Self adaptive differential evolution

Huang, Ling

2010

Huang, L. (2010). Self adaptive differential evolution. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/47016>

<https://doi.org/10.32657/10356/47016>

Nanyang Technological University

Downloaded on 26 Apr 2023 21:36:08 SGT

SELF ADAPTIVE DIFFERENTIAL EVOLUTION

HUANG LING

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University in partial
fulfillment of the requirements for the degree of
Doctor of Philosophy

2010

Acknowledgement

First of all, I would like to express my sincere and utmost gratitude to my supervisor, Associate Professor Ponnuthurai Nagaratnam Suganthan, for his invaluable support, advice and encouragement throughout my Ph.D. studies at Nanyang Technological University (NTU).

I would also like to thank my colleagues at the Instrumentation and System Engineering lab for their valuable input, discussions, and most importantly, friendship. I am also very grateful to NTU for providing me with such a generous research scholarship to support my Ph.D. studies.

Last but not the least, I would like to thank my parents and friends in China for their warmth, understanding and most importantly, love.

Summary

This thesis investigates a relatively new technique in the global optimization field, namely Differential Evolution (DE). A thorough literature survey is provided on the DE algorithm. This survey starts from DE's origin, goes through the basic mutation, crossover and selection operations, as well as the many variants that aim to improve the performance of the original DE, before ending with DE's real-life applications.

A Self-adaptive DE algorithm (SaDE) is developed to avoid the time-consuming exhaustive search for the most suitable trial vector generation strategy and the associated control parameters scaling factor (F) and Crossover Rate (CR). In SaDE, the trial vector generation strategies were probabilistically assigned to each target vector in the current population according to probabilities obtained from previous attempts to generate improved solutions. Experiments on the self-adaptive characteristics of the CR value and trial vector generation strategies show that the SaDE algorithm could evolve suitable strategies and parameter values as evolution progresses. The performance of the SaDE is evaluated over a suite of 26 bound constrained numerical optimization problems. The comprehensive experimental results verified the promising performance of the SaDE to handle problems with distinct properties such as unimodality and multi-modality. Furthermore, SaDE is more effective and robust when compared with the conventional DE and three adaptive state-of-the-art DE variants.

Besides optimizing the bound constrained problems, SaDE is generalized to solve complex constrained problems by introducing constraint handling methods. This allows SaDE to be easily applied to many real world optimization problems. The performance of SaDE is evaluated on a suite of constrained benchmarks.

Inspired by the excellent performance of SaDE on single-objective (bound constrained and constrained) problems, we extended the SaDE to solve numerical optimization problems with multiple conflicting objectives, resulting in Multi-Objective SaDE (MOSaDE) and Objective-Wise Multi-Objective SaDE (OW-MOSaDE). MOSaDE adapts the strategy and parameters according to non-domination, while OW-MOSaDE

adapts according to individual objective as well as non-domination. Experimental results demonstrate that both MOSaDE and OW-MOSaDE solve multi-objective problems well. However, OW-MOSaDE improves upon the search ability of the original MOSaDE by using the new objective-wise learning strategy in addition to the non-domination-wise learning strategy.

Table of Contents

Acknowledgement	I
Summary	II
Table of Contents	IV
List of Figures	VI
List of Tables.....	VII
Chapter 1 Introduction	1
1.1 Motivation	1
1.1.1 Evolutionary Computation	1
1.1.2 Why DE?	2
1.1.3 Limitations of DE	3
1.2 Objectives	4
1.3 Major Contributions of the Thesis.....	5
1.4 Organization of the Thesis	6
Chapter 2 Background and Literature Study	7
2.1 Global Optimization	7
2.2 Evolutionary Algorithm	7
2.3 The Origin of Differential Evolution.....	9
2.4 The Basics of Differential Evolution.....	9
2.4.1 Mutation operation	10
2.4.2 Crossover operation.....	11
2.4.3 Selection operation	13
2.5 The Development of DE.....	17
2.5.1 Control parameters	17
2.5.2 Trial vector generation strategy.....	19
2.5.3 Hybridization.....	21
2.5.4 Multi-modal optimization.....	21
2.5.5 Comparative Study	22
2.6 The Applications of DE.....	23
Chapter 3 Differential Evolution Algorithm with Strategy and Parameter Adaptation for Global Numerical Optimization	25
3.1 Introduction	25
3.2 Differential Evolution Algorithm with Strategy Adaptation.....	26
3.2.1 Trial Vector Generation Strategy Adaptation	27
3.2.2 Parameter Adaptation.....	32
3.3 Performance Evaluation of SaDE on Bound Constrained Optimization Problems	36
3.3.1 Introduction to the Test Functions.....	36
3.3.2 Algorithms for Comparison.....	41
3.3.3 Experimental Results and Discussions.....	42
3.4 Analysis of self-adaptation property of SaDE.....	68

3.4.1 Self-adaptation of crossover probability	68
3.4.2 Self-adaptation of trial vector generation strategy	70
3.5 Learning period parameter study	72
3.6 Conclusion	79
Chapter 4 SaDE for Constrained Optimization Problems	81
4.1 Introduction.....	81
4.2 Constrained Optimization	82
4.3 SaDE for Constrained Optimization	83
4.4 Experiments	86
4.4.1 Test Problems	86
4.4.2 Performance Evaluation Criteria.....	87
4.4.3 Experimental Results	87
4.5 Conclusion	95
Chapter 5 Multi-objective Optimization based on Self-adaptive Differential Evolution Algorithm	96
5.1 Introduction.....	96
5.2 Multi-objective Optimization	97
5.3 Multi-objective Self-adaptive DE	101
5.4 Multi-objective Self-adaptive DE with objective-wise learning strategies (OW- MOSaDE)	107
5.5 Experimental Results	111
5.5.1 Experimental results of MOSaDE	111
5.5.2 Experimental results of OW-MOSaDE	136
5.6 Conclusion	144
Chapter 6 Conclusion and Future Work.....	145
6.1 Conclusion	145
6.2 Recommendations of future research	147
List Of Publications	149
APPENDIX.....	151
BIBLIOGRAPHY	171

List of Figures

Figure 1-1: Number of publications per year in ISI web of knowledge database	3
Figure 2-1 Coordinate rotation changes the location of trial vectors generated by the crossover operator defined in Equation (2-7).	12
Figure 2-2: Illustration of DE/rand/1/bin in 2-dimensional space	17
Figure 3-1: Success Memory and Failure Memory	31
Figure 3-2: Progress of Success Memory	32
Figure 3-3: The median convergence characteristics of rand/1 CR=0.1, rand/1 CR=0.9,	54
Figure 3-4 The median convergence characteristics of SaDE, ADE (Zaharie), SDE and jDE on 10-D test functions f_1 - f_{14}	65
Figure 3-5: Empirical distribution of normalized success performance on all 26 test problems	68
Figure 3-6: Self-adaptation characteristics of CRm on Rosenbrock and Rastrigin function (10-D)	70
Figure 3-7: Self-adaptation characteristics of strategy (10-D)	72
Figure 3-8: SaDE's results with different learning period LP on 11 test functions (10-D)	79
Figure 4-1: Convergence Graph for Function 1-6	91
Figure 4-2: Convergence Graph for Function 7-12	92
Figure 4-3: Convergence Graph for Function 13-18	93
Figure 4-4: Convergence Graph for Function 19-24	94
Figure 5-1: Globally and locally Pareto-optimal set	99
Figure 5-2: An example of non-dominated target vector and trial vector	102
Figure 5-3: Pareto optimal front and 0%, 50%, 100% attainment surfaces after $5e+5$ FES	132
Figure 5-4: 50% attainment surfaces after $5e+5$ FES on test function s 8-13(M=3).	135
Figure 5-5: Pareto front after $5e+5$ on test functions 12 and 13 (M=5)	136
Figure 5-6: The plot of the final approximation set with the smallest IGD value in the objective space for each test instances UF1-UF13	143
Figure 6-1 Pareto Front of an example of unconstrained portfolio optimization	148
Figure 6-2 Pareto Front of an example of constrained portfolio optimization	148
Figure B-1 Empirical Distribution of Normalized Success Performance	168

List of Tables

Table 2-1: The algorithmic description of DE	13
Table 3-1: The algorithmic description of SaDE	34
Table 3-2: Global optimum, search ranges and initialization ranges of the test functions	40
Table 3-3: Results of rand/1(F=0.9, CR=0.1), rand/1(F=0.9, CR=0.9), rand/1(F=0.5,CR=0.3),.....	44
Table 3-4: Results of rand/1(F=0.9, CR=0.1), rand/1(F=0.9, CR=0.9), rand/1(F=0.5,CR=0.3),.....	46
Table 3-5: Comparison of rand/1(F=0.9 CR=0.1), rand/1(F=0.9 CR=0.9), rand/1(F=0.5,CR=0.3),.....	54
Table 3-6: Results of SaDE, ADE, SDE and jDE on $f_1 - f_{14}$ (10-D)	56
Table 3-7: Results of SaDE, ADE, SDE and jDE on $f_1 - f_{14}$ (30-D).....	57
Table 3-8: SaDE's results with different learning period (LP) on test functions f_8, f_{10} and f_{14} (10-D)	73
Table 4-1: Details of the 24 test problems.....	86
Table 4-2: Error Values Achieved When FES= 5×10^3 , FES= 5×10^4 , FES= 5×10^5 for Problems 1-6	88
Table 4-3: Error Values Achieved When FES= 5×10^3 , FES= 5×10^4 , FES= 5×10^5 for Problems 7-12	88
Table 4-4: Error Values Achieved When FES= 5×10^3 , FES= 5×10^4 , FES= 5×10^5 for Problems 13-18	88
Table 4-5: Error Values Achieved When FES= 5×10^3 , FES= 5×10^4 , FES= 5×10^5 for Problems 19-24	89
Table 4-6: Number of FES to achieve the fixed accuracy level ($(f(\bar{x}) - f(\bar{x}^*)) \leq 0.0001$),	89
Table 5-1: Truncating the archive solutions by using harmonic mean distance.....	103
Table 5-2: The algorithmic description of MOSaDE	103
Table 5-3: Update_archive.....	106
Table 5-4: Local search Procedure	107
Table 5-5: The algorithmic description of OW-MOSaDE	109
Table 5-6: Properties of the test functions [32]	123
Table 5-7: The experimental settings for the test functions.....	123
Table 5-8: The results for R indicator on test functions 1-7.....	127
Table 5-9: The results for R indicator on test functions 8-13 when M=3.	127
Table 5-10: The results for R indicator on test functions 8-13 when M=5.	127
Table 5-11: The results for Hypervolume indicator $I_{\bar{H}}$ on test functions 1-7.	128
Table 5-12: The results for Hypervolume indicator $I_{\bar{H}}$ on test functions 8-13 when M=3.....	128
Table 5-13: The results for Hypervolume indicator $I_{\bar{H}}$ on test functions 8-13 when M=5.....	128
Table 5-14: The results for Covered sets CS for test function SYMPART.....	129
Table 5-15: The average IGD value of the 30 final approximation sets obtained for each test problem.	138

Table A-1 Data set for test problem g19 160

Table A-2 Data set for test problem g20 162

Table A-3 Details of the 24 test problems..... 165

Table A-4 $f(x^*)$ and the bounds for the 24 problems 166

Chapter 1 Introduction

Optimization problems exist everywhere. Manufacturers aim to maximize the efficiency of their production processes. Engineers optimize the parameters of their designs. Investors seek portfolios that give them the maximum return with the minimum risk exposure.

To optimize these problems, we must first identify some objective, a quantitative measure of the performance of the system under study. This objective could be profit, cost, time, potential energy, or any quantity or combination of quantities that can be represented by a single number. The objective depends on certain characteristics of the system, called variables or parameters. Our goal is to find values of the variables that minimize or maximize one or several such objectives. Over the years, many optimization techniques have been developed to achieve this goal. This thesis investigates a relatively new optimization technique known as Differential Evolution (DE).

1.1 Motivation

1.1.1 Evolutionary Computation

Evolutionary computation is a subfield of artificial intelligence. It uses computational models of evolutionary processes as the key elements in design and implementation of computer-based problem solving system. A number of evolutionary computational models (which are referred to as evolutionary algorithms) have been developed, including genetic algorithms, evolutionary programming, and evolution strategies.

Genetic Algorithms (GAs) were introduced by Holland. He summarized his work on adaptive and reproduction plans in 1975 in a book that serves as the starting point of nearly all known applications and implementations of genetic algorithms [1]. After Holland's pioneering work, there have been substantial improvements in the

understanding of what simple GAs can and cannot do. Holland's theory of schemata suggests that GAs work by combining highly fit sub-solutions or building blocks. Unfortunately, the simple GA can only process well a small fraction of these building blocks, those that are compact. For this reason, there has been extensive research in designing GAs that can effectively process building blocks, regardless of whether they are compact or not. Another important area of research is the investigation of how a GA's performance is affected by the parameters and operators. Work along these lines, included studies on selection, population sizing models, and systematic controlled experiments for analyzing the relationships between various GA parameters.

Evolution Strategies (ES) [1] are a joint development of Bienert, Rechenberg and Schwefel, who did preliminary work in this area in the 1960s at the Technical University of Berlin in Germany, to solve the shape optimization of a bended pipe, the drag minimization of a jointed plate, and the shape optimization of a flashing nozzle. Due to how time-consuming this evaluation took in early studies, a simple two-member ES was used. Later, multi-member ES, recombinative ES and self-adaptive ES were suggested.

Evolutionary Programming (EP) was first used by Lawrence J. Fogel [2] [3] in 1960 in order to use simulated evolution as a learning process aiming to generate artificial intelligence. However, this approach remained underused for approximately thirty years. In the late 1980s, David B. Fogel extended evolutionary programming to applications involving real-parameter optimization problems [4]. Real-parameter EP is similar to ES, in that the mutations in both algorithms are normally distributed. Both algorithms encode mutation strength for each decision variable and self-adapt to the mutation strengths.

1.1.2 Why DE?

DE is a population-based nonlinear continuous global optimization algorithm. One of the first articles published on DE was *Differential evolution - A simple and efficient heuristic for global optimization over continuous Spaces*. As the title of this article suggests, the simplicity and efficiency of DE is what impressed many researchers and

engineers. Moreover, with self-referential mutation, DE adapts the step length intrinsically along the evolutionary process, unlike the pre-defined probability distribution used by the Evolution Strategy. Being inherently parallel, DE can save much computational time and space by executing on many different processing devices when solving large optimization problems.

Most importantly, many comparative studies report the high robustness, convergence speed, and solution quality of DE, both on benchmark functions and real-world applications. These studies are reviewed in Section 2.5.5. All of the abovementioned reasons make DE very popular among scientists and engineers, especially in the last 3-4 years. The growth in usage and adoptability is shown in Figure 1-1.

Figure 1-1: Number of publications per year in ISI web of knowledge database

1.1.3 Limitations of DE

Since the objective of this thesis is to extensively improve DE for optimization tasks, we are particularly motivated by the following limitations of conventional DE.

DE has various trial vector generation strategies, where a particular strategy may be apt to problems with different properties. The control parameters of each strategy, *i.e.* the values of crossover rate, scaling factor and population size, are also sensitive to

these problems. To achieve the most satisfactory optimization performance by applying the conventional DE to a given problem, it is common to perform an extensive trial-and-error search for the most appropriate trial vector generation strategy and then fine-tune its associated control parameter values. This expends a huge amount of computational resources. Moreover, one trial vector generation strategy with certain control parameter values may not always be the optimal choice throughout the whole evolution (or search) process. In fact, at different stages of the search process, a specific strategy with some parameter values may be more preferred than others in order to yield the best performance. Therefore, several inspirations come into our minds:

Is it possible to avoid the expensive computational costs spent on selecting the most appropriate strategy and its associated parameter values in DE implementation? Can we make full use of more potential strategies at different stages of evolution?

The original DE is designed for single-objective optimization. However, optimization problems in the real-world are more complicated, and always or possibly contain various constraints, or more than one objective.

Can we extend the newly proposed algorithm to handle constrained optimization and multi-objective optimization problems so that users can utilize the algorithm to solve real-world problems?

1.2 Objectives

The research in this thesis concentrates on the improvements of DE. The main objectives of this research are:

- 1) To self-adapt the strategies and the associated parameters of the conventional DE.
- 2) To extend the self-adaptive DE algorithm for constrained optimization problems.
- 3) To extend the self-adaptive DE algorithm for multi-objective optimization problems.

1.3 Major Contributions of the Thesis

The contributions of this thesis can be summarized as follows:

1) A comprehensive review of Differential Evolution is presented, covering the origin, basics, development and applications of DE. Where the development of DE is concerned, we also explore in detail how researchers set control parameters and choose trial vector generation strategies for different problems.

2) A novel Self-adaptive Differential Evolution (SaDE) algorithm is proposed, in which both trial vector generation strategies and their associated control parameter values are gradually self-adapted by learning from their previous experiences in generating promising solutions. Consequently, a more suitable generation strategy along with its parameter settings can be determined adaptively to match different phases of the search process or evolution. The performance of the SaDE algorithm is extensively evaluated on a suite of bound-constrained numerical optimization problems and compares favorably with conventional DE and several state-of-the-art parameter adaptive DE variants. Therefore, for researchers who are unfamiliar with DE, the SaDE algorithm successfully removes the headache of choosing the strategy or parameters. Furthermore, it saves on computational costs that researchers, who want to achieve the most satisfactory optimization performance, would spend on trial-and-error searches for the appropriate strategy and parameters.

3) Furthermore, we extend the SaDE algorithm to solve optimization problems with constraints. The performance of the proposed method is reported on 12 benchmark problems chosen from the Congress on Evolutionary Computation 2006 (CEC2006) special session on constrained real parameter optimization.

4) The limitations of the Zitzler-Deb-Thiele (ZDT) and Deb-Thiele-Laumanns-Zitzler (DTLZ) test suites are analyzed and modifications to overcome the shortcomings are proposed. These modified ZDT and DTLZ test suites and several selected benchmark functions for multi-objective optimization were provided by the CEC2007 special session on Performance Assessment of Multi-Objective Optimization Algorithms.

5) An extension of the SaDE to solve multi-objective optimization problems called Multi-Objective SaDE (MOSaDE) is proposed, and the performance evaluation is conducted.

6) An objective-wise learning strategy is proposed and applied to MOSaDE, resulting in Objective-Wise MOSaDE (OW-MOSaDE).

1.4 Organization of the Thesis

This thesis is structured into six chapters, beginning with this introduction.

Chapter 2 briefly introduces the evolutionary algorithms and gives the problem definitions. It then reviews the origin of the DE, followed by an extensive overview of the basics, development and applications of DE.

Chapter 3 presents the details of the proposed SaDE algorithm. Performance evaluations of the SaDE are conducted on a comprehensive set of benchmarks, together with comparative studies, investigations of self-adaptive characteristic, and sensitivity analysis of the parameter LP .

Chapter 4 generalizes the SaDE algorithm, which allows it to solve optimization problems with multiple constraints.

Chapter 5 is devoted to extending the SaDE to solve numerical optimization problems with multiple conflicting objectives. After reviewing preliminary concepts of multi-objective optimization and related work of differential evolution in multi-objective optimization, we simply extend SaDE to multi-objective optimization or MOSaDE. Consequently, we propose Objective-Wise MOSaDE (OW-MOSaDE) by learning the strategy and parameters accorded to each objective. A suite of benchmark problems is provided to evaluate the performance of the proposed MOSaDE and OW-MOSaDE algorithm.

Finally, the conclusions and recommendations for future research are addressed in Chapter 6.

Chapter 2 Background and Literature Study

2.1 Global Optimization

Global optimization is the process of finding the global minimum or maximum within the search space.

Definition 2.1 (Global minimum): Given a function $f : M \subseteq \mathbb{R}^n \rightarrow \mathbb{R}, M \neq \emptyset$, for $\mathbf{x}^* \in M$ the value $f^* := f(\mathbf{x}^*) > -\infty$ is called a global minimum, iff

$$\forall \mathbf{x} \in M : f(\mathbf{x}^*) \leq f(\mathbf{x})$$

Then, $\mathbf{x}$ is a **global minimum** point, f is called *objective function*, and the set M is called the feasible region. The problem of determining a global minimum point is called the **global minimization** problem.

Usually, maximization problems are transformed into minimization ones as follows:

$$\max \{f(\mathbf{x}) \mid \mathbf{x} \in M\} = -\min \{-f(\mathbf{x}) \mid \mathbf{x} \in M\}$$

Definition 2.2 (Local minimum): For $\hat{\mathbf{x}} \in M$ the value $\hat{f} := f(\hat{\mathbf{x}})$ is called a **local minimum**, iff $\exists \varepsilon \in \mathbb{R}, \varepsilon > 0 : \forall \mathbf{x} \in M : \|\mathbf{x} - \hat{\mathbf{x}}\| < \varepsilon \Rightarrow \hat{f} \leq f(\mathbf{x})$

In other words, an ε -environment $U_\varepsilon(\hat{\mathbf{x}}) = \{\mathbf{x} \in M \mid \|\mathbf{x} - \hat{\mathbf{x}}\| < \varepsilon\}$ exists such that $\hat{f}$ is the smallest feasible objective function value within this environment.

2.2 Evolutionary Algorithm

The term Evolutionary Algorithm (EA) represents a class of stochastic optimization methods that emulate the process of natural evolution, inspired by biological evolution processes including reproduction, mutation, recombination, natural selection and survival of the fittest. The origins of EAs can be traced back to the late 1950s. Since the 1970s, several evolutionary methodologies have been proposed, such as genetic algorithms, evolutionary programming, and evolution strategies [6]. All of these approaches operate on a population of individuals. The population of EA is initialized,

Chapter 2: Background and Literature Study

often randomly. The objective value, also called the fitness, is evaluated for each individual in the population. After completing these two steps, the algorithm begins its evolutionary cycle. Iterations, also called generations, last until a stopping condition is satisfied. In each evolutionary cycle, the population passes through the following three steps:

1. Selection of the individuals that are more apt to reproduce offspring.
2. Variations of the selected individuals in a random manner. Mainly crossover and mutation operations are used.
3. Replacement operation determines which solution instances are retained in the population of the next generation.

Although the underlying principles are simple, these algorithms have proven themselves as general, robust and powerful search methods. Back, Hammel, and Schwefel [6] argue that “The most significant advantage of using evolutionary search lies in the gain of flexibility and adaptability to the task at hand, in combination with robust performance (although this depends on the problem class) and global search characteristics.”

At the 2005 Genetic and Evolutionary Computation Conference, Libo and Lima organized a workshop entitled *Parameter setting in evolutionary algorithms*. Libo, Lima and Michalewicz also presented a book with the same name as the workshop, covering topics on various areas of evolutionary computation, including genetic algorithms, evolution strategies, genetic programming and estimation of distribution algorithms. They also discussed the parameters used in parallel implementations, multi-objective EAs, and real-world applications [8].

Estimation of Distribution Algorithms (EDAs), also called Probabilistic Model-Building Genetic Algorithms (PMBGA), are a novel class of evolutionary optimization algorithms that were developed as a natural alternative to genetic algorithms over the last twenty years. EDAs assume that it is possible to build a model of the promising areas of the search space and to use this model to guide the search for the optimum. In EDAs, modeling is achieved by building a probabilistic graphical model that

Chapter 2: Background and Literature Study

represents a condensed representation of the features shared by the selected solutions. Once the model is obtained, it can be sampled to produce more candidate solutions, which are then used to adapt or regenerate the model. Such a model can capture different patterns of interactions between subsets of the problem variables, and can conveniently use this knowledge to sample new solutions.

The principal advantages of EDAs over GAs are the absence of multiple parameters to be tuned (e.g. crossover and mutation probabilities) and the expressiveness and transparency of the probabilistic model that guides the search process.

EDAs are typically classified according to the level of variable interaction that their probabilistic model includes. They can be classed as univariate (no interactions), bivariate (interactions between pairs of variables) or multivariate (interactions between more than two variables) [7].

2.3 The Origin of Differential Evolution

Differential evolution grew out of the attempts to use genetic annealing [9] to solve the Chebyshev polynomial fitting problem. Genetic annealing is a population-based, combinatorial optimization algorithm that implements a thermodynamic annealing criterion via thresholds. Although successfully applied to solve many combinatorial tasks, genetic annealing cannot solve the Chebyshev problem well. Price then changed bit-string encoding into floating-point and logical operations into arithmetic and the differential mutation operator was born. Later, Storn suggested creating separate parent and children populations to accommodate the parallel machine architectures. Furthermore, Price and Storn removed the annealing criterion. Eventually, they solved the Chebyshev problem effectively and efficiently, and the obtained algorithm was named as **differential evolution**.

2.4 The Basics of Differential Evolution

The DE algorithm aims at evolving a population of NP D -dimensional parameter vectors representing the candidate solutions (also called individuals) at G generation,

i.e. $\mathbf{X}_{i,G} = \{x_{i,G}^1, \dots, x_{i,G}^D\}$, $i = 1, \dots, NP$ towards the global optima. The initial population covers the entire search space as much as possible by uniformly randomizing individuals within the search space constrained by the prescribed minimum and maximum parameter bounds $\mathbf{X}_{\min} = \{x_{\min}^1, \dots, x_{\min}^D\}$ and $\mathbf{X}_{\max} = \{x_{\max}^1, \dots, x_{\max}^D\}$. For example, the initial value of the j^{th} parameter in the i^{th} individual at the generation $G = 0$ is initialized as follows:

$$x_{i,0}^j = x_{\min}^j + rand_j(0,1) \cdot (x_{\max}^j - x_{\min}^j) \quad j = 1, 2, \dots, D \quad (2-1)$$

where $rand_j(0,1)$ represents a uniformly distributed random variable within the range $[0,1]$. The superscript j indicates that a new random value is generated for each parameter.

2.4.1 Mutation operation

After initialization, DE employs the mutation operation to produce a mutant vector $\mathbf{V}_{i,G}$ for each individual $\mathbf{X}_{i,G}$, also called the target vector, in the current population. For each target vector $\mathbf{X}_{i,G}$ at generation G , its associated mutant vector $\mathbf{V}_{i,G} = \{v_{i,G}^1, v_{i,G}^2, \dots, v_{i,G}^D\}$ can be generated by employing a mutation strategy. Five most frequently used mutation strategies implemented in the DE codes publicly available online at <http://www.icsi.berkeley.edu/~storn/code.html> are listed below:

$$\text{"DE/rand/1": } \mathbf{V}_{i,G} = \mathbf{X}_{r_1^i,G} + F \cdot (\mathbf{X}_{r_2^i,G} - \mathbf{X}_{r_3^i,G}) \quad (2-2)$$

$$\text{"DE/best/1": } \mathbf{V}_{i,G} = \mathbf{X}_{best,G} + F \cdot (\mathbf{X}_{r_1^i,G} - \mathbf{X}_{r_2^i,G}) \quad (2-3)$$

$$\text{"DE/rand to best/1": } \mathbf{V}_{i,G} = \mathbf{X}_{i,G} + F \cdot (\mathbf{X}_{best,G} - \mathbf{X}_{i,G}) + F \cdot (\mathbf{X}_{r_1^i,G} - \mathbf{X}_{r_2^i,G}) \quad (2-4)$$

$$\text{"DE/best/2": } \mathbf{V}_{i,G} = \mathbf{X}_{best,G} + F \cdot (\mathbf{X}_{r_1^i,G} - \mathbf{X}_{r_2^i,G}) + F \cdot (\mathbf{X}_{r_3^i,G} - \mathbf{X}_{r_4^i,G}) \quad (2-5)$$

$$\text{"DE/rand/2": } \mathbf{V}_{i,G} = \mathbf{X}_{r_1^i,G} + F \cdot (\mathbf{X}_{r_2^i,G} - \mathbf{X}_{r_3^i,G}) + F \cdot (\mathbf{X}_{r_4^i,G} - \mathbf{X}_{r_5^i,G}) \quad (2-6)$$

The indices $r_1^i, r_2^i, r_3^i, r_4^i, r_5^i$ are mutually exclusive integers randomly generated within the range $[1, NP]$, which are also different from the index i . These indices are randomly generated for each mutant vector. The scaling factor F is a positive control

parameter used to scale the difference vector. $\mathbf{X}_{best,G}$ is the best individual vector with the best fitness value in the population at generation G .

