension D_X, scenario dimension D_S, maximum generation number $G_{\max}$ and control parameters for each algorithmic component
2: Set generation $g = 0$ and randomly initialize the population $P_g = \{(X_{1,g}, S_{1,g}), (X_{2,g}, S_{2,g}), \ldots, (X_{N,g}, S_{N,g})\}$, where $X_{i,g} = \{x_{i,g}^1, x_{i,g}^2, \ldots, x_{i,g}^{D_X}\}$ and $S_{i,g} = \{s_{i,g}^1, s_{i,g}^2, \ldots, s_{i,g}^{D_S}\}$, $i = 1, 2, \ldots, N$
3: **for** $i = 1$ to N **do**
4: Evaluate objective value for individual $(X_{i,g}, S_{i,g})$
5: **end for**
6: **for** $g = 1$ to $G_{\max}$ **do**
7: Construct a min heap with each individual in P_g as one node, the key of each node is the objective value of corresponding individual.
8: Perform Bottom-Boosting Scheme based on the constructed min heap
9: **for** $i = 1$ to N **do**
10: Update individual $(X_{i,g}, S_{i,g})$ with the individual stored in the root node of the min heap
11: Pop the root node from the min heap
12: **end for**
13: $X_{\text{best}} = X_{1,g}$
14: Perform Partial-Regeneration Strategy to update P_g
15: $P_{g+1} = P_g$
16: **end for**

each node is the objective value of the corresponding individual. The main feature of a min heap is that the node with the minimum key value will always be placed at the root, and it only takes $O(\log n)$ time to delete the current root node and $O(1)$ time to insert a new node [170], where n is the total number of nodes inside the heap. Based on the constructed min heap, a Bottom-Boosting Scheme will be performed to update the scenarios associated with the nodes. After that, the current population will be updated by continuously extracting root nodes from the min heap. This will lead to a population wherein all the individuals are sorted according to their corresponding objective values in non-descending order. The associated solution of the first individual is recorded as the best solution found so far. Finally, a Partial-Regeneration Strategy will be conducted to further update current population, and the algorithm will then proceed to the

next generation. The whole optimization process will be terminated when the maximum generation number is reached.

Bottom-Boosting Scheme

Algorithm 5.2 Bottom-Boosting Scheme

Require:

A min heap constructed from current population P_g

K_S: Parameter to control the total number of objective function evaluations

F and Cr: Control parameters for DE operators

Ensure:

An updated min heap

1: $k = 0$
2: **while** $k < K_S$ **do**
3: Read the root node of the min heap, suppose the individual stored inside the root node is $(X_{i,g}, S_{i,g})$
4: generate mutant scenario as $SV_{i,g} = S_{r_1,g} + F \cdot (S_{r_2,g} - S_{r_3,g})$ where indices r_1, r_2, r_3 are randomly selected integers from $\{1, 2, \ldots, N\}$ that are distinct from i and mutually different, and N is the population size
5: Perform binomial recombination on $S_{i,g}$ and $SV_{i,g}$ to generate the trial scenario $SU_{i,g}$
6: **if** $f(X_{i,g}, SU_{i,g}) > f(X_{i,g}, S_{i,g})$ **then**
7: Pop the root node from the min heap
8: $S_{i,g} = SU_{i,g}$
9: Push the updated individual $(X_{i,g}, S_{i,g})$ into the min heap
10: **end if**
11: $k = k + 1$
12: **end while**

Algorithm 5.2 describes the internal procedure of the proposed Bottom-Boosting Scheme. A new parameter K_S is introduced to control the total number of objective function evaluations incurred. Given the min heap constructed from current population, the individual stored in the root node will be extracted. Based on the scenario associated with this individual, a traditional "DE/rand/1" mutation strategy [9] and binomial recombination will be performed to generate the trial scenario. Afterward, the original solution will be evaluated with

the trial scenario to obtain a new objective value. If the new objective value is larger than the original objective value, the scenario associated with the original individual will be updated using the trial scenario. The corresponding node will also undergo a pop-update-push process to ensure the min heap is updated. The above steps will be repeated K_S times so totally K_S objective function evaluations are involved. Please note that the updating of scenarios will be performed on both current population and the min heap.

Partial-Regeneration Strategy

Algorithm 5.3 Partial-Regeneration Strategy

Require:

Current population P_g with population size N, all the individuals have already been sorted according to their corresponding objective values in non-descending order

T: Parameter to control the number of regenerated individuals

F and Cr: Control parameters for DE operators

Ensure:

Updated population P_g

1: $i = 1$
2: **while** $i \leq T$ **do**
3: $V_{i,g} = X_{i,g} + F \cdot (X_{r_1,g} - X_{r_2,g})$
where indices r_1, r_2 are randomly selected integers from $\{1, 2, \ldots, N\}$ that are distinct from i and mutually different
4: Perform binomial recombination on $X_{i,g}$ and $V_{i,g}$ to generate the trial solution $U_{i,g}$
5: $X_{N+1-i,g} = U_{i,g}$
6: Randomly reinitialize $S_{N+1-i,g}$
7: Evaluate the objective value of the updated individual $f(X_{N+1-i,g}, S_{N+1-i,g})$
8: $i = i + 1$
9: **end while**

Algorithm 5.3 shows the basic steps of the proposed Partial-Regeneration Strategy. A new parameter T is introduced to control the total number of regenerated individuals. Before performing Partial-Regeneration Strategy, all the individuals have already been sorted according to their corresponding objective

values in non-descending order. For the first T individuals, a new mutation strategy, namely "DE/current/1", will be performed on each associated solution to generate T mutant solutions. Suppose $X_{i,g}$ is the associated solution for ith individual, its corresponding mutant solution will be generated via "DE/current/1" as

$$V_{i,g} = X_{i,g} + F \cdot (X_{r_1,g} - X_{r_2,g}) \tag{5.3}$$

where indices r_1, r_2 are randomly selected integers from $\{1, 2, \ldots, N\}$ that are distinct from i and mutually different, and N is the population size. After that, T trial solutions will be generated by performing binomial recombination on each mutant solution and its corresponding original solution. Next, the associated solutions of the last T individuals will be updated using these trial solutions, and their associated scenarios will also be randomly reinitialized. The last step is to evaluate all these regenerated individuals and record their new objective values. Totally T objective function evaluations are carried out in the whole process.

5.2.3 Underlying Rationale

Different from traditional coevolutionary EAs, there is only one population in MMDE and each individual is represented as a pair of solution and scenario. Based on this representation, the optimization process in each generation is divided into two phases, in which the scenarios and solutions are updated respectively. Briefly speaking, the scenario updating phase aims at searching for the promising solutions in terms of worst-case performance, and the solution updating phase tries to further exploit based on these solutions.

In scenario updating phase, the Bottom-Boosting Scheme is employed to identify the promising solutions. Naturally, this can be done by searching for the worst-case scenarios for each solution. However, a large number of objective function evaluations will be wasted because our target is actually to find those solutions with good worst-case performance rather than obtain the worst-case scenarios for all the solutions. If the performance of solution X_1 under certain scenario has already been worse than the worst-case performance of X_2, it becomes pointless to further explore X_1 over scenario space. Following this consideration, the Bottom-Boosting Scheme will focus on evolving scenarios for the best-performing individuals only. A heap data structure is utilized to minimize the computation time for finding the new best-performing individual after each updating. Compared to a normal sorting approach, the computational complexity for each best-finding operation can be reduced from $O(n \log n)$ to $O(\log n)$ by using a heap. The root node of the min heap always stores the individual with the lowest objective value. The Bottom-Boosting Scheme will keep performing traditional "DE/rand/1" mutation operation and binomial recombination to search for worse scenarios for current best-performing solution. Due to the strong explorative ability of "DE/rand/1" operator, it will be very difficult for one solution to keep staying in the root node unless this solution is truly superior in terms of worst-case performance. Those solutions with poor performance under certain scenario will be eliminated from the optimization process in the early stage so that a large number of unnecessary evaluations can be skipped. In contrast, the solutions with robust performance under various scenarios will be more frequently challenged during their stay in the root node,

thereby further strengthening the reliability of the final outstanding solutions.

In solution updating phase, the Partial-Regeneration Strategy is adopted to evolve the whole population. Based on the min heap constructed in previous step, a heap sort is carried out to sort all the existing individuals based on their objective values. The worst T individuals will then be replaced by the offspring of the best T individuals. The proposed "DE/current/1" mutation strategy generates the mutant solution by adding a scaled perturbation into the original solution. In this way, the offspring will keep exploiting around the promising solutions so that the overall optimization efficiency is enhanced. The associated scenarios of the regenerated solutions are randomly reinitialized because maintaining the diversity of the scenarios is beneficial to both explorative ability and reliability of the scenario updating phase. Same as the best T individuals, the remaining intermediate individuals are kept unchanged so that their robustness can be further judged in the next scenario updating phase.

With the specially designed algorithmic structure, MMDE is able to avoid the infinite optimization cycles that coevolutionary approaches will encounter in handling asymmetrical problems. The key point is that MMDE successfully avoid the behaviors of finding good solutions for any particular scenarios. MMDE only searches for bad scenarios for particular solutions. The evolution of solutions in MMDE is only based on their performance during scenario updating phase. The individuals survive to next generation are the ones providing robust performance under various scenarios instead of the ones with superior performance under particular scenario.

5.3 Empirical Study

5.3.1 Comparison with Existing Algorithms

The optimization performance of the proposed MMDE is evaluated by comparing with seven famous minimax optimization EAs, namely, Aging Sampled GA (ASGA) [21], Alternating Coevolutionary PSO (ACPSO) [24], Alternating Coevolutionary GA (ACGA) [26], Parallel Coevolutionary GA (PCGA) [27], Best Remaining Coevolutionary GA (BRCGA) [105], Rank-Based Coevolutionary GA (RBCGA) [104] and Stackelberg Strategy Coevolutionary GA (SSCGA) [21]. For all the existing algorithms, parametric settings suggested in [21] are utilized. For MMDE, the control parameters are set as follows: $F = 0.7$ and $Cr = 0.5$ for both Bottom-Boosting Scheme and Partial-Regeneration Strategy, $N = 100$, $K_S = 190$, $T = 10$.

Following the recent minimax optimization studies [21, 28, 108], the six most commonly used benchmark problems [104, 171] are tested in our experiments. Table 5.1 shows the objective functions, searching domains and global optima of the six benchmarks. F1, F5 and F6 are symmetrical problems, and F2, F3 and F4 are asymmetrical problems. Mean square error (MSE) [104] is used as the performance metric to quantitatively compare the optimization performances of algorithms. MSE is calculated according to the best solution obtained by the algorithm and the global optimal solution as

$$MSE(X_{\text{best}}, X_{\text{opt}}) = \frac{1}{D_X} \sum_{j=1}^{D_X} (x_{\text{best}}^j - x_{\text{opt}}^j)^2 \tag{5.4}$$

where X_{best} is the best solution found by the algorithm, X_{opt} is the global optimal solution, D_X is dimensionality of solution space, x_{best}^j and x_{opt}^j are the

Table 5.1: Description of Benchmark Problems

F1	Objective	$f(x,s) = (x-5)^2 - (s-5)^2$
	Domain	$x \in [0,10], s \in [0,10]$
	Optimum	$(x^*, s^*) = (5,5)$
F2	Objective	$f(x,s) = \min\{3 - 0.2x + 0.3s, 3 + 0.2x - 0.1s\}$
	Domain	$x \in [0,10], s \in [0,10]$
	Optimum	$(x^*, s^*) = (0,0)$
F3	Objective	$f(x,s) = \frac{\sin(x-s)}{\sqrt{x^2+s^2}}$
	Domain	$x \in (0,10], s \in (0,10]$
	Optimum	$(x^*, s^*) = (10, 2.125683)$
F4	Objective	$f(x,s) = \frac{\cos\sqrt{x^2+s^2}}{\sqrt{x^2+s^2}+10}$
	Domain	$x \in [0,10], s \in [0,10]$
	Optimum	$(x^*, s^*) = (7.044146333751212, 10 \text{ or } 0)$
F5	Objective	$f(X,S) = 100(x_2 - x_1^2)^2 + (1-x_1)^2$ $-s_1(x_1 + x_2^2) - s_2(x_1^2 + x_2)$
	Domain	$X \in [-0.5, 0.5] \times [0,1], S \in [0,10]^2$
	Optimum	$(X^*, S^*) = (0.5, 0.25, 0, 0)$
F6	Objective	$f(X,S) = (x_1-2)^2 + (x_2-1)^2$ $+s_1(x_1^2 - x_2) + s_2(x_1 + x_2 - 2)$
	Domain	$X \in [-1,3]^2, S \in [0,10]^2$
	Optimum	$(X^*, S^*) = (1, 1, \text{any}, \text{any})$

variables in dimension j. All of the simulations were done on an Intel (R) Core (TM) i7 machine with 16-GB RAM and 3.40-GHz speed. Microsoft (R) Visual Studio (R) 2012 Express is used to develop the coding and run the experiments.