2.4.2 Crossover operation

After the mutation phase, crossover operation is applied to each pair of the target vector $\mathbf{X}_{i,G}$ and its corresponding mutant vector $\mathbf{V}_{i,G}$ to generate a trial vector:

$\mathbf{U}_{i,G} = (u_{i,G}^1, u_{i,G}^2, \dots, u_{i,G}^D)$. In the basic version, DE employs the binomial (uniform) crossover defined as follows:

$$u_{i,G}^j = \begin{cases} v_{i,G}^j, & \text{if } (rand_j[0, 1] \leq CR) \text{ or } (j = j_{rand}) \\ x_{i,G}^j, & \text{otherwise} \end{cases} \quad j = 1, 2, \dots, D \quad (2-7)$$

In equation (2-7), the crossover rate CR is a user-specified constant within the range $[0, 1]$, which controls the fraction of parameter values copied from the mutant vector. j_{rand} is a randomly chosen integer in the range $[1, D]$. The binomial crossover operator copies the j^{th} parameter of the mutant vector $\mathbf{V}_{i,G}$ to the corresponding element in the trial vector $\mathbf{U}_{i,G}$ if $rand_j[0,1] \leq CR$ or $j = j_{rand}$. Otherwise, it is copied from the corresponding target vector $\mathbf{X}_{i,G}$. There exists exponential crossover operator, in which the parameters of trial vector $\mathbf{U}_{i,G}$ are inherited from the corresponding mutant vector $\mathbf{V}_i$ starting from a randomly chosen parameter index till the first time $rand_j[0,1] > CR$. The remaining parameters of the trial vector $\mathbf{U}_{i,G}$ are copied from the corresponding target vector $\mathbf{X}_{i,G}$. The condition $j = j_{rand}$ is introduced to ensure that the trial vector $\mathbf{U}_{i,G}$ will differ from its corresponding target vector $\mathbf{X}_{i,G}$ by at least one parameter. DE's exponential crossover operator is functionally equivalent to the circular two-point crossover operator.

Figure 2-1 Coordinate rotation changes the location of trial vectors generated by the crossover operator defined in Equation (2-7).

The crossover operator used in the conventional DE is a discrete recombination. Figure 2-1 illustrates a 2D example of recombining the parameters of two vectors $\mathbf{X}_{i,G}$ and $\mathbf{V}_{i,G}$ according to this crossover operator, in which the potential trial vectors are generated at the corners of a rectangle. As shown in Figure 2-1, the discrete recombination is a rotationally variant operation. A rotation transforms the coordinates of both vectors and thus changes the shape of the rectangle. Consequently, the potential location of the trial vector moves from $\mathbf{U}_{i,G}$ to $\mathbf{U}'_{i,G}$. To overcome this limitation, a new trial vector generation strategy ‘DE/current-to-rand/1’ is proposed in [48], which replaces the crossover operator prescribed in equation (2-7) with the rotationally invariant arithmetic crossover operator to generate the trial vector $\mathbf{U}_{i,G}$. This is achieved by linearly combining the target vector $\mathbf{X}_{i,G}$ and the corresponding mutant vector $\mathbf{V}_{i,G}$ as follows:

$$\mathbf{U}_{i,G} = \mathbf{X}_{i,G} + K \cdot (\mathbf{V}_{i,G} - \mathbf{X}_{i,G}) \quad (2-10)$$

The mutant vector $\mathbf{V}_{i,G}$ can be derived by equation (2-2), and thus we can obtain:

$$\mathbf{U}_{i,G} = \mathbf{X}_{i,G} + K \cdot (\mathbf{X}_{r_1,G} + F \cdot (\mathbf{X}_{r_2,G} - \mathbf{X}_{r_3,G}) - \mathbf{X}_{i,G}) \quad (2-11)$$

The ‘DE/current-to-rand/1’ strategy in (2-11) can be further expressed by:

$$\mathbf{U}_{i,G} = \mathbf{X}_{i,G} + K \cdot (\mathbf{X}_{r_1,G} - \mathbf{X}_{i,G}) + F \cdot (\mathbf{X}_{r_2,G} - \mathbf{X}_{r_3,G}) \quad (2-12)$$

where K is the combination coefficient, which has been proven in [48] to be effective when it is chosen with a uniform random distribution from $[0, 1]$. The rotationally

invariant property of the strategy DE/current-to-rand/1 has been verified when solving the multi-objective optimization problem in [12].

2.4.3 Selection operation

If the values of some parameters of a newly generated trial vector exceed the corresponding upper and lower bounds, we randomly and uniformly reinitialize them within the pre-specified range. Then, the objective function values of all trial vectors are evaluated. After that, a selection operation is performed. The objective function value of each trial vector $f(\mathbf{U}_{i,G})$ is compared to that of its corresponding target vector $f(\mathbf{X}_{i,G})$ in the current population. If the trial vector has less or equal objective function value than the corresponding target vector (minimization objective), the trial vector will replace the target vector and enter the population of the next generation. Otherwise, the target vector will remain in the population for the next generation and the trial vector is discarded. The selection operation can be expressed as follows:

$$\mathbf{X}_{i,G+1} = \begin{cases} \mathbf{U}_{i,G}, & \text{if } f(\mathbf{U}_{i,G}) \leq f(\mathbf{X}_{i,G}) \\ \mathbf{X}_{i,G}, & \text{otherwise} \end{cases} \quad (2-13)$$

The above three steps are repeated generation after generation until one of the specific termination criteria is satisfied.

The algorithmic description of DE is presented in Table 2-1. Furthermore, the Illustration of classic DE (DE/rand/1/bin) in 2-dimensional space is shown in Figure 2-2.

Table 2-1: The algorithmic description of DE

Step 1	Set the generation number $G = 0$, and randomly initialize a population of NP individuals $\mathbf{P}_G = \{\mathbf{X}_{1,G}, \dots, \mathbf{X}_{NP,G}\}$ with $\mathbf{X}_{i,G} = \{x_{i,G}^1, \dots, x_{i,G}^D\}$, $i = 1, \dots, NP$ uniformly distributed in the range $[\mathbf{x}_{\min}, \mathbf{x}_{\max}]$, where $\mathbf{x}_{\min} = \{x_{\min}^1, \dots, x_{\min}^D\}$ and $\mathbf{x}_{\max} = \{x_{\max}^1, \dots, x_{\max}^D\}$.
Step 2	WHILE stopping criterion is not satisfied
	DO
	----- For 'DE/rand/1, DE/best/1, DE/rand-to-best/1, DE/best/2, DE/rand/2' --
	Step 2.1 Mutation step

Chapter 2: Background and Literature Study

//Generate a mutated vector $\mathbf{V}_{i,G} = \{v_{i,G}^1, \dots, v_{i,G}^D\}$ for each target vector $\mathbf{X}_{i,G}$

FOR $i = 1$ to NP

Generate a mutated vector $\mathbf{V}_{i,G} = \{v_{i,G}^1, \dots, v_{i,G}^D\}$ corresponding to the target vector $\mathbf{X}_{i,G}$ via one of the equations (2-2)-(2-6).

END FOR

Step 2.2 Crossover step

//Generate a trial vector $\mathbf{U}_{i,G} = \{u_{i,G}^1, \dots, u_{i,G}^D\}$ for each target vector $\mathbf{X}_{i,G}$

a) Binomial crossover

FOR $i = 1$ to NP

$$j_{rand} = \lfloor rand[0,1) \times D \rfloor$$

FOR $j = 1$ to D

$$u_{i,G}^j = \begin{cases} v_{i,G}^j, & \text{if } (rand[0,1) \leq CR) \text{ or } (j = j_{rand}) \\ x_{i,G}^j, & \text{otherwise} \end{cases}$$

END FOR

END FOR

b) Exponential crossover

FOR $i = 1$ to NP

$$j = \lfloor rand[0,1) \times D \rfloor, L=0$$

$$\mathbf{U}_{i,G} = \mathbf{X}_{i,G}$$

DO

$$u_{i,G}^j = v_{i,G}^j$$

$$j = \langle j+1 \rangle_D^*$$

$$L=L+1$$

WHILE ($rand[0,1) < CR$ & $L < D$)

END FOR

----- For 'DE/current-to-rand/1' -----

FOR $i = 1$ to NP

Chapter 2: Background and Literature Study

Generate a trial vector $\mathbf{U}_{i,G} = \{u_{i,G}^1, \dots, u_{i,G}^D\}$ corresponding to the target vector $\mathbf{X}_{i,G}$ via the equation (2-12)

END FOR

Step 2.3 Selection step

//Selection

FOR $i=1$ to NP

Evaluate the trial vector $\mathbf{U}_{i,G}$

IF $f(\mathbf{U}_{i,G}) \leq f(\mathbf{X}_{i,G})$, THEN $\mathbf{X}_{i,G+1} = \mathbf{U}_{i,G}$, $f(\mathbf{X}_{i,G+1}) = f(\mathbf{U}_{i,G})$

IF $f(\mathbf{U}_{i,G}) < f(\mathbf{X}_{best,G})$, THEN $\mathbf{X}_{best,G} = \mathbf{U}_{i,G}$, $f(\mathbf{X}_{best,G}) = f(\mathbf{U}_{i,G})$

END IF

END IF

END FOR

Step 2.4 Increment the generation count $G = G + 1$

Step 3 END WHILE

* Acute brackets $\langle \rangle_D$ denote modulo function with modulus D .

(a) Initialized population and four operating vectors (in red circle) in 2D continuous space

Chapter 2: Background and Literature Study

(b) Mutation

(c) Crossover

(d) Replacement

Figure 2-2: Illustration of DE/rand/1/bin in 2-dimensional space

2.5 The Development of DE

Fascinated by the prospect and potential of DE, many researchers are now working on developing improved variants. This has resulted in many variants of the algorithm. A brief overview of these contemporary research efforts is presented in this section.

2.5.1 Control parameters

Researchers have suggested certain guidelines for choosing the control parameter values for DE due to their high influence on the ultimate performance. Storn and Price [58] suggested that a reasonable value for NP should be between $5D$ and $10D$, a good initial choice of F would be 0.5, and the effective range of F should be in $[0.4, 1]$. The first reasonable choice of CR value can be 0.1. However, since the large CR value can speed up convergence, the value of 0.9 for CR may be a good initial choice if the problem is near uni-modal or fast convergence is desired. Moreover, if the population converges prematurely, either F or NP can be increased.

It was recommended in [48] to use the trial vector generation strategy DE/current-to-rand/1 and parameter settings $NP = 20D$, $K = 0.5$, $F = 0.8$. If the DE converges prematurely, one should increase the value of NP and F or decrease the value of K . If

Chapter 2: Background and Literature Study

the population stagnates, one should increase the value of NP or F , or randomly choose K within the range $[0, 1]$. If none of the above configurations work, one may try the strategy DE/rand/1/bin along with a small CR value. Gämperle *et al.* [59] examined different parameter settings for DE on Sphere, Rosenbrock and Rastrigin functions. Their experimental results showed that the searching capability and convergence speed are very sensitive to the choice of control parameters NP , F and CR . They recommended that the population size, the scaling factor, and the crossover rate should be between $[3D, 8D]$, 0.6, and between $[0.3, 0.9]$, respectively.

Recently, authors in [40] suggested using F values between $[0.4, 0.95]$ with $F = 0.9$ being a good initial choice. The CR values should lie in $[0, 0.2]$ when the function is separable or should be 0.9 when the function is non-separable and multi-modal. However, when solving real-world problems, the characteristics of the problems are usually unknown. Consequently, it is difficult to choose the most appropriate CR value in advance.

In DE literature, conflicting conclusions have been drawn with regard to the rules for manually choosing the strategy and control parameters. This may be due to the variations in benchmark problem sets, chosen strategies, and the combinations of parameter values used. These conflicting conclusions and rules undesirably confuse researchers who employ DE to solve real-world problems.

To combat this problem, researchers have developed a few techniques to avoid manual tuning of the control parameters. For example, Das *et al.* [54] linearly reduced the scaling factor F with increasing generation count from a maximum to a minimum value or randomly varied F in the range $[0.5, 1]$. He also employed a uniform distribution between 0.5 and 1.5 (with a mean value of 1) to obtain a new hybrid DE variant [65]. In addition, several researchers focused on the adaptation of the control parameters F and CR . Liu and Lampinen introduced Fuzzy Adaptive Differential Evolution (FADE) using fuzzy logic controllers whose inputs incorporate the relative function values and individuals of successive generations to adapt the parameters for the mutation and crossover operations [39]. Based on the experimental results on test functions, the FADE algorithm outperformed the conventional DE on higher

dimensional problems. Zaharie proposed a parameter adaptation for DE (ADE), based on the idea of controlling the population diversity, and implemented a multi-population approach [12]. Following the same ideas, Zaharie and Petcu designed an adaptive Pareto DE algorithm for multi-objective optimization and analyzed its parallel implementation [13]. Abbass [17] self-adapted the crossover rate of DE for multi-objective optimization, by encoding the crossover rate into each individual, to simultaneously evolve with other parameters. The scaling factor F is generated for each variable from a Gaussian distribution $N(0, 1)$.

Subsequently, new research papers focusing on the adaptation of control parameters in DE have been published. Omran *et al.* [52] introduced a self-adapted scaling factor parameter F analogous to the adaptation of crossover rate CR in [17]. The crossover rate CR in [52] is generated for each individual from a normal distribution $N(0.5, 0.15)$. This approach, known as SDE, was tested on four benchmark functions and performed better than other versions of DE. Besides adapting the control parameters F and CR , Teo proposed Differential Evolution with Self-Adapting Populations (DESAP) [42], based on Abbass's self-adaptive Pareto DE. Recently, Brest *et al.* [46] encoded control parameters F and CR into the individuals and adjusted by introducing two new parameters τ_1 and τ_2 . In their algorithm, known as jDE, a set of F values was assigned to individuals in the population. Then, a random number $rand$ was uniformly generated in the range of $[0, 1]$. If $rand < \tau_1$, the F was reinitialized to a new random value in the range of $[0.1, 1.0]$, otherwise it was kept unchanged. The CR was adapted in the same manner but with a different re-initialization range of $[0, 1]$.

2.5.2 Trial vector generation strategy

Besides the trial vector generation strategies included in Storn's DE codes, more strategies have been proposed to improve the performance of the conventional DE. In the conventional DE, the base vector is simply a randomly chosen population vector without considering the objective function value. To guide differential evolution, Price [53] obtained the base vector by combining the target vector with one randomly selected vector whose objective function value is less than or equal to that of its target

vector. Later, Fan and Lampinen [82] also applied the objective function value in their proposed trigonometric mutation operation. The Trigonometric mutation DE (TDE) algorithm employed the new trigonometric mutation and the original differential mutation stochastically according to an additional control parameter Mt . The resulting DE variant would outperform the classic DE algorithm on certain benchmark and real world problems.

Chakraborty *et al.* [65] proposed a neighborhood-based local mutation operator that drew inspiration from Particle Swarm Optimization (PSO), and combined the local mutation with global mutation using a time-varying weight to balance the exploration and exploitation abilities of the DE. Noman and Iba applied an EA-like generational model to help accelerate the search capability of the DE algorithm [81]. Price and Rönkkönen [83] presented two mutation-only DE approaches and benchmarked their scaling performances to discover why large population sizes are needed to support optimizations with small-scale factors. Rahnamayan *et al.* [84] employed Opposition-Based Learning (OBL) for DE population initialization and also for generation jumping to improve the convergence rate. The experiments conducted on 58 complex benchmark functions, showed that the ODE outperformed the original DE and FADE [39].

Mezura-Montes *et al.* [98] presented an empirical comparison of eight DE trial vector generation strategies in order to identify which strategy is more suitable to solve an optimization problem depending on the problem's characteristics. Five strategies with discrete recombination: rand/1/bin, rand/1/exp, best/1/bin, best/1/exp, current-to-rand/1/bin [48]; two strategies with arithmetic recombination: current-to-rand/1, current-to-best/1 [50]; and one strategy which incorporates objective function information: rand/2/dir [97]. The authors selected 13 scalable test functions from [87], and grouped them as follows: uni-modal & separable, uni-modal & non-separable, multi-modal & separable and multi-modal & non-separable. From the experimental results, they came to the following conclusions: 1) Binomial crossover always performed better than the exponential crossover. 2) best/1/bin was the most competitive for uni-modal & separable and uni-modal & non-separable problems. However, for multi-modal & separable problems, this variant performed similarly as

rand/1/bin and rand/2/dir. For multi-modal & non-separable problems, rand/2/dir and rand/1/bin were much better than best/1/bin.

Recently, the strategies of DE were elaborated in the Chapter *Strategies of Search* in [96]. The author classified all the strategies into four groups according to their search behavior: random, directed, local, and hybrid. All the strategies were tested on standard benchmarks with problem dimensions from 2 to 100 and control parameters within a systematic range. Their conclusions are as follows:

- “1. There are strategies less sensitive to the control parameters. Generally, these are strategies that are similar to arithmetic crossover. And there are strategies more sensitive to the control parameters. These are basically the strategies that realize the local search around the best found solution. The practice shows that the more sensitive the strategy is, the better the solution achieved.
2. With increasing dimensionality of a problem, increasing number of randomly extracted individuals is needed. It could be explained by the need to increase the dimensionality of subspaces, where the search is performed.
3. The more complex the function is, the more random strategy is needed.”

2.5.3 Hybridization

Hybridizing with different algorithms is another direction to improve the DE algorithm. Zhang and Xie combined PSO with the classical DE [69]. Das improved particle swarm optimization with differentially perturbed velocity [79]. Sun *et al.* [78] hybridized DE and Estimation of Distribution Algorithm (EDA). The DE/EDA uses the local information obtained by DE mutation and global information extracted from a population of solutions by EDA modeling. Noman and Iba proposed the Fittest Individual Refinement (FIR) scheme, which accelerated DE by applying a fixed-length crossover based search in the neighborhood of the best solution [80].

2.5.4 Multi-modal optimization

In some real-world optimization problems, it may be necessary to locate not only one optimum but also the entire set of global or local optima. Such problems are called *multi-modal optimization problems*. Some investigations have been conducted to extend the DE to handle multi-modal optimization. Rumpler and Moore [71] modified

the DE to automatically determine the number of subpopulations as well as the minimal spanning distance between individuals of each population. Later, multi-resolution multi-population DE by Zaharie [73] and MultiDE by Hendershot [74] both adopted the ‘island model’ approach. Thomsen extended the DE with a crowding scheme to track and maintain multiple optima [75]. This variation is called crowding DE and outperformed the well-known fitness sharing scheme that penalizes similar candidate solutions. Recently, Li proposed species-based DE [76], which was more computationally efficient than the crowding DE.

2.5.5 Comparative Study

The first comparative study of DE was conducted by Storn and Price [54]. They compared DE/rand/1/bin to two annealing methods, two EAs and the stochastic differential equations. In most instances, the DE outperformed other approaches. Later, Ali and Törn [51] compared four DE versions to several versions of the Controlled Random Search (CRS) algorithm and two floating-point GAs on two sets of test problems. While one test set was composed of simple, low-dimensional problems, and the other, of relatively difficult ones. On the easy test set, DE versions solved all the problems, but converged slower than CRS. On the difficult test set, DE versions outperformed in terms of CPU time and success rate.

Vesterstrøm and Thomsen [45] compared DE/rand/1/exp with two PSO versions (PSO and arPSO) and a Simple EA (SEA) on a suite of 23 benchmark problems. The results showed that DE/rand/1/exp generally outperformed the other two algorithms. Rönkkönen *et al.* [40] also extensively evaluated DE/rand/1/bin on 25 benchmark problems. Price *et al.* provided a comprehensive comparison in [50], where DE was compared to 16 other optimizers (listed as following) on five well-known test problems.

- Evolutionary Programming with Adaptive Levy mutations (ALEP) [85]
- Attractive and repulsive Particle Swarm Optimization (arPSO) [45]
- Cooperative Co-evolutionary Genetic Algorithm (CCGA) [86]
- Classical Evolutionary Programming (CEP) [87]
- Conventional Evolutionary Programming with Adaptive Mutations (CEP/AM) [88]
- Classical Evolution Strategies (CES) [89]

- Cooperative Particle Swarm Optimization (CPSO-S6) [90]
- Evolutionary Optimization (EO) [91]
- Fast Evolution Programming (FEP) [87]
- Fast Evolutionary Strategies (FES) [89]
- Hybrid Taguchi-Genetic Algorithm (HTGA) [93]
- Orthogonal Genetic Algorithm (OGA) [92]
- Particle Swarm Optimization (PSO) [91]
- Quantum Evolutionary Algorithm with Rotation (QEA/R) [94]
- Simple Evolutionary Algorithm (SEA) [45]
- Stochastic Genetic Algorithm (StGA) [95].

For most problems, DE usually obtained the best result. However, for certain problems, it may be slower than some of the other methods.

2.6 The Applications of DE

Scientists and engineers have successfully applied DE to solve many optimization problems:

- Engineering design: Digital Filter Design [56], aerodynamic design [110], robust design of gas circuit breaker [112].
- Control: Optimal control and optimal time location problems [107], fuzzy controller design [108], fuzzy controller tuning [109].
- Neural network and fuzzy system: Neural network learning [99], training for feed-forward networks [38], online training [103], tuning fuzzy membership functions [104].
- Scheduling: plant scheduling and planning [105], heterogeneous multiprocessor scheduling [106].
- Chemical engineering and bio-systems: Heat transfer parameter estimation in a trickle bed reactor [100], optimal control of bioprocess systems [116], fermentation process [115].
- Bioinformatics: Find predictive gene subsets in microarray data [111], gene regulatory networks [114].
- Image processing: Image registration [101], unsupervised image classification [102].

Chapter 2: Background and Literature Study

Moreover, due to increasing interest in the DE algorithm, it has been built in MATHEMATICA's function minimization (since version 4.2) and MATLAB's GA toolbox.

Chapter 3 Differential Evolution Algorithm

with Strategy and Parameter Adaptation for

Global Numerical Optimization

3.1 Introduction

Evolutionary algorithms (EAs), inspired by the natural evolution of species, have been successfully applied to solve numerous optimization problems in diverse fields. However, when implementing EAs, users not only need to determine the appropriate evolutionary operators, but also need to choose the suitable parameter settings to ensure the success of the algorithm. This often leads to demanding computational costs due to the time-consuming trial and error parameter and operator tuning process. To overcome such problems, researchers have actively investigated the adaptation of parameters and operators in EAs [117]-[119]. Different categorizations of parameter adaptation methods have been presented in [120]-[121]. Angeline [120] summarized two types of parameter updating rules in an adaptive EA, namely absolute and empirical rules. Absolute updating rules usually pre-specify how the parameter modifications are made, while empirical updating rules adapt parameters according to the competition inherent in EAs. Literature [121] divided the parameter adaptation techniques into three categories: deterministic, adaptive and self-adaptive control rules. Deterministic rules modify the parameters according to certain pre-determined rationales without utilizing any feedback from the search process. Adaptive rules incorporate some form of feedback from the search procedure to guide the parameter adaptation. Self-adaptive rules directly encode parameters into the individuals and evolve them together with the encoded solutions. Parameter values involved in individuals with better fitness values will survive, which fully utilize the feedback from the search process. Generally speaking, self-adaptive rules can also refer to the

rules that mainly utilize the feedback from the search process, such as fitness values, to guide the updating of parameters.

Differential Evolution (DE) is an efficient and powerful population-based stochastic search technique for solving global optimization problems over continuous space, which has been widely applied in many scientific and engineering fields. However, the success of DE to handle a specific problem crucially depends on the proper choice of trial vector generation strategies and their associated control parameter values. Employing the trial and error scheme to search for the most suitable strategy and its associated parameter settings requires high computational costs. Moreover, at different stages of evolution, different strategies coupled with different parameter settings may be required in order to achieve the optimal performance. In this paper, we propose a Self-adaptive DE (SaDE) algorithm, in which both trial vector generation strategies and their associated control parameter values are gradually self-adapted by learning from their previous experiences in generating promising solutions. Consequently, a more suitable generation strategy along with its parameter settings can be determined adaptively to match different phases of the search process or evolution. The performance of the SaDE algorithm is extensively evaluated on a suite of bound-constrained numerical optimization problems and compares favorably with the conventional DE and several state-of-the-art parameter adaptive DE variants.

3.2 Differential Evolution Algorithm with Strategy Adaptation

As we reviewed in Chapter 2, DE has many different trial vector generation strategies, which may be suitable to solve problems with different properties. Moreover, the control parameters of each strategy are also sensitive to the specific problems.

To achieve the most satisfactory optimization performance by applying conventional DE to a given problem, it is common to perform a trial-and-error search for the most appropriate trial vector generation strategy, then fine-tune its associated control parameter values, *i.e.* the values of CR , F and NP . This, however, may expend a huge amount of computational costs. Moreover, during different stages of evolution, different trial vector generation strategies coupled with specific control parameter

values can be more effective than others. Motivated by these observations, we develop a Self-adaptive DE (SaDE) algorithm, in which both trial vector generation strategies and their associated control parameter values can be gradually self-adapted according to their previous experiences of generating promising solutions. The core idea behind the proposed SaDE algorithm is elucidated below.

3.2.1 Trial Vector Generation Strategy Adaptation

DE realization that employs different trial vector generation strategies usually performs differently when solving different optimization problems. Instead of employing the computationally expensive trial-and-error search for the most suitable strategy and its associated parameter values, we maintain a strategy candidate pool that includes several trial vector generation strategies with effective yet diverse characteristics. During evolution, with respect to each target vector in the current population, one strategy will be chosen from the candidate pool according to a probability learnt from its previous experience of generating promising solutions. This probability is then applied to perform the mutation operation. The more successful a particular strategy behaves in previous generations to generate promising solutions, the higher probability it will be chosen in the current generation to generate solutions. We investigated several effective trial vector generation strategies commonly referred to in DE literatures and chose a subset of them to construct the strategy candidate pool.

- Strategies relying on the best solution found so far, such as ‘DE/rand-to-best/1/bin’, ‘DE/best/1/bin’ and ‘DE/best/2/bin’, usually have the fast convergence speed and perform well when solving uni-modal problems. However, when solving multi-modal problems, they are more likely to get stuck at a local optimum, thereby leading to a premature convergence.
- The ‘DE/rand/1/bin’ strategy usually demonstrates slow convergence speed and bears stronger exploration capability. Therefore, it is better suited for solving multi-modal problems than the strategies relying on the best solution found so far.
- The ‘DE/best/1/bin’ strategy is a degenerated case of the “DE/rand-to-best/1/bin” strategy with F equal to 1.

- Two difference vectors based strategies may result in better perturbation than one difference vector based strategies. Storn [55] claimed that according to the central limit theorem, the random variation of the summation of difference vectors of all target vector pairs in the current population was shifted slightly towards the Gaussian direction, which is the most commonly used mutation operator in EAs. The advantage of using two difference vectors based strategies was also discussed in [69], which empirically demonstrated that the statistical distribution of the summation of one difference vector had a triangle shape, while the statistical distribution of the summation of all two difference vectors had a bell shape, which is generally regarded as a better perturbation mode.
- DE/current-to-rand/1 is a rotation-invariant strategy. Its effectiveness has been verified when it was applied to solve multi-objective optimization problems [12].

Considering these observations, we incorporate four strategies into the strategy candidate pool, *i.e.* ‘DE/rand/1/bin’, ‘DE/rand-to-best/2/bin’, ‘DE/rand/2/bin’ and ‘DE/current-to-rand/1’. ‘DE/rand/1/bin’ is frequently employed in many DE literatures. ‘DE/rand-to-best/2/bin’ and ‘DE/rand/2/bin’ can have a better exploration capability due to the Gaussian like perturbation while ‘DE/current-to-rand/1’ enables the algorithm to solve rotated problems more effectively. The four trial vector generation strategies constituting the strategy candidate pool in the proposed SaDE algorithm are listed below. The binomial type crossover operator is utilized in the first three strategies due to its popularity in many DE literatures [54] [50].