Table 5.2 and Table 5.3 present the mean, median and standard deviation of the MSEs for each algorithm on each benchmark problem over 100 independent runs. For all the algorithms, the maximum number of objective function evaluations or fitness evaluations (FEs) is fixed as 100000. Apart from the limited number of FEs, another termination condition is that the mean of the MSEs over 100 independent runs reaches 10^{-20} level or lower. For each problem, the total number of FEs performed by each algorithm in each run is also shown in Table

Table 5.2: Summarized Results over 100 Independent Runs for F1-F3

Problems		Mean	Median	Std	P-value	FEs
F1	MMDE	**0.0000E+00**	**0.0000E+00**	0.0000E+00	NA	**48500**
	ASGA	6.7677E-17	3.2221E-20	4.3812E-16	5.6400E-39	100000
	ACPSO	3.8636E-02	9.3041E-03	7.1026E-02	5.6400E-39	100000
	ACGA	3.8149E-05	6.9977E-15	2.7335E-04	5.6400E-39	100000
	PCGA	1.4495E-04	1.7041E-16	9.6055E-04	8.0734E-32	100000
	BRCGA	3.2412E-05	1.9274E-17	1.5046E-04	8.0708E-32	100000
	RBCGA	3.1323E-04	8.5099E-21	1.7825E-03	8.3171E-29	100000
	SSCGA	4.8554E-06	5.4367E-17	3.2735E-05	2.7251E-30	100000
F2	MMDE	**0.0000E+00**	**0.0000E+00**	0.0000E+00	NA	**68500**
	ASGA	8.0707E-05	1.1160E-12	7.5651E-04	5.6400E-39	100000
	ACPSO	1.7507E+01	1.5118E-02	3.5838E+01	5.6400E-39	100000
	ACGA	3.0871E+01	1.0128E-04	4.6208E+01	5.6400E-39	100000
	PCGA	3.0010E+01	1.1536E-04	4.5080E+01	5.6400E-39	100000
	BRCGA	2.8730E+01	1.5118E-02	3.5838E+01	5.6400E-39	100000
	RBCGA	9.4512E+00	9.2164E-09	2.8471E+01	5.6400E-39	100000
	SSCGA	4.8793E+01	1.7764E+01	4.9080E+01	5.6400E-39	100000
F3	MMDE	**0.0000E+00**	**0.0000E+00**	0.0000E+00	NA	**2700**
	ASGA	1.3934E-04	2.3755E-09	8.0636E-04	5.6400E-39	100000
	ACPSO	2.9861E+00	6.4173E-01	3.5121E+00	5.6400E-39	100000
	ACGA	2.5216E+01	1.4978E+01	2.7667E+01	5.6400E-39	100000
	PCGA	2.1888E+01	1.2116E+01	2.7996E+01	5.6400E-39	100000
	BRCGA	2.8629E+01	1.0285E+01	3.6638E+01	5.6400E-39	100000
	RBCGA	1.4610E+01	6.2826E-02	2.8781E+01	5.6400E-39	100000
	SSCGA	2.7145E+01	9.0744E+00	3.6543E+01	5.6400E-39	100000

Table 5.3: Summarized Results over 100 Independent Runs for F4-F6

Problems		Mean	Median	Std	P-value	FEs
F4	MMDE	**1.2098E-21**	**7.0997E-30**	1.2020E-20	NA	**59900**
	ASGA	8.7356E-03	8.4006E-04	6.7046E-02	1.9751E-34	100000
	ACPSO	5.8835E+00	1.4031E+00	9.1577E+00	1.9751E-34	100000
	ACGA	1.1798E+01	6.7570E+00	1.3040E+01	1.9751E-34	100000
	PCGA	1.1210E+01	3.1082E+00	1.5225E+01	1.9751E-34	100000
	BRCGA	7.3399E+00	2.7788E+00	1.0651E+01	1.9751E-34	100000
	RBCGA	1.7349E+00	2.8796E-05	4.9994E+00	1.9751E-34	100000
	SSCGA	1.1879E+01	5.0549E+00	1.5445E+01	1.9751E-34	100000
F5	MMDE	**9.9702E-20**	**8.1885E-20**	8.7226E-20	NA	**27300**
	ASGA	5.8738E-03	8.5873E-04	1.2063E-02	2.5621E-34	100000
	ACPSO	3.5519E-02	2.2115E-02	4.1838E-02	2.5621E-34	100000
	ACGA	1.4323E-02	4.4641E-03	2.2798E-02	2.5621E-34	100000
	PCGA	1.1241E-02	2.1755E-03	2.0305E-02	2.5621E-34	100000
	BRCGA	1.6391E-02	2.8940E-03	2.9882E-02	2.5621E-34	100000
	RBCGA	1.5835E-02	4.9840E-03	2.4590E-02	2.5621E-34	100000
	SSCGA	1.4666E-02	4.0267E-03	2.3430E-02	2.5621E-34	100000
F6	MMDE	**1.6830E-13**	**1.6055E-17**	8.4048E-13	NA	100000
	ASGA	8.0846E-04	8.4348E-05	1.6708E-03	4.2663E-34	100000
	ACPSO	1.2959E-01	9.4149E-02	1.2564E-01	2.5621E-34	100000
	ACGA	1.3966E-02	3.7605E-03	2.3574E-02	2.5621E-34	100000
	PCGA	9.2942E-03	4.5820E-04	2.3378E-02	2.5621E-34	100000
	BRCGA	2.1609E-02	2.7748E-04	9.7750E-02	2.5621E-34	100000
	RBCGA	2.9893E-03	1.1057E-04	1.5591E-02	2.5621E-34	100000
	SSCGA	4.2151E-02	1.8814E-03	1.2522E-01	2.5621E-34	100000

5.2 and Table 5.3. In order to judge whether the results of the best-performing algorithm in terms of mean of MSEs differ from the results of the competitors in a statistically significant way, a non-parametric statistical test called Wilcoxons rank-sum test [118] is conducted at the 5% significance level. The P-values obtained through the rank sum test between the best algorithm and each of the remaining algorithms over all the benchmark functions are presented in Table 5.2 and Table 5.3. NA stands for $not\ applicable$ and occurs for the best performing algorithm itself in each case. If the P-values are less than 0.05, it indicates that the better performances achieved by the best algorithm in each case are statistically significant and have not occurred by chance [172]. In Table 5.2 and Table 5.3, the best entries in terms of mean of MSEs, median of MSEs and number of FEs are marked in boldface.

Based on the experimental results in Table 5.2 and Table 5.3, the proposed MMDE is able to $significantly$ outperform all the other algorithms in all the benchmark problems in terms of both mean and median of MSEs over 100 independent runs. For the first three problems, MMDE can reach the global optimal solution without any errors using only 48500, 68500 and 2700 FEs in all the 100 independent runs. For problem F4 and F5, the number of FEs required for MMDE to reduce the mean of MSEs into 10^{-20} level or lower is only 59900 and 27300, respectively. For problem F6, MMDE also achieves considerably lower MSEs than other approaches in all the 100 independent runs. For all the co-evolutionary EAs, poor performances were observed in the three asymmetrical problems (F2, F3 and F4). This is in accordance with our theoretical analysis in subsection 5.2.1. Besides MMDE, ASGA is the only algorithm that solves both

symmetrical and asymmetrical problems correctly. However, from the perspective of optimization efficiency, MMDE shows substantially better performance than ASGA. To summarize, MMDE is capable of solving both symmetrical and asymmetrical minimax optimization problems not only reliably but also efficiently.

5.3.2 Effectiveness of Bottom-Boosting Scheme

In MMDE, the Bottom-Boosting Scheme plays a very critical role in increasing the efficiency of the whole algorithm. A large number of objective function evaluations are skipped during the proposed mechanism. In order to further investigate the effectiveness of Bottom-Boosting Scheme, the performance of the original MMDE is compared with that of a simplified version. In the simplified MMDE, the only modification is that during the scenario updating phase, instead of only optimizing the current best-performing individual, the mutation, recombination and updating operations will be conducted on each individual one by one. For a more comprehensive investigation, the number of FEs used in each scenario updating phase of the simplified version (also defined by K_S) will be set from 200 to 2000, and all these different variants will be tested for comparison. For all the other control parameters and experimental setups, same settings as described in subsection 5.3.1 are employed. Fig. 5.3 shows the boxplots of the MSE values obtained by each algorithm over 100 independent runs on problem F2, F4 and F5.

From the experimental results, the original MMDE achieves consistently better performance than all the variants of the simplified version. For problem

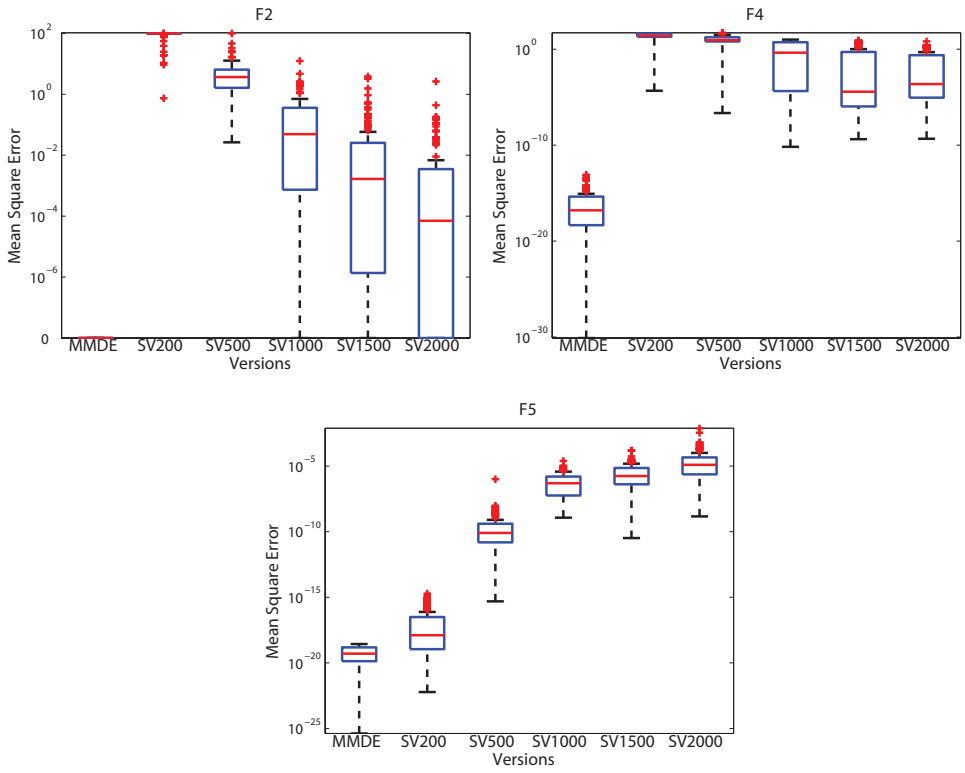

Figure 5.3: Boxplots of the MSE values over 100 independent runs. "SV200" represents the simplified version with $K_S = 200$. Same rule applies to all the remaining labels. On each box, the red line is the median, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points not consider outliers, and outliers are plotted individually using red markers.

F2, a K_S value less then 500 will lead to a very poor performance of the simplified version. The reason is that due to the evenly distribution of computational budget, each individual can only perform a limited number of FEs, which may not be sufficient to correctly evaluate their true quality. According to the results, a great number of FEs are required for the simplified version to provide an acceptable performance in F2. In comparison, the proposed Bottom-Boosting Scheme successfully reaches the global optimal solution with no errors in all the 100 independent runs with only 190 FEs in each scenario updating phase. Similar pattern can also be observed from the experimental results on problem F4. This implies that the proposed mechanism successfully enhances both the

efficiency and reliability of the whole algorithm. For problem F5, a small K_S is preferred by the simplified version. This is because F5 is a symmetrical problem that satisfies (1.14) and (1.15). All the individuals will gradually evolve their associated scenarios towards the same one (e.g., S^* in (1.15)), thereby reducing the difficulty for scenario optimization. Since a smaller number of FEs in each scenario updating phase allows more exploration over solution space, the solution quality is further enhanced. Under this circumstance, Bottom-Boosting Scheme still provides better overall performance than the simplified version with similar K_S values. This indicates the consistent effectiveness of the proposed scheme in solving problems with different properties.

5.3.3 Sensitivity Study for K_S and T

This subsection aims at studying the influences of the two new control parameters K_S and T on the overall optimization performance.

Fig. 5.4 shows the boxplots of MSEs over 100 independent runs for different K_S settings. All the other control parameters and experimental setups are identical with those in subsection 5.3.1. For the three symmetrical problems F1, F5 and F6, relatively lower K_S values are desirable. As discussed in subsection 5.3.2, with the reduced difficulty in scenario optimization, the exploration over solution space becomes more important in handling symmetrical problems. A better trade-off between the scenario optimization, which is related to solution reliability, and solution optimization, which is relevant to solution quality, will be achieved by selecting a relatively smaller K_S. For the three asymmetrical problems F2, F3 and F4, the difficulty of the scenario optimization has increased

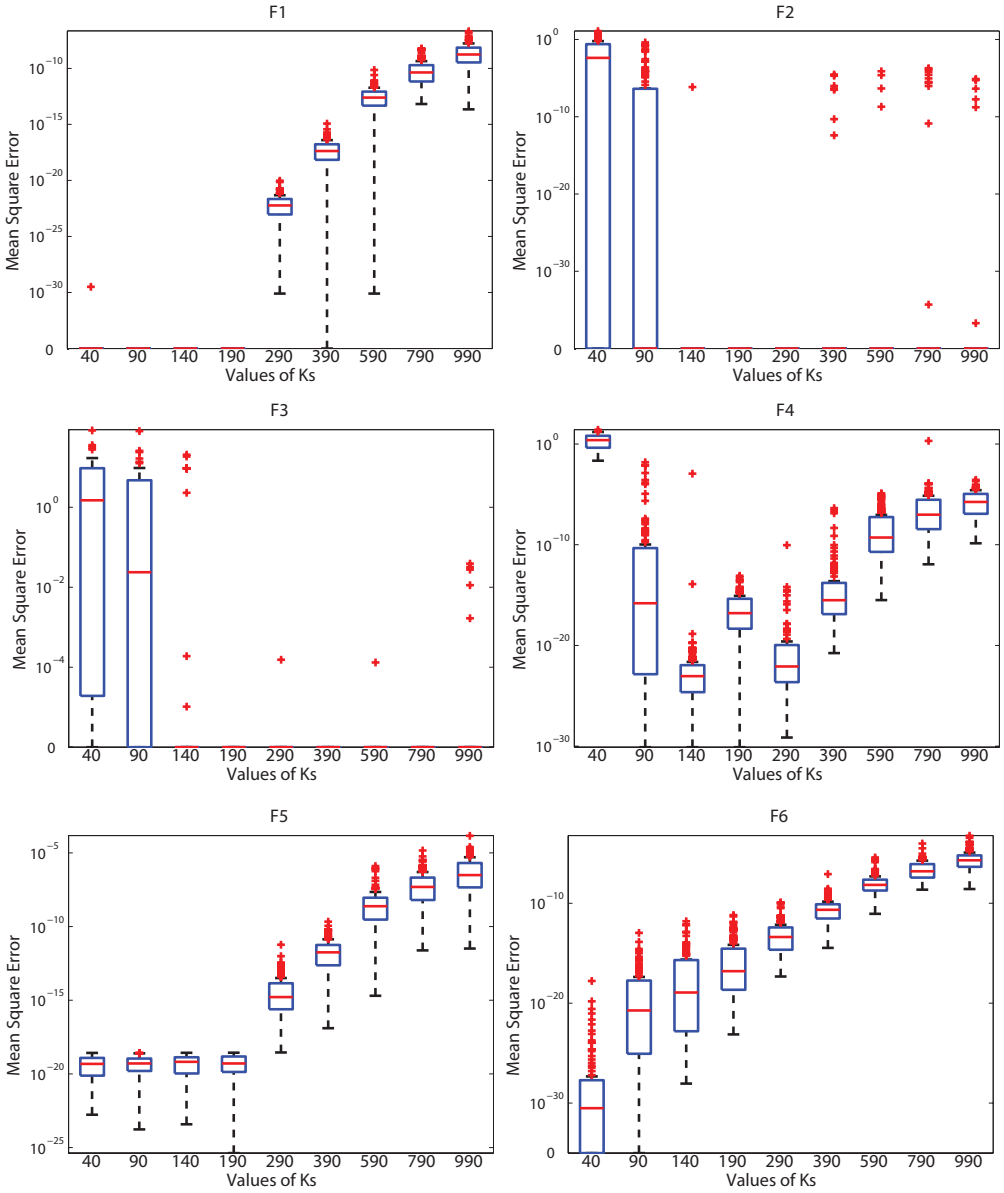

Figure 5.4: Boxplots of the MSE values over 100 independent runs for different K_S settings. On each box, the red line is the median, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points not consider outliers, and outliers are plotted individually using red markers.

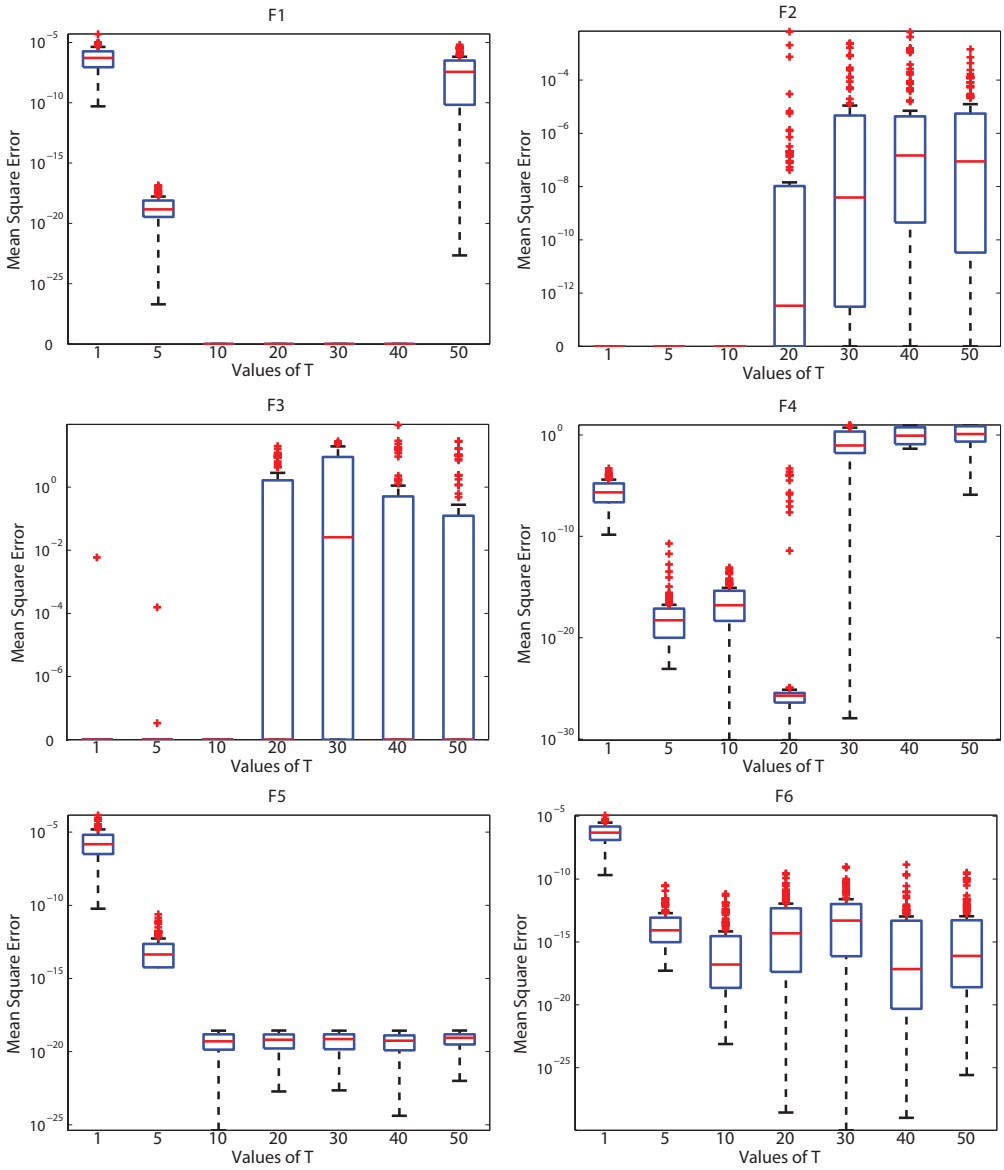

Figure 5.5: Boxplots of the MSE values over 100 independent runs for different T settings. On each box, the red line is the median, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points not consider outliers, and outliers are plotted individually using red markers.

because each individual has disparate worst-case scenarios. Based on the experimental results, a K_S value less than 100 is insufficient for the scenario updating phase to correctly identify the true promising solutions, and an over large K_S value may increase the solution reliability by sacrificing the solution quality. A moderate K_S value is therefore more suitable for asymmetrical problems.

Fig. 5.5 presents the boxplots of MSEs over 100 independent runs for different T settings. All the other control parameters and experimental setups are identical with those in subsection 5.3.1. For both symmetrical and asymmetrical problems, a T value of 10 provides the most robust performance. Either an over small or over large T may lead to unsatisfactory optimization performance regardless of the problem properties. This is because the value of T decides the number of regenerated individuals during each solution updating phase. If T is too small, the exploration over solution space will be conducted in a very slow manner. If T is too large, the regeneration of solutions will become excessively frequent, and the probability to discard promising solutions is increased. In order to avoid these two issues, a T value that is around 10% of the population size is recommended.

5.4 Applications

5.4.1 Robust Optimal Design of Iterative Learning Control

In this subsection, the proposed MMDE algorithm is applied to address an open problem in iterative learning control (ILC): how to systematically design an appropriate learning control gain matrix for nonlinear multi-input-multi-output

(MIMO) systems. An appropriately chosen learning gain matrix can speed up learning convergence in the presence of the system nonlinearities and uncertainties. Targeting at time-optimal (fastest convergence) and robustness properties concurrently, the ILC design task is formulated into a minimax optimization problem.

Consider an ILC for MIMO dynamic systems

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t), t) \qquad x(0) = x_0 \\ y(t) &= g(x(t), u(t), t) \end{aligned} \tag{5.5}$$

where $t \in [0, T]$, $x(t) \in \mathbb{X} \subset \mathbb{R}^n$, $y(t) \in \mathbb{Y} \subset \mathbb{R}^m$, $u(t) \in \mathbb{U} \subset \mathbb{R}^m$, $\mathbb{X}$, $\mathbb{Y}$, $\mathbb{U}$ are compact convex subsets. Nonlinear functions $f(\cdot)$ and $g(\cdot)$ satisfy the global Lipschitz continuity condition with respect to state x and input u.