DE/rand/1/bin:

$$u_{i,j} = \begin{cases} x_{r_1,j} + F \cdot (x_{r_2,j} - x_{r_3,j}) & \text{if } rand[0,1) < CR \text{ or } j = j_{rand} \\ x_{i,j} & \text{otherwise} \end{cases}$$

DE/rand-to-best/2/bin:

$$u_{i,j} = \begin{cases} x_{i,j} + F \cdot (x_{best,j} - x_{i,j}) + F \cdot (x_{r_1,j} - x_{r_2,j}) + F \cdot (x_{r_3,j} - x_{r_4,j}) & \text{if } rand[0,1) < CR \text{ or } j = j_{rand} \\ x_{i,j} & \text{otherwise} \end{cases}$$

DE/rand/2/bin:

$$u_{i,j} = \begin{cases} x_{r_1,j} + F \cdot (x_{r_2,j} - x_{r_3,j}) + F \cdot (x_{r_4,j} - x_{r_5,j}) & \text{if } rand[0,1) < CR \text{ or } j = j_{rand} \\ x_{i,j} & \text{otherwise} \end{cases}$$

$$\text{DE/current-to-rand/1: } U_{i,G} = X_{i,G} + K \cdot (X_{r_1,G} - X_{i,G}) + F \cdot (X_{r_2,G} - X_{r_3,G})$$

Generally speaking, a good candidate pool should be restrictive so that the unfavorable influences of less effective strategies can be suppressed. Moreover, a set of effective strategies contained in a good candidate pool should have diverse characteristics, that is, the used strategies should demonstrate distinct capabilities when dealing with a specific problem at different stages of evolution. The theoretical study on the choice of the optimal pool size and the selection of strategies used in the pool are attractive research issues and deserve further investigation.

In the SaDE algorithm, with respect to each target vector in the current population, one trial vector generation strategy is selected from the candidate pool according to the probability learned from its success rate in generating improved solutions within a certain number of previous generations. The selected strategy is subsequently applied to the corresponding target vector to generate a trial vector. More specifically, at each generation, the probabilities of choosing each strategy in the candidate pool are summed to 1. These probabilities are gradually adapted during evolution in the following manner: Assume that the probability of applying the k^{th} strategy in the candidate pool to a target vector in the current population is p_k , $k = 1, 2, \dots, K$, where K is the total number of strategies contained in the pool. The probabilities with respect to each strategy are initialized as $1/K$, i.e. all strategies have an equal probability of being chosen. We use the Stochastic Universal Selection method [43] to select one trial vector generation strategy for each target vector in the current population. At

generation G , after evaluating all the generated trial vectors, the number of trial vectors generated by the k^{th} strategy that can successfully enter the next generation is recorded as $ns_{k,G}$ while the number of trial vectors generated by the k^{th} strategy that are discarded in the next generation is recorded as $nf_{k,G}$. We introduce *Success and Failure Memories* to store these numbers within a fixed number of previous generations hereby named Learning Period (LP). As illustrated in

Figure 3-1, at generation G , the number of trial vectors generated by different strategies that can enter or fail to enter the next generation over the previous LP generations is stored in different columns of the *Success and Failure Memories*. Once the memories overflow after LP generations, the earliest records stored in the memories, i.e. ns_{G-LP} or nf_{G-LP} will be removed so that the numbers calculated in the current generation can be stored in those memories, as shown in Figure 3-2.

After the initial LP generations, the probabilities of choosing different strategies will be updated at each subsequent generation based on the *Success and Failure Memories*. For example, at generation G , the probability of choosing the k^{th} ($k = 1, 2, \dots, K$) strategy is updated by:

$$p_{k,G} = \frac{S_{k,G}}{\sum_{k=1}^K S_{k,G}}, \quad (3-1)$$

$$S_{k,G} = \frac{\sum_{g=G-LP}^{G-1} ns_{k,g}}{\sum_{g=G-LP}^{G-1} ns_{k,g} + \sum_{g=G-LP}^{G-1} nf_{k,g}} + \epsilon, \quad (k=1,2,\dots,K; G > LP) \quad (3-2)$$

where $S_{k,G}$ represents the success rate of the trial vectors generated by the k^{th} strategy that successfully entered the next generation within the previous LP generations with respect to generation G . The small constant value $\epsilon = 0.01$ is used to avoid the possible null success rates. To ensure that the probabilities of choosing strategies are always summed to 1, we further divide $S_{k,G}$ by $\sum_{k=1}^K S_{k,G}$ to calculate $p_{k,G}$. The larger the success rate for the k^{th} strategy within the previous LP generations, the larger the probability of applying it to generate the trial vectors at the current generation.

Success Memory				
Index	Strategy 1	Strategy 2	...	Strategy K
1	$ns_{1,G-LP}$	$ns_{2,G-LP}$	...	$ns_{k,G-LP}$
2	$ns_{1,G-LP+1}$	$ns_{2,G-LP+1}$	...	$ns_{k,G-LP+1}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
LP	$ns_{1,G-1}$	$ns_{2,G-1}$	...	$ns_{k,G-1}$

Failure Memory				
Index	Strategy 1	Strategy 2	...	Strategy K
1	$nf_{1,G-LP}$	$nf_{2,G-LP}$	...	$nf_{k,G-LP}$
2	$nf_{1,G-LP+1}$	$nf_{2,G-LP+1}$	...	$nf_{k,G-LP+1}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
LP	$nf_{1,G-1}$	$nf_{2,G-1}$	...	$nf_{k,G-1}$

Figure 3-1: Success Memory and Failure Memory

Figure 3-2: Progress of Success Memory

3.2.2 Parameter Adaptation

In conventional DE, the choice of numerical values for the three control parameters F , CR and NP , highly depends on the problem under consideration. Some empirical guidelines for choosing reasonable parameter settings have been discussed in Chapter 2. In the proposed SaDE algorithm, we leave NP as a user-specified parameter since it highly relies on the complexity of a given problem. However, the population size NP does not need to be fine-tuned and just a few typical values can be tried according to the pre-estimated complexity of the given problem.

Regarding the other two parameters, CR is usually more sensitive to problems with different characteristics, e.g. the uni-modality and multi-modality, while F is closely related to the convergence speed. Usually, transforming F into a random variable effectively increases the pool of potential trial vectors and minimizes the risk of

stagnation without increasing the population size. After our empirical study, the parameter F is approximated by a normal distribution with mean value 0.5 and standard deviation 0.3, denoted by $N(0.5, 0.3)$. A set of F values are randomly sampled from the normal distribution and applied to each target vector in the current population. It is easy to verify that values of F must fall into the range $[-0.4, 1.4]$ with the probability of 0.997. By doing so, we attempt to maintain both exploitation (with small F values) and exploration (with large F values) power throughout the entire evolution process. Following suggestions in [48], the control parameter K in the strategy “DE/current-to-rand/1” is hereby randomly generated within $[0, 1]$ so as to eliminate one additional parameter.

As demonstrated by a suite of extensive experiments in [50], the proper choice of CR can lead to successful optimization performance while a wrong choice may deteriorate the performance. In fact, good values of CR generally fall into a small range for a given problem, with which the algorithm can perform consistently well. Therefore, we consider gradually adjusting the range of CR values for a given problem according to previous CR values that have generated trial vectors successfully entering the next generation. Specifically, we assume that CR obeys a normal distribution with mean value CRm and standard deviation 0.1, denoted by $N(CRm, 0.1)$. In our preliminary work in [66], the same CRm value was used for all the trial vector generation strategies. However, it is possible that different strategies can perform well by using different ranges of CR values. Hence, it is reasonable to adapt the value of CRm with respect to each trial vector generation strategy. Without loss of generality, with respect to the k^{th} strategy, the value of CRm_k is initialized to 0.5. A set of CR values are randomly generated according to $N(CRm_k, 0.1)$ and then applied to those target vectors to which the k^{th} strategy is assigned. To adapt the crossover rate CR , we establish memories named $CRMemory_k$ to store those CR values with respect to the k^{th} strategy that have generated trial vectors successfully entering the next generation within the previous LP generations. Specifically, during the first LP generations, CR values with respect to k^{th} strategy are generated by $N(CRm_k, 0.1)$. At each generation after LP generations, the median value stored in $CRMemory_k$ will be calculated to overwrite CRm_k . Then, CR values can be generated according to $N(CRm_k, 0.1)$ when applying the k^{th} strategy. After evaluating the newly generated trial vectors, CR values

in $CRMemory_k$ that correspond to earlier generations will be replaced by promising CR values obtained at the current generation with respect to the k^{th} strategy.

By incorporating the above-mentioned trial vector generation strategy and control parameter adaptation schemes into the conventional DE framework, a Self-adaptive Differential Evolution (SaDE) algorithm is developed. In the SaDE algorithm, both trial vector generation strategies and their associated parameter values are gradually self-adapted by learning from their previous experiences of generating promising solutions. Consequently, a more suitable strategy along with its parameter setting can be determined adaptively to suit different phases of the search process. The algorithmic description of the SaDE is presented in Table 3-1.

Table 3-1: The algorithmic description of SaDE

<p>Step 1 Set the generation counter $G = 0$, and randomly initialize a population of NP individuals $\mathbf{P}_G = \{\mathbf{X}_{1,G}, \dots, \mathbf{X}_{NP,G}\}$ with $\mathbf{X}_{i,G} = \{x_{i,G}^1, \dots, x_{i,G}^D\}$, $i = 1, \dots, NP$ uniformly distributed in the range $[\mathbf{X}_{\min}, \mathbf{X}_{\max}]$, where $\mathbf{X}_{\min} = \{x_{\min}^1, \dots, x_{\min}^D\}$ and $\mathbf{X}_{\max} = \{x_{\max}^1, \dots, x_{\max}^D\}$. Initialize the median value of CR (CRm_k), strategy probability ($p_{k,G}$, $k = 1, \dots, K$, K is the no. of available strategies), learning period (LP)</p>
<p>Step 2 Evaluate the population</p>
<p>Step 3 WHILE stopping criterion is not satisfied DO</p>
<p>Step 3.1 Calculate strategy probability $p_{k,G}$ and update the <i>Success</i> and <i>Failure Memory</i> IF $G > LP$ For $k = 1$ to K Update the $p_{k,G}$ by equation (3-1) Remove $ns_{k,G-LP}$ and $nf_{k,G-LP}$ out of the <i>Success</i> and <i>Failure Memory</i> respectively. END</p>
<p>Step 3.2 Assign trial vector generation strategy and parameter to each target vector $\mathbf{X}_{i,G}$ /* Assign trial vector generation strategy */ Using stochastic universal sampling to select one strategy k for each target vector $\mathbf{X}_{i,G}$ /* Assign control parameter F */ FOR $i = 1$ to NP $F_i = \text{Normrnd}(0.5, 0.3)$ END FOR /* Assign control parameter CR */ IF $G \geq LP$</p>

```

    FOR  $k = 1$  to  $K$ ,  $CRm_k = \text{median}(CRMemory_k)$ , END FOR
  END IF
  FOR  $k = 1$  to  $K$ 
    FOR  $i = 1$  to  $NP$ 
       $CR_{k,i} = \text{Normrnd}(CRm_k, 0.1)$ 
      WHILE  $CR_{k,i} < 0$  or  $CR_{k,i} > 1$ 
         $CR_{k,i} = \text{Normrnd}(CRm_k, 0.1)$ 
      END
    END FOR
  END FOR
END FOR

```

Step 3.3 Generate a new population where each trial vector $U_{i,G}^k$ is generated according to associated trial vector generation strategy k and parameters F_i and $CR_{k,i}$ in Step 3.2.

Step 3.4 Randomly reinitialize the trial vector $U_{i,G}^k$ within the search space if any variable is outside its boundaries.

Step 3.5 Selection:

```

  FOR  $i = 1$  to  $NP$ 
    Evaluate the trial vector  $U_{i,G}^k$ 
    IF  $f(U_{i,G}^k) \leq f(X_{i,G})$ 
       $X_{i,G+1} = U_{i,G}^k$ ,  $f(X_{i,G+1}) = f(U_{i,G}^k)$ 
       $ns_{k,G} = ns_{k,G} + 1$ 
      Store  $CR_{k,i}$  into  $CRMemory_k$ 
      IF  $f(U_{i,G}^k) < f(X_{best,G})$ 
         $X_{best,G} = U_{i,G}^k$ ,  $f(X_{best,G}) = f(U_{i,G}^k)$ 
      END IF
    ELSE
       $nf_{k,G} = nf_{k,G} + 1$ 
    END IF
  END FOR
  Store  $ns_{k,G}$  and  $nf_{k,G}$  ( $k = 1, \dots, K$ ) into the Success and Failure Memory respectively.

```

Step 3.6 Increment the generation count $G = G + 1$

Step 4 END WHILE

3.3 Performance Evaluation of SaDE on Bound Constrained Optimization Problems

3.3.1 Introduction to the Test Functions

As discussed in [44], many benchmark numerical functions commonly used to evaluate and compare optimization algorithms may suffer from two problems. Firstly, the global optimum lies at the center of the search range. Secondly, the local optima lie along the coordinate axes or no linkage among the variables or dimensions exists. To solve these problems, we can shift or rotate the conventional benchmark functions. For benchmark functions suffering from the first problem, we may shift the global optimum to a random position so that the global optimum position has different numerical values for different dimensions, *i.e.* $F(\mathbf{x}) = f(\mathbf{x} - \mathbf{o}_{new} + \mathbf{o}_{old})$, where $F(\mathbf{x})$ is the new function, $f(\mathbf{x})$ is old function, $\mathbf{o}_{old}$ is the old global optimum and $\mathbf{o}_{new}$ is the new global optimum with different values for different dimensions and not lying at the center of the search range. For the second problem, we can rotate the function $F(\mathbf{x}) = f(\mathbf{M}\mathbf{x})$, where $\mathbf{M}$ is an orthogonal rotation matrix, to avoid local optima lying along the coordinate axes while retaining the properties of the test function. We hereby shift 9 commonly used benchmark functions (f_1 - f_5 , f_7 , f_9 , f_{11-12}), and further rotate three of them (f_6 , f_8 , f_{10}).

Two composition functions f_{13} and f_{14} are chosen from [44], which are constructed by using some basic benchmark functions to obtain more challenging problems. Gaussian function is used to combine the simple benchmark functions and blur the functions' structures. The composition functions are asymmetrical multimodal problems, with different properties in different areas. The detailed principle of constructing this class of functions is presented in [44], which are not repeated here.

Below, 14 test functions are listed, among which functions f_1 - f_4 are unimodal and functions f_5 - f_{14} are multimodal. These 14 test functions (f_1 - f_{14}) are dimension-wise scalable:

- 1) Shifted Sphere function

$$f_1(x) = \sum_{i=1}^D z_i^2, \quad \mathbf{z} = \mathbf{x} - \mathbf{o}, \quad \mathbf{o} = [o_1, o_2, \dots, o_D]: \text{the shifted global optimum.}$$

2) Shifted Schwefel's Problem 1.2

$$f_2(x) = \sum_{i=1}^D \left(\sum_{j=1}^i z_j \right)^2, \quad \mathbf{z} = \mathbf{x} - \mathbf{o}, \quad \mathbf{o} = [o_1, o_2, \dots, o_D]: \text{the shifted global}$$

optimum.

3) Rosenbrock's function

$$f_3(x) = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$$

4) Shifted Schwefel's Problem 1.2 with Noise in Function

$$f_4(x) = \left(\sum_{i=1}^D \left(\sum_{j=1}^i z_j \right)^2 \right) \left(1 + 0.4 |N(0,1)| \right), \quad \mathbf{z} = \mathbf{x} - \mathbf{o}, \quad \mathbf{o} = [o_1, o_2, \dots, o_D]: \text{the}$$

shifted global optimum.

5) Shifted Ackley's function

$$f_5(x) = -20 \exp(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D z_i^2}) - \exp\left(\frac{1}{D} \sum_{i=1}^D \cos(2\pi z_i)\right) + 20 + e, \quad \mathbf{z} = \mathbf{x} - \mathbf{o},$$

$\mathbf{o} = [o_1, o_2, \dots, o_D]: \text{the shifted global optimum.}$

6) Shifted Rotated Ackley's function

$$f_6(x) = -20 \exp(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D z_i^2}) - \exp\left(\frac{1}{D} \sum_{i=1}^D \cos(2\pi z_i)\right) + 20 + e,$$

$$\mathbf{z} = \mathbf{M}(\mathbf{x} - \mathbf{o}), \quad \text{cond}^l(\mathbf{M})=1$$

$\mathbf{o} = [o_1, o_2, \dots, o_D]: \text{the shifted global optimum.}$

7) Shifted Griewank's function

$$f_7(x) = \sum_{i=1}^D \frac{z_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1, \quad \mathbf{z} = \mathbf{x} - \mathbf{o}, \quad \mathbf{o} = [o_1, o_2, \dots, o_D]: \text{the}$$

shifted global optimum.

¹ $\text{Cond}(\mathbf{M})$ means the condition number of rotation matrix $\mathbf{M}$.

8) Shifted Rotated Griewank's function

$$f_8(x) = \sum_{i=1}^D \frac{z_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1, \quad \mathbf{z} = \mathbf{M}(\mathbf{x} - \mathbf{o}), \quad \text{cond}(\mathbf{M})=3$$

$\mathbf{o} = [o_1, o_2, \dots, o_D]$: the shifted global optimum.

9) Shifted Rastrigin's function

$$f_9(x) = \sum_{i=1}^D (z_i^2 - 10 \cos(2\pi z_i) + 10), \quad \mathbf{z} = \mathbf{x} - \mathbf{o}, \quad \mathbf{o} = [o_1, o_2, \dots, o_D]: \text{the shifted global optimum.}$$

10) Shifted Rotated Rastrigin's function

$$f_{10}(x) = \sum_{i=1}^D (z_i^2 - 10 \cos(2\pi z_i) + 10), \quad \mathbf{z} = \mathbf{M}(\mathbf{x} - \mathbf{o}), \quad \text{cond}(\mathbf{M})=2$$

$\mathbf{o} = [o_1, o_2, \dots, o_D]$: the shifted global optimum.

11) Shifted Non-continuous Rastrigin's function

$$f_{11}(x) = \sum_{i=1}^D (y_i^2 - 10 \cos(2\pi y_i) + 10), \quad y_i = \begin{cases} z_i & |z_i| < 1/2 \\ \text{round}(2z_i)/2 & |z_i| \geq 1/2 \end{cases} \text{ for } i = 1, 2, \dots, D, \quad \mathbf{z} = \mathbf{x} - \mathbf{o}, \quad \mathbf{o} = [o_1, o_2, \dots, o_D]: \text{the shifted global optimum.}$$

12) Schwefel's function

$$f_{12}(x) = 418.9829 \times D - \sum_{i=1}^D x_i \sin(|x_i|^{1/2})$$

13) Composition function 1 (CF1) in [44]

The function f_{13} (CF1) is composed by using ten sphere functions. The global optimum is easy to find once the global basin is located. The details of constructing such functions are presented in [44].

14) Composition function 6 (CF6) in [44]

The function f_{14} (CF6) is composed by using ten different benchmark functions, *i.e.* two rotated Rastrigin's functions, two rotated Weierstrass functions, two rotated Griewank's functions, two rotated Ackley's functions and two rotated Sphere functions.

To make our test suite more comprehensive, we also chose an additional set of 12 test functions from [87].

15) Schwefel's Problem 2.22

$$f_{15}(x) = \sum_{i=1}^D |x_i| + \prod_{i=1}^D |x_i|$$

16) Schwefel's Problem 2.21

$$f_{16}(x) = \max_i \{|x_i|, 1 \leq i \leq D\}$$

17) Generalized Penalized Function 1

$$\begin{aligned} f_{17}(x) &= \frac{\pi}{D} \left\{ 10 \sin^2(\pi y_1) + \sum_{i=1}^{D-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] + (y_D - 1)^2 \right\} \\ &+ \sum_{i=1}^D u(x_i, 10, 100, 4) \\ y_i &= 1 + \frac{1}{4}(x_i + 1) \\ u(x_i, 10, 100, 4) &= \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a \leq x_i \leq a, \\ k(-x_i - a)^m, & x_i < -a. \end{cases} \end{aligned}$$

18) Generalized Penalized Function 2

$$\begin{aligned} f_{18}(x) &= 0.1 \left\{ \sin^2(3\pi x_1) + \sum_{i=1}^{D-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] + (x_D - 1) [1 + \sin^2(2\pi x_D)] \right\} \\ &+ \sum_{i=1}^D u(x_i, 5, 100, 4) \end{aligned}$$

19) Kowalik's Function

$$f_{19}(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2$$

20) Six-Hump Camel-Back Function

$$f_{20}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$$

21) Branin Function

$$f_{21}(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos x_1 + 10$$

22) Hartman's Function 1

$$f_{22}(x) = -\sum_{i=1}^4 c_i \exp\left[-\sum_{j=1}^3 a_{ij}(x_j - p_{ij})^2\right]$$

23) Hartman's Function 2

$$f_{23}(x) = -\sum_{i=1}^4 c_i \exp\left[-\sum_{j=1}^6 a_{ij}(x_j - p_{ij})^2\right]$$

24) Shekel's Family

$$f(x) = -\sum_{i=1}^m \left[(x - a_i)^T (x - a_i) + c_i \right]^{-1} \text{ with } m=5, 7 \text{ and } 10 \text{ for } f_{24}(x), f_{25}(x) \text{ and } f_{26}(x).$$

$$f_{26}(x)$$

Table 3-2: Global optimum, search ranges and initialization ranges of the test functions

f	Dimension	Global optimum x^*	$f(x^*)$	Search Range	Initialization Range
f_1		o	0	$[-100, 100]^D$	$[-100, 100]^D$
f_2		o	0	$[-100, 100]^D$	$[-100, 100]^D$
f_3		$(1, 1, \dots, 1)$	0	$[-100, 100]^D$	$[-100, 100]^D$
f_4		o	0	$[-32, 32]^D$	$[-32, 32]^D$
f_5		o	0	$[-32, 32]^D$	$[-32, 32]^D$
f_6		o	0	$\Re$	$[0, 600]^D$
f_7	10 and 30	o	0	$\Re$	$[0, 600]^D$
f_8		o	0	$[-5, 5]^D$	$[-5, 5]^D$
f_9		o	0	$[-5, 5]^D$	$[-5, 5]^D$
f_{10}		o	0	$[-5, 5]^D$	$[-5, 5]^D$
f_{11}		$(420.96, \dots, 420.96)$	0	$[-500, 500]^D$	$[-500, 500]^D$
f_{12}		$(420.96, \dots, 420.96)$	0	$[-500, 500]^D$	$[-500, 500]^D$
f_{13}		o_I	0	$[-5, 5]^D$	$[-5, 5]^D$
f_{14}		o_I	0	$[-5, 5]^D$	$[-5, 5]^D$
f_{15}	30	$(0, \dots, 0)$	0	$[-10, 10]^D$	Same as search range
f_{16}	30	$(0, \dots, 0)$	0	$[-100, 100]^D$	
f_{17}	30	$(1, \dots, 1)$	0	$[-50, 50]^D$	
f_{18}	30	$(1, \dots, 1)$	0	$[-50, 50]^D$	
f_{19}	4	$(0.1928, 0.1908, 0.1231, 0.1358)$	0.0003075	$[-5, 5]^D$	
f_{20}	2	$(0.8983, -0.7126), (-0.08983, 0.7126)$	-1.0316285	$[-5, 5]^D$	

f_{21}	2	$(-3.142, 12.275), (3.142, 2.275), (9.425, 2.425)$	0.398	$[-5, 10] \times [0, 15]$
f_{22}	4	$(0.114, 0.556, 0.852)$	-3.86	$[0, 1]^D$
f_{23}	6	$(0.201, 0.150, 0.477, 0.275, 0.311, 0.657)$	-3.32	$[0, 1]^D$
f_{24}	4	$[4, 4, 4, 4]$	-10.2	$[0, 10]^D$
f_{25}	4	$[4, 4, 4, 4]$	-10.4	$[0, 10]^D$
f_{26}	4	$[4, 4, 4, 4]$	-10.5	$[0, 10]^D$

$\mathbf{o}$ is the shifted vector.
 $\mathbf{o}_1$ is the shifted vector for the first basic function in the Composition function.

3.3.2 Algorithms for Comparison

In order to investigate the performance of the SaDE algorithm compared with the original DE, we choose five DE variants:

- DE /rand/1/bin, $F=0.9$, $CR=0.1$
- DE /rand/1/bin, $F=0.9$, $CR=0.9$
- DE/rand/1/bin, $F=0.5$, $CR=0.3$
- DE/rand-to-best/1/bin, $F=0.5$, $CR=0.3$
- DE/rand-to-best/2/ bin, $F=0.5$, $CR=0.3$

Here ‘DE/rand/1/bin’ is chosen since it employs a most commonly used trial vector generation strategy, with three commonly suggested sets of control parameters: 1) $F=0.9$, $CR=0.1$; 2) $F=0.9$, $CR=0.9$; 3) $F=0.5$, $CR=0.3$ [58][40][45]. ‘DE/rand-to-best/1/bin’ and ‘DE/rand-to-best/2/bin’ employ more reliable trial vector generation strategies, and the control parameters are both set as $F=0.5$, $CR=0.3$ in our experiments [50][96].

We also chose three most representative adaptive DE variants proposed recently to compare with our proposed SaDE algorithm:

- Adaptive DE algorithm [12]
- SDE algorithm [52]
- jDE [46]

For functions $f_1 - f_{14}$, both 10-dimensional (10-D) and 30-dimensional (30-D) functions were tested. The maximum number of Function Evaluations (FEs) is set to 100,000 when solving 10-D problems, and 300,000 when solving 30-D counterpart. For the remaining functions $f_{15} - f_{26}$, the function dimensions are listed in Table 3-2, and the maximum number of FEs is set to 500,000 [87]. All experiments are run 30 times, independently. The population sizes are all 50 for both 10-dimensional and 30-dimensional functions.

3.3.3 Experimental Results and Discussions

Table 3-3, Table 3-4, Table 3-6 and Table 3-7 report the mean and standard deviation of function values as well as the success rates by applying the nine algorithms to optimize the 10-dimensional and 30-dimensional numerical functions $f_1 - f_{14}$, respectively. The best results are typed in bold. The success of an algorithm means that this algorithm can result in a function value no worse than the pre-specified optimal value, i.e. $f(x^*) + 1e-5$ for all problems with the number of FEs less than the pre-specified maximum number. The success rate is calculated as the number of successful runs divided by the total number of runs.

Figure 3- and Figure 3-4 illustrate the convergence characteristics in terms of the best fitness value of the median run of each algorithm for functions $f_1 - f_{14}$ with $D=10$. As the convergence graphs of the 30-dimensional problems are similar to their 10-dimensional counterparts, they are omitted here.

For the functions $f_{15} - f_{26}$, since most algorithms can locate the global optima with 100% success rate within the specified maximum FEs, it is unnecessary to present the mean and standard deviation of function values. Instead, to compare the convergence speed, we report the average Number of Function Evaluations (NFE) required to find the global optima when an algorithm solves the problem with 100% success rate.

1) Comparing SaDE with conventional DE

In this section, we intend to show how well the proposed SaDE algorithm performs when compared to the conventional DE.

With respect to the 10-dimensional $f_1 - f_{14}$, Table 3-3 shows that f_1 and f_5 are easily optimized by five DE variants and the SaDE algorithm with 100% success rate. Except for these two functions, the performances of DE/rand/1($CR=0.1$) and DE/rand/1($CR=0.9$) are almost contrary. For example, DE/rand/1/bin ($CR=0.9$) obtains better results on unimodal functions f_2 and f_3 where DE/rand/1/bin ($CR=0.1$) performs poorly. On the contrary, DE/rand/1/bin ($CR=0.1$) performs efficiently and robustly on multimodal functions f_7, f_9, f_{11} and f_{12} , while DE/rand/1/bin ($CR=0.9$) fails entirely on f_7, f_9 , and f_{11} , with only a 13% success rate on f_{12} . Therefore, different values of CR for DE/rand/1/bin demonstrate diverse performances on different problems. In general, DE/rand-to-best/1/bin and DE/rand-to-best/2/bin are more reliable than DE/rand/1/bin. However, they also entirely or partially fail in optimizing some problems where SaDE can locate the global optima in every run. On the whole, SaDE obtains a smaller mean value and a higher success rate than the five DE variants for all problems.