The target of ILC is to find a sequence of appropriate control inputs such that the system output $y_i(t)$ can track the reference trajectory $y_r(t)$. A typical ILC is given as

$$u_{i+1}(t) = u_i(t) + Q\Delta y_i(t) \tag{5.6}$$

where $Q \in \mathbb{R}^{m \times m}$ is the learning gain matrix. According to [173–175], The ILC convergence condition is

$$\| \Delta y_{i+1} \| \leq \| I_{m \times m} - G(\xi)Q \| \| \Delta y_i \| \tag{5.7}$$

where $G \triangleq \frac{\partial g}{\partial u}$ is the direct feed-through matrix, ξ is a point in the compact set $\Omega = \mathbb{X} \times \mathbb{U} \times [0, T]$, $\| \cdot \|$ is an appropriate vector norm and the induced matrix norm. Clearly, to warrant a convergent learning sequence of $\| \Delta y_i \|$, a sufficient condition is

$$\| I_{m \times m} - G(\xi)Q \| \leq \gamma, \forall \xi \in \Omega \tag{5.8}$$

where $\gamma \in (0, 1)$ is a constant. Moreover, the smaller is $\parallel I_{m\times m} - G(\xi)Q \parallel$, the faster is the learning convergence speed. Hence, the ILC design task becomes to find an appropriate gain matrix $Q \in \mathbb{R}^{m\times m}$, such that $\parallel I_{m\times m} - G(\xi)Q \parallel$ is minimal under the worst-case $G(\xi)$, $\forall \xi \in \Omega$.

Now the robust optimal design can be formulated as the following minimax optimization problem

$$\min_{Q\in\mathbb{X}} \max_{\xi\in\Omega} \parallel I_{m\times m} - G(\xi)Q \parallel \tag{5.9}$$

where $\mathbb{X} = \mathbb{R}^{m\times m}$.

Due to the existence of nonlinearities, uncertainties and non-symmetry in the system direct feed-through matrix, it is in general a very difficult task to directly solve the original minimax optimization problem with a closed-form solution. Nevertheless, the proposed MMDE does not make any prior assumption about the mathematical properties of the problem. It is practicable to apply MMDE to solve this mathematically intractable optimization problem.

To verify the effectiveness of MMDE, the proposed algorithm is applied to the robust optimal design of a two-link robotic manipulator, which is a nonlinear dynamic system described by

$$M(x)\ddot{x} + f(x, \dot{x}) = u \tag{5.10}$$

where

$$M(x) = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \tag{5.11}$$

is the inertia matrix with

$$\begin{aligned} m_{11} &= m_1 l_{c1}^2 + I_1 + m_2[l_1^2 + l_{c2}^2 + 2l_1 l_{c2}\cos(x_2) + I_2] \\ m_{12} &= m_{21} = m_2 l_1 l_{c2} cos(x_2) + m_2 l_{c2}^2 + I_2 \\ m_{22} &= m_2 l_{c2}^2 + I_2 \end{aligned} \tag{5.12}$$

and

$$f(x, \dot{x}) = \begin{bmatrix} f_{11} \\ \\ f_{12} \end{bmatrix} \tag{5.13}$$

represents all Centrifugal, Corioli's and gravity terms with

$$\begin{aligned} f_{11} &= -2h\dot{x}_1\dot{x}_2 - h\dot{x}_2^2 + m_1 l_{c1} g\cos(x_1) \\ &\quad + m_2 g[l_{c2} cos(x_1 + x_2) + l_1\cos(x_1)] \\ f_{12} &= h\dot{x}_1^2 + m_2 l_{c2} g\cos(x_1 + x_2) \end{aligned} \tag{5.14}$$

$x = [x_1, x_2]^T$ are the two joint angles, $u = [u_1, u_2]^T$ are the joint inputs.

The angular velocities are selected to be the system output, $y = \dot{x}$. The desired trajectories are

$$y_{r,1} = y_{r,2} = 20(60\tau^3 - 30\tau^4 - 30\tau^2) \tag{5.15}$$

where $y_r = [y_{r,1}, y_{r,2}]^T$ and $\tau = t/T_c$. We choose $T_c = 1$ second to be the period of learning cycle.

The system parameters are given as: link masses $m1 = 4$Kg; $m2 = 3 + \Delta m_2$Kg; link lengths $l_1 = 0.5$meter; center of gravity co-ordinates $l_{c1} = 0.3$meter; $l_{c2} = 0.25$meter; and moments of inertia $I_1 = 0.4$Kg-meter^2; $I_2 = 0.25 + \Delta I_x$Kg-meter^2. In addition, there exist parametric uncertainties in m_2 and I_2: $\pm 50\%$ deviations from their nominal values.

Since there is no direct feed-through item in the input-output mapping, a D-type ILC scheme is employed

$$u_{i+1}(t) = u_i(t) + Q[\dot{y}_r(t) - \dot{y}_i(t)] \tag{5.16}$$

where $Q \in \mathbb{R}^{2\times 2}$. The resulting direct feed-through matrix is $G = M^{-1}$, and G is positive definite. With the above definitions of Q and G, the robust optimal design can be formulated into a minimax optimization problem using (5.9), and MMDE is applied to solve it.

Conventionally, the gain matrix Q is a diagonal matrix. Therefore in this example, Q can be represented as

$$Q = \begin{bmatrix} x_{11} & 0 \\ 0 & x_{22} \end{bmatrix}. \tag{5.17}$$

Thus, x_{11} and x_{22} are the two variables to be optimized over solution space. In the scenario space, there are three variables, namely, $m_2 \in [1.5, 4.5]$, $I_2 \in [0.125, 0.375]$ and $x_2 \in [-\pi, \pi]$. The objective values are calculated using $\| I_{m\times m} - G(\xi)Q \|$. The control parameters are set as follows: $F = 0.7$ and $Cr = 0.5$ for both Bottom-Boosting Scheme and Partial-Regeneration Strategy, $N = 100$, $K_S = 990$, $T = 10$, the maximum number of FEs is 2×10^7.

Based on the experimental results, the proposed MMDE is able to find numerous solutions that satisfy condition (5.8). Based on the solutions provided by MMDE, the feasible region in solution space have been successfully located, in which all the solutions are valid. Fig. 5.6 plots the worst-case objective values of all the solutions within or around the feasible region in solution space. All the solutions with worst-case objective values less than 1 are valid solutions for this design problem. Within the feasible region, the solutions with lower worst-case objective values are preferred since they provide generally faster learning convergence speed.

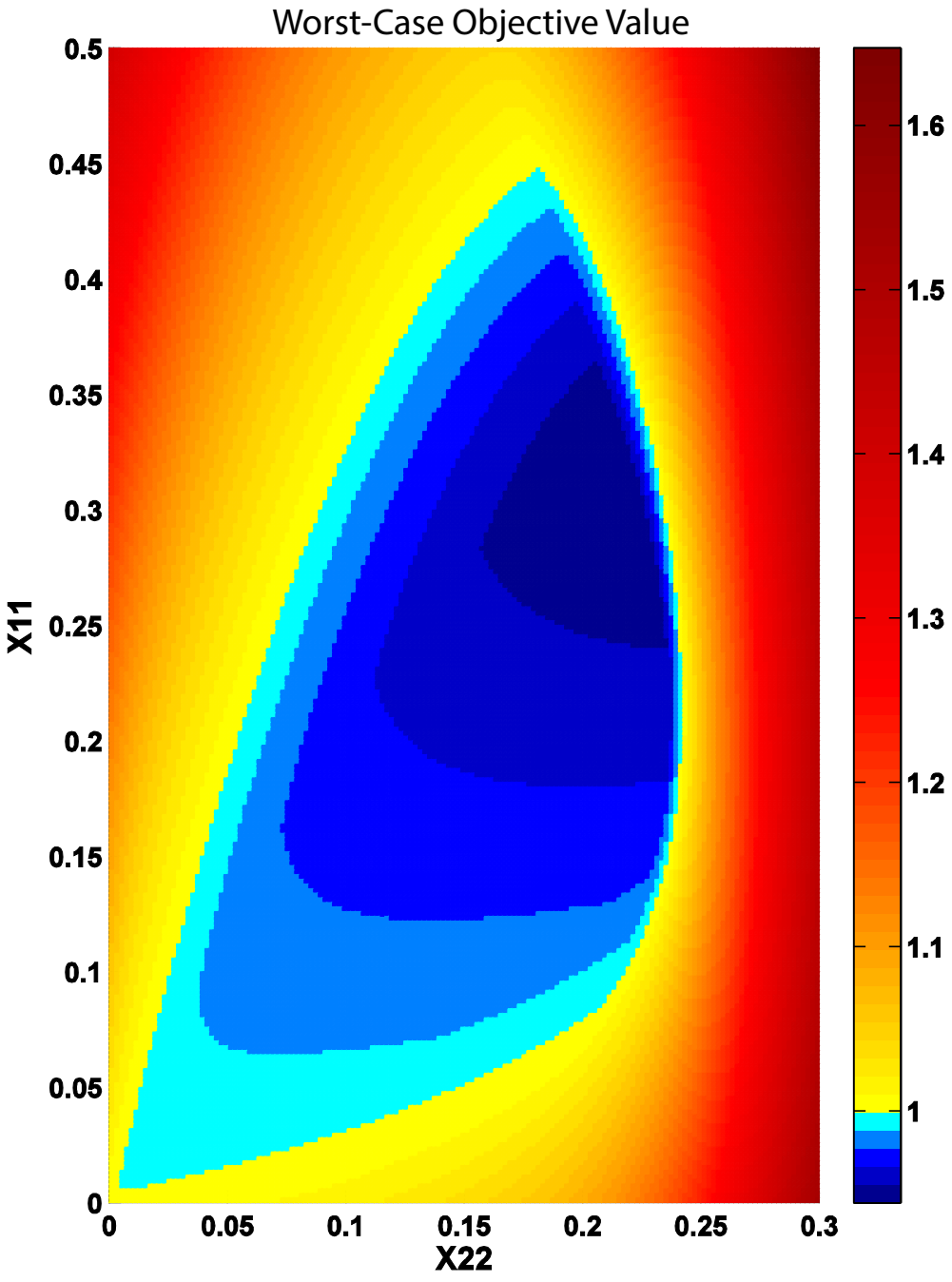

Figure 5.6: This figure plots the worst-case objective values of the solutions within or around the feasible region in solution space.

5.4.2 Robust Stabilization of Uncertain Time-Delay Systems

Time-delays are inevitable in many natural and industrial applications, and they often give rise to oscillations or instability of the entire systems. The robust stabilization of time-delay systems with uncertainties have been a very challenging problem in the realm of control theory [176, 177]. In [177], a linear matrix inequality (LMI) based approach was proposed to solve this robust stabilization problems, and it was claimed that the LMI approach obtained less conservative results compared to other traditional methods. In order to further examine the effectiveness of MMDE, this Robust Stabilization task will be formulated into a minimax optimization problem, and the results generated by MMDE will be compared with those of LMI method.

Consider an uncertain time-delay system containing nonlinear saturating actuators

$$\begin{aligned} &\dot{x}(t) = A(\delta)x(t) + A_1(\delta)x(t-d) + B(\delta)u'(t) \\ &u'(t) = sat(u(t)) \\ &sat(u(t)) = [sat(u_1(t))sat(u_2(t))\ldots sat(u_m(t))] \\ &x(t) = \phi(t), t \in [-\tau, 0] \end{aligned} \tag{5.18}$$

where $x(t) \in \mathbb{R}^n$ is the state vector, $u(t) \in \mathbb{R}^m$ is the control input vector of the actuator, $u'(t) \in \mathbb{R}^m$ is the control input vector to the plant, $A(\delta) = A + \Delta A(\delta)$, $A_1(\delta) = A_1 + \Delta A_1(\delta)$, $B(\delta) = B + \Delta B(\delta)$, $A \in \mathbb{R}^{n\times n}$, $A_1 \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ are constant matrices, $\Delta A(\cdot)$, $\Delta A_1(\cdot)$ and $\Delta B(\cdot)$ represent parameter uncertainties, $sat(\cdot)$ is the nonlinear saturation function, $\phi(t)$ is a smooth vector-valued continuous initial function.

The admissible uncertainties are assumed to be of the form

$$\begin{aligned} \Delta A(\delta) &= H_1 F_1(\delta) E_1 \\ \Delta A_1(\delta) &= H_2 F_2(\delta) E_2 \\ \Delta B(\delta) &= H_3 F_3(\delta) E_3 \end{aligned} \tag{5.19}$$

where $F_i(\delta) \in \mathbb{R}^{s_i \times q_i}$, $i = 1, 2, 3$ are unknown real matrices with Lebesgue measurable elements that satisfy

$$F_i^T(\delta) F_i(\delta) \leq I, i = 1, 2, 3 \tag{5.20}$$

and H_i, E_i, $i = 1, 2, 3$ are known real constant matrices.

The purpose of our task is to extend the upper bound τ of time-delay d such that the uncertain linear time-delay system is robustly stabilizable for any $0 < d \leq \tau$. In order to convert the robust stabilization task into a minimax optimization problem, the original system described in (5.18) is written as

$$\dot{x}(t) = \bar{A}x(t) + \bar{A}_1 x(t-d) + \bar{B}Lx(t) \tag{5.21}$$

where $\bar{A} = A + \Delta A(\delta)$, $\bar{A}_1 = A_1 + \Delta A_1(\delta)$, $\bar{B} = B + \Delta B(\delta)$, $Lx(t) = u'(t)$ and L is a predefined real-valued matrix. After performing Laplace transform on (5.21), we will have

$$sIX(s) = \bar{A}X(s) + \bar{A}_1 e^{-ds} X(s) + \bar{B}LX(s) \tag{5.22}$$

To ensure the system stability, all poles of the system must be negative, and this is equivalent to ensuring all the roots s of equation

$$det(sI - \bar{A} - \bar{A}_1 e^{-ds} - \bar{B}L) = 0 \tag{5.23}$$

are negative. $det(\cdot)$ means the determinant of a matrix. Now the target becomes to find a matrix L such that all the roots s of (5.23) are negative under any

uncertainties resulted from $\bar{A}$, $\bar{A}_1$ and $\bar{B}$. This can be achieved by solving a minimax optimization problem that aims at minimizing the value of the largest root of (5.23) under the worst-case uncertainty.

$$\min_{L} \max_{\bar{A},\bar{A}_1,\bar{B}} \{s_{\max} | det(sI - \bar{A} - \bar{A}_1 e^{-ds} - \bar{B}L) = 0\} \tag{5.24}$$

where $s_{\max}$ is the largest root of the equation. If the obtained largest root is less than 0, then it is guaranteed that all the remaining poles are also negative so that the system is successfully stabilized.

The same numerical example in [177] is tested here:

$$A = \begin{bmatrix} -2 & 0 \\ 1 & -3 \end{bmatrix}, A_1 = \begin{bmatrix} -1 & 0 \\ -0.8 & -1 \end{bmatrix}, B = \begin{bmatrix} 1 & 2 \\ -1 & 4 \end{bmatrix}$$
$$H_i = \begin{bmatrix} 0.2 & 0 \\ 0 & 0.2 \end{bmatrix}, E_i = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, i = 1, 2, 3$$
$$F_i = \begin{bmatrix} \sin(\delta) & 0 \\ 0 & \cos(\delta) \end{bmatrix}, i = 1, 2, 3 \tag{5.25}$$

Based on the above assumptions, L is a 2 by 2 matrix, and the 4 elements of L will be optimized by MMDE as variables in solution space. The searching domain for all the solution variables are $[-100, 100]$. δ will be treated as the variable in scenario space, and the search domain is $[0, 2\pi]$. Other control parameters of MMDE are set as follows: $F = 0.7$ and $Cr = 0.5$ for both Bottom-Boosting Scheme and Partial-Regeneration Strategy, $N = 100$, $K_S = 990$, $T = 10$, the maximum number of FEs is 1×10^6. Newton-Raphson method is employed to obtain the largest roots of (5.23) with an extremely big initial guess. For each L found by MMDE, an exhaustive search will be performed

over scenario space to ensure that the system is stabilized by L under all the possible uncertainties.

During the experiments, the time-delay d will be gradually increased if MMDE is able to find a matrix L that can robustly stabilized the system. Finally, the d value has been increased to 1.33, and it has been validated that MMDE is able to solve this robust stabilization task for any $0 < d \leq 1.33$. In comparison, the LMI approach can only robustly stabilize this system for time-delay $d \leq 0.2961$ [177]. Using MMDE, the upper bound for feasible time-delay has increased by 349.17%.

5.5 Conclusion

A MMDE algorithm is proposed in this chapter to overcome the limitations of existing approaches in solving minimax optimization problems. In scenario updating phase, the Bottom-Boosting Scheme successfully skips a large number of unnecessary objective function computations while maintaining the reliability of solution evaluations. In solution updating phase, the Partial-Regeneration Strategy and "DE/current/1" mutation operator allow an efficient solution exploration based on current population. Moreover, the overall algorithmic structure enables the proposed method to properly handle the asymmetrical problems. Experimental results show that MMDE outperforms all the other tested algorithms in both symmetrical and asymmetrical benchmarks. The effectiveness of MMDE is further validated by solving two open problems in robust design.

Chapter 6

Conclusion & Future Work

6.1 Conclusion

The primary focus of this thesis is to develop new DE-based operators and algorithms for solving various types of numerical optimization problems, e.g., problems with unknown variable interrelations, multi-objective problems and minimax problems. The proposed approaches in this thesis have successfully addressed several open issues in EA community, and the effectiveness of the new methods have been validated via both theoretical analysis and empirical studies.

In chapter 3, a multiple exponential recombination is proposed to overcome the limitations of existing DE crossover operators in handling dependent variables. By exchanging multiple segments among individuals simultaneously, the new operator is able to preserve all the main advantages of traditional binomial recombination and exponential recombination while showing a more robust performance in tackling different types of dependent variables. Implementations

in 6 classical DE variants and 1 adaptive DE algorithm demonstrate the strength of the proposed operator in solving problems with unknown variable interrelations.

In chapter 4, two novel mutation operators and one new parameter adaptation mechanism are proposed to circumvent the following issues for existing MO-DEs: slow convergence speed, parametric sensitivity and lack of flexibility in implementation. A new ACGDE algorithm is developed by systematically combining the proposed operators and mechanisms. ACGDE utilizes the information across different generations to predict the promising searching directions and favorable control parameters. Experimental results show that ACGDE is able to trigger a delicate trade-off between diversity and convergence. Comparisons with other state-of-the-art MOEAs verify the superiority of ACGDE in solving MOPs.

In chapter 5, a new DE-based algorithm is devised for solving several fundamental problems with existing minimax optimization EAs: incapability of solving asymmetrical problems, high computational cost and low optimization efficiency. The proposed Bottom-Boosting Scheme significantly reduce the computational cost for searching promising solutions without sacrificing the reliability. The new Partial-Regeneration Strategy and "DE/current/1" mutation operator substantially enhance the exploration efficiency for solution optimization. Proper integration of all the new mechanisms generates an algorithm that can successfully avoid the infinite optimization cycle while handling asymmetrical problems. The proposed algorithm is proved to be very powerful in solving both symmetrical and asymmetrical problems by comparing with several

famous minimax optimization EAs. Applications in robust optimal design of ILC and robust stabilization of uncertain time-delay systems further validate the effectiveness of the proposed method.

Besides the above specific conclusions, some high-level thoughts about the area of evolutionary computation are also inspired by the works in this thesis: first, the mathematical analysis of the behaviors of EAs is very critical in solidifying the theoretical basis of EAs. However, because of the involvement of large numbers of stochastic processes, most researchers in EA community found it too difficult to systematically theorize the behaviors of EAs. Similar to the studies in Chapter 3, a combination of mathematical proof and experimental verification may be an effective way for EA community to overcome this issue; second, during the design of EAs, researchers should take into account the usability and extensibility of the algorithm in addition to the optimization efficiency. Like the proposed operators in Chapter 3 and Chapter 4, the convenience in controlling the algorithm behaviors and the easy steps for implementing to different frameworks have substantially increase the practical values of the new operators; third, the EA researchers should consider comparing EAs with traditional approaches from other areas, e.g., the LMI method in Chapter 5. These comparisons will further validate the effectiveness of EAs and provide more useful insights for the whole scientific community.

6.2 Future Work

Similar to all the traditional crossover operators, the proposed multiple exponential recombination is sensitive to the selection of control parameter Cr. The

value of Cr has a significant impact on the average lengths of the exchanged segments among individuals. Development of an automatic adaptation mechanism exclusively for multiple exponential recombination would further improve the optimization efficiency of the proposed operator. The adaptation mechanism could be based on the detection of variable interrelations. By analyzing the dependency relationships among variables, a proper Cr value could be employed to achieve the desired exchanging length. Another limitation with the current research works is the lack of new benchmark problems that possess more different variable dependency configurations. Design of a benchmark suite with various variable interrelations could provide more convenience for the testing and comparison of related approaches.

In the proposed ACGDE algorithm, the utilization probability of NCG mutation and PCG mutation are both fixed as 50%. It would be interesting if a dynamic mechanism could be introduced to intelligently adapt the ratio between NCG mutation and PCG mutation. Online performance assessment of the two mutation strategies can be conducted to dynamically decide the utilization ratio. Another potential enhancement of ACGDE could be the involvement of more than two generations. The current ACGDE only uses the information from two consecutive generations, which may not be enough for a reliable estimation of correct searching directions. Extracting information from more generations is expected to increase the accuracy of predictions. Last, in Chapter 4, ACGDE is implemented in dominance-based and decomposition-based MO frameworks. It will provide more insightful results to implement ACGDE in many other types of MO frameworks, e.g., indicator-based MO frameworks.

From the empirical results in Chapter 5, different types of problems may require different choices of K_S values. In order to introduce a self-adaptive mechanism for K_S in MMDE, the properties of the minimax optimization problem need to be analyzed. One possible way is to decide whether the problem is symmetrical or asymmetrical by comparing the associated scenarios among different solutions. After that, relatively smaller K_S values can be used for symmetrical problems and larger K_S values can be employed for asymmetrical ones. As a result, the robustness of the proposed algorithm will be enhanced. In section 5.4, MMDE is successfully applied to solve two practical problems in robust control. In future works, the new algorithm will be extended to more application areas. One promising domain is the minimax optimal design in biostatistics. Most of the biostatistics researchers are solving minimax optimization problems with basic EAs [178], which may suffer from low efficiency and poor robustness. The proposed MMDE will provide the biostatistics community a more powerful solver for minimax optimal design problems.

List of publications

The publications that was published, accepted, and submitted during the course of my PhD study are listed as follows.

Journals

1. X. Qiu, J. X. Xu and K. C. Tan, "Minimax Differential Evolution for Robust Design", *IEEE Transactions on Cybernetics*, submitted.

2. X. Qiu, K. C. Tan, and J. X. Xu, "Multiple exponential recombination for differential evolution", *IEEE Transactions on Cybernetics*, accepted.

3. X. Qiu, J. X. Xu, K. C. Tan and H. A. Abbass, "Adaptive cross-generation differential evolution operators for multi-objective optimization", *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 2, pp. 232-244, April 2016.

Conferences

1. X. Qiu, W. N. Xu, J. X. Xu and K. C. Tan, "Enhancing Exploration in Differential Evolution via Exponential Recombination", *IEEE Congress on Evolutionary Computation (CEC)*, Vancouver, Canada, July 24-29, 2016, accepted.

2. X. Qiu, W. N. Xu, J. X. Xu and K. C. Tan, "A new framework for self-adapting control parameters in multi-objective optimization", *Genetic and Evolutionary Computation Conference 2015 (GECCO2015)*, Madrid, Spain, July 11-15, 2015, *GECCO '15 Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, pp. 743-750, 2015.

3. X. Qiu, Y. Huang, J. X. Xu and K. C. Tan, "A Novel Hybrid Multi-objective Optimization Framework: Rotating the Objective Space", *The Tenth International Conference on Simulated Evolution And Learning (SEAL2014)*, Dunedin, New Zealand, December 15-18, 2014, *Lecture Notes in Computer Science*, volume 8886, pp 192-203, 2014.

4. X. Qiu, Y. Huang and K. C. Tan, "A Novel Multi-objective Optimization Framework Combining NSGA-II and MOEA/D", *the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems (IES2014)*, Singapore, November 10-12, 2014, *Proceedings in Adaptation, Learning and Optimization*, Vol. 2, pp 227-237, 2014.