Furthermore, the convergence maps of DE/rand/1/bin ($F=0.9, CR=0.1$), DE/rand/1/bin ($F=0.9, CR=0.9$), DE/rand/1/bin ($F=0.5, CR=0.3$), DE/rand-to-best/1/bin ($F=0.5, CR=0.3$), DE/rand-to-best/2/bin ($F=0.5, CR=0.3$), and SaDE in Figure 3- show that the SaDE algorithm always converges faster than others on six problems ($f_2, f_3, f_4, f_8, f_{10}$, and f_{14}), while slightly slower than DE/rand-to-best/1/bin ($F=0.5, CR=0.3$) or DE/rand/1/bin ($F=0.5, CR=0.3$) on the remaining problems. It can be observed that with respect to f_9 DE/rand/1/bin ($CR=0.1$), DE/rand-to-best/1/bin ($F=0.5, CR=0.3$) and DE/rand/1/bin ($F=0.5, CR=0.3$) converge fast, followed by SaDE. This is because a small CR value ($CR=0.1$, or 0.3) is an effective value to optimize Rastrigin problem, while our SaDE, with an initial $CR=0.5$, need some generations to self-adapt the parameters to suitable values. This issue will be discussed in Section 3.4 .

For the 30-dimensional problems $f_1 - f_{14}$, DE/rand/1/bin ($CR=0.9$) has great difficulty in finding the global optima on all problems. DE/rand/1/bin ($CR=0.1$) yields 100% success rate only on $f_1, f_5, f_7, f_9, f_{11}$ and f_{12} . The SaDE algorithm performs much better with success rates of 100% on most problems, and 90% on f_3 where other DE variants totally fail in finding the global optima. When the dimension of variables was increased to 30, some problems such as functions f_4, f_{10} , and f_{14} become so difficult that

none of the six approaches could find the optimal solutions within the maximum FEs. Regarding the speed of different algorithms in reaching the optimal solutions, we further compare the NFEs on problems with 100% success rate in Table 3-5, and find that SaDE's convergence is the fastest among these algorithms on eight problems except f_7 .

For the test problems $f_{15} - f_{26}$, both SaDE and DE/rand/1/bin ($F=0.5$, $CR=0.3$) successfully solve all problems in each run, while SaDE shows an overall better convergence speed than DE/rand/1/bin ($F=0.5$, $CR=0.3$).

Table 3-3: Results of rand/1($F=0.9$, $CR=0.1$), rand/1($F=0.9$, $CR=0.9$), rand/1($F=0.5$, $CR=0.3$), rand-to-best/1, rand-to-best/2 and SaDE on $f_1 - f_{14}$ (10-D)

Algorithm	f_1 . Sphere			f_2 . Schwefel's Problem 1.2			f_3 . Rosenbrock		
	Mean	Std	Success Rate	Mean	Std	Success Rate	Mean	Std	Success Rate
DE/rand/1/bin ($F=0.9$ $CR=0.1$)	0	0	100%	8.89E-01 [†]	4.96E-01	0%	9.01E-01 [†]	7.94E-01	0%
DE/rand/1/bin ($F=0.9$ $CR=0.9$)	4.95E-13	5.27E-13	100%	1.44E-05 [†]	1.13E-05	43%	7.11E-03 [†]	2.74E-02	0%
DE/rand/1/bin ($F=0.5$ $CR=0.3$)	0	0	100%	9.63E-09	5.99E-09	100%	1.76E+00 [†]	1.54E+00	0%
DE/rand-to-best/1/bin ($F=0.5$ $CR=0.3$)	0	0	100%	0	0	100%	2.57E+00 [†]	1.86E+00	0%
DE/rand-to-best/2/bin ($F=0.5$ $CR=0.3$)	0	0	100%	9.45E-13	9.90E-13	100%	2.37E+00 [†]	2.23E+00	0%
SaDE	0	0	100%	0	0	100%	0	0	100%

[†]indicates that SaDE performs better than the other algorithm with 95% certainty by t -test.

[‡]means that corresponding algorithm is better than SaDE.

Table 3-3 (Cont.): Results of rand/1($F=0.9$, $CR=0.1$), rand/1($F=0.9$, $CR=0.9$), rand/1($F=0.5$, $CR=0.3$), rand-to-best/1, rand-to-best/2 and SaDE on $f_1 - f_{14}$ (10-D)

Algorithm	f_4 . Schwefel's Problem 1.2 with noise			f_5 . Ackley			f_6 . Rotated Ackley		
	Mean	Std	Success Rate	Mean	Std	Success Rate	Mean	Std	Success Rate
DE/rand/1/bin ($F=0.9$ $CR=0.1$)	2.41E+01 [†]	1.28E+01	0%	0	0	100%	3.81E-05 [†]	0.00013	90%
DE/rand/1/bin ($F=0.9$ $CR=0.9$)	2.42E-04 [†]	1.38E-04	0%	4.59E-07 [†]	2.41E-07	100%	6.86E-07 [†]	3.89E-07	100%
DE/rand/1/bin ($F=0.5$ $CR=0.3$)	5.42E-06	4.44E-06	83%	0	0	100%	3.32E-15	9.01E-16	100%
DE/rand-to-best/1/bin ($F=0.5$ $CR=0.3$)	0	0	100%	4.97E-15	1.77E-15	100%	4.26E-15	1.45E-15	100%
DE/rand-to-best/2/bin ($F=0.5$ $CR=0.3$)	1.04E-08	1.20E-08	100%	3.55E-15	1.87E-15	100%	3.55E-15	0	100%
SaDE	0	0	100%	0	0	100%	0	0	100%

Chapter 3: SaDE

Table 3-3 (Cont.): Results of rand/1(F=0.9, CR=0.1), rand/1(F=0.9, CR=0.9), rand/1(F=0.5,CR=0.3), rand-to-best/1, rand-to-best/2 and SaDE on $f_1 - f_{14}$ (10-D)

Algorithm	f_7 . Griewank			f_8 . Rotated Griewank			f_9 . Rastrigin		
	Mean	Std	Success Rate	Mean	Std	Success Rate	Mean	Std	Success Rate
DE/rand/1/bin (F=0.9 CR=0.1)	0	0	100%	1.22E-01 [†]	2.77E-02	0%	0	0	100%
DE/rand/1/bin (F=0.9 CR=0.9)	3.05E-01 [†]	2.02E-01	0%	2.41E-01 [†]	2.00E-01	0%	8.71E+00 [†]	5.53E+00	0%
DE/rand/1/bin (F=0.5 CR=0.3)	0	0	100%	1.60E-01 [†]	3.75E-02	0%	0	0	100%
DE/rand-to-best/1/bin (F=0.5 CR=0.3)	4.67E-03 [†]	8.13E-03	70%	2.91E-01 [†]	3.14E-01	0%	6.63E-02	2.52E-01	93%
DE/rand-to-best/2/bin (F=0.5 CR=0.3)	0	0	100%	1.44E-01 [†]	3.97E-02	0%	0	0	100%
SaDE	0	0	100%	1.37E-02	1.18E-02	20%	0	0	100%

Table 3-3 (Cont.): Results of rand/1(F=0.9, CR=0.1), rand/1(F=0.9, CR=0.9), rand/1(F=0.5,CR=0.3), rand-to-best/1, rand-to-best/2 and SaDE on $f_1 - f_{14}$ (10-D)

Algorithm	f_{10} . Rotated Rastrigin			f_{11} . Non-continuous Rastrigin			f_{12} . Schwefel		
	Mean	Std	Success Rate	Mean	Std	Success Rate	Mean	Std	Success Rate
DE/rand/1/bin (F=0.9 CR=0.1)	1.33E+01 [†]	3.00E+00	0%	0	0	100%	0	0	100%
DE/rand/1/bin (F=0.9 CR=0.9)	1.63E+01 [†]	1.10E+01	0%	8.20E+00 [†]	3.37E+00	0%	2.82E+00 [†]	1.41E+01	13%
DE/rand/1/bin (F=0.5 CR=0.3)	1.65E+01 [†]	2.99E+00	0%	0	0	100%	0	0	100%
DE/rand-to-best/1/bin (F=0.5 CR=0.3)	1.00E+01 [†]	2.32E+00	0%	1.33E-01 [†]	3.46E-01	87%	1.18E+01 [†]	3.61E+01	90%
DE/rand-to-best/2/bin (F=0.5 CR=0.3)	1.63E+01 [†]	3.36E+00	0%	0	0	100%	6.06E-14	2.31E-13	100%
SaDE	3.80E+00	1.35E+00	0%	0	0	100%	0	0	100%

Table 3-3 (Cont.): Results of rand/1(F=0.9, CR=0.1), rand/1(F=0.9, CR=0.9), rand/1(F=0.5,CR=0.3), rand-to-best/1, rand-to-best/2 and SaDE on $f_1 - f_{14}$ (10-D)

Algorithm	f_{13} . Composition Function1			f_{14} . Composition Function6		
	Mean	Std	Success Rate	Mean	Std	Success Rate
DE/rand/1/bin (F=0.9 CR=0.1)	3.41E-02 [†]	5.98E-02	0%	6.62E+00 [†]	1.33E+00	0%
DE/rand/1/bin (F=0.9 CR=0.9)	4.27E-13	3.74E-13	100%	2.17E-01	5.64E-01	53%
DE/rand/1/bin (F=0.5 CR=0.3)	6.50E-01	3.56E+00	97%	9.48E+00 [†]	2.49E+01	0%
DE/rand-to-best/1/bin (F=0.5 CR=0.3)	1.83E+01 [†]	4.62E+01	80%	2.36E+01 [†]	4.31E+01	17%
DE/rand-to-best/2/bin (F=0.5 CR=0.3)	1.02E-09	3.95E-09	100%	1.37E+00 [†]	8.26E-01	0%
SaDE	0	0	100%	2.54E-01	5.21E-01	80%

Chapter 3: SaDE

Table 3-4: Results of rand/1(F=0.9, CR=0.1), rand/1(F=0.9, CR=0.9), rand/1(F=0.5,CR=0.3), rand-to-best/1, rand-to-best/2 and SaDE on $f_1 - f_{14}$ (30-D)

Algorithm	f_1 . Sphere			f_2 . Schwefel's Problem1.2			f_3 . Rosenbrock		
	Mean	Std	Success Rate	Mean	Std	Success Rate	Mean	Std	Success Rate
DE/rand/1/bin (F=0.9 CR=0.1)	0	0	100%	3.62E+03 [†]	8.22E+02	0%	3.39E+01 [†]	1.51E+01	0%
DE/rand/1/bin (F=0.9 CR=0.9)	4.50E-02 [†]	6.15E-02	0%	1.73E+03 [†]	1.40E+03	0%	1.04E+02 [†]	6.25E+01	0%
DE/rand/1/bin (F=0.5 CR=0.3)	0	0	100%	1.38E+03 [†]	2.53E+02	0%	2.14E+01 [†]	1.98E+00	0%
DE/rand-to-best/1/bin (F=0.5 CR=0.3)	1.90E-07	1.04E-06	97%	1.92E+01 [†]	2.27E+01	0%	6.37E+01 [†]	4.01E+01	0%
DE/rand-to-best/2/bin (F=0.5 CR=0.3)	0	0	100%	1.04E+02 [†]	8.25E+01	0%	2.08E+01 [†]	1.19E+01	0%
SaDE	0	0	100%	0	0	100%	3.99E-01	1.22E+00	90%

Table 3-4 (Cont.): Results of rand/1(F=0.9, CR=0.1), rand/1(F=0.9, CR=0.9), rand/1(F=0.5,CR=0.3), rand-to-best/1, rand-to-best/2 and SaDE on $f_1 - f_{14}$ (30-D)

Algorithm	f_4 . Schwefel's Problem 1.2 with noise			f_5 . Ackley			f_6 . Rotated Ackley		
	Mean	Std	Success Rate	Mean	Std	Success Rate	Mean	Std	Success Rate
DE/rand/1/bin (F=0.9 CR=0.1)	1.24E+04 [†]	2.15E+03	0%	0	0	100%	3.81E-05 [†]	0.00013	90%
DE/rand/1/bin (F=0.9 CR=0.9)	8.98E+03 [†]	5.74E+03	0%	3.86E-02 [†]	2.18E-02	0%	7.64E-02 [†]	5.11E-02	0%
DE/rand/1/bin (F=0.5 CR=0.3)	5.19E+03 [†]	1.24E+03	0%	4.03E-15	1.23E-15	100%	3.67E-15	6.49E-16	100%
DE/rand-to-best/1/bin (F=0.5 CR=0.3)	2.36E+00	5.47E+00	0%	3.10E-02 [†]	1.70E-01	93%	4.82E-03 [†]	2.64E-02	97%
DE/rand-to-best/2/bin (F=0.5 CR=0.3)	1.51E+03 [†]	2.07E+02	0%	7.58E-15	1.80E-15	100%	7.34E-15	1.30E-15	100%
SaDE	3.37E+00	1.37E+01	0%	0	0	100%	0	0	100%

Table 3-4 (Cont.): Results of rand/1(F=0.9, CR=0.1), rand/1(F=0.9, CR=0.9), rand/1(F=0.5,CR=0.3), rand-to-best/1, rand-to-best/2 and SaDE on $f_1 - f_{14}$ (30-D)

Algorithm	f_7 . Griewank			f_8 . Rotated Griewank			f_9 . Rastrigin		
	Mean	Std	Success Rate	Mean	Std	Success Rate	Mean	Std	Success Rate
DE/rand/1/bin (F=0.9 CR=0.1)	0 [†]	0	100%	9.12E-02 [†]	3.08E-02	0%	0	0	100%
DE/rand/1/bin (F=0.9 CR=0.9)	1.52E-01 [†]	1.15E-01	0%	9.01E-01 [†]	1.40E-01	0%	8.54E+01 [†]	3.30E+01	0%
DE/rand/1/bin (F=0.5 CR=0.3)	0 [†]	0	100%	2.24E-05 [†]	1.19E-04	93%	3.10E+01 [†]	3.24E+00	0%
DE/rand-to-best/1/bin (F=0.5 CR=0.3)	1.08E+01	1.00E+01	0%	1.82E+02 [†]	5.47E+01	0%	9.58E+00 [†]	3.88E+00	0%
DE/rand-to-best/2/bin (F=0.5 CR=0.3)	0 [†]	0	100%	3.97E-03	1.85E-02	37%	4.03E+01 [†]	3.73E+00	0%
SaDE	2.38E-03	5.03E-03	80%	8.54E-03	9.09E-03	40%	0	0	100%

Chapter 3: SaDE

Table 3-4 (Cont.): Results of rand/1(F=0.9, CR=0.1), rand/1(F=0.9, CR=0.9), rand/1(F=0.5,CR=0.3), rand-to-best/1, rand-to-best/2 and SaDE on $f_1 - f_{14}$ (30-D)

Algorithm	f_{10} . Rotated Rastrigin			f_{11} . Non-continuous Rastrigin			f_{12} . Schwefel		
	Mean	Std	Success Rate	Mean	Std	Success Rate	Mean	Std	Success Rate
DE/rand/1/bin (F=0.9 CR=0.1)	1.68E+02 [†]	1.43E+01	0%	0	0	100%	0	0	100%
DE/rand/1/bin (F=0.9 CR=0.9)	2.45E+02 [†]	2.20E+01	0%	6.93E+01 [†]	2.37E+01	0%	4.98E+03 [†]	1.75E+03	0%
DE/rand/1/bin (F=0.5 CR=0.3)	1.87E+02 [†]	1.09E+01	0%	2.88E+01 [†]	1.94E+00	0%	0	0	100%
DE/rand-to-best/1/bin (F=0.5 CR=0.3)	1.44E+02 [†]	1.09E+01	0%	1.18E+01 [†]	2.70E+00	0%	4.18E+02 [†]	1.75E+02	3%
DE/rand-to-best/2/bin (F=0.5 CR=0.3)	1.88E+02 [†]	7.15E+00	0%	3.17E+01 [†]	2.25E+00	0%	2.48E+03 [†]	4.01E+02	0%
SaDE	1.67E+01	5.26E+00	0%	0	0	100%	0	0	100%

Table 3-4 (Cont.): Results of rand/1(F=0.9, CR=0.1), rand/1(F=0.9, CR=0.9), rand/1(F=0.5,CR=0.3), rand-to-best/1, rand-to-best/2 and SaDE on $f_1 - f_{14}$ (30-D)

Algorithm	f_{13} . Composition Function1			f_{14} . Composition Function6		
	Mean	Std	Success Rate	Mean	Std	Success Rate
DE/rand/1/bin (F=0.9 CR=0.1)	3.78E-03 [†]	1.15E-02	47%	1.33E+01 [†]	3.08E+00	0%
DE/rand/1/bin (F=0.9 CR=0.9)	5.06E-03 [†]	5.54E-03	0%	1.56E+01 [†]	5.46E+00	0%
DE/rand/1/bin (F=0.5 CR=0.3)	4.49E-06	4.62E-06	100%	1.15E+01 [†]	1.54E+00	0%
DE/rand-to-best/1/bin (F=0.5 CR=0.3)	5.00E+01 [†]	8.20E+01	53%	2.86E+01 [†]	7.69E+01	0%
DE/rand-to-best/2/bin (F=0.5 CR=0.3)	2.17E-09	1.19E-08	100%	1.22E+01 [†]	2.99E+00	0%
SaDE	0	0	100%	2.17E+00	9.23E-01	0%

(1) f_1 : Sphere

(2) f_2 : Schwefel's 1.2

(3) f_3 : Rosenbrock

(4) f_4 : Schwefel's Problem 1.2 with Noise in Function

(5) f_5 : Ackley

(6) f_6 : Rotated Ackley

(7) f_7 : Griewank

(8) f_8 : Rotated Griewank

(11) f_{11} : Non-continuous Rastrigin

(12) f_{12} : Schwefel

(13) f_{13} : Composition Function 1

(14) f_{14} : Composition Function 6

Figure 3-3: The median convergence characteristics of rand/1 CR=0.1, rand/1 CR=0.9, rand/1 F=0.5,CR=0.3, rand-to-best/1, rand-to-best/2 and SaDE on f_1 - f_{14} (10-D)

Table 3-5: Comparison of rand/1(F=0.9 CR=0.1), rand/1(F=0.9 CR=0.9), rand/1(F=0.5,CR=0.3),

rand-to-best/1, rand-to-best/2 and SaDE (The best Results for each case are highlighted in Boldface)

	rand/1/bin (F=0.9 CR=0.1)		rand/1/bin (F=0.9 CR=0.9)		rand/1/bin (F=0.5 CR=0.3)		rand-to-best/1/bin (F=0.5 CR=0.3)		rand-to-best/2/bin (F=0.5 CR=0.3)		SaDE	
10D	NFE	SR	NFE	SR	NFE	SR	NFE	SR	NFE	SR	NFE	SR
f_1	16770	100%	53298	100%	10291	100%	6318	100%	10058	100%	8375	100%
f_2	-	0%	-	43%	72436	100%	23383	100%	53658	100%	14867	100%
f_3	-	0%	-	0%	-	0%	-	0%	-	0%	42446	100%
f_4	-	0%	-	0%	-	83%	30925	100%	71278	100%	15754	100%
f_5	25335	100%	82919	100%	15157	100%	9436	100%	15045	100%	12123	100%
f_6	74601	90%	85272	100%	16682	100%	9923	100%	16980	100%	12244	100%
f_7	41247	100%	-	0%	29961	100%	-	70%	59205	100%	35393	100%
f_9	19200	100%	-	0%	23155	100%	-	93%	30621	100%	23799	100%
f_{11}	20898	100%	-	0%	29559	100%	-	87%	46592	100%	26945	100%
f_{12}	19094	100%	-	13%	14698	100%	-	90%	33091	100%	16663	100%
f_{13}	-	0%	53131	100%	-	97%	-	80%	38290	100%	9740	100%
30D												
f_1	66339	100%	-	0%	34687	100%	-	97%	31470	100%	20184	100%
f_2	-	0%	-	0%	-	0%	-	0%	-	0%	118743	100%
f_5	92421	100%	-	0%	49822	100%	-	93%	45948	100%	26953	100%
f_6	-	0%	-	0%	55108	100%	-	97%	49961	100%	33014	100%
f_7	80741	100%	-	0%	39436	100%	-	0%	41314	100%	-	80%
f_9	90391	100%	-	0%	-	0%	-	0%	-	0%	58723	100%
f_{11}	108406	100%	-	0%	-	0%	-	0%	-	0%	77920	100%
f_{12}	78056	100%	-	0%	73756	100%	-	3%	-	0%	44283	100%
f_{13}	-	47%	-	0%	34012	100%	-	53%	38386	100%	19031	100%
f_{15}	80944	100%	-	0%	39460	100%	41729	100%	18617	100%	25137	100%
f_{16}	-	0%	-	0%	149511	100%	112445	100%	-	0%	88934	100%
f_{17}	62766	100%	-	90%	32420	100%	32442	100%	14289	100%	18742	100%
f_{18}	62388	100%	-	87%	31346	100%	28690	100%	-	93%	19390	100%
f_{19}	-	90%	7782	100%	31121	100%	33900	100%	-	93%	6426	100%
f_{20}	2744	100%	2628	100%	2178	100%	2086	100%	1416	100%	2076	100%
f_{21}	3659	100%	2713	100%	2246	100%	-	97%	1593	100%	2614	100%
f_{22}	1314	100%	996	100%	926	100%	672	100%	1004	100%	802	100%
f_{23}	5656	100%	-	60%	3970	100%	-	67%	4759	100%	3080	100%
f_{24}	31615	100%	7720	100%	10468	100%	-	90%	12381	100%	4947	100%
f_{25}	35503	100%	6596	100%	8653	100%	5085	100%	12921	100%	4173	100%
f_{26}	33905	100%	6974	100%	8742	100%	4853	100%	14933	100%	4267	100%

2) Comparing SaDE with other adaptive DE variants

In this section, we compare the SaDE with three other adaptive DE variants: Adaptive Differential Evolution (ADE) [12], Species-based Differential Evolution (SDE) [52] and jDE [46] on 10-dimensional and 30-dimensional problems $f_1 - f_{14}$.

For 10-dimensional problems, the best results of uni-modal functions f_1 – f_4 are obtained by SaDE and jDE, where SaDE demonstrated a slight superiority on efficiency, which can be observed from the convergence graphs (Figure 3-4 (1)–(4)). Among the multi-modal functions, f_5 Ackley and f_6 rotated Ackley were easily solved by all the adaptive DE variants with 100% success rate. For f_7 Griewank and f_9 Rastrigin, before rotation, SaDE, ADE and jDE had high success rates as shown in Table 3-6. After rotation (f_8 and f_{10}), only SaDE and jDE successfully found the global optimum on f_8 in some run, and SaDE offered a higher success rate than jDE. f_{10} is that no algorithm could find the global optimum. For the last two composition problems f_{13} and f_{14} , SaDE was able to locate the optimal solutions with smaller standard deviations and higher success rates, demonstrating better efficiency and stability than the other three algorithms.

From the results of 30-dimensional problems shown in Table 3-7, we can observe that the algorithms achieved similar ranking as in the 10-dimensional problems, and were not very successful in optimizing f_4 , f_{10} and f_{14} . However, with respect to the means and standard deviations, SaDE obtained smallest values on f_{10} and f_{14} , and the 2nd smallest values on f_4 where jDE got the smallest values.

Table 3-6: Results of SaDE, ADE, SDE and jDE on f_1 - f_{14} (10-D)

Algorithm	f_1 . Sphere			f_2 . Schwefel's Problem1.2			f_3 . Rosenbrock		
	Mean	Std	Success Rate	Mean	Std	Success Rate	Mean	Std	Success Rate
SaDE	0	0	100%	0	0	100%	0	0	100%
ADE(Zaharie)	0	0	100%	1.44E-04 [†]	2.48E-04	3%	1.56E+00 [†]	2.64E+00	0%
SDE	0	0	100%	0	0	100%	2.05E+00 [†]	1.68E+00	0%
jDE	0	0	100%	0	0	100%	1.34E-13	7.32E-13	100%

[†]indicates that SaDE performs better than the other algorithm with 95% certainty by *t*-test.

[‡]means that corresponding algorithm is better than SaDE.

Table 3-6 (Cont.): Results of SaDE, ADE, SDE and jDE on f_1 - f_{14} (10-D)

Algorithm	f_4 . Schwefel's Problem 1.2 with noise			f_5 . Ackley			f_6 . Rotated Ackley		
	Mean	Std	Success Rate	Mean	Std	Success Rate	Mean	Std	Success Rate
SaDE	0	0	100%	0	0	100%	0	0	100%
ADE(Zaharie)	7.00E-02 [†]	5.84E-02	0%	0	0	100%	0	0	100%
SDE	0	0	100%	0	0	100%	0	0	100%
jDE	0	0	100%	0	0	100%	0	0	100%

Chapter 3: SaDE

Table 3-6 (Cont.): Results of SaDE, ADE, SDE and jDE on $f_1 - f_{14}$ (10-D)

Algorithm	f_7 . Griewank			f_8 . Rotated Griewank			f_9 . Rastrigin		
	Mean	Std	Success Rate	Mean	Std	Success Rate	Mean	Std	Success Rate
SaDE	0	0	100%	1.37E-02	1.18E-02	20%	0	0	100%
ADE(Zaharie)	2.55E-07	1.40E-06	100%	7.93E-02 [†]	4.24E-02	0%	0	0	100%
SDE	7.39E-03 [†]	7.59E-03	40%	3.81E-02 [†]	3.06E-02	0%	6.96E-01 [†]	8.72E-01	50%
jDE	5.75E-04	2.21E-03	93%	2.26E-02 [†]	1.77E-02	7%	0	0	100%

Table 3-6 (Cont.): Results of SaDE, ADE, SDE and jDE on $f_1 - f_{14}$ (10-D)

Algorithm	f_{10} . Rotated Rastrigin			f_{11} . Non-continuous Rastrigin			f_{12} . Schwefel		
	Mean	Std	Success Rate	Mean	Std	Success Rate	Mean	Std	Success Rate
SaDE	3.80E+00	1.35E+00	0%	0	0	100%	0	0	100%
ADE(Zaharie)	9.41E+00 [†]	2.20E+00	0%	0	0	100%	0	0	100%
SDE	7.79E+00 [†]	3.18E+00	0%	1.22E+00 [†]	9.99E-01	27%	2.37E+01 [†]	4.82E+01	80%
jDE	5.78E+00 [†]	2.10E+00	0%	0	0	100%	0	0	100%

Table 3-6 (Cont.): Results of SaDE, ADE, SDE and jDE on $f_1 - f_{14}$ (10-D)

Algorithm	f_{13} . Composition Function1			f_{14} . Composition Function6		
	Mean	Std	Success Rate	Mean	Std	Success Rate
SaDE	0	0	100%	2.54E-01	5.21E-01	80%
ADE(Zaharie)	1.48E-01	5.97E-01	90%	5.87E+00 [†]	4.85E+00	0%
SDE	3.00E+01 [†]	4.66E+01	70%	8.24E+00 [†]	2.50E+01	23%
jDE	1.33E+01 [†]	3.46E+01	87%	1.27E+00 [†]	3.20E+00	53%

Table 3-7: Results of SaDE, ADE, SDE and jDE on $f_1 - f_{14}$ (30-D)

Algorithm	f_1 . Sphere			f_2 . Schwefel's Problem1.2			f_3 . Rosenbrock		
	Mean	Std	Success Rate	Mean	Std	Success Rate	Mean	Std	Success Rate
SaDE	0	0	100%	0	0	100%	3.99E-01	1.22E+00	90%
ADE(Zaharie)	0	0	100%	3.04E+02 [†]	6.86E+01	0%	4.69E+01 [†]	2.64E+01	0%
SDE	4.56E-01 [†]	2.08E+00	50%	1.58E+00 [†]	4.48E+00	0%	7.73E+03 [†]	3.27E+04	0%
jDE	0	0	100%	8.91E-11	1.27E-10	100%	5.77E-01	1.38E+00	40%

Table 3-7(Cont.): Results of SaDE, ADE, SDE and jDE on $f_1 - f_{14}$ (30-D)

Algorithm	f_4 . Schwefel's Problem 1.2 with noise			f_5 . Ackley			f_6 . Rotated Ackley		
	Mean	Std	Success Rate	Mean	Std	Success Rate	Mean	Std	Success Rate
SaDE	3.37E+00	1.37E+01	0%	0	0	100%	0	0	100%
ADE(Zaharie)	6.75E+04 [†]	1.02E+04	0%	0	0	100%	0	0	100%

Chapter 3: SaDE

SDE	4.67E+02 [†]	5.09E+02	0%	2.19E-01 [†]	3.87E-01	40%	1.01E-01 [†]	3.04E-01	63%
jDE	2.15E-01[‡]	4.91E-01	0%	0	0	100%	0	0	100%

Table 3-7 (Cont.): Results of SaDE, ADE, SDE and jDE on $f_1 - f_{14}$ (30-D)

Algorithm	f_7 . Griewank			f_8 . Rotated Griewank			f_9 . Rastrigin		
	Mean	Std	Success Rate	Mean	Std	Success Rate	Mean	Std	Success Rate
SaDE	2.38E-03	5.03E-03	80%	8.54E-03	9.09E-03	40%	0	0	100%
ADE(Zaharie)	0[‡]	0	100%	2.93E-03	5.65E-03	10%	2.32E-01 [†]	5.01E-01	80%
SDE	1.59E+00 [†]	2.23E+00	13%	1.39E+00 [†]	4.24E+00	13%	1.09E+01 [†]	4.23E+00	0%
jDE	0[‡]	0	100%	5.17E-03	6.64E-03	57%	0	0	100%

Table 3-7 (Cont.): Results of SaDE, ADE, SDE and jDE on $f_1 - f_{14}$ (30-D)

Algorithm	f_{10} . Rotated Rastrigin			f_{11} . Non-continuous Rastrigin			f_{12} . Schwefel		
	Mean	Std	Success Rate	Mean	Std	Success Rate	Mean	Std	Success Rate
SaDE	1.67E+01	5.26E+00	0%	0	0	100%	0	0	100%
ADE(Zaharie)	1.21E+02 [†]	1.28E+01	0%	0	0	100%	3.95E+00	2.16E+01	97%
SDE	3.63E+01 [†]	6.78E+00	0%	1.56E+01 [†]	3.52E+00	0%	4.68E+02 [†]	2.08E+02	0%
jDE	3.65E+01 [†]	8.29E+00	0%	6.67E-02	2.54E-01	93%	6.32E+01 [†]	1.07E+02	63%

Table 3-7 (Cont.): Results of SaDE, ADE, SDE and jDE on $f_1 - f_{14}$ (30-D)

Algorithm	f_{13} . Composition Function1			f_{14} . Composition Function6		
	Mean	Std	Success Rate	Mean	Std	Success Rate
SaDE	0	0	100%	2.17E+00	9.23E-01	0%
ADE(Zaharie)	1.67E-03	9.13E-03	97%	4.18E+00 [†]	1.41E+00	0%
SDE	1.29E-01 [†]	3.62E-01	67%	1.16E+01 [†]	1.86E+01	0%
jDE	0	0	100%	1.24E+01 [†]	3.06E+01	0%

(1) f_1 : Sphere

(2) f_2 : Schwefel's 1.2

(3) f_3 : Rosenbrock

(4) f_4 : Schwefel's Problem 1.2 with Noise in Function

(5) f_5 : Ackley

(6) f_6 : Rotated Ackley

(7) f_7 : Griewank

(8) f_8 : Rotated Griewank

(9) f_9 : Rastrigin

(10) f_{10} : Rotated Rastrigin

(11) f_{11} : Non-continuous Rastrigin

(12) f_{12} : Schwefel

(13) f_{13} : Composition Function1

(14) f_{14} : Composition Function6

Figure 3-4 The median convergence characteristics of SaDE, ADE (Zaharie), SDE and jDE on 10-D test functions f_1 - f_{14}

3) Overall Performance Comparison

In this section, we intend to further compare the overall performances of nine algorithms on all functions by plotting empirical distribution of normalized success performance [123]. The success performance is defined as: Success Performance (SP) = $\text{mean (FEs for successful runs)} * (\# \text{ of total runs}) / (\# \text{ of successful runs})$.

We first calculated the success performance of nine algorithms on each test function, and then normalized the SP by dividing all SP's by the SP of the best algorithm on the respective function. Results of all functions are used where at least one algorithm was successful at least once. Therefore, for 10-dimensional f_1 - f_{14} problems, we plot the results of all functions except f_{10} , and exclude functions f_4 , f_{10} , and f_{14} for 30-dimensional f_1 - f_{14} problems. The resulting test problems f_{15} - f_{26} are also plotted. Small values of SP and large values of the empirical distribution in graphs are preferable. The first one that reaches the top of the graph will be regarded as the best algorithm.

From Figure 3-5, we can observe that SaDE outperforms other approaches on overall performance on 36 (=40-4) of the test problems.

(a)

Figure 3-5: Empirical distribution of normalized success performance on all 26 test problems

3.4 Analysis of self-adaptation property of SaDE

3.4.1 Self-adaptation of crossover probability

Usually when DE solves problems, there is no specific value of CR or F which is effective in all search phases. Instead possibly several combinations of different CR and F values can be effective in different search phases. However, Rosenbrock and Rastrigin functions are exceptions to the rule. From the experiments of DE/rand/1/bin on these two functions, we know that usually DE/rand/1/bin with a large CR could obtain a good result on Rosenbrock function and a small CR is beneficial to Rastrigin function, which could also be concluded from comparing the results of DE/rand/1/bin $CR=0.1$ and $CR=0.9$. As we know the suitable CR values for Rosenbrock and Rastrigin

functions, we can observe the variation of CR values in SaDE algorithm to check whether the self-adaptation of CR is effective.

Since the CR value in SaDE is mostly depended on the mean value CRm of normal distribution, we plot the CRm value when the SaDE algorithm optimized Rosenbrock and Rastrigin functions as the generation increases in Figure 3-6. We found that for Rosenbrock function, the CRm values of strategy ‘rand/1/bin’, ‘rand-to-best/2/bin’ and ‘rand/2/bin’ keep increasing during evolution, as we expected. On the contrary, for Rastrigin function, the CRm values of all three strategies keep decreasing during evolution. Therefore, we can say that our proposed SaDE algorithm self-adapts the crossover probability effectively.

(1) Rosenbrock function

(2) Rastrigin function

Figure 3-6: Self-adaptation characteristics of *CRm* on Rosenbrock and Rastrigin function (10-D)

3.4.2 Self-adaptation of trial vector generation strategy

To investigate the self-adaptive selection of trial vector generation strategy in SaDE algorithm, we plot the variations of the four different strategies’ probabilities as the evolution progresses. As we know that the strategy ‘current-to-rand’ is rotationally invariant [48] and has superior performance on rotated problem as stated in section 2.4.2 , this strategy should occupy more proportion if it yields good results when dealing with rotated problems. As shown in Figure 3-7, for Griewank’s function, the strategy ‘current-to-rand’ occupied a very small proportion during the whole evolution progress. At the beginning, we assigned each strategy an equal probability. Since the strategy ‘current-to-rand’ could not yield better results, our self-adaptive mechanism of learning is unlikely to choose the strategy ‘current-to-rand’ in next few generations. As this strategy always performed badly, its proportion almost lowered to 0. The individuals were mostly mutated by the other three strategies that always occupy proportions above 0.25 as shown in Figure 3-7 After 800 generations, the algorithm

converged, and these four strategies performed competitively so as to all occupy around 0.25 proportion. On the contrary, after we rotated the Griewank's function, the strategy 'current-to-rand' demonstrated a good performance at the beginning so as to occupy a higher proportion in the following 300 generations. During the 300th to 550th generations, the proportion of strategy 'current-to-rand' slightly decreased under 0.25 since fewer good solutions are found by this strategy during this stage of evolution. After the 550th generation, strategy 'current-to-rand' occupied a higher proportion again till the algorithm converged and four strategies were almost stable.

(1) Griewank

(2) Rotated Griewank

Figure 3-7: Self-adaptation characteristics of strategy (10-D)

3.5 Learning period parameter study

In SaDE, both trial vector generation strategies and CR values are self-adapted by learning from their previous experiences in generating promising solutions. Whether the strategy and CR value are beneficial should not be determined solely based on one generation result. It is necessary to evaluate the consistency of good performance (i.e. generating promising solutions) in several generations. Therefore, the performances of the strategy and CR value within several generations are recorded. The Learning Period (LP) is the number of generations over which the information is recorded. It controls how much historical information will be collected to adapt the strategy and CR value. If the learning period is too short, the information collected may be sensitive and biased to one generation result. On the contrary, if the learning period is too long, the adaption process will be slowed down. However, the performance of SaDE is supposed to be insensitive when setting LP as various values within a reasonable range. To justify this observation, we design five scenarios by setting $LP = 20$ as the

starting point, and increase by 10 generations until $LP = 60$. Here LP is increased by 10 generations, and not by 1 or 2 generations, since it is insensitive to minor change. Thus five scenarios were investigated, $LP = 20, 30, 40, 50, 60$.

We use 10-dimensional test functions $f_1 - f_{14}$ to investigate the impact of this parameter on the SaDE algorithm. The SaDE algorithm runs 30 times on each function with five different learning periods LP of 20, 30, 40, 50, and 60. Since functions f_{1-7}, f_9 , and f_{11-13} are solved with 100% success rate with 5 different LP values, we omit the results of the mean and standard deviation for these functions, and only the results of f_8, f_{10} and f_{14} are shown in Table 3-8. However, for functions f_{1-7}, f_9 , and f_{11-13} , we investigate the influence of LP 's on convergence speed of the SaDE algorithm, by comparing the FEs that the SaDE algorithm cost to obtain the global optima with different LP s.

Figure 3-8 shows the box plots of FEs that the SaDE algorithm cost to obtain the global optima with five different LP s. The box has lines at the lower quartile, median, and upper quartile values. The whiskers are lines extending from each end of the box to show the extent of the rest of the data. Outliers are data with values beyond the ends of the whiskers. If there is no data outside the whisker, a dot is placed at the bottom whisker. From Figure 3-8, we found that SaDE algorithm succeeds on all functions with similar FEs by using five different LP s. Therefore, the convergence speed of SaDE algorithm is less sensitive to the parameter LP values between 20 and 60. It means that LP can be set as any value between 20 and 60. As LP increases from 20 to 60, there is no obvious improvement. Therefore it is no need to investigate the scenario of more than 60 generations, since longer learning periods will consume more time.

Table 3-8: SaDE’s results with different learning period (LP) on test functions f_8, f_{10} and f_{14} (10-D)

LP	f_8 . Rotated Griewank			f_{10} . Rotated Rastrigin			f_{14} . Composition Function 6		
	Mean	Std	Success Rate	Mean	Std	Success Rate	Mean	Std	Success Rate
20	6.67E-03	6.41E-03	27%	4.18E+00	1.21E+00	0%	1.01E+01	3.05E+01	83%
30	9.65E-03	7.61E-03	17%	3.98E+00	1.35E+00	0%	1.66E-01	4.32E-01	87%
40	5.17E-03	6.27E-03	43%	4.24E+00	9.96E-01	0%	3.45E+00	1.82E+01	87%
50	1.37E-02	1.18E-02	20%	3.80E+00	1.35E+00	0%	2.54E-01	5.21E-01	80%
60	1.16E-02	1.62E-02	27%	3.89E+00	1.24E+00	0%	9.22E-02	3.54E-01	93%

f_7 : Griewank

f_9 : Rastrigin

f_{11} : Non-continuous Rastrigin

f_{12} : Schwefel

Figure 3-8: SaDE’s results with different learning period *LP* on 11 test functions (10-D)

3.6 Conclusion

This chapter presented the SaDE algorithm, which eliminates the time-consuming exhaustive search for the most suitable trial vector generation strategy and the associated control parameters *F* and *CR*. In SaDE, the trial vector generation strategies were probabilistically assigned to each target vector in the current population according to the probabilities gradually learnt from the experience to generate improved solutions.

I evaluated the performance of SaDE over a suite of 26 bound constrained numerical optimization problems, and compared it with conventional DE and three other adaptive DE variants. It was concluded that SaDE is more effective in obtaining better quality solutions, more stable with the relatively smaller standard deviation, and had higher success rates, which is also demonstrated by the overall performance comparison provided in section 4.3.3.

Furthermore, I have investigated the self-adaptive characteristics of the *CR* value and trial vector generation strategies. The experimental results verified the promising

performance of SaDE to handle problems with distinct properties such as uni-modality and multi-modality, and showed that the SaDE algorithm could evolve suitable strategies and parameter values during evolution progress.

Finally, I studied the impact of the parameter, learning period LP , on the performance of SaDE. The sensitivity analysis of the parameter LP indicated that the convergence speed of SaDE algorithm has no significant change when the parameter LP values are between 20 and 60.

Chapter 4 SaDE for Constrained Optimization Problems

4.1 Introduction

The optimization performance of the Self-adapted Differential Evolution (SaDE) algorithm was evaluated on a comprehensive set of benchmarks in the previous chapter. However those benchmarks are only bound- constrained single objective problems. Most real world optimization problems involve searching for a solution that is not only optimal, but also satisfies one or more constraints. To make our SaDE algorithm more applicable to real world problems, this chapter extends the SaDE algorithm to solve single objective optimization problems with multiple constraints. A simple constraints handling mechanism without adding any more parameters is incorporated into the original SaDE. In comparison with the original SaDE algorithm, only the replacement criterion needs to be modified for handling constraints. The self-adaptive mechanism remains unchanged. Finally, the performance of the proposed method is reported on a set of benchmark problems chosen from Congress on Evolutionary Computation 2006 (CEC2006) special session on constrained real parameter optimization.

4.2 Constrained Optimization

Most optimization problems in science and engineering have complex constraints. A constrained optimization problem is usually written as a nonlinear programming problem of the following form:

Minimize: $f(\mathbf{x})$, $\mathbf{x} = (x_1, x_2, \dots, x_n)$ and $\mathbf{x} \in \Sigma$

Subject to:

inequality constraints: $g_i(\mathbf{x}) \leq 0$, $i = 1, \dots, q$

equality constraints: $h_j(\mathbf{x}) = 0$, $j = q+1, \dots, m$

bound constraints: $x_{k,l} \leq x_k \leq x_{k,u}$, $k = 1, \dots, n$

Σ is the whole search space. q is the number of inequality constraints. The number of equality constraints is $m-q$. $x_{k,l}$, $x_{k,u}$ are lower and upper bounds of x_k .

Although evolutionary algorithms have been successful in a wide range of applications, they naturally perform unconstrained searches. When used for solving constrained optimization problems, evolutionary algorithms require additional mechanisms to handle constraints in their objective function. In various Differential Evolution (DE) literature, several constraints handling techniques have been suggested for solving constrained optimization by using evolutionary algorithms.

Michalewicz and Schoenauer [125] grouped the methods for handling constraints by evolutionary algorithm into four categories: 1) preserving feasibility of solutions, 2) penalty functions, 3) make a separation between feasible and infeasible solutions, and 4) other hybrid methods. The most common approach when dealing with constraints is the method based on penalty functions, which penalizes infeasible solutions. However, penalty functions have, in general, several limitations. They require a careful tuning to determine the most appropriate penalty factors. Also, they tend to not behave well when trying to solve a problem in which the optimum is at the boundary between the feasible and the infeasible regions or when the feasible region is disjoint. For some difficult problems in which it is extremely difficult to locate a feasible solution due to inappropriate representation scheme, researchers designed special representations and operators to preserve the feasibility of solutions at all times. Recently a few methods have been proposed which emphasize the distinction between feasible and infeasible

Chapter 4: SaDE for Constrained Optimization Problems

solutions in the search space, such as behavioral memory method, superiority of feasible solutions to infeasible solutions, and repairing infeasible solutions. Also, researchers developed hybrid methods, which combine evolutionary algorithm with another technique (normally a numerical optimization approach) to handle constraints.

4.3 SaDE for Constrained Optimization

Most real world optimization problems involve searching for a solution that is not only optimal, but also satisfies one or more constraints. To make our SaDE algorithm more suited to solve to real world applications, we generalize the SaDE algorithm to solve optimization problems with multiple constraints.

For a constrained optimization problem in a form described in Section 4.2, the equality constraints are transformed into the inequality form for convenience, and then we can combine all the constraints as

$$G_i(\mathbf{x}) = \begin{cases} \max\{g_i(\mathbf{x}), 0\} & i = 1, \dots, q \\ |h_i(\mathbf{x})| & i = q + 1, \dots, m \end{cases}$$

Therefore, the objective of our algorithm is to minimize the objective function $f(\mathbf{x})$, but at the same time, the optimum solutions obtained must satisfy all the constraints $G_i(\mathbf{x})$.

Among various constraints handling methods mentioned in Section 4.2, some methods based on the superiority of feasible solutions, such as the approach proposed by Deb [126], have demonstrated promising performance, as indicated in [127][128], which deal with constraints using DE. However, Deb's selection criterion [126] has no parameter to fine-tune, which is the motivation of our SaDE, *i.e.* saving the cost of fine-tuning the parameters. Hence, we incorporate this constraints handling technique into our SaDE.

Chapter 4: SaDE for Constrained Optimization Problems

During the selection procedure, the trial vector is compared to that of its corresponding target vector in the current population considering both the fitness value and constraints. The trial vector will replace the target vector and enter the population of the next generation if any of the following conditions is true:

- 1) The trial vector is feasible and the target vector is not.
- 2) The trial vector and target vector are both feasible and trial vector has smaller or equal fitness value (for minimization problems) than the corresponding target vector.
- 3) The trial vector and target vector are both infeasible, but trial vector has a smaller overall constrain violation.

The overall constrain violation $v(\mathbf{x})$ is a weighted mean value of all the constraints $G_i(\mathbf{x}) (i=1, \dots, m)$, which is expressed as following,

$$v(\mathbf{x}) = \frac{\sum_{i=1}^m w_i (G_i(\mathbf{x}))}{\sum_{i=1}^m w_i}$$

where $w_i = \frac{1}{G_{\max_i}}$ is a weighted parameter, $G_{\max_i}$ is the maximum violation of the constraints $G_i(\mathbf{x})$ obtained so far. Here, we set w_i as $\frac{1}{G_{\max_i}}$ which varies during the evolution in order to accurately normalize the constraints of the problem so that the overall constrain violation can represent all constraints more equally.

Based on the idea that any obtained solution can be taken as a starting point for some local search process to further search for a better solution within its neighborhood, we use a local search to further refine the solutions found by the SaDE algorithm.

Usually, local search is applied on the best individual, but according to our experimental experience, searching within the neighborhood of other non-best individual could also yield a refined solution. Therefore, besides the best individual, local search will also be applied to other promising individuals in the population. Here, the best 50% of individuals are regarded as promising individuals. Among these promising individuals, the best individual is confirmed to be a starting point of local search, and others are regarded equivalently promising in finding better solution within

Chapter 4: SaDE for Constrained Optimization Problems

the neighborhood. For the sake of reducing computational cost, the local search is applied on 5% of the individuals.

Finally, the following individuals are selected as starting points of the local search:

- 1) The current best individual
- 2) The remaining individuals are randomly selected out of the 50% best individuals (excluding the best individual)

If the local search has already been applied to one particular individual but without finding any refined solution, this particular individual will not be set as a starting point any more.

Since the local search is only a supplementary to the SaDE algorithm, it is not supposed to be the main contributor for finding a good solution. Therefore, the local search will be used only once every 500 generations. Sequential Quadratic Programming (SQP) method implemented in the MATLAB statistical toolbox is used as the local search method.

4.4 Experiments

4.4.1 Test Problems

We evaluated the performance of the SaDE algorithm 24 benchmark functions with constraints [124], which include linear, nonlinear, quadratic, cubic, polynomial constraints.

Table 4-1: Details of the 24 test problems

Prob.	n	Type of function	ρ	LI	NI	LE	NE	a
g01	13	quadratic	0.0111%	9	0	0	0	6
g02	20	nonlinear	99.9971%	0	2	0	0	1
g03	10	polynomial	0.0000%	0	0	0	1	1
g04	5	quadratic	52.1230%	0	6	0	0	2
g05	4	cubic	0.0000%	2	0	0	3	3
g06	2	cubic	0.0066%	0	2	0	0	2
g07	10	quadratic	0.0003%	3	5	0	0	6
g08	2	nonlinear	0.8560%	0	2	0	0	0
g09	7	polynomial	0.5121%	0	4	0	0	2
g10	8	linear	0.0010%	3	3	0	0	6
g11	2	quadratic	0.0000%	0	0	0	1	1
g12	3	quadratic	4.7713%	0	1	0	0	0
g13	5	nonlinear	0.0000%	0	0	0	3	3
g14	10	nonlinear	0.0000%	0	0	3	0	3
g15	3	quadratic	0.0000%	0	0	1	1	2
g16	5	nonlinear	0.0204%	4	34	0	0	4
g17	6	nonlinear	0.0000%	0	0	0	4	4
g18	9	quadratic	0.0000%	0	13	0	0	6
g19	15	nonlinear	33.4761%	0	5	0	0	0
g20	24	linear	0.0000%	0	6	2	12	16
g21	7	linear	0.0000%	0	1	0	5	6
g22	22	linear	0.0000%	0	1	8	11	19
g23	9	linear	0.0000%	0	2	3	1	6
g24	2	linear	79.6556%	0	2	0	0	2

n is the number of decision variables, $\rho = |F|/|S|$ is the estimated ratio between the feasible region and the search space, LI is the number of Linear Inequality constraints, NI the number of Nonlinear Inequality constraints, LE is the number of Linear Equality constraints and NE is the number of Nonlinear Equality constraints. a is the number of active constraints at $\bar{x}$.

Chapter 4: SaDE for Constrained Optimization Problems

4.4.2 Performance Evaluation Criteria

For each function, the SaDE algorithm was run 25 times. The population size was set at 50. We used the fitness value of best known solutions ($f(\mathbf{x}^*)$) newly updated in [124]. The steps we undertook are as follows:

- 1) Record the function error value ($f(\bar{x}) - f(\bar{x}^*)$) for the achieved best solution $\bar{x}$ after $5e+3$, $5e+4$, $5e+5$ FES for each run.
- 2) Record the FES needed in each run for finding a solution satisfying the following condition: $(f(\bar{x}) - f(\bar{x}^*)) \leq 0.0001$ and $\bar{x}$ is feasible.
- 3) Feasible Rate, Success Rate & Success Performance for Each Problem
- 4) Feasible Run: A run during which at least one feasible solution is found in Max FES.
- 5) Successful Run: A run during which the algorithm finds a feasible solution $\bar{x}$ satisfying $(f(\bar{x}) - f(\bar{x}^*)) \leq 0.0001$.

Feasible Rate = (Number of feasible runs) / total runs

Success Rate = (Number of successful runs) / total runs

Success Performance = mean (FEs for successful runs) * (Number of total runs) / (Number of successful runs)

4.4.3 Experimental Results

The error values ($f(\bar{x}) - f(\bar{x}^*)$) for the achieved best solution $\bar{x}$ after FES= $5e+3$, FES= $5e+4$, FES= $5e+5$ for the 24 test functions are listed in Tables 4-2, 4-3, 4-4, and 4-5. We recorded the FES needed in each run to find a solution that satisfies the successful condition [124] in Table 4-6. The success rate, feasible rate, and success performance are also listed in that table. The convergence maps of SaDE on functions 1-6, functions 7-12, functions 13-18, and functions 19-24 are plotted in Figures 4-1, 4-2, 4-3 and 4-4 respectively.

More comparison results between SaDE and other algorithms submitted to the CEC 2006 special session on constrained real-parameter optimization are shown in the Appendix.

Chapter 4: SaDE for Constrained Optimization Problems

Table 4-2: Error Values Achieved When FES= 5×10^3 , FES= 5×10^4 , FES= 5×10^5 for Problems 1-6

FES \ Prob.		g01	g02	g03	g04	g05	g06
5×10^3	Best	2.9525e+000(0)	3.2958e-001(0)	3.3141e-001(0)	3.5714e+001(0)	2.2578e+001(3)	6.4548e+001(0)
	Median	4.2828e+000(0)	3.7275e-001(0)	8.4884e-001(0)	7.5864e+001(0)	2.7489e+002(3)	3.4622e+002(0)
	Worst	5.2143e+000(0)	4.3843e-001(0)	8.8551e-001(1)	1.1877e+002(0)	1.0948e+002(3)	1.7502e+003(0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 1, 3	0, 0, 0
	$\bar{v}$	0	0	0	0	4.7389e-003	0
	Mean	4.1779e+000	3.7920e-001	7.8931e-001	7.7970e+001	1.5908e+002	4.3473e+002
	Std	5.2257e-001	2.9111e-002	1.5394e-001	2.1465e+001	1.1344e+002	3.6649e+002
5×10^4	Best	2.8414e-010(0)	4.2504e-003(0)	5.9753e-005(0)	2.5043e-007(0)	1.0004e-011(0)	4.5475e-011(0)
	Median	2.9675e-010(0)	2.2353e-002(0)	9.9222e-004(0)	2.9835e-007(0)	5.2481e-004(0)	4.5475e-011(0)
	Worst	3.0000e-010(0)	3.8493e-002(0)	7.1664e-001(0)	3.3795e-007(0)	1.3955e-003(0)	4.5475e-011(0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	0	0	0	0	0	0
	Mean	2.9488e-010	2.1825e-002	6.2425e-002	2.9795e-007	6.3977e-004	4.5475e-011
	Std	4.9526e-012	8.2730e-003	1.5421e-001	2.5148e-008	5.4286e-004	0
5×10^5	Best	0(0)	8.0719e-010(0)	1.3749e-010(0)	2.1667e-007(0)	0(0)	4.5475e-011(0)
	Median	0(0)	3.0800e-009(0)	1.7770e-008(0)	2.1667e-007(0)	0(0)	4.5475e-011(0)
	Worst	0(0)	1.8353e-002(0)	1.3389e-004(0)	2.1667e-007(0)	0(0)	4.5475e-011(0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	0	0	0	0	0	0
	Mean	0	2.0560e-003	1.3532e-005	2.1667e-007	0	4.5475e-011
	Std	0	4.9786e-003	3.4743e-005	1.8550e-012	1.8190e-013	0

Table 4-3: Error Values Achieved When FES= 5×10^3 , FES= 5×10^4 , FES= 5×10^5 for Problems 7-12

FES \ Prob.		g07	g08	g09	g10	g11	g12
5×10^3	Best	3.8845e+000(0)	8.1964e-011(0)	4.1400e-001(0)	9.2363e+002(0)	3.3473e-004(0)	1.9651e-014(0)
	Median	6.7238e+000(0)	8.1964e-011(0)	7.0402e-001(0)	1.4735e+003(0)	9.6840e-002(0)	1.2552e-012(0)
	Worst	1.2048e+001(0)	8.1964e-011(0)	1.8411e+000(0)	2.0997e+003(0)	2.1645e-001(0)	1.2816e-010(0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	0	0	0	0	0	0
	Mean	7.1258e+000	8.1964e-011	8.1345e-001	1.4603e+003	1.0756e-001	1.0822e-011
	Std	2.6450e+000	1.0599e-017	3.5365e-001	2.7973e+002	6.8547e-002	2.6763e-011
5×10^4	Best	6.8221e-008(0)	8.1964e-011(0)	3.7440e-007(0)	1.9910e-006(0)	0(0)	0(0)
	Median	2.7655e-003(0)	8.1964e-011(0)	7.1033e-007(0)	1.3888e-001(0)	0(0)	0(0)
	Worst	2.0290e-002(0)	8.1964e-011(0)	3.8943e-005(0)	3.4904e+000(0)	9.9985e-005(0)	0(0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	0	0	0	0	0	0
	Mean	4.9297e-003	8.1964e-011	4.4545e-006	3.9758e-001	9.3997e-006	0
	Std	5.9290e-003	6.0492e-018	8.7853e-006	7.4290e-001	2.7549e-005	0
5×10^5	Best	6.8180e-008(0)	8.1964e-011(0)	3.7440e-007(0)	6.6393e-011(0)	0(0)	0(0)
	Median	1.4608e-007(0)	8.1964e-011(0)	3.7440e-007(0)	1.8120e-006(0)	0(0)	0(0)
	Worst	6.3431e-005(0)	8.1964e-011(0)	3.7440e-007(0)	7.8300e-006(0)	0(0)	0(0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	0	0	0	0	0	0
	Mean	4.7432e-006	8.1964e-011	3.7440e-007	1.6907e-006	0	0
	Std	1.4993e-005	3.8426e-018	7.9851e-014	1.5244e-006	0	0

Table 4-4: Error Values Achieved When FES= 5×10^3 , FES= 5×10^4 , FES= 5×10^5 for Problems 13-18

FES \ Prob.		g13	g14	g15	g16	g17	g18
-------------	--	-----	-----	-----	-----	-----	-----