5. X. Qiu, J. X. Xu and K. C. Tan, “A Novel Differential Evolution (DE) Algorithm for Multi-objective Optimization”, *IEEE Congress on Evolutionary Computation (CEC)*, Beijing, China, July 6-11, 2014, pp. 2391-2396, 2014.

6. J. K. Chong and X. Qiu, “An Opposition-Based Self-adaptive Differential Evolution with Decomposition for Solving the Multiobjective Multiple Salesman Problem”, *IEEE Congress on Evolutionary Computation (CEC)*, Vancouver, Canada, July 24-29, 2016, accepted.

7. S. B. Gee, X. Qiu, and K. C. Tan, “A novel diversity maintenance scheme for evolutionary multi-objective optimization”, *The International Conference on Intelligent Data Engineering And Automated Learning 2013*, Hefei, Anhui, China, October 20-23, pp. 270-277, 2013.

8. Y. S. Liau, K. C. Tan, J. Hu, X. Qiu, S. B. Gee, “Machine learning enhanced multi-objective evolutionary algorithm based on decomposition”, *The International Conference on Intelligent Data Engineering and Automated Learning 2013*, Hefei, Anhui, China, October 20-23, pp. 553-560, 2013.

9. A. Muruganantham, Y. Zhao, S. B. Gee, X. Qiu and K. C. Tan, “Dynamic multiobjective optimization using evolutionary algorithm with Kalman filter”, *Asia Pacific Symposium on Intelligent and Evolutionary Systems*, Seoul, Korea, November 7-9, pp 66-75, 2013.

Bibliography

[1] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York: Springer, 2nd ed., 2006.

[2] A. E. Eiben and J. Smith, "From evolutionary computation to the evolution of things," *Nature*, vol. 521, pp. 476–482, May 2015.

[3] D. L. Fraga, L. Gerardo, and C. A. Coello Coello, "A review of applications of evolutionary algorithms in pattern recognition," *Pattern Recognition, Machine Intelligence and Biometrics*, pp. 3–28, 2011.

[4] O. Cordn, E. Herrera-Viedma, C. Lpez-Pujalte, M. Luque, and C. Zarco, "A review on the application of evolutionary computation to information retrieval," *International Journal of Approximate Reasoning*, vol. 34, no. 2, pp. 241 – 264, 2003.

[5] A. A. Freitas, "A review of evolutionary algorithms for data mining," *Soft Computing for Knowledge Discovery and Data Mining*, pp. 79–111, 2008.

[6] S. K. Pal, S. Bandyopadhyay, and S. S. Ray, "Evolutionary computation in bioinformatics: a review," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 36, pp. 601–615, Sept 2006.

[7] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, and C. Coello Coello, "A survey of multiobjective evolutionary algorithms for data mining: Part i," *Evolutionary Computation, IEEE Transactions on*, vol. 18, pp. 4–19, Feb 2014.

[8] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, and C. Coello, "Survey of multiobjective evolutionary algorithms for data mining: Part ii," *Evolutionary Computation, IEEE Transactions on*, vol. 18, pp. 20–35, Feb 2014.

[9] R. Storn and K. V. Price, "Differential evolution -a simple and efficient heuristic for global optimization over continuous spaces," *Global Optimization, Journal of*, vol. 11, pp. 341 –359, Dec. 1997.

[10] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *Evolutionary Computation, IEEE Transactions on*, vol. 15, pp. 4 –31, feb. 2011.

[11] S. Das, A. Abraham, U. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *Evolutionary Computation, IEEE Transactions on*, vol. 13, pp. 526–553, June 2009.

[12] H. Wang, C. Weng, and J. Yuan, "Multi-feature spectral clustering with minimax optimization," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pp. 4106–4113, June 2014.

[13] B. Chen, J. Wang, L. Wang, Y. He, and Z. Wang, "Robust optimization for transmission expansion planning: Minimax cost vs. minimax regret," *IEEE Transactions on Power Systems*, vol. 29, pp. 3069–3077, Nov 2014.

[14] C. Ho, B. K. Ling, L. Benmesbah, T. Kok, W. C. Siu, and K. L. Teo, "Two-channel linear phase FIR QMF bank minimax design via global nonconvex optimization programming," *IEEE Transactions on Signal Processing*, vol. 58, pp. 4436–4441, Aug 2010.

[15] V. Grushkovskaya and A. Zuyev, "Optimal stabilization problem with minimax cost in a critical case," *IEEE Transactions on Automatic Control*, vol. 59, pp. 2512–2517, Sept 2014.

[16] D. Agnew, "Improved minimax optimization for circuit design," *IEEE Transactions on Circuits and Systems*, vol. 28, pp. 791–803, Aug 1981.

[17] J. W. Bandler, W. Kellermann, and K. Madsen, "A superlinearly convergent minimax algorithm for microwave circuit design," *IEEE Transactions on Microwave Theory and Techniques*, vol. 33, pp. 1519–1530, Dec 1985.

[18] A. V. Sebald and J. Schlenzig, "Minimax design of neural net controllers for highly uncertain plants," *IEEE Transactions on Neural Networks*, vol. 5, pp. 73–82, Jan 1994.

[19] Y.-S. Ong, P. B. Nair, and K. Y. Lum, "Max-min surrogate-assisted evolutionary algorithm for robust design," *IEEE Transactions on Evolutionary Computation*, vol. 10, pp. 392–404, Aug 2006.

[20] M. T. Jensen, *Robust and Flexible Scheduling With Evolutionary Computation*. PhD thesis, Dept. Comp. Sci., Univ. Aarhus, Aarhus, Denmark, 2001.

[21] A. M. Cramer, S. D. Sudhoff, and E. L. Zivi, "Evolutionary algorithms for minimax problems in robust design," *IEEE Transactions on Evolutionary Computation*, vol. 13, pp. 444–453, April 2009.

[22] M.-J. Tahk and B.-C. Sun, "Coevolutionary augmented lagrangian methods for constrained optimization," *IEEE Transactions on Evolutionary Computation*, vol. 4, pp. 114–124, Jul 2000.

[23] J. Kim and M.-J. Tahk, "Co-evolutionary computation for constrained min-max problems and its applications for pursuit-evasion games," in *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, vol. 2, pp. 1205–1212, 2001.

[24] Y. Shi and R. A. Krohling, "Co-evolutionary particle swarm optimization to solve min-max problems," in *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, vol. 2, pp. 1682–1687, 2002.

[25] R. A. Krohling, F. Hoffmann, and L. S. Coelho, "Co-evolutionary particle swarm optimization for min-max problems using gaussian distribution," in *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 1, pp. 959–964, June 2004.

[26] H. J. C. Barbosa, "A coevolutionary genetic algorithm for constrained optimization," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 3, pp. 1605–1611, 1999.

[27] J. W. Herrmann, "A genetic algorithm for minimax optimization problems," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 2, p. 1103 Vol. 2, 1999.

[28] R. I. Lung and D. Dumitrescu, "A new evolutionary approach to minimax problems," in *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pp. 1902–1905, June 2011.

[29] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *Evolutionary Computation, IEEE Transactions on*, vol. 10, pp. 646–657, Dec 2006.

[30] A. K. Qin, V. L. Huang, and P. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 13, pp. 398–417, April 2009.

[31] J. Zhang and A. Sanderson, "Jade: Adaptive differential evolution with optional external archive," *Evolutionary Computation, IEEE Transactions on*, vol. 13, pp. 945–958, Oct 2009.

[32] Y.-L. Li, Z.-H. Zhan, Y.-J. Gong, W.-N. Chen, J. Zhang, and Y. Li, "Differential evolution with an evolution path: A deep evolutionary algorithm," *Cybernetics, IEEE Transactions on*, vol. 45, pp. 1798–1810, Sept 2015.

[33] U. Halder, S. Das, and D. Maity, "A cluster-based differential evolution algorithm with external archive for optimization in dynamic environments," *Cybernetics, IEEE Transactions on*, vol. 43, pp. 881–897, June 2013.

[34] W.-F. Gao, G. Yen, and S.-Y. Liu, "A dual-population differential evolution with coevolution for constrained optimization," *Cybernetics, IEEE Transactions on*, vol. 45, pp. 1094–1107, May 2015.

[35] S. Das, A. Mandal, and R. Mukherjee, "An adaptive differential evolution algorithm for global optimization in dynamic environments," *Cybernetics, IEEE Transactions on*, vol. 44, pp. 966–978, June 2014.

[36] M. Yang, C. Li, Z. Cai, and J. Guan, "Differential evolution with auto-enhanced population diversity," *Cybernetics, IEEE Transactions on*, vol. 45, pp. 302–315, Feb 2015.

[37] M. Epitropakis, D. Tasoulis, N. Pavlidis, V. Plagianakos, and M. Vrahatis, "Enhancing differential evolution utilizing proximity-based mutation operators," *Evolutionary Computation, IEEE Transactions on*, vol. 15, pp. 99–119, Feb 2011.

[38] S. Islam, S. Das, S. Ghosh, S. Roy, and P. Suganthan, "An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, pp. 482–500, April 2012.

[39] H. Wang, S. Rahnamayan, H. Sun, and M. Omran, "Gaussian bare-bones differential evolution," *Cybernetics, IEEE Transactions on*, vol. 43, pp. 634–647, April 2013.

[40] Q. Fan and X. Yan, "Self-adaptive differential evolution algorithm with zoning evolution of control parameters and adaptive mutation strategies," *Cybernetics, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.

[41] W. Gong and Z. Cai, "Differential evolution with ranking-based mutation operators," *Cybernetics, IEEE Transactions on*, vol. 43, pp. 2066–2081, Dec 2013.

[42] Y. Cai and J. Wang, "Differential evolution with neighborhood and direction information for numerical optimization," *Cybernetics, IEEE Transactions on*, vol. 43, pp. 2202–2215, Dec 2013.

[43] J. Wang, J. Liao, Y. Zhou, and Y. Cai, "Differential evolution enhanced with multiobjective sorting-based mutation operators," *Cybernetics, IEEE Transactions on*, vol. 44, pp. 2792–2805, Dec 2014.

[44] L. Tang, Y. Dong, and J. Liu, "Differential evolution with an individual-dependent mechanism," *Evolutionary Computation, IEEE Transactions on*, vol. 19, pp. 560–574, Aug 2015.

[45] W. Gong, Z. Cai, and D. Liang, "Adaptive ranking mutation operator based differential evolution for constrained optimization," *Cybernetics, IEEE Transactions on*, vol. 45, pp. 716–727, April 2015.

[46] S. Biswas, S. Kundu, and S. Das, "An improved parent-centric mutation with normalized neighborhoods for inducing niching behavior in differential evolution," *Cybernetics, IEEE Transactions on*, vol. 44, pp. 1726–1737, Oct 2014.

[47] Y.-J. Gong, Q. Zhou, Y. Lin, and J. Zhang, "Orthogonal predictive differential evolution," in *Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems*, vol. 1, pp. 141–154, 2015.

[48] W.-J. Yu, M. Shen, W.-N. Chen, Z.-H. Zhan, Y.-J. Gong, Y. Lin, O. Liu, and J. Zhang, "Differential evolution with two-level parameter adaptation," *Cybernetics, IEEE Transactions on*, vol. 44, pp. 1080–1099, July 2014.

[49] W. Gao, G. Yen, and S. Liu, "A cluster-based differential evolution with self-adaptive strategy for multimodal optimization," *Cybernetics, IEEE Transactions on*, vol. 44, pp. 1314–1327, Aug 2014.

[50] R. Storn and K. V. Price, "Differential evolutiona simple and efficient adaptive scheme for global optimization over continuous spaces," International Computer Science Institute, Berkeley, 1995, TR-95-012.

[51] D. Zaharie, "Influence of crossover on the behavior of differential evolution algorithms," *Applied Soft Computing*, vol. 9, pp. 1126–1138, Jun. 2009.

[52] C. Lin, A. Qing, and Q. Feng, "A comparative study of crossover in differential evolution," *Journal of Heuristics*, vol. 17, pp. 675 –703, Dec. 2011.