Chapter 4: SaDE for Constrained Optimization Problems

5×10^3	Best	9.4326e-001(3)	4.3915e+000(3)	6.6347e-002(2)	2.8438e-002(0)	9.1761e+001(4)	1.2814e-001(0)
	Median	8.3030e-001(3)	5.4314e+000(3)	9.4042e-001(2)	5.7669e-002(0)	9.3311e+001(4)	2.6913e-001(0)
	Worst	5.2140e-001(3)	2.4726e+000(3)	1.6126e+000(2)	1.1087e-001(0)	9.5509e+001(4)	4.1448e-001(0)
	c	0, 2, 3	0, 3, 3	0, 1, 2	0, 0, 0	0, 3, 4	0, 0, 0
	$\bar{v}$	5.6738e-002	3.7325e-002	1.3524e-002	0	5.5918e-002	0
	Mean	1.1713e+000	4.9550e+000	1.4324e+000	5.9573e-002	1.0664e+002	2.7886e-001
	Std	1.2511e+000	1.7242e+000	1.5740e+000	1.9716e-002	6.1893e+001	7.7546e-002
5×10^4	Best	5.3102e-006(0)	1.5331e-005(0)	6.0822e-011(0)	6.5214e-011(0)	3.9816e+001(0)	2.7581e-011(0)
	Median	8.3300e-006(0)	1.4460e-004(0)	6.8150e-005(0)	6.5251e-011(0)	7.4058e+001(0)	1.0178e-010(0)
	Worst	3.8491e-001(0)	4.4106e-004(0)	1.4563e-004(0)	6.5455e-011(0)	7.4058e+001(0)	1.9163e-001(0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	0	0	0	0	0	0
	Mean	1.0778e-001	1.6058e-004	6.3430e-005	6.5277e-011	7.2688e+001	1.5312e-002
	Std	1.7638e-001	1.0951e-004	6.0058e-005	6.7594e-014	6.8484e+000	5.2992e-002
5×10^5	Best	4.1898e-011(0)	2.9050e-006(0)	6.0822e-011(0)	6.5214e-011(0)	8.1858e-011(0)	1.5561e-011(0)
	Median	4.1898e-011(0)	1.4793e-005(0)	6.0822e-011(0)	6.5214e-011(0)	7.4058e+001(0)	1.5561e-011(0)
	Worst	1.0696e-006(0)	2.5233e-004(0)	6.0822e-011(0)	6.5214e-011(0)	7.4058e+001(0)	1.9104e-001(0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	0	0	0	0	0	0
	Mean	8.0263e-008	4.6979e-005	6.0822e-011	6.5214e-011	6.9670e+001	1.5284e-002
	Std	2.7832e-007	6.4986e-005	0	0	1.6168e+001	5.2898e-002

Table 4-5: Error Values Achieved When FES= 5×10^3 , FES= 5×10^4 , FES= 5×10^5 for Problems 19-24

FES \ Prob.		g19	g20	g21	g22	g23	g24
5×10^3	Best	9.9842e+001(0)	1.2960e+001(20)	4.6515e+002(5)	1.6363e+004(19)	6.1159e+002(4)	1.7199e-005(0)
	Median	1.5038e+002(0)	1.5603e+001(20)	3.9781e+002(5)	7.8714e+003(19)	2.8993e+002(4)	9.3936e-005(0)
	Worst	1.9265e+002(0)	1.3569e+001(20)	4.0739e+002(5)	1.3480e+004(19)	4.4529e+002(5)	1.7135e-004(0)
	c	0, 0, 0	2, 18, 20	0, 3, 5	14, 19, 19	0, 2, 4	0, 0, 0
	$\bar{v}$	0	5.6112e+000	6.3055e-002	1.7472e+007	6.0069e-003	0
	Mean	1.4462e+002	1.4265e+001	4.1002e+002	1.2034e+004	4.0600e+002	8.4004e-005
	Std	2.2065e+001	1.7690e+000	1.2050e+002	2.9316e+003	8.0627e+001	4.5396e-005
5×10^4	Best	5.8659e-007(0)	1.8020e-002(6)	4.6858e-002(0)	3.0199e+001(0)	7.6219e-003(0)	4.6372e-012(0)
	Median	8.5695e-004(0)	5.1702e-001(19)	6.1550e-002(0)	1.0372e+002(0)	5.5090e-002(0)	4.6372e-012(0)
	Worst	1.1303e-001(0)	4.5755e+000(19)	6.2397e-002(0)	1.1282e+004(16)	5.5144e-002(0)	4.6372e-012(0)
	c	0, 0, 0	1, 15, 19	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	0	1.5836e-001	0	0	0	0
	Mean	6.0660e-003	2.1193e+000	6.0340e-002	5.3477e+003	5.0237e-002	4.6372e-012
	Std	2.2429e-002	2.4755e+000	3.8090e-003	6.2993e+003	1.3633e-002	0
5×10^5	Best	5.4456e-011(0)	1.8020e-002(6)	0(0)	1.9483e+000(0)	0(0)	4.6372e-012(0)
	Median	1.3868e-010(0)	2.3757e-001(20)	2.5785e-008(0)	4.6907e+001(0)	3.9790e-013(0)	4.6372e-012(0)
	Worst	3.1141e-009(0)	5.3597e-001(20)	6.0712e-003(0)	1.2204e+002(0)	1.3788e-003(0)	4.6372e-012(0)
	c	0	0, 16, 20	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	0	8.8255e-002	0	0	0	0
	Mean	4.2052e-010	2.5564e-001	7.6753e-004	5.0203e+001	1.2061e-004	4.6372e-012
	Std	7.2976e-010	1.0638e-001	1.5058e-003	3.0415e+001	3.4116e-004	0

Table 4-6: Number of FES to achieve the fixed accuracy level $((f(\bar{x}) - f(\bar{x}^*)) \leq 0.0001)$,

Success Rate, Feasible Rate and Success Performance

Prob.	Best	Median	Worst	Mean	Std	Feasible Rate	Success Rate	Success Performance
-------	------	--------	-------	------	-----	---------------	--------------	---------------------

Chapter 4: SaDE for Constrained Optimization Problems

g01	25115	25115	25115	25115	0	100%	100%	25115
g02	76915	128970	-	188990	142570	100%	84%	183850
g03	30000	261000	-	243520	135990	100%	96%	298960
g04	25107	25107	25113	25107	2	100%	100%	25107
g05	35500	65000	152000	74340	30484	100%	100%	73000
g06	12546	14404	18347	14394	1242	100%	100%	12546
g07	25195	101240	422860	143090	124800	100%	100%	27637
g08	782	1272	1775	1268	242	100%	100%	1323
g09	12960	16787	33166	18560	5170	100%	100%	21446
g10	26000	52000	153000	58760	33968	100%	100%	44167
g11	12643	25111	25120	23353	3482	100%	100%	25111
g12	463	1717	2576	1611	582	100%	100%	2576
g13	25161	25219	126080	42372	29775	100%	100%	25168
g14	32000	77000	-	154320	179040	100%	80%	45000
g15	25500	41500	97000	45240	18928	100%	100%	27000
g16	13144	14433	15797	14545	748	100%	100%	14948
g17	443000	-	-	497720	11400	100%	4%	12500000
g18	26000	26000	-	65400	130870	100%	92%	28261
g19	25531	51588	78048	48733	13552	100%	100%	52165
g20	-	-	-	-	-	0%	0%	-
g21	98500	320000	-	327660	166180	100%	60%	164170
g22	-	-	-	-	-	100%	0%	-
g23	82500	298500	-	294540	137090	100%	88%	129550
g24	4280	4843	5657	4847	360	100%	100%	4624

Chapter 4: SaDE for Constrained Optimization Problems

Figure 4-1: Convergence Graph for Function 1-6

Chapter 4: SaDE for Constrained Optimization Problems

(2-a) $\log(f(x) - f(x^*))$ vs FES

(2-b) $\log(v)$ vs FES

Figure 4-2: Convergence Graph for Function 7-12

Chapter 4: SaDE for Constrained Optimization Problems

(3-a) $\log(f(x)-f(x^*))$ vs FES

(3-b) $\log(v)$ vs FES

Figure 4-3: Convergence Graph for Function 13-18

Chapter 4: SaDE for Constrained Optimization Problems

Figure 4-4: Convergence Graph for Function 19-24

From the results, we observed that, the SaDE algorithm could reach the newly updated best known solutions for all problems except problems 20 and 22.

As shown in Table 4-6, the feasible rates of all problems are 100%, except problem 20

Chapter 4: SaDE for Constrained Optimization Problems

and 22. Problem 20 is highly constrained and no algorithm in the available literature found feasible solutions. The success rates are very encouraging, with most problems having a success rate of 100%. Problems 2, 3, 10, 14, 18, 21 and 23 have a success rate of 84%, 96%, 80%, 92%, 60% and 88% respectively. Problem 17's success rate is 4%, and successfully found a better solution only once. Although the success rate of problem 22 is 0, the result we obtained is much better than previous best known solutions, and approximates the newly updated best known solutions.

4.5 Conclusion

This chapter generalizes the SaDE algorithm so that it can handle optimization problems with constraints, without introducing any additional parameters. The performance of our approach was evaluated on the testbed for CEC2006 special session on constrained real parameter optimization. The SaDE algorithm demonstrated effectiveness and robustness.

Chapter 5 Multi-objective Optimization based on Self-adaptive Differential Evolution Algorithm

5.1 Introduction

Inspired by the good performance of Self-adapted Differential Evolution (SaDE) on single-objective (unconstrained and constrained) problems in Chapters 3 and 4, we are interested in extending SaDE to solve numerical optimization problems with multiple conflicting objectives.

This chapter firstly introduces preliminary concepts of multi-objective optimization and presents related work of differential evolution in multi-objective optimization. Secondly, the extension of the SaDE algorithm to solve multi-objective optimization problems, called Multi-Objective Self-adapted Differential Evolution (MOSaDE), is proposed. To evaluate the performance of the proposed MOSaDE algorithm, a suite of 19 benchmark problems is used. This suite was also used for the Congress of Evolutionary Computation 2007 (CEC2007) special session (<http://www.ntu.edu.sg/home/epnsugan/>) on Performance Assessment of Multi-Objective Optimization Algorithms.

As discussed in Chapters 2 and 3, different trial vector generation strategies and control parameters may be better suited to solve problems with different properties. Similarly for multi-objective problems, different objective functions may also possess various properties so that one set of strategy and parameters may benefit one objective but deteriorate others. Hence, it can be more meaningful to learn different sets of strategy and parameters for each objective. Therefore, we apply objective-wise

learning strategies to the MOSaDE algorithm, resulting in Objective-Wise MOSaDE (OW-MOSaDE).

5.2 Multi-objective Optimization

Many real-world problems involve simultaneous optimization of several conflicting objectives or criteria. For example, when an investor invests in a portfolio, he wants high return, but at the same time, low risk. The task of finding one or more optimum solutions for such a type of problem is known as multi-objective optimization. While in single-objective optimization the optimal solution is usually a single point, this does not hold for multi-objective optimization problems. Instead of a single optimum, there is a set of trade-off solutions known as *Pareto-optimal* solutions. These solutions are optimal in the wider sense that no other solutions in the search space are superior to them when all objectives are considered.

In the early days, some researchers solved multi-objective optimization problems by aggregating all the objectives to form a single function, thereby converting it to a single-objective optimization problem. Such a method only obtains one particular optimal solution at a time. It works efficiently only when the user knows the exact trade-off among objective functions and assigns appropriate weights to different objective functions. However, users of this form of optimization are usually not sure of the exact trade-offs among objectives. In this case, it is better to find a set of Pareto-optimal solutions first and then choose one solution from the set using some other high-level information or consideration. A set of Pareto-optimal solutions could give users an overall perspective and allow a user to choose a solution according to some desired degree of importance given to each objective. Given this knowledge, a multi-objective approach with the following principles is very useful:

1. No knowledge of any weight is assumed at first;
2. A population-based optimization algorithm is used to find a set of Pareto-optimal solutions in a single run.

Definition 5.1: The Multi-objective Optimization Problem (MOP) can be defined as:

$$\begin{aligned} \text{Minimize/Maximize } & \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_m(\mathbf{x})) \in \mathbf{Y} \\ \text{Subject to } & \mathbf{G}(\mathbf{x}) = (g_1(\mathbf{x}), g_2(\mathbf{x}), \dots, g_j(\mathbf{x})) \geq 0 \\ & \mathbf{H}(\mathbf{x}) = (h_1(\mathbf{x}), h_2(\mathbf{x}), \dots, h_k(\mathbf{x})) = 0 \\ \text{where } & \mathbf{x} = (x_1, x_2, \dots, x_n) \in \mathbf{X} \end{aligned}$$

and $\mathbf{x}$ is the decision vector, $\mathbf{F}(\mathbf{x})$ is the objective vector, $\mathbf{X}$ is denoted as the decision space, and $\mathbf{Y}$ is called the objective space. The constraints $\mathbf{G}(\mathbf{x}) \geq 0$ and $\mathbf{H}(\mathbf{x}) = 0$ determine the feasible region.

Definition 5.2: Pareto Dominance

For any two decision vectors $\mathbf{u}$ and $\mathbf{v}$, $\mathbf{u}$ is said to dominate $\mathbf{v}$, if :

- 1) $\mathbf{u}$ is no worse than $\mathbf{v}$ in all objectives;
- 2) $\mathbf{u}$ is strictly better than $\mathbf{v}$ in at least one objective.

Definition 5.3: Non-dominated set

Among a set of solutions P , the non-dominated set of solutions P' are those that are not dominated by any member of the set P .

Definition 5.4: Pareto Optimality

When the set P is the entire search space, or $P = S$, the resulting non-dominated set P' is called the *Pareto-optimal set*.

Definition 5.5: Globally and locally Pareto-optimal set

The non-dominated set of the entire feasible search space S is the **globally Pareto-optimal set**. A **locally Pareto-optimal set** can be defined as follows [47].

If $\forall \mathbf{x} \in \underline{P}$, there exists no $\mathbf{y}$ (in the neighborhood of $\mathbf{x}$, such that, $\|\mathbf{x} - \mathbf{y}\|_{\infty} \leq \varepsilon$, where ε is a small positive number) dominating any member of the set $\underline{P}$, then solutions belonging to set $\underline{P}$ constitute a locally Pareto-optimal set.

In Figure 5-1 the shorter curve represents a local Pareto-optimal (PO) front. None of its points have any neighbor that dominates any member of this set.

Figure 5-1: Globally and locally Pareto-optimal set

Since the first attempt to solve multi-objective optimization by Evolutionary Algorithm (EA) [37], Evolutionary Multi-objective Optimization (EMO) methods have attracted many researchers, are widely used in a lot of applications [47] in recent years. EMO methods benefit from the characteristic of EA, obtaining a set of solutions concurrently in a single run.

Differential Evolution (DE) has demonstrated good performance on single-objective test problems and applications. Recently, researchers and engineers introduced DE to the multi-objective optimization domain. One approach was presented to optimize train movement by tuning fuzzy membership functions in mass transit systems [129]. Abbass *et al.* [34] introduced a Pareto-frontier Differential Evolution (PDE) algorithm to solve multi-objective problems by incorporating Pareto dominance. Madavan [130] extended DE to solve multi-objective optimization problems by incorporating a non-dominated sorting and ranking selection scheme of Non-dominated Sorting Genetic Algorithm version II (NSGA-II). Another approach involves applying Pareto-based evaluation to DE to solve multi-objective decision problems. This approach been applied to an enterprise planning problem with two objectives namely, cycle time and cost [16] [17]. Recently, a group of researchers developed parallel multi-population DE algorithm [18]. Kukkonen and Lampinen extended DE for constrained multi-

objective optimization by modifying the selection rule of the basic DE and called it Generalized Differential Evolution (GDE) [19]. Furthermore, they proposed two improved version, GDE2 [20] and GDE3 [21].

There also exists some work to adapt the control parameters of DE when solving the multi-objective problems. Abbass [22] self-adapted the crossover rate of PDE [34], by encoding the crossover rate into each individual, while simultaneously evolving it with other parameters. The scaling factor F is generated for each variable from a Gaussian distribution $N(0,1)$ [22]. Zaharie proposed a parameter Adaptation for DE (ADE) based on the idea of controlling the population diversity [12]. Following the same ideas, Zaharie and Petcu designed an adaptive Pareto DE algorithm for multi-objective optimization and also analyzed its parallel implementation [13]. Xue *et al.* used Fuzzy Logic Controller (FLC) to dynamically adjust the parameters of Multi-Objective Differential Evolution [17].

5.3 Multi-objective Self-adaptive DE

From single-objective optimization to multi-objective optimization

The Multi-Objective SaDE (MOSaDE) algorithm is an extension of SaDE to optimize problems with multiple objectives. Similar to SaDE, the MOSaDE algorithm automatically adapts the trial vector generation strategies and their associated parameters according to their previous experiences of generating promising or inferior individuals. However, when extending the single-objective algorithm to multi-objective domain, the evaluation criteria of promising or inferior solutions must be changed. In our MOSaDE algorithm, we use the following criterion:

Individual A is better than individual B, if

- (1) individual A dominates B, or
- (2) individual A and individual B are non-dominated with each other, but A is less crowded than individual B.

If, according to this criterion, the trial vector is better than the target vector, we will record the associated parameter and strategy as we did with the SaDE in Chapter 3. Besides the evaluation criteria, there exists a set of solutions, as opposed to a single optimum, when extending the SaDE algorithm to multi-objective optimization. We use an external archive to keep the non-dominated solutions generated so far by the MOSaDE algorithm. Initially, this archive is empty. As the evolution progresses, good solutions enter the archive. However, the size of the true non-dominated set can be huge. Since the computational complexity of maintaining the archive increases with the archive's size, the size of the archive will be restricted to a pre-specified value, as many other researchers have done [23][24][25][26].

In our MOSaDE approach, we apply ϵ -dominance [27] to get a well-distributed set of non-domination solutions.

Definition 5.6: ϵ -dominance

For any two decision vectors $\mathbf{u}$ and $\mathbf{v}$, $\mathbf{u}$ is said to ε -dominate $\mathbf{v}$, if the following conditions are true in an M -objective minimization problem:

- 1) $f_i^u \leq f_i^v$ for all $i = 1, 2, \dots, M$;
- 2) $f_i^u < f_i^v - \varepsilon_i$ for at least one $i \in \{1, M\}$.

Here, ε_i is defined as $\varepsilon_i = \delta(f_i^{\max} - f_i^{\min})$, where $f_i^{\max}$ and $f_i^{\min}$ are respectively the maximum and minimum value of the i -th objective in the population.

Crowding Degree Estimation

To ensure good diversity among the non-dominated solutions in the external archive, we need a good measure to evaluate the crowding degree around each non-dominated solution. The crowding degree estimation method is invoked in two situations. Firstly, when the target vector and trial vector do not dominate each other, we evaluate the crowding degree of the target vector and trial vector with respect to the non-dominated solutions in the external archive. The less crowded one is chosen as the new target vector of the next generation. Figure 5-2 shows an example where the trial vector is in a less crowded region. Hence, the trial vector is chosen as the new target vector. Secondly, when the external archive exceeds the pre-specified size, the solutions located at the most crowded place should be detected and eliminated. In our approach, we adopt the harmonic mean distance to estimate the crowding degree, which is defined as follows:

Figure 5-2: An example of non-dominated target vector and trial vector

Definition 5.7: Harmonic Mean

The harmonic mean H of the positive real numbers $a_1, \dots, a_n$ is defined to be

$$H = \frac{n}{\frac{1}{a_1} + \frac{1}{a_2} + \dots + \frac{1}{a_n}}$$

That is, the harmonic mean of a group of terms is the reciprocal of the arithmetic mean of the terms' reciprocals. The procedure of truncating the archive solution by using the harmonic mean distance is shown in Table 5-1. The harmonic mean distance was proved to be effective in [67].

Table 5-1: Truncating the archive solutions by using harmonic mean distance

Step 1: Calculate the distance matrix $\mathbf{Dist}_{N \times N}$ using all the solutions in the external

$$\text{archive } \mathbf{A}_G, \mathbf{Dist}_{N \times N} = \begin{pmatrix} d_{11} & d_{12} & \dots & d_{1N} \\ d_{12} & d_{22} & \dots & d_{2N} \\ \vdots & & & \\ d_{N1} & d_{N2} & \dots & d_{NN} \end{pmatrix},$$

Step 2: Sort the distance matrix along row

Step 3: Compute the harmonic mean distance of the first k positive values in each row and construct a column vector of these harmonic mean distances.

Step 4: The index corresponding to the smallest element of the vector is obtained.

Step 5: Remove the solution corresponding to the index in the original solution set, and set all the distances corresponding to this solution in the distance matrix $\mathbf{Dist}$ calculated in step 1 to be -1.

Step 6: Repeat step 3 until the $N=N_{max}$.

The strategies incorporated into our proposed MOSaDE algorithm are 'DE/rand/1/bin' and 'DE/best/2/bin', which are usually used when solving multi-objective problems in literature [67][34][22][13][14]. The $\mathbf{X}_{best}$ in 'best/2/bin' is randomly selected from the external archive.

The main steps of MOSaDE are described in Table 5-2:

Table 5-2: The algorithmic description of MOSaDE

Chapter 5: MOSaDE

Step 1 Set the generation counter $G = 0$, and randomly initialize a population of NP individuals

$\mathbf{P}_G = \{\mathbf{X}_{1,G}, \dots, \mathbf{X}_{NP,G}\}$ with $\mathbf{X}_{i,G} = \{x_{i,G}^1, \dots, x_{i,G}^D\}$, $i = 1, \dots, NP$ uniformly distributed in the range $[\mathbf{X}_{\min}, \mathbf{X}_{\max}]$, where $\mathbf{X}_{\min} = \{x_{\min}^1, \dots, x_{\min}^D\}$ and $\mathbf{X}_{\max} = \{x_{\max}^1, \dots, x_{\max}^D\}$.

Initialize a second population the external archive $\mathbf{A}_G = \mathbf{P}_G$.

Initialize the median value of CR (CRm_k), strategy probability ($p_{k,G}$, $k = 1, \dots, K$, K is the no. of available strategies), learning period (LP).

Step 2 Evaluate the population

Step 3 WHILE stopping criterion is not satisfied
DO

Step 3.1 Calculate strategy probability $p_{k,G}$ and update the *Success* and *Failure Memory*

IF $G > LP$

FOR $k = 1$ to K

Update the $p_{k,G}$ by equation $p_{k,G} = \frac{S_{k,G}}{\sum_{k=1}^K S_{k,G}}$, ($k = 1, 2, \dots, K$; $G > LP$)

where $S_{k,G} = \frac{\sum_{g=G-LP}^{G-1} ns_{k,g}}{\sum_{g=G-LP}^{G-1} ns_{k,g} + \sum_{g=G-LP}^{G-1} nf_{k,g}} + \varepsilon$, ($\varepsilon = 0.01$)

Remove $ns_{k,G-LP}^2$ and $nf_{k,G-LP}$ out of the *Success* and *Failure Memory* respectively.

END FOR

END IF

Step 3.2 Assign trial vector generation strategy and parameters to each target vector $\mathbf{X}_{i,G}$

/* Assign trial vector generation strategy */

Using stochastic universal sampling to select one strategy k for each target vector

$\mathbf{X}_{i,G}$

/* Assign control parameter F */

FOR $i = 1$ to NP

$F_i = \text{Normrnd}(0.3, 0.1)$

END FOR

/* Assign control parameter CR */

IF $G \geq LP$

FOR $k = 1$ to K ,

$CRm_k = \text{median}(CRMemory_k)$,

END FOR

END IF

FOR $k = 1$ to K

FOR $i = 1$ to NP

$CR_{k,i} = \text{Normrnd}(CRm_k, 0.1)$

WHILE $CR_{m,i} < 0$ or $CR_{m,i} > 1$

$CR_{k,i} = \text{Normrnd}(CRm_k, 0.1)$

² The definition of $ns_{k,G}$ and $nf_{k,G}$ was introduced in Chapter 3.2.2. At the generation G , after evaluating all the generated trial vectors, the number of trial vectors generated by the k^{th} strategy that can successfully enter the next generation is recorded as $ns_{k,G}$ while the number of trial vectors generated by the k^{th} strategy that are discarded in the next generation is recorded as $nf_{k,G}$.

END
END FOR
END FOR

Step 3.3 Generate a new population where each trial vector $\mathbf{U}_{i,G}^k$ is generated according to associated trial

vector generation strategy k and parameter F_i and $CR_{k,i}$ in Step 3.1.

Step 3.4 Randomly reinitialize the trial vector $\mathbf{U}_{i,G}^k$ within the search space if any variable is outside its boundaries.

Step 3.5 Selection:

FOR $i=1$ to NP

Evaluate the trial vector $\mathbf{U}_{i,G}^k$

Find the target vector $\mathbf{X}_{n(i),G}$ nearest to $\mathbf{U}_{i,G}^k$

IF $\mathbf{X}_{n(i),G}$ dominates $\mathbf{U}_{i,G}^k$

$$nf_{k,G} = nf_{k,G} + 1$$

ELSE

IF $\mathbf{U}_{i,G}^k$ dominates $\mathbf{X}_{n(i),G}$

$$X_{n(i),G+1} = U_{i,G}^k, F(X_{n(i),G+1}) = F(U_{i,G}^k)$$

$$ns_{k,G} = ns_{k,G} + 1$$

Store $CR_{k,i}$ into $CRMemory_k$

ENDIF

IF $\mathbf{X}_{n(i),G}$ and $\mathbf{U}_{i,G}^k$ are non-dominated with each other

Choose less crowded between $\mathbf{X}_{n(i),G}$ and $\mathbf{U}_{i,G}^k$ to be

$\mathbf{X}_{n(i),G+1}$

ENDIF

Update the external archive $\mathbf{A}_G$ by Table 5-3:

Update_archive

END IF

END FOR

Store $ns_{k,G}$ and $nf_{k,G}$ ($k=1,\dots,K$) into the *Success* and *Failure Memory* respectively.

When $\mathbf{A}_G$ exceeds the maximum size, we select the less crowded vectors based on harmonic mean distance to keep the archive size at

$$N_{\max}$$

Step 3.6 Increment the generation count $G = G+1$

Step 4 END WHILE

Table 5-3: Update_archive³

```

IF  $U_{i,G}$  is dominated by any member of  $A_G$ ,
    discard  $U_{i,G}$ 
ELSE
    IF  $U_{i,G}$  dominates a set of members  $D(U_{i,G})$  from  $A_G$ 
         $A_G = A_G \setminus D(U_{i,G}); A_G = A_G \cup \{U_{i,G}\}$ 
    ELSE
         $A_G = A_G \cup \{U_{i,G}\}$ 
    END IF
END IF

```

Based on the idea that any obtained solution can be taken as a starting point for some local search process to further search for a better solution within its neighborhood, we use local search to further improve solutions found by the MOSaDE algorithm.

More specifically, we introduce a local search flag (LS_flag) taking a binary value to indicate whether the individual in the external archive has been chosen as the start point for the local search, i.e.,

$$LS_flag = \begin{cases} 0 & \text{individual not experiencing local search} \\ 1 & \text{individual being applied with local search} \end{cases}$$

When one individual newly enters into the external archive, its local search flag i.e. LS_flag , is set to 0. When it is chosen to apply local search, its local search flag i.e. LS_flag , is set to 1.

Here, we employ the Quasi-Newton method implemented in the MATLAB statistical toolbox as the local search method, considering only one objective randomly selected each time. The local search procedure is applied once every 200 generations, on 10 individuals randomly selected among the non-dominated solutions that were not applied local search previously, i.e. their corresponding local search flags are equal to 0. If the number of those individuals is smaller than 10, all of them will be applied

³ When a non-dominated solution $U_{i,G}^k$ is obtained in Step 3.3 in Table 5-2, the archive A_G will be updated according to Table 5-3.

local search. The individuals obtained after local search will be compared with the original starting point, and compared with external archive if needed. This procedure is shown in Table 5-4.

Table 5-4: Local search Procedure

$X_{LS} = \text{Local_search} (X)$
IF X dominates X_{LS} ,
Discard X_{LS}
ELSE
Update external archive with X_{LS} by Table 5-3:
Update_archive
END

5.4 Multi-objective Self-adaptive DE with objective-wise learning strategies (OW-MOSaDE)

As discussed in Chapters 2 and 3, different trial vector generation strategies and control parameters may be suitable to solve problems with different properties. Similarly for multi-objective problems, different objective functions may also possess various properties so that one set of strategy and parameters may benefit one objective but deteriorate others. Therefore, it can be more meaningful to learn different sets of strategy and parameters for each objective.

In MOSaDE, a strategy or a CR value is regarded as successful when under which the:

- 1) Trial vector dominates the target vector; or
- 2) Trial vector is less crowded when the trial and target vectors are non-dominated with each other.

In these cases, the strategy and CR value will be recorded for learning. Such a learning scheme is called **nondomination-based learning**.

Here, we propose a new learning scheme, called **objective-wise learning**, in which different sets of strategy and parameters are assigned to each objective. For certain objectives, a strategy or a CR value is regarded as successful when the trial vector is fitter than the target vector with respect to these objective.

Trial Vector Generation Strategy Adaptation

In our SaDE and MOSaDE, we introduced **Success Memory** to record the successful times of each strategy, and **Failure Memory** to record the failure times of each strategy. Hereby, to learn different sets of strategy for each objective, we will create $SuccMem_j$ and $FailMem_j$ for objective f_j ($j=1,2,\dots,n_obj$, n_obj is the number of objectives.) to record the successful counts of $ns_{j,k}$ and failure counts of $nf_{j,k}$ of strategy k .

Parameter Adaptation

In SaDE, to adapt the crossover rate CR , we establish memories named $CRMemory_k$ to store those successful CR values with respect to the k^{th} strategy. Same as the alteration to strategy adaption, we will establish $CRMemory_{j,k}$ for objective f_j of strategy k .

The main steps of OW-MOSaDE are described in Table 5-5.

Table 5-5: The algorithmic description of OW-MOSaDE

Step 1 Set the generation counter $G = 0$, and randomly initialize a population of NP individuals $\mathbf{P}_G = \{\mathbf{X}_{1,G}, \dots, \mathbf{X}_{NP,G}\}$ with $\mathbf{X}_{i,G} = \{x_{i,G}^1, \dots, x_{i,G}^D\}$, $i = 1, \dots, NP$ uniformly distributed in the range $[\mathbf{X}_{\min}, \mathbf{X}_{\max}]$, where $\mathbf{X}_{\min} = \{x_{\min}^1, \dots, x_{\min}^D\}$ and $\mathbf{X}_{\max} = \{x_{\max}^1, \dots, x_{\max}^D\}$.

Initialize a second population the external archive $\mathbf{A}_G = \mathbf{P}_G$.

For objective f_j , initialize the median value of CR ($CRm_{j,k}$),

strategy probability ($p_{j,k,G}$, $j = 1, \dots, n_obj$, $k = 1, \dots, K$, n_obj is the no. of objectives, K is the no. of available strategies), learning period (LP).

Step 2 Evaluate the population

Step 3 WHILE stopping criterion is not satisfied
DO

Step 3.1 Calculate strategy probability $p_{j,k,G}$ and update the *Success* and *Failure Memory*

IF $G > LP$

For $j = 1$ to n_obj

For $k = 1$ to K

Update the $p_{k,G}$ by equation $p_{j,k,G} = \frac{S_{j,k,G}}{\sum_{k=1}^K S_{j,k,G}}$, ($k = 1, 2, \dots, K$) where

$$S_{k,G} = \frac{\sum_{g=G-LP}^{G-1} ns_{j,k,g}}{\sum_{g=G-LP}^{G-1} ns_{j,k,g} + \sum_{g=G-LP}^{G-1} nf_{j,k,g}} + \varepsilon, (\varepsilon = 0.01)$$

Remove $ns_{j,k,G-LP}$ and $nf_{j,k,G-LP}$ out of the *Success Memory* ($SucMem_j$), and *Failure Memory* ($FailMem_j$) respectively.

END

END

Step 3.2 Assign trial vector generation strategy and parameters to each target vector $\mathbf{X}_{i,G}$

/* Assign trial vector generation strategy */

Randomly choose one objective f among n_obj objectives

For each target vector $\mathbf{X}_{i,G}$, using stochastic universal sampling to select one strategy k from the strategy vector of the chosen objective f .

/* Assign control parameter F */

FOR $i = 1$ to NP

$F_i = \text{Normrnd}(0.3, 0.1)$

END FOR

/* Assign control parameter CR */

IF $G \geq LP$

FOR $j = 1$ to n_obj

FOR $k = 1$ to K ,

$CRm_{j,k} = \text{median}(CRMemory_{j,k})$,

END FOR

END FOR

END IF

FOR $j = 1$ to n_obj

FOR $k = 1$ to K

FOR $i = 1$ to NP

```

 $CR_{k,i} = \text{Normrnd}(CRM_{j,k}, 0.1)$ 
WHILE  $CR_{m,i} < 0$  or  $CR_{m,i} > 1$ 
 $CR_{k,i} = \text{Normrnd}(CRM_k, 0.1)$ 
END WHILE
END FOR
END FOR
END FOR

```

Step 3.3 Generate a new population where each trial vector $\mathbf{U}_{i,G}^k$ is generated according to associated trial vector generation strategy k and parameter F_i and $CR_{k,i}$ in Step 3.1.

Step 3.4 Randomly reinitialize the trial vector $\mathbf{U}_{i,G}^k$ within the search space if any variable is outside its boundaries.

Step 3.5 Selection:

```

FOR  $i=1$  to  $NP$ 
    Evaluate the trial vector  $\mathbf{U}_{i,G}^k$ 
    Find the target vector  $\mathbf{X}_{n(i),G}$  nearest to  $\mathbf{U}_{i,G}^k$ 
    FOR  $j=1$  to  $n\_obj$ 
        IF  $f_j(\mathbf{U}_{i,G}^k) < f_j(\mathbf{X}_{i,G}^k)$ 
             $ns_{j,k,G} = ns_{j,k,G} + 1$ 
            Store  $CR_{k,i}$  into  $CRM_{Memory_{j,k}}$ 
        ELSE
             $nf_{j,k,G} = nf_{j,k,G} + 1$ 
        END IF
    END FOR

    IF  $\mathbf{U}_{i,G}^k$  dominates  $\mathbf{X}_{n(i),G}$ 
         $\mathbf{X}_{n(i),G+1} = \mathbf{U}_{i,G}^k$ ,  $F(\mathbf{X}_{n(i),G+1}) = F(\mathbf{U}_{i,G}^k)$ 
    ENDIF
    IF  $\mathbf{X}_{n(i),G}$  and  $\mathbf{U}_{i,G}^k$  are non-dominated with each other
        Choose less crowded between  $\mathbf{X}_{n(i),G}$  and  $\mathbf{U}_{i,G}^k$  to be  $\mathbf{X}_{n(i),G+1}$ 
    ENDIF

    Update the external archive  $\mathbf{A}_G$  by Table 5-3
END IF
END FOR

```

Store $ns_{j,k,G}$ and $nf_{j,k,G}$ ($k=1, \dots, K$) into the *Success* and *Failure Memory* $SucMem_j$, $FailMem_j$ respectively.

When A_G exceeds the maximum size, we select the less crowded vectors based on harmonic mean distance to keep the archive size at

$$N_{\max}.$$

Step 3.6 Increment the generation count $G = G + 1$

Step 4 END WHILE

5.5 Experimental Results

5.5.1 Experimental results of MOSaDE

Test Problems

In this section, a set of multi-objective optimization problems is described. We assume that the optimization problems under consideration involve M objective functions $f_1, \dots, f_M$ that are all to be minimized.

First, we choose test functions OKA2 [28] and SYMPART [31] proposed recently. Second, we select some problems from two immensely popular test suites, Zitzler-Deb-Thiele (ZDT) and Deb-Thiele-Laumanns-Zitzler (DTLZ) [29][30]. Finally, three test functions of the newly proposed Word Financial Group (WFG) test suite are included [32].

However, the ZDT and DTLZ test suites have the following limitations:

- For all problems, the global optimum has the same parameter values for different variables/dimensions
- The global optimum lies in the center of the search range
- The global optimum lies on the bounds
- All of these problems are separable

To overcome these shortcomings, we shifted or rotated the original ZDT and DTLZ problem.

□ Extending and Shifting

$f(\mathbf{z})$: original function. Search range $[\mathbf{zmin}, \mathbf{zmax}]$

$F(\mathbf{x})$: new extended function. Search range $[\mathbf{xmin}, \mathbf{xmax}]$

D: dimension

$\mathbf{o} = [o_1, o_2, \dots, o_D]$: the shifted vector in parameter space

$\mathbf{d} = [d_1, d_2, \dots, d_D]$: the extended length of the lower bound

$\lambda = [\lambda_1, \lambda_2, \dots, \lambda_D]$: the scale factor

$\mathbf{p} = [p_1, p_1, \dots, p_D]$: the penalty value

1) To overcome the shortcomings of the ZDT test functions for which the global optimum lies on the lower bound, or in the center of the search range, we extend the lower bound $\mathbf{zmin}$ by $\mathbf{d}$. Then, for the solution in the extended region, the function value is obtained by mapping and stretching.

$$f'(\mathbf{z}') = S(psum)(f(\mathbf{z}') + f_bias)$$

$$\text{where } z'_i = \begin{cases} z_i, & z_i \geq z \min_i \\ z \min_i + \lambda_i (z \min_i - z_i), & z_i < z \min_i \end{cases},$$

$$S(psum) = \frac{2}{1 + \exp(-psum)}, \quad psum = \sqrt{\sum_{i \in I} p_i^2}, \quad I \subseteq \{1, 2, \dots, D\} \quad (I \text{ is a set of all variables included in the objective function } f(x))$$

Here the constant parameter vector λ is used to make the searching region not symmetric with respect to the variable. The stretching function S is used to guarantee that the objective function values of solutions in the extended region are always worse than those in the original region, i.e., the Pareto Optimal front remains unchanged. This assumption holds true on the condition that $f > 0$. Therefore we shift f to $f + f_bias$ to make sure that all function values are positive. The range of the function S is $[1, 2]$. When one solution in the extended region is near the mapping center, there will be $psum \rightarrow 0$ and $S \rightarrow 1$. On the contrary, if the solution is far from the mapping center, $S \rightarrow 2$. Thus, we enlarge the objective value in the extended region whilst at the same time keeping the function connected.

The penalty value p_i in each variable is calculated as:

$$p_i = \begin{cases} 0, & z_i \geq z \min_i \\ |z \min_i - z_i| / d_i, & z_i < z \min_i \end{cases}, \quad i = 1, 2, \dots, D$$

2) After extending the region, shift the parameter space by vector $\mathbf{o}$, and then the new function $F(\mathbf{x}) = f'(\mathbf{x})$, $\mathbf{x} = \mathbf{z} + \mathbf{o}$

Therefore, the extended shifted function will be:

$$F_m(\mathbf{x}) = \begin{cases} f_m(\mathbf{z}') + 1 & \text{for all } z_i \geq z \min_i \\ S(psum_m)(f_m(\mathbf{z}') + 1) & \text{otherwise} \end{cases}, \quad m = 1, 2, \dots, M, \quad \mathbf{z} = \mathbf{x} - \mathbf{o}$$

$$\text{where } S(psum) = \frac{2}{1 + \exp(-psum)}, \quad psum = \sqrt{\sum_{i \in I} p_i^2}, \quad I \subseteq \{1, 2, \dots, D\}$$

$$z'_i = \begin{cases} z_i, & z_i \geq z \min_i \\ z \min_i + \lambda_i(z \min_i - z_i), & z_i < z \min_i \end{cases}$$

$$p_i = \begin{cases} 0, & z_i \geq z \min_i \\ |z \min_i - z_i| / d_i, & z_i < z \min_i \end{cases}, \quad i = 1, 2, \dots, D$$

According to the above description, we extended and shifted ZDT1, ZDT2, ZDT4, ZDT6, DTLZ2, DTLZ3, obtaining S_ZDT1, S_ZDT2, S_ZDT4, S_ZDT6, S_DTLZ2, and S_DTLZ3 [68].

□ Extending and Rotation

The detailed principles of extending are the same as the previous description.

$$F_m(\mathbf{x}) = \begin{cases} f_m(\mathbf{z}') + 1 & \text{for all } x \min_i \leq x_i \leq x \max_i \\ S(psum_m)(f_m(\mathbf{z}') + 1) & \text{otherwise} \end{cases}, \quad m = 1, 2, \dots, M, \quad \mathbf{z} = \mathbf{M}\mathbf{x}$$

$$\text{where } S(psum) = \frac{2}{1 + \exp(-psum)}, \quad psum = \sqrt{\sum_{i \in I} p_i^2}, \quad I \subseteq \{1, 2, \dots, D\}$$

$$z'_i = \begin{cases} z \min_i + \lambda_i(z \min_i - z_i), & z_i < z \min_i \\ z_i, & z \min_i \leq z_i \leq z \max_i \\ z \max_i - \lambda_i(z_i - z \max_i), & z_i > z \max_i \end{cases}$$

$$p_i = \begin{cases} z \min_i - z_i, & z_i < z \min_i \\ 0, & z \min_i \leq z_i \leq z \max_i \\ z_i - z \max_i, & z_i > z \max_i \end{cases}$$

$$i = 1, 2, \dots, D$$

$\mathbf{M}$: linear transformation orthogonal matrix, with condition number=1.

According to the above description, we extended and rotated ZDT4, DTLZ2, obtaining R_ZDT4, R_DTLZ2.

1) OKA2

$$f_1(\mathbf{x}) = x_1$$

$$f_2(\mathbf{x}) = 1 - \frac{1}{4\pi^2} (x_1 + \pi)^2 + |x_2 - 5 \cos(x_1)|^{\frac{1}{3}} + |x_3 - 5 \sin(x_1)|^{\frac{1}{3}}$$

$$x_1 \in [-\pi, \pi], \quad x_2, x_3 \in [-5, 5]$$

$$\text{Pareto Set } (x_1, x_2, x_3) = (\zeta, 5 \cos(\zeta), 5 \sin(\zeta)), \quad \zeta \in [-\pi, \pi]$$

2) SYM-PART

$$f_1(\mathbf{x}) = (x_1' + a - t_1 c_2)^2 + (x_2' - t_2 b)^2 + \dots + (x_{D-1}' + a - t_1 c_2)^2 + (x_D' - t_2 b)^2$$

$$f_2(\mathbf{x}) = (x_1' - a - t_1 c_2)^2 + (x_2' - t_2 b)^2 + \dots + (x_{D-1}' - a - t_1 c_2)^2 + (x_D' - t_2 b)^2$$

where

$$\mathbf{x}' = \begin{pmatrix} \cos \omega & -\sin \omega & 0 & 0 & \dots \\ \sin \omega & \cos \omega & 0 & 0 & \dots \\ 0 & 0 & \cos \omega & -\sin \omega & \dots \\ 0 & 0 & \sin \omega & \cos \omega & \dots \\ \dots & \dots & \dots & \dots & \dots \end{pmatrix} \mathbf{x}$$

$$t_1 = \text{sgn}(x_1') \times \left\lceil \frac{|x_1'| - c_2 / 2}{c_2} \right\rceil, \quad t_2 = \text{sgn}(x_2') \times \left\lceil \frac{|x_2'| - b / 2}{b} \right\rceil$$

$$a = 1, b = 10, c = 8, c_2 = c + 2a = 10;$$

D : dimension

$$\mathbf{x} \in [-20, 20]^D$$

3) Extended shifted ZDT1 (S_ZDT1)

$$f_1(x) = \begin{cases} z_1' + 1, & z_1 \geq 0 \\ S(p_1)(z_1' + 1), & z_1 < 0 \end{cases}$$

$$f_2(x) = \begin{cases} g(x)[1 - \sqrt{z_1' / g(x)}] + 1, & \text{all } z_i \geq 0 \\ S(\sqrt{\sum_{i=1}^D p_i^2}) \left(g(x)[1 - \sqrt{z_1' / g(x)}] + 1 \right), & \text{otherwise} \end{cases}$$

$$g(x) = 1 + 9 \cdot (\sum_{i=2}^D z_i') / (D - 1)$$

$$S(t) = \frac{2}{1 + \exp(-t)}$$

$$\text{where } z'_i = \begin{cases} z_i, & z_i \geq 0 \\ -\lambda_i z_i, & z_i < 0 \end{cases}, \quad p_i = \begin{cases} 0, & z_i \geq 0 \\ |z_i|/d_i, & z_i < 0 \end{cases}, \quad i = 1, 2, \dots, D$$

$$\mathbf{z} = \mathbf{x} - \mathbf{o}, \quad \mathbf{x} = [x_1, x_2, \dots, x_D], \mathbf{z} = [z_1, z_2, \dots, z_D]$$

D : dimension

$\mathbf{o} = [o_1, o_2, \dots, o_D]$: the shifted vector in parameter space

$\mathbf{d} = [d_1, d_2, \dots, d_D]$: the extended length of the lower bound

$\lambda = [\lambda_1, \lambda_2, \dots, \lambda_D]$: the scale factor

$\mathbf{p} = [p_1, p_2, \dots, p_D]$: the penalty value

$$x_i \in [x \min_i, x \max_i], \quad \mathbf{xmin} = [x \min_1, x \min_2, \dots, x \min_D] \quad \text{and}$$

$$\mathbf{xmax} = [x \max_1, x \max_2, \dots, x \max_D]$$

The optimal solutions: $x_1 \in [o_1, 1+o_1]$ and $x_i = o_i, \quad i = 2, \dots, D$

4) Extended Shifted ZDT2 (S_ ZDT2)

$$f_1(x) = \begin{cases} z'_1 + 1, & z_1 \geq 0 \\ S(p_1)(z'_1 + 1), & z_1 < 0 \end{cases}$$

$$f_2(x) = \begin{cases} g(x)[1 - (z'_1 / g(x))^2] + 1, & z_i \geq 0 \\ S\left(\sqrt{\sum_{i=1}^D p_i^2}\right)(g(x)[1 - (z'_1 / g(x))^2] + 1), & \text{otherwise} \end{cases}$$

$$g(x) = 1 + 9 \cdot (\sum_{i=2}^D z'_i) / (D - 1)$$

$$\text{where } z'_i = \begin{cases} z_i, & z_i \geq 0 \\ -\lambda_i z_i, & z_i < 0 \end{cases}, \quad p_i = \begin{cases} 0, & z_i \geq 0 \\ |z_i|/d_i, & z_i < 0 \end{cases}, \quad i = 1, 2, \dots, D,$$

$$\mathbf{z} = \mathbf{x} - \mathbf{o}, \quad \mathbf{x} = [x_1, x_2, \dots, x_D], \mathbf{z} = [z_1, z_2, \dots, z_D]$$

D : dimension

$\mathbf{o} = [o_1, o_2, \dots, o_D]$: the shifted vector in parameter space

$\mathbf{d} = [d_1, d_2, \dots, d_D]$: the extended length of the lower bound

$\lambda = [\lambda_1, \lambda_2, \dots, \lambda_D]$: the scale factor

$\mathbf{p} = [p_1, p_1, \dots, p_D]$: the penalty value

$$x_i \in [x \min_i, x \max_i], \quad \mathbf{xmin} = [x \min_1, x \min_2, \dots, x \min_D] \quad \text{and}$$

$$\mathbf{xmax} = [x \max_1, x \max_2, \dots, x \max_D]$$

The optimal solutions $x_1 \in [o_1, 1+o_1]$ and $x_i = o_i, i = 2, \dots, D$

5) Extended Shifted ZDT4 (S_ZDT4)

$$f_1(x) = \begin{cases} z_1' + 1, & z_1 \geq 0 \\ S(p_1)(z_1' + 1), & z_1 < 0 \end{cases}$$

$$f_2(x) = \begin{cases} g(x)[1 - \sqrt{z_1' / g(x)}] + 1, & \text{all } z_i \geq -5 \\ S\left(\sqrt{\sum_{i=1}^D p_i^2}\right)\left(g(x)[1 - \sqrt{z_1' / g(x)}] + 1\right), & \text{otherwise} \end{cases}$$

$$g(x) = 1 + 10(D-1) + \sum_{i=2}^D [z_i'^2 - 10 \cos(4\pi z_i')]$$

$$\text{where } z_1' = \begin{cases} z_1, & z_1 \geq 0 \\ -\lambda_1 z_1, & z_1 < 0 \end{cases}, \quad p_1 = \begin{cases} 0, & z_1 \geq 0 \\ |z_1| / d_1, & z_1 < 0 \end{cases}$$

$$z_i' = \begin{cases} z_i, & z_i \geq -5 \\ -5 - \lambda_i(z_i + 5), & z_i < -5 \end{cases}, \quad p_i = \begin{cases} 0, & z_i \geq -5 \\ |z_i + 5| / d_i, & z_i < -5 \end{cases}, \quad i = 2, \dots, D$$

$$\mathbf{z} = \mathbf{x} - \mathbf{o}, \quad \mathbf{x} = [x_1, x_2, \dots, x_D], \quad \mathbf{z} = [z_1, z_2, \dots, z_D]$$

D : dimension

$\mathbf{o} = [o_1, o_2, \dots, o_D]$: the shifted vector in parameter space

$\mathbf{d} = [d_1, d_2, \dots, d_D]$: the extended length of the lower bound

$\lambda = [\lambda_1, \lambda_2, \dots, \lambda_D]$: the scale factor

$\mathbf{p} = [p_1, p_1, \dots, p_D]$: the penalty value

$$x_i \in [x \min_i, x \max_i], \quad \mathbf{xmin} = [x \min_1, x \min_2, \dots, x \min_D] \quad \text{and}$$

$\mathbf{xmax} = [x \max_1, x \max_2, \dots, x \max_D]$ are included in the data file.

The optimal solutions $x_1 \in [o_1, 1+o_1]$ and $x_i = o_i, i = 2, \dots, D$

6) Extended rotated ZDT4 (R_ZDT4)

$$f_1(x) = \begin{cases} z_1' + 1, & z_1 \geq 0 \\ S(p_1)(z_1' + 1), & z_1 < 0 \end{cases}$$

$$f_2(x) = \begin{cases} g(x)[1 - \sqrt{z_1' / g(x)}] + 1, & \text{all } z_i \geq -5 \\ S\left(\sqrt{\sum_{i=1}^D p_i^2}\right)\left(g(x)[1 - \sqrt{z_1' / g(x)}] + 1\right), & \text{otherwise} \end{cases}$$

$$g(x) = 1 + 10(D-1) + \sum_{i=2}^D [z_i'^2 - 10 \cos(4\pi z_i')]$$

$$\text{where } z_1' = \begin{cases} -\lambda_1 z_1, & z_1 < 0 \\ z_1, & 0 \leq z_1 \leq 1 \\ \lambda_1 z_1, & z_1 > 1 \end{cases}, \quad p_i = \begin{cases} -z_1, & z_1 < 0 \\ 0, & 0 \leq z_1 \leq 1 \\ z_1 - 1, & z_1 > 1 \end{cases}$$

$$z_i' = \begin{cases} -5 - \lambda_i(z_i + 5), & z_i < -5 \\ z_i, & -5 \leq z_i \leq 5 \\ 5 - \lambda_i(z_i - 5), & z_i > 5 \end{cases}, \quad p_i = \begin{cases} -5 - z_i, & z_i < -5 \\ 0, & -5 \leq z_i \leq 5 \\ z_i - 5, & z_i > 5 \end{cases}, \quad i = 2, \dots, D$$

$$\mathbf{z} = \mathbf{M}\mathbf{x}, \quad \mathbf{x} = [x_1, x_2, \dots, x_D], \mathbf{z} = [z_1, z_2, \dots, z_D]$$

D : dimension

$\lambda = [\lambda_1, \lambda_2, \dots, \lambda_D]$: the scale factor

$\mathbf{p} = [p_1, p_1, \dots, p_D]$: the penalty value

$$x_i \in [x \min_i, x \max_i], \quad \mathbf{xmin} = [x \min_1, x \min_2, \dots, x \min_D] \quad \text{and}$$

$$\mathbf{xmax} = [x \max_1, x \max_2, \dots, x \max_D]$$

7) Extended shifted ZDT6 (S_ZDT6)

$$f_1(x) = \begin{cases} 1 - \exp(-4z_1') \sin^6(6\pi z_1') + 1, & z_1 \geq 0 \\ S(p_1)(1 - \exp(-4z_1') \sin^6(6\pi z_1') + 1), & z_1 < 0 \end{cases}$$

$$f_2(x) = \begin{cases} g(x)[1 - (z_1' / g(x))^2] + 1, & z_i \geq 0 \\ S(\sqrt{\sum_{i=1}^D p_i^2})(g(x)[1 - (z_1' / g(x))^2] + 1), & \text{otherwise} \end{cases}$$

$$g(x) = 1 + 9 \left[\left(\sum_{i=2}^D z_i' \right) / (D - 1) \right]^{0.25}$$

$$\text{where } z_i' = \begin{cases} z_i, & z_i \geq 0 \\ -\lambda_i z_i, & z_i < 0 \end{cases}, \quad p_i = \begin{cases} 0, & z_i \geq 0 \\ |z_i| / d_i, & z_i < 0 \end{cases}, \quad i = 1, 2, \dots, D$$

$$\mathbf{z} = \mathbf{x} - \mathbf{o}, \quad \mathbf{x} = [x_1, x_2, \dots, x_D], \mathbf{z} = [z_1, z_2, \dots, z_D]$$

D : dimension

$\mathbf{o} = [o_1, o_2, \dots, o_D]$: the shifted vector in parameter space

$\mathbf{d} = [d_1, d_2, \dots, d_D]$: the extended length of the lower bound

$\lambda = [\lambda_1, \lambda_2, \dots, \lambda_D]$: the scale factor

$\mathbf{p} = [p_1, p_1, \dots, p_D]$: the penalty value

$$x_i \in [x \min_i, x \max_i], \quad \mathbf{xmin} = [x \min_1, x \min_2, \dots, x \min_D] \quad \text{and}$$

$$\mathbf{xmax} = [x \max_1, x \max_2, \dots, x \max_D]$$

The optimal solutions: $x_i \in [o_i, 1+o_i]$ and $x_i = o_i, i = 2, \dots, D$

8) Extended shifted DTLZ2 (S_DTLZ2)

$$\begin{aligned}
 f_1(\mathbf{x}) &= \begin{cases} (1 + g(\mathbf{x}_M)) \cos(z'_1 \pi/2) \cos(z'_2 \pi/2) \dots \cos(z'_{M-2} \pi/2) \cos(z'_{M-1} \pi/2) + 1, & z_i \geq 0 \\ S(psum_1) \left((1 + g(\mathbf{x}_M)) \cos(z'_1 \pi/2) \cos(z'_2 \pi/2) \dots \cos(z'_{M-2} \pi/2) \cos(z'_{M-1} \pi/2) + 1 \right), & \text{otherwise} \end{cases} \\
 f_2(\mathbf{x}) &= \begin{cases} (1 + g(\mathbf{x}_M)) \cos(z'_1 \pi/2) \cos(z'_2 \pi/2) \dots \cos(z'_{M-2} \pi/2) \sin(z'_{M-1} \pi/2) + 1, & z_i \geq 0 \\ S(psum_2) \left((1 + g(\mathbf{x}_M)) \cos(z'_1 \pi/2) \cos(z'_2 \pi/2) \dots \cos(z'_{M-2} \pi/2) \sin(z'_{M-1} \pi/2) + 1 \right), & \text{otherwise} \end{cases} \\
 f_3(\mathbf{x}) &= \begin{cases} (1 + g(\mathbf{x}_M)) \cos(z'_1 \pi/2) \cos(z'_2 \pi/2) \dots \sin(z'_{M-2} \pi/2) + 1, & z_i \geq 0 \\ S(psum_3) \left((1 + g(\mathbf{x}_M)) \cos(z'_1 \pi/2) \cos(z'_2 \pi/2) \dots \sin(z'_{M-2} \pi/2) + 1 \right), & \text{otherwise} \end{cases} \\
 \vdots & \\
 f_{M-1}(\mathbf{x}) &= \begin{cases} (1 + g(\mathbf{x}_M)) \cos(z'_1 \pi/2) \sin(z'_2 \pi/2) + 1, & z_i \geq 0 \\ S(psum_{M-1}) \left((1 + g(\mathbf{x}_M)) \cos(z'_1 \pi/2) \sin(z'_2 \pi/2) + 1 \right), & \text{otherwise} \end{cases} \\
 f_M(\mathbf{x}) &= \begin{cases} (1 + g(\mathbf{x}_M)) \sin(z'_1 \pi/2) + 1, & z_i \geq 0 \\ S(psum_M) \left((1 + g(\mathbf{x}_M)) \sin(z'_1 \pi/2) + 1 \right), & \text{otherwise} \end{cases} \\
 g(\mathbf{x}_M) &= \sum_{x_i \in \mathbf{x}_M} (z'_i - 0.5)^2
 \end{aligned}$$

$$\text{where } z'_i = \begin{cases} z_i, & z_i \geq 0 \\ -\lambda_i z_i, & z_i < 0 \end{cases}, \quad p_i = \begin{cases} 0, & z_i \geq 0 \\ |z_i|/d_i, & z_i < 0 \end{cases}, \quad i = 1, 2, \dots, D$$

$$\mathbf{z} = \mathbf{x} - \mathbf{o}, \quad \mathbf{x} = [x_1, x_2, \dots, x_D], \quad \mathbf{z} = [z_1, z_2, \dots, z_D]$$

D : dimension

$\mathbf{o} = [o_1, o_2, \dots, o_D]$: the shifted vector in parameter space

$\mathbf{d} = [d_1, d_2, \dots, d_D]$: the extended length of the lower bound

$\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \dots, \lambda_D]$: the scale factor

$\mathbf{p} = [p_1, p_1, \dots, p_D]$: the penalty value

$$x_i \in [x \min_i, x \max_i], \quad \mathbf{xmin} = [x \min_1, x \min_2, \dots, x \min_D] \quad \text{and}$$

$\mathbf{x} \max = [x \max_1, x \max_2, \dots, x \max_D]$ are included in the data file.