[53] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, "A comparative study of differential evolution variants for global optimization," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, pp. 485–492, 2006.

[54] J. Tvrdik, "Adaptive differential evolution and exponential crossover," in *Computer Science and Information Technology, 2008. IMCSIT 2008. International Multiconference on*, pp. 927–931, Oct 2008.

[55] G. Jeyakumar and C. Shanmugavelayutham, "Convergence analysis of differential evolution variants on unconstrained global optimization functions," *International Journal of Artificial Intelligence & Applications*, vol. 2, pp. 116–127, April 2011.

[56] G. Jeyakumar and C. Shunmuga Velayutham, "A comparative performance analysis of differential evolution and dynamic differential evolution variants," in *Nature Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pp. 463–468, Dec 2009.

[57] D. Zaharie, "A comparative analysis of crossover variants in differential evolution," in *Proceedings of the International Multiconference on Computer Science and Information Technology*, pp. 171–181, 2007.

[58] S.-M. Guo and C.-C. Yang, "Enhancing differential evolution utilizing eigenvector-based crossover operator," *Evolutionary Computation, IEEE Transactions on*, vol. 19, pp. 31–49, Feb 2015.

[59] S. Zhao and P. N. Suganthan, "Empirical investigations into the exponential crossover of differential evolutions," *Swarm and Evolutionary Computation*, vol. 9, pp. 27 –36, Apr. 2013.

[60] C. S. Chang, D. Y. Xu, and H. B. Quek, "Pareto-optimal set based multiobjective tuning of fuzzy automatic train operation for mass transit system," *IEEE Proceeding on Electric Power Applications*, vol. 146(5), pp. 577–583, 1999.

[61] B. V. Babu and M. M. L. Jehan, "Differential evolution for multi-objective optimization," *Proceedings of the 2003 Congress on Evolutionary Computation (CEC2003)*, vol. 4, pp. 2696–2703, 2003.

[62] H. Li and Q. Zhang, "A multiobjective differential evolution based on decomposition for multiobjective optimization with variable linkages," *Parallel Problem Solving from Nature - PPSN IX, 9th International Conference*, vol. 4193, pp. 583–592, 2006.

[63] J. Lampinen, "DE's selection rule for multiobjective optimization," *Technical report, Lappeenranta University of Technology, Department of Information Technology*, 2001.

[64] S. Kukkonen and J. Lampinen, "A differential evolution algorithm for constrained multi-objective optimization: Initial assessment," *IASTED International Conference on Artificial Intelligence and Applications (AIA 2004)*, pp. 96–102, 2004.

[65] S. Kukkonen and J. Lampinen, "An extension of generalized differential evolution for multi-objective optimization with constraints," *Parallel Problem Solving from Nature - PPSN VIII*, vol. 3242, pp. 752–761, 2004.

[66] S. Kukkonen and J. Lampinen, "GDE3: The third evolution step of generalized differential evolution," *IEEE Congress on Evolutionary Computation (CEC'2005)*, vol. 1, pp. 443–450, 2005.

[67] H. A. Abbass, R. Sarker, and C. Newton, "PDE:a pareto-frontier differential evolution approach for multi-objective optimization problems," *Proceedings of the Congress on Evolutionary Computation 2001*, vol. 2, pp. 971–978, 2001.

[68] H. A. Abbass, "The self-adaptive pareto differential evolution algorithm," *Congress on Evolutionary Computation 2002*, vol. 1, pp. 831–836, 2002.

[69] H. A. Abbass, "A memetic pareto evolutionary approach to artificial neural networks," *The Australian Joint Conference on Artificial Intelligence*, vol. 2256, pp. 1–12, 2001.

[70] L. V. Santana-Quintero and C. A. C. Coello, "An algorithm based on differential evolution for multi-objective problems," *International Journal of Computational Intelligence Research*, vol. 1(2), pp. 151–169, 2005.

[71] N. K. Madavan, "Multiobjective optimization using apareto differential evolution approach," *Congress on Evolutionary Computation (CEC'2002)*, vol. 2, pp. 1145–1150, 2002.

[72] A. W. Iorio and X. Li, "Solving rotated multi-objective optimization problems using differential evolution.," *Advances in Artificial Intelligence, Proceedings*, vol. 3339, pp. 861–872, 2004.

[73] T. Robic and B. Filipic, "DEMO: Differential evolution for multiobjective optimization," *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO*, vol. 3410, pp. 520–533, 2005.

[74] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II," *Evolutionary Computation, IEEE Transactions on*, vol. 13, no. 2, pp. 284–302, 2009.

[75] L. T. Bui, J. Liu, A. Bender, M. Barlow, S. Wesolkowski, and H. A. Abbass, "DMEA: a direction-based multiobjective evolutionary algorithm," *Memetic Comp.*, vol. 3, pp. 271–285, 2011.

[76] H. Fan and J. Lampinen, "A trigonometric mutation operation to differential evolution," *J. Global Optim.*, vol. 17, no. 1, pp. 105–129, 2003.

[77] H. Y. Fan and J. Lampinen, "A directed mutation operation for the differential evolution algorithm," *Int. J. Ind. Eng.*, vol. 10, no. 1, pp. 6–15, 2003.

[78] V. Feoktistov and S. Janaqi, "Generalization of the strategies in differential evolution," *in Proc. Parallel Distrib. Process. Symp.*, pp. 165–170, 2004.

[79] V. Feoktistov, "Differential evolution: In search of solutions," *Secaucus, NJ, USA: Springer-Verlag*, 2006.

[80] Y. X.Wang and Q. L. Xiang, "Exploring new learning strategies in differential evolution algorithm," *in Proc. IEEE Congr. Evol. Comput.*, pp. 204–209, 2008.

[81] P. Kaelo and M. Ali, "Differential evolution algorithms using hybrid mutation," *Comput. Optim. Appl.*, vol. 37, no. 2, pp. 231–246, 2007.

[82] X. J. Bi and J. Xiao, "Classification-based self-adaptive differential evolution with fast and reliable convergence performance," *Soft Comput.*, vol. 15, no. 8, pp. 1581–1599, 2011.

[83] J. Zhang and A. C. Sanderson, "Self-adaptive multi-objective differential evolution with direction information provided by archived inferior solutions," *in Proc. IEEE Congr. Evol. Comput.*, pp. 2801–2810, 2008.

[84] A. Iorio and X. Li, "Incorporating directional information within a differential evolution algorithm for multi-objective optimization," *in Proc. 8th Annu. Conf. Genetic Evol. Comput.*, pp. 691–698, 2006.

[85] J. Liu, Z. Fan, and E. Goodman, "SRDE: An improved differential evolution based on stochastic ranking," *in Proc. 1st ACM/SIGEVO*, pp. 345–352, 2009.

[86] K. Li, S. Kwong, R. Wang, J. Cao, and I. Rudas, "Multi-objective differential evolution with self-navigation," in *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, pp. 508–513, 2012.

[87] L. Nguyen, L. Bui, and H. Abbass, "Dmea-ii: the direction-based multi-objective evolutionary algorithm-ii," *Soft Computing*, vol. 18, no. 11, pp. 2119–2134, 2014.

[88] L. Nguyen, L. T. Bui, and H. Abbass, "A new niching method for the direction-based multi-objective evolutionary algorithm," in *Computational Intelligence in Multi-Criteria Decision-Making (MCDM), 2013 IEEE Symposium on*, pp. 1–8, April 2013.

[89] L. T. Bui, H. A. Abbass, and D. Essam, "Local modelsan approach to distributed multi-objective optimization," *Comput Optim Appl*, vol. 42, pp. 105–139, 2009.

[90] L. T. Bui, K. Deb, H. A. Abbass, and D. Essam, "Interleaving guidance in evolutionary multi-objective optimization," *Computer Science and Technology*, vol. 23, no. 1, pp. 44–63, 2008.

[91] H. Li and D. Landa-Silva, "Evolutionary multi-objective simulated annealing with adaptive and competitive search direction," in *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, pp. 3311–3318, June 2008.

[92] C. Goh, Y. Ong, K. Tan, and E. Teoh, "An investigation on evolutionary gradient search for multi-objective optimization," in *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, pp. 3741–3746, 2008.

[93] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," in *TENCON '02. Proceedings. 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering*, vol. 1, pp. 606–611, Oct 2002.

[94] F. Caraffini, F. Neri, J. Cheng, G. Zhang, L. Picinali, G. Iacca, and E. Mininno, "Super-fit multicriteria adaptive differential evolution," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pp. 1678–1685, June 2013.

[95] O. Soliman and L. T. Bui, "A self-adaptive strategy for controlling parameters in differential evolution," in *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, pp. 2837–2842, June 2008.

[96] W. Jin, L. Gao, Y. Ge, and Y. Zhang, "An improved self-adapting differential evolution algorithm," in *Computer Design and Applications (ICCDA), 2010 International Conference on*, vol. 3, pp. 341–344, June 2010.

[97] E. Mezura-Montes and A. Palomeque-Ortiz, "Parameter control in differential evolution for constrained optimization," in *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, pp. 1375–1382, May 2009.

[98] D. Zaharie, "Control of population diversity and adaptation in differential evolution algorithms," *in Proc. 9th Int. Conf. MENDEL*, pp. 41–46, 2003.

[99] D. Zaharie and D. Petcu, "Adaptive pareto differential evolution and its parallelization," *in Proc. 5th Int. Conf. Parallel Process. Appl. Math.*, pp. 261–268, 2003.

[100] A. Zamuda, J. Brest, B. Boskovic, and V. Zumer, "Differential evolution for multiobjective optimization with self adaptation," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pp. 3617–3624, Sept 2007.

[101] T. Tusar and B. Filipic, "Differential evolution versus genetic algorithms in multiobjective optimization," *in Proceedings of the Fourth International Conference on Evolutionary Multi-Criterion Optimization-EMO2007,ser.LectureNotesinComputerScience, Springer*, vol. 4403, pp. 257–271, 2007.

[102] V. L. Huang, A. K. Qin, P. Suganthan, and M. Tasgetiren, "Multi-objective optimization based on self-adaptive differential evolution algorithm," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pp. 3601–3608, Sept 2007.

[103] V. L. Huang, S.-Z. Zhao, R. Mallipeddi, and P. Suganthan, "Multi-objective optimization using self-adaptive differential evolution algorithm," in *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, pp. 190–194, May 2009.

[104] M. T. Jensen, *Metaheuristics: Computer Decision-Making*, ch. A New Look at Solving Minimax Problems with Coevolutionary Genetic Algorithms, pp. 369–384. Boston, MA: Springer US, 2004.

[105] J. Hur, H. Lee, and M.-J. Tahk, "Parameter robust control design using bimatrix co-evolution algorithms," *Engineering Optimization*, vol. 35, no. 4, pp. 417–426, 2003.

[106] J. Branke and J. Rosenbusch, "New approaches to coevolutionary worst-case optimization," in *Parallel Problem Solving from Nature - PPSN X*, pp. 144–153, September 2008.

[107] E. C. Laskari, K. E. Parsopoulos, and M. N. Vrahatis, "Particle swarm optimization for minimax problems," in *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, vol. 2, pp. 1576–1581, 2002.

[108] A. Zhou and Q. Zhang, "A surrogate-assisted evolutionary algorithm for minimax optimization," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pp. 1–7, July 2010.

[109] R. Sarker, S. Elsayed, and T. Ray, "Differential evolution with dynamic parameters selection for optimization problems," *Evolutionary Computation, IEEE Transactions on*, vol. 18, pp. 689–707, Oct 2014.

[110] L. Tang, Y. Zhao, and J. Liu, "An improved differential evolution algorithm for practical dynamic scheduling in steelmaking-continuous casting production," *Evolutionary Computation, IEEE Transactions on*, vol. 18, pp. 209–225, April 2014.

[111] S. Biswas, S. Kundu, and S. Das, "Inducing niching behavior in differential evolution through local information sharing," *Evolutionary Computation, IEEE Transactions on*, vol. 19, pp. 246–263, April 2015.

[112] S. Bandyopadhyay and A. Mukherjee, "An algorithm for many-objective optimization with reduced objective computations: A study in differential evolution," *Evolutionary Computation, IEEE Transactions on*, vol. 19, pp. 400–413, June 2015.

[113] D. R. Newman, "The use of linkage learning in genetic algorithms," September 2006.

[114] Y. Chen, T. Yu, K. Sastry, and D. Goldberg, "A survey of linkage learning techniques in genetic and evolutionary algorithms," Illinois Genetic Algorithms Library, Tech. Rep., 2007.

[115] K. De Jong and W. Spears, "A formal analysis of the role of multi-point crossover in genetic algorithms," *Annals of Mathematics and Artificial Intelligence*, vol. 5, no. 1, pp. 1–26, 1992.

[116] A. A. Markov, "Extension of the limit theorems of probability theory to a sum of variables connected in a chain," *Dynamic Probabilistic Systems, volume 1: Markov Chains*, vol. 1, 1971.

[117] P. Suganthan, N. Hansen, J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC2005 special session on real-parameter optimization," *Nanyang Technological University*, May 2005.

[118] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics*, vol. 1, pp. 80 –83, Dec. 1945.

[119] R. Joshi and A. C. Sanderson, "Minimal representation multisensor fusion using differential evolution," *IEEE Trans. Syst., Man Cybern. Part A*, vol. 29, no. 1, pp. 63–76, 1999.

[120] R. Storn, "System design by constraint adaptation and differential evolution," *IEEE Trans. Evol. Comput.*, vol. 3, pp. 22–34, Apr 1999.

[121] A. Qing, "Dynamic differential evolution strategy and applications in electromagnetic inverse scattering problems," *IEEE Trans. Geosci. Remote Sensing*, vol. 44, pp. 116–125, Jan 2006.

[122] D. Datta and S. Dutta, "A binary-real-coded differential evolution for unit commitment problem," *International Journal of Electrical Power and Energy Systems*, vol. 42, no. 1, pp. 517 – 524, 2012.

[123] D. Datta and J. R. Figueira, "A real-integer-discrete-coded differential evolution," *Applied Soft Computing*, vol. 13, no. 9, pp. 3884 – 3893, 2013.

[124] K. Deb *Multi-Objective Optimization Using Evolutionary Algorithms*, Wiley, 2001.

[125] C. Coello, G. Lamont, and D. V. Veldhuizen *Evolutionary algorithms for solving multi-objective problems*, vol. 5, Springer, 2007.

[126] K. Miettinen, *Nonlinear Multiobjective Optimization*. Springer, 2008.

[127] E. Mezura-Montes, M. Reyes-Sierra, and C. A. C. Coello, "Multi-objective optimization using differential evolution: A survey of the state-of-the-art," *Advances in Differential Evolution , Studies in Computational Intelligence*, vol. 143, pp. 173–196, 2008.

[128] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm Evol. Comput.*, vol. 1, pp. 32–49, Mar. 2011.

[129] S. B. Gee, K. C. Tan, V. A. Shim, and N. R. Pal, "On-line diversity assessment in evolutionary multiobjective optimization: A geometrical perspective," Submitted to IEEE Transactions on Evolutionary Computation.

[130] P. Bosman and D. Thierens, "The balance between proximity and diversity in multiobjective evolutionary algorithms," *Evolutionary Computation, IEEE Transactions on*, vol. 7, pp. 174–188, April 2003.

[131] B. Xue, W. Fu, and M. Zhang, "Differential evolution (DE) for multi-objective feature selection in classification," in *Proceedings of the 2014 Conference Companion on Genetic and Evolutionary Computation Companion*, pp. 83–84, 2014.

[132] D. Datta and J. R. Figueira, "Graph partitioning by multi-objective real-valued metaheuristics: A comparative study," *Applied Soft Computing*, vol. 11, no. 5, pp. 3976 – 3987, 2011.

[133] B. Xue, W. Fu, and M. Zhang, "Multi-objective feature selection in classification: A differential evolution approach," in *Simulated Evolution and Learning*, vol. 8886, pp. 516–528, 2014.

[134] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182–197, 2002.

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

[136] O. Grodzevich and O. Romanko, "Normalization and other topics in multi-objective optimization," in *Proceedings of the FieldsMITACS Industrial Problems Workshop*, pp. 89–101, 2006.

[137] J. Branke, K. Deb, K. Miettinen, and R. Slowinski, "Multiobjective optimization: Interactive and evolutionary approaches," Springer, 2008.

[138] R. Marler and J. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and Multidisciplinary Optimization*, vol. 26, no. 6, pp. 369–395, 2004.

[139] I. Kim and O. de Weck, "Adaptive weighted sum method for multiobjective optimization: a new method for pareto front generation," *Structural and Multidisciplinary Optimization*, vol. 31, no. 2, pp. 105–116, 2006.

[140] S. Kukkonen and J. Lampinen, "Ranking-dominance and many-objective optimization," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pp. 3983–3990, Sept 2007.

[141] K. V. Price, "New ideas in optimization," ch. An Introduction to Differential Evolution, pp. 79–108, Maidenhead, UK, England: McGraw-Hill Ltd., UK, 1999.

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

[143] C. Goh, K. Tan, D. Liu, and S. Chiam, "A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design," *European Journal of Operational Research*, vol. 202, no. 1, pp. 42–54, 2010.

[144] E. Ziztler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization," *Evolutionary Methods for Design, Optimization, and Control*, pp. 95–100, 2002.

[145] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, and S. Tiwari, "Multiobjective optimization test instances for the CEC 2009 special session and competition," *Technical Report*, 2008.

[146] L. Bradstreet, L. Barone, L. While, S. Huband, and P. Hingston, "Use of the WFG toolkit and PISA for comparison of MOEAs," in *Computational Intelligence in Multicriteria Decision Making, IEEE Symposium on*, pp. 382–389, 2007.

[147] C. A. C. Coello and N. C. Cortes, "Solving multiobjective optimization problems using an artificial immune system," *Genetic Programming and Evolvable Machines*, vol. 6, no. 2, pp. 163–190, 2005.

[148] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *Evolutionary Computation, IEEE Transactions on*, vol. 3, pp. 257–271, Nov 1999.

[149] D. Datta and J. R. Figueira, "Some convergence-based M-ary cardinal metrics for comparing performances of multi-objective optimizers,"

Computers and Operations Research, vol. 39, no. 7, pp. 1754 – 1762, 2012.

[150] J. Bader and E. Zitzler, "Hype: An algorithm for fast hypervolume-based many-objective optimization," *Evolutionary Computation*, vol. 19, pp. 45–76, March 2011.

[151] B. Li, Y.-S. Ong, M. N. Le, and C. K. Goh, "Memetic gradient search," in *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, pp. 2894–2901, June 2008.

[152] A. Lara, C. A. Coello Coello, and O. Schuetze, "Using gradient information for multi-objective problems in the evolutionary context," in *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '10, (New York, NY, USA), pp. 2011–2014, ACM, 2010.

[153] C.-M. Hong, C.-M. Chen, and H.-K. Fan, "A new gradient-based search method: Grey-gradient search method," in *Multiple Approaches to Intelligent Systems* (I. Imam, Y. Kodratoff, A. El-Dessouki, and M. Ali, eds.), vol. 1611 of *Lecture Notes in Computer Science*, pp. 185–194, Springer Berlin Heidelberg, 1999.

[154] D. Zaharie, "On the explorative power of differential evolution," in *Proc. 3rd Int. Workshop Symbol. Numer. Algorithms Sci. Comput.*, Timisoara, Romania, Oct. 2-4 2001.

[155] S. Dasgupta, S. Das, A. Biswas, and A. Abraham, "On stability and convergence of the population-dynamics in differential evolution," *AI Commun.*, vol. 22, no. 1, pp. 1–20, 2009.

[156] P. Kouvelis and G. Yu, "Robust discrete optimization and its applications," in *Nonconvex Optimization and Its Applications*, vol. 14, 1997.

[157] R. Montemanni, L. Gambardella, and A. Donati, "A branch and bound algorithm for the robust shortest path problem with interval data," *Operations Research Letters*, vol. 32, no. 3, pp. 225 – 232, 2004.

[158] R. Montemanni and L. Gambardella, "A branch and bound algorithm for the robust spanning tree problem with interval data," *European Journal of Operational Research*, vol. 161, no. 3, pp. 771 – 779, 2005.

[159] M. Inuiguchi and M. Sakawa, "Minimax regret solution to linear programming problems with an interval objective function," *European Journal of Operational Research*, vol. 86, no. 3, pp. 526 – 536, 1995.

[160] H. E. Mausser and M. Laguna, "A new mixed integer formulation for the maximum regret problem," *International Transactions in Operational Research*, vol. 5, no. 5, pp. 389 – 403, 1998.

[161] G. Yu, "Min-max optimization of several classical discrete optimization problems," *Journal of Optimization Theory and Applications*, vol. 98, no. 1, pp. 221–242.

[162] B. Lu, Y. Cao, M. j. Yuan, and J. Zhou, "Reference variable methods of solving min–max optimization problems," *Journal of Global Optimization*, vol. 42, no. 1, pp. 1–21, 2007.

[163] M. . Sainz, P. Herrero, J. Armengol, and J. Veh, "Continuous minimax optimization using modal intervals," *Journal of Mathematical Analysis and Applications*, vol. 339, no. 1, pp. 18 – 30, 2008.

[164] Y. Feng, L. Hongwei, Z. Shuisheng, and L. Sanyang, "A smoothing trust-region Newton-CG method for minimax problem," *Applied Mathematics and Computation*, vol. 199, no. 2, pp. 581 – 589, 2008.

[165] H. Aissi, C. Bazgan, and D. Vanderpooten, "Minmax and minmax regret versions of combinatorial optimization problems: A survey," *European Journal of Operational Research*, vol. 197, no. 2, pp. 427 – 438, 2009.

[166] C. Charalambous and A. R. Conn, "An efficient method to solve the minimax problem directly," *SIAM Journal on Numerical Analysis*, vol. 15, pp. 162–187, Feb 1978.

[167] R. A. Polyak, "Smooth optimization methods for minimax problems," *SIAM Journal on Control and Optimization*, vol. 26, pp. 1274–1286, Nov 1988.

[168] X. Qiu, J. X. Xu, K. C. Tan, and H. A. Abbass, "Adaptive cross-generation differential evolution operators for multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, pp. 232–244, April 2016.

[169] J. Williams, "Algorithm 232: Heapsort," *Communications of the ACM*, vol. 7, no. 6, pp. 347–348, 1964.

[170] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *J. ACM*, vol. 34, pp. 596–615, July 1987.

[171] H. J. C. Barbosa, "A genetic algorithm for min-max problems," in *Proc. 1st Int. Conf. Evol. Comput. and Applicat.*, pp. 99–109, 1996.

[172] J. Derrac, S. Garca, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, pp. 3 –18, Mar. 2011.

[173] S. Arimoto, S. Kawamura, and F. Miyazak, "Bettering operation of robust by learning," *Journal of Robotic Systems*, vol. 1, pp. 123–140, 1985.

[174] Y. Y. Kuc, S. Lee, and K. Nam, "An iterative learning control theory for a class of nonlinear dynamic systems," *Automatica*, vol. 28, no. 10, pp. 1215–1221, 1992.

[175] T. J. Jang, C. H. Choi, and H. S. Ahn, "Iterative learning control in feedback system," *Automatica*, vol. 31, pp. 243–248, 1995.

[176] S. I. Niculescu, J. M. Dion, and L. Dugard, "Robust stabilization for uncertain time-delay systems containing saturating actuators," *IEEE Transactions on Automatic Control*, vol. 41, pp. 742–747, May 1996.

[177] H. Y. Su, J. B. Hu, J. Lam, and J. Chu, "Robust stabilization of uncertain time-delay systems containing nonlinear saturating actuators," *Journal of Zhejiang University SCIENCE A*, vol. 1, no. 3, pp. 241–248, 2000.

[178] R.-B. Chen, S.-P. Chang, W. Wang, H.-C. Tung, and W. K. Wong, "Minimax optimal designs via particle swarm optimization methods," *Statistics and Computing*, vol. 25, no. 5, pp. 975–988, 2015.