The Pareto-optimal solutions $x_i^* = 0.5 + o_i (x_i^* \in \mathbf{x}_M)$ and the objective function values

$$\text{must satisfy: } \sum_{m=1}^M (f_m^*)^2 = 0.5$$

9) Extended Rotated DTLZ2 (R_DTLZ2)

$$\begin{aligned}
f_1(\mathbf{x}) &= \begin{cases} (1 + g(\mathbf{x}_M)) \cos(z'_1 \pi/2) \cos(z'_2 \pi/2) \dots \cos(z'_{M-2} \pi/2) \cos(z'_{M-1} \pi/2) + 1, & z_i \geq 0 \\ S(psum_1) \left((1 + g(\mathbf{x}_M)) \cos(z'_1 \pi/2) \cos(z'_2 \pi/2) \dots \cos(z'_{M-2} \pi/2) \cos(z'_{M-1} \pi/2) + 1 \right), & \text{otherwise} \end{cases} \\
f_2(\mathbf{x}) &= \begin{cases} (1 + g(\mathbf{x}_M)) \cos(z'_1 \pi/2) \cos(z'_2 \pi/2) \dots \cos(z'_{M-2} \pi/2) \sin(z'_{M-1} \pi/2) + 1, & z_i \geq 0 \\ S(psum_2) \left((1 + g(\mathbf{x}_M)) \cos(z'_1 \pi/2) \cos(z'_2 \pi/2) \dots \cos(z'_{M-2} \pi/2) \sin(z'_{M-1} \pi/2) + 1 \right), & \text{otherwise} \end{cases} \\
f_3(\mathbf{x}) &= \begin{cases} (1 + g(\mathbf{x}_M)) \cos(z'_1 \pi/2) \cos(z'_2 \pi/2) \dots \sin(z'_{M-2} \pi/2) + 1, & z_i \geq 0 \\ S(psum_3) \left((1 + g(\mathbf{x}_M)) \cos(z'_1 \pi/2) \cos(z'_2 \pi/2) \dots \sin(z'_{M-2} \pi/2) + 1 \right), & \text{otherwise} \end{cases} \\
\vdots & \\
f_{M-1}(\mathbf{x}) &= \begin{cases} (1 + g(\mathbf{x}_M)) \cos(z'_1 \pi/2) \sin(z'_2 \pi/2) + 1, & z_i \geq 0 \\ S(psum_{M-1}) \left((1 + g(\mathbf{x}_M)) \cos(z'_1 \pi/2) \sin(z'_2 \pi/2) + 1 \right), & \text{otherwise} \end{cases} \\
f_M(\mathbf{x}) &= \begin{cases} (1 + g(\mathbf{x}_M)) \sin(z'_1 \pi/2) + 1, & z_i \geq 0 \\ S(psum_M) \left((1 + g(\mathbf{x}_M)) \sin(z'_1 \pi/2) + 1 \right), & \text{otherwise} \end{cases} \\
g(\mathbf{x}_M) &= \sum_{x_i \in \mathbf{x}_M} (z'_i - 0.5)^2
\end{aligned}$$

$$\text{where } z'_i = \begin{cases} -\lambda_i z_i, & z_i < 0 \\ z_i, & 0 \leq z_i \leq 1 \\ \lambda_i z_i, & z_i > 1 \end{cases}, \quad p_i = \begin{cases} -z_i, & z_i < 0 \\ 0, & 0 \leq z_i \leq 1 \\ z_i - 1, & z_i > 1 \end{cases}, \quad i = 1, 2, \dots, D$$

$$\mathbf{z} = \mathbf{M}\mathbf{x}, \quad \mathbf{x} = [x_1, x_2, \dots, x_D], \quad \mathbf{z} = [z_1, z_2, \dots, z_D]$$

D : dimension

$\mathbf{o} = [o_1, o_2, \dots, o_D]$: the shifted vector in parameter space

$\lambda = [\lambda_1, \lambda_2, \dots, \lambda_D]$: the scale factor

$\mathbf{p} = [p_1, p_1, \dots, p_D]$: the penalty value

$$x_i \in [x \min_i, x \max_i], \quad \mathbf{xmin} = [x \min_1, x \min_2, \dots, x \min_D] \quad \text{and}$$

$$\mathbf{x} \max = [x \max_1, x \max_2, \dots, x \max_D]$$

10) Extended shifted DTLZ3 (S_DTLZ3)

Chapter 5: MOSaDE

$$\begin{aligned}
f_1(\mathbf{x}) &= \begin{cases} (1 + g(\mathbf{x}_M)) \cos(z'_1 \pi/2) \cos(z'_2 \pi/2) \dots \cos(z'_{M-2} \pi/2) \cos(z'_{M-1} \pi/2), & z_i \geq 0 \\ S(psum_1) \left((1 + g(\mathbf{x}_M)) \cos(z'_1 \pi/2) \cos(z'_2 \pi/2) \dots \cos(z'_{M-2} \pi/2) \cos(z'_{M-1} \pi/2) + 1 \right), & \text{otherwise} \end{cases} \\
f_2(\mathbf{x}) &= \begin{cases} (1 + g(\mathbf{x}_M)) \cos(z'_1 \pi/2) \cos(z'_2 \pi/2) \dots \cos(z'_{M-2} \pi/2) \sin(z'_{M-1} \pi/2), & z_i \geq 0 \\ S(psum_2) \left((1 + g(\mathbf{x}_M)) \cos(z'_1 \pi/2) \cos(z'_2 \pi/2) \dots \cos(z'_{M-2} \pi/2) \sin(z'_{M-1} \pi/2) + 1 \right), & \text{otherwise} \end{cases} \\
f_3(\mathbf{x}) &= \begin{cases} (1 + g(\mathbf{x}_M)) \cos(z'_1 \pi/2) \cos(z'_2 \pi/2) \dots \sin(z'_{M-2} \pi/2), & z_i \geq 0 \\ S(psum_3) \left((1 + g(\mathbf{x}_M)) \cos(z'_1 \pi/2) \cos(z'_2 \pi/2) \dots \sin(z'_{M-2} \pi/2) + 1 \right), & \text{otherwise} \end{cases} \\
&\vdots \\
f_{M-1}(\mathbf{x}) &= \begin{cases} (1 + g(\mathbf{x}_M)) \cos(z'_1 \pi/2) \sin(z'_2 \pi/2), & z_i \geq 0 \\ S(psum_{M-1}) \left((1 + g(\mathbf{x}_M)) \cos(z'_1 \pi/2) \sin(z'_2 \pi/2) + 1 \right), & \text{otherwise} \end{cases} \\
f_M(\mathbf{x}) &= \begin{cases} (1 + g(\mathbf{x}_M)) \sin(z'_1 \pi/2), & z_i \geq 0 \\ S(psum_M) \left((1 + g(\mathbf{x}_M)) \sin(z'_1 \pi/2) + 1 \right), & \text{otherwise} \end{cases}
\end{aligned}$$

$$\text{where } g(\mathbf{x}_M) = 100 \left(\|\mathbf{x}_M\| + \sum_{x_i \in \mathbf{x}_M} (z_i - 0.5)^2 - \cos(20\pi(z_i - 0.5)) \right)$$

$$z'_i = \begin{cases} z_i, & z_i \geq 0 \\ -\lambda_i z_i, & z_i < 0 \end{cases}, \quad p_i = \begin{cases} 0, & z_i \geq 0 \\ |z_i|/d_i, & z_i < 0 \end{cases}, \quad i = 1, 2, \dots, D$$

$$\mathbf{z} = \mathbf{x} - \mathbf{o}, \quad \mathbf{x} = [x_1, x_2, \dots, x_D], \mathbf{z} = [z_1, z_2, \dots, z_D]$$

D : dimension

$\mathbf{o} = [o_1, o_2, \dots, o_D]$: the shifted vector in parameter space

$\mathbf{d} = [d_1, d_2, \dots, d_D]$: the extended length of the lower bound

$\lambda = [\lambda_1, \lambda_2, \dots, \lambda_D]$: the scale factor

$\mathbf{p} = [p_1, p_1, \dots, p_D]$: the penalty value

$$x_i \in [x \min_i, x \max_i], \quad \mathbf{xmin} = [x \min_1, x \min_2, \dots, x \min_D] \quad \text{and}$$

$\mathbf{xmax} = [x \max_1, x \max_2, \dots, x \max_D]$ are included in the data file.

The Pareto-optimal solutions $x_i^* = 0.5 + o_i$ ($x_i^* \in \mathbf{x}_M$).

WFG [32]

Given $\mathbf{z} = \{z_1, \dots, z_k, z_{k+1}, \dots, z_n\}$

Minimize $f_{m=1:M}(\mathbf{x}) = D x_M + S_m h_m(x_1, \dots, x_{M-1})$

Where

$$\begin{aligned}
\mathbf{x} &= \{x_1, \dots, x_M\} \\
&= \{\max(t_M^p, A_1)(t_1^p - 0.5) + 0.5, \dots, \max(t_M^p, A_{M-1})(t_{M-1}^p - 0.5) + 0.5, t_M^p\} \\
\mathbf{t}^p &= \{t_1^p, \dots, t_M^p\} \leftarrow \mathbf{t}^{p-1} \leftarrow \dots \leftarrow \mathbf{t}^1 \leftarrow \mathbf{z}_{[0,1]}
\end{aligned}$$

$$\begin{aligned}\mathbf{z}_{[0,1]} &= \{z_{1,[0,1]}, \dots, z_{n,[0,1]}\} \\ &= \{z_1 / z_{1,\max}, \dots, z_n / z_{n,\max}\}\end{aligned}$$

where M is the number of objectives, $\mathbf{x}$ is a set of M underlying parameters (where x_M is an underlying distance parameter and $x_{1:M-1}$ are underlying position parameters), $\mathbf{z}$ is a set of $k + l = n \geq M$ working parameters (the first k and the last l working parameters are position-and distance-related parameters, respectively).

$D > 0$ is a distance scaling constant, $A_{1:M-1} \in \{0,1\}$ are degeneracy constants (for each $A_i = 0$, the dimensionality of the Pareto optimal front is reduced by one), $h_{1:M}$ are shape functions, $S_{1:M} > 0$ are scaling constants, and $\mathbf{t}^{1:p}$ are transition vectors, where “ $\leftarrow$ ” indicates that each transition vector is created from another vector via transformation functions. The domain of all $z_i \in \mathbf{z}$ is $[0, 2i]$, $i = 1, \dots, n$. Note that all $x_i \in \mathbf{x}$ will have domain $[0,1]$.

$$\text{Constants} \quad S_{m=1:M} = 2m, \quad D = 1, \quad A_1 = 1, \quad A_{2:M-1} = 1$$

11) WFG1

Shape $h_{m=1:M-l} = \text{convex}_m$

$$h_M = \text{mixed}_M(\text{with } \alpha=1 \text{ and } A=5)$$

$$\mathbf{t}^1 \quad t_{i=1:k}^1 = y_i$$

$$t_{i=k+1:n}^1 = \text{s_linear}(y_i, 0.35)$$

$$\mathbf{t}^2 \quad t_{i=1:k}^2 = y_i$$

$$t_{i=k+1:n}^2 = \text{b_flat}(y_i, 0.8, 0.75, 0.85)$$

$$\mathbf{t}^3 \quad t_{i=1:n}^3 = \text{b_poly}(y_i, 0.02)$$

$$\mathbf{t}^4 \quad t_{i=1:M-l}^4 = \text{r_sum}(\{y_{(i-1)k/(M-1)+1}, \dots, y_{ik/(M-1)}\},$$

$$\{2((i-1)k/(M-1)+1), \dots, 2ik/(M-1)\})$$

$$t_M^4 = \text{r_sum}(\{y_{k+1}, \dots, y_n\}, \{2(k+1), \dots, 2n\})$$

Optimal solutions:

$$z_{i=1:k} : \text{any combination of values in the range } [0, 2i]$$

$$z_{i=k+1:n} = 2i \times 0.35$$

12) WFG8

Shape $h_{m=1:M} = \text{concave}_m$

$$\begin{aligned}
t^1 \quad t^1_{i=1:k} &= y_i \\
t^1_{i=k+1:n} &= b_param(y_i, r_sum(\{y_1, \dots, y_{i-1}\}, \{1, \dots, 1\}), 0.98/49.98, 0.02, 50) \\
t^2 \quad t^2_{i=1:k} &= y_i \\
t^2_{i=k+1:n} &= s_linear(y_i, 0.35) \\
t^3 \quad t^3_{i=1:M-1} &= r_sum(\{y_{(i-1)k/(M-1)+1}, \dots, y_{ik/(M-1)}\}, \{1, \dots, 1\}) \\
t^3_M &= r_sum(\{y_{k+1}, \dots, y_n\}, \{1, \dots, 1\})
\end{aligned}$$

Optimal solutions:

$z_{i=1:k}$: any combination of values in the range $[0, 2i]$

$$\begin{aligned}
z_{i=k+1:n} &= 2i \times 0.35^{(0.02+49.98(\frac{0.98}{49.98} - (1-2u)) \lfloor 0.5-u \rfloor + \frac{0.98}{49.98})^{-1}} \\
u &= r_sum(\{z_1, \dots, z_{i-1}\}, \{1, \dots, 1\}).
\end{aligned}$$

To obtain a Pareto optimal solution, the position should first be determined by setting $z_{1:k}$ appropriately. The required distance-related parameter values can then be calculated by first determining z_{k+1} (which is trivial given $z_{1:k}$ have been set), then z_{k+2} , and so on, until z_n has been calculated.

13) WFG9

Shape $h_{m=1:M} = \text{concave}_m$

$$\begin{aligned}
t^1 \quad t^1_{i=1:n-1} &= b_param(y_i, r_sum(\{y_{i+1}, \dots, y_n\}, \{1, \dots, 1\}), 0.98/49.98, 0.02, 50) \\
t^1_n &= y_n \\
t^2 \quad t^2_{i=1:k} &= s_decept(y_i, 0.35, 0.001, 0.05) \\
t^2_{i=k+1:n} &= s_multi(y_i, 30, 95, 0.35) \\
t^3 \quad t^3_{i=1:M-1} &= r_nonsep(\{y_{(i-1)k/(M-1)+1}, \dots, y_{ik/(M-1)}\}, k/(M-1)) \\
t^3_M &= r_nonsep(\{y_{k+1}, \dots, y_n\}, l)
\end{aligned}$$

Optimal solutions:

$$\begin{aligned}
z_{i=k+1:n} &= 2i \times \begin{cases} 0.35^{(0.02+1.96u)^{-1}}, & i \neq n \\ 0.35, & i = n \end{cases} \\
u &= r_sum(\{z_{i+1}, \dots, z_n\}, \{1, \dots, 1\}).
\end{aligned}$$

Which can be found by first determining z_n , then z_{n-1} , and so on, until the required value for z_{k+1} is determined. Once the optimal values for $z_{k+1:n}$ are determined, the position-related parameters $z_{1:k}$ can be varied arbitrarily, in the range $[0, 2i]$, to obtain different Pareto optimal solutions.

Table 5-6: Properties of the test functions [32]

Test functions	Objective	# Parameters	Separability	Modality	No Extremal	No Medial	Optima Known	Geometry	Pareto many-to-one	Flat Regions
1.OKA2	f_1	1	S	U	✓	✓	✓	concave	X	X
	f_2	1	NS	M						
2.SYMPART	$f_{1:2}$	✓	NS	M	✓	✓	✓	concave	✓	X
3.S_ZDT1	f_1	1	S	U	✓	✓	✓	convex	X	X
	f_2	✓	S	U						
4.S_ZDT2	f_1	1	S	U	✓	✓	✓	concave	X	X
	f_2	✓	S	U						
5.S_ZDT4	f_1	1	S	U	✓	✓	✓	convex	X	X
	f_2	✓	S	M						
6.R_ZDT4	$f_{1:2}$	✓	NS	M	✓	✓	✓	convex	X	X
7.S_ZDT6	f_1	1	S	M	✓	✓	✓	concave	✓	X
	f_2	✓	S	M						
8.S_DTLZ2	$f_{1:M}$	✓	S	U	✓	✓	✓	concave	✓	X
9.R_DTLZ2	$f_{1:M}$	✓	NS	M	✓	✓	✓	concave	✓	X
10.S_DTLZ3	$f_{1:M}$	✓	S	M	✓	✓	✓	concave	✓	X
11.WFG1	$f_{1:M}$	✓	S	U	✓	✓	✓	convex,mixed	✓	✓
12.WFG8	$f_{1:M}$		NS	U				concave		
13.WFG9	$f_{1:M}$		NS	M,D				concave		

S: Separable; NS: non-separable; U: Uni-modal; M: Multi-modal; D: Deceptive.

Table 5-7: The experimental settings for the test functions

Test Function	No. of Objectives	No.of Parameter	Approximation set size
1.OKA2	2	3	100
2.SYMPART	2	30	
3.S_ZDT1	2	30	
4.S_ZDT2	2	30	
5.S_ZDT4	2	30	
6.R_ZDT4	2	10	
7.S_ZDT6	2	30	
8.S_DTLZ2	3	30	150
	5	30	3000
9.R_DTLZ2	3	30	150

	5	30	3000
10.S_DTLZ3	3	30	150
	5	30	3000
11.WFG1	3	24*	150
	5	28*	3000
12.WFG8	3	24*	150
	5	28*	3000
13.WFG9	3	24*	150
	5	28*	3000

* The dimension of WFG test function is decided by $k+l$:

k : If $M = 2$, then let $k = 4$. Otherwise, if $M \geq 3$, let $k = 2 \times (M-1)$.

l : It is recommended that $l=20$ should be sufficient for the number of distance related parameters.

Therefore, for test functions (11, 12, 13) with 3 objectives ($M=3$), $k = 4$, $l = 20$, $D = k+l = 24$; with 5 objectives ($M=5$), $k = 8$, $l = 20$, $D = k+l = 28$.

Performance Metrics

1) R indicator (I_{R2}) [31]

$$I_{R2} = \frac{\sum_{\lambda \in \Lambda} u^*(\lambda, A) - u^*(\lambda, R)}{|\Lambda|}$$

where R is a reference set, u^* is the maximum value reached by the utility function u with weight vector λ on an approximation set A , i.e., $u^* = \max_{z \in A} u_\lambda(z)$. We choose the augmented Tchebycheff function as the utility function.

2) Hypervolume difference to a reference set ($I_{\bar{H}}$)[31]

The hypervolume indicator I_H measures the hypervolume of that portion of the objective space that is weakly dominated by an approximation set A , and is to be maximized. Here we consider the hypervolume difference to a reference set R , and we will refer to this indicator as $I_{\bar{H}}$, which is defined as $I_{\bar{H}} = I_H(R) - I_H(A)$ where smaller values correspond to higher quality - in contrast to the original hypervolume I_H . When calculating these two indicators, we need to normalize the objective values first.

3) Covered sets CS [33]

The above two indicators only refer to the objective space, and therefore we need one additional indicator that regards the Pareto sets of the SYMPART test function. The formal definitions refer to a population P of points (**ind**) in decision space and a set S of Pareto subsets (**set**). The Boolean function **near(ind,set)** becomes true if the tested individual reaches the vicinity of the tested set.

The number of covered Pareto subsets (which comprises of at least one individual in their vicinity) is:

$$CS(P, S) = |\{\text{set} \in S : \exists \text{ind} \in P, \text{near}(\text{ind}, \text{set})\}|$$

4) Attainment surface [31]

An approximation set A is called the $k\%$ -approximation set of an EAF $\alpha_r(z)$, if it weakly dominates exactly those objective vectors that have been attained in at least k percent of the r runs. Formally, $\forall z \in Z : \alpha_r(z) \geq k / 100 \Leftrightarrow A \preceq \{z\}$

$$\text{where } \alpha_r(z) = \frac{1}{r} \sum_{i=1}^r I(A^i \preceq \{z\})$$

A^i is the i th approximation set (run) of the optimizer and $I(\cdot)$ is the indicator function, which evaluates to one if its argument is true and zero if its argument is false.

An **attainment surface** of a given approximation set A is the union of all tightest goals that are known to be attainable as a result of A . Formally, this is the set $\{z \in \square^n : A \preceq z \wedge \neg A \prec z\}$

PC Configuration:

Windows XP Professional

Intel Pentium® 4 CPU 3.00 GHZ

1 GB of memory

Language: MATLAB 6.5

Parameters Setting:

The population size is set at 50.

The maximum external archive size is set at 100, 150 and 800 for 2, 3 and 5 objectives respectively, as specified in [68].

The maximum FES is set as $5e+5$.

Results Achieved:

For all the test functions, we recorded the approximate set after $5e+3$, $5e+4$, $5e+5$ FES in each run. According to these approximate sets, we calculated the performance metrics, R indicator I_{R2} and Hypervolume difference to a reference set $I_{\bar{H}}$. The best, median, worst result, mean value and standard deviation for the 25 runs are presented in **Table 5-8**, **Table 5-9**, **Table 5-10**, **Table 5-11**, **Table 5-12** and **Table 5-13**. Smaller $I_{\bar{H}}$ and I_{R2} values mean better results. For SYPMART problem, the numbers of covered Pareto subsets are shown in **Table 5-14**.

The 0%, 50% and 100% attainment surfaces after $5e+5$ FES on test functions 1-7 are presented in Figure 5-3. The 50% attainment surfaces after $5e+5$ FES on test functions 8-13(M=3) are shown in Figure 5-4. Finally, for test functions 12 and 13 (M=5), the median approximation sets after $5e+5$ FES with respect to the R indicator are used to plot the Pareto front in Figure 5-5. The objective function values are normalized to [1, 2].

Among the two-objective problems, S_ZDT1, S_ZDT2, S_ZDT4, and S_ZDT6, all the attainment surfaces obtained by MOSaDE algorithm overlapped the Pareto optimal front as shown in Figure 5-3. It demonstrates that MOSaDE successfully finds the non-dominated sets on these test problems.

With the characteristic of Pareto many-to-one, SYMPART with 30D is a very difficult problem. The MOSaDE failed to find the nine subsets, and only obtained one subset at $5e+5$ FES. R_ZDT4 is a rotated version based on the difficult problem ZDT4. After rotation, separable parameters become non-separable, so it is also hard to converge to the Pareto optimal front in $5e+5$ FES. The 0% attainment surface obtained approximates nearly to the Pareto optimal front after $5e+5$ FES, and the MOSaDE can converge to the Pareto optimal front if we enlarge FES.

Chapter 5: MOSaDE

With respect to three-objective problems, we only plot the 50% attainment surfaces. Comparing with the Pareto Optimal front provided by [68], we find that the 50% attainment surfaces of S_DTLZ2, S_DTLZ3, WFG1, WFG8 and WFG9 almost approximate to the Pareto optimal front. However, the MOSaDE algorithm could not find the non-dominated set on R_DTLZ2 in $5e+5$ FES. The difficulty in this problem is similar to R_ZDT4, due to the rotation. For five-objective problems, the Pareto fronts of WFG8 and WFG9 are similar to the Pareto Optimal fronts provided by [68].

Table 5-8: The results for R indicator on test functions 1-7.

FES		1.OKA2	2.SYMPART	3.S_ZDT1	4.S_ZDT2	5.S_ZDT4	6.R_ZDT4	7.S_ZDT6
5e+3	Best	1.2177E-01	1.7684E-03	1.9663E-02	4.4899E-02	4.3182E-02	1.4915E-02	1.0962E-01
	Media	1.2306E-01	3.9466E-03	2.3770E-02	5.5264E-02	5.9843E-02	2.1182E-02	1.1359E-01
	Worst	1.4756E-01	9.5696E-03	3.5158E-02	6.5406E-02	6.7866E-02	2.9358E-02	1.1937E-01
	Mean	1.2813E-01	4.2596E-03	2.5002E-02	5.5486E-02	5.8496E-02	2.1604E-02	1.1388E-01
	Std	9.0825E-03	1.9180E-03	4.2724E-03	4.8212E-03	6.7373E-03	4.0411E-03	2.4268E-03
5e+4	Best	1.2172E-01	7.2771E-05	-5.1796E-03	-8.2067E-03	3.3781E-03	-3.8100E-04	2.0186E-02
	Media	1.2221E-01	1.1314E-04	-5.0892E-03	-7.6591E-03	5.4790E-03	7.0833E-04	2.9874E-02
	Worst	1.2396E-01	3.7643E-04	-4.8313E-03	-6.9376E-03	7.7048E-03	2.4221E-03	4.0448E-02
	Mean	1.2234E-01	1.3952E-04	-5.0790E-03	-7.6630E-03	5.3110E-03	8.9153E-04	3.0298E-02
	Std	5.3280E-04	8.4855E-05	9.3273E-05	3.6085E-04	1.0551E-03	7.9243E-04	5.5756E-03
5e+5	Best	1.2165E-01	1.3164E-05	-5.2415E-03	-8.2830E-03	-2.5687E-04	-1.1048E-03	-8.9721E-06
	Media	1.2178E-01	2.1163E-05	-5.2002E-03	-8.2736E-03	-1.8372E-04	-7.6458E-04	-8.9721E-06
	Worst	1.2206E-01	4.7535E-05	-5.0933E-03	-8.1664E-03	-1.0172E-04	7.1316E-04	-8.7923E-06
	Mean	1.2181E-01	2.5315E-05	-5.1914E-03	-8.2653E-03	-1.5774E-04	-7.1405E-04	-8.9649E-06
	Std	1.0745E-04	9.6293E-06	3.6006E-05	2.9059E-05	2.1885E-05	3.8317E-04	3.5966E-08

Table 5-9: The results for R indicator on test functions 8-13 when M=3.

FES		8.S_DTLZ2	9.R_DTLZ2	10.S_DTLZ3	11.WFG1	12.WFG8	13.WFG9
5e+3	Best	3.0769E-05	6.0706E-04	4.1889E-04	8.3391E-02	1.1104E-02	1.0533E-02
	Median	7.8227E-05	7.5672E-04	5.7885E-04	8.6207E-02	1.6708E-02	1.5448E-02
	Worst	2.6172E-04	8.4139E-04	9.4378E-04	8.7996E-02	2.3806E-02	1.9425E-02
	Mean	1.0316E-04	7.4496E-04	6.0347E-04	8.6235E-02	1.7493E-02	1.4920E-02
	Std	5.7497E-05	5.1642E-05	1.2572E-04	1.2373E-03	3.6919E-03	2.9145E-03
5e+4	Best	-8.0685E-05	3.6652E-04	6.7448E-05	7.2435E-02	-8.3665E-04	6.1755E-03
	Median	-5.5214E-05	4.1699E-04	1.0706E-04	7.5951E-02	5.0192E-03	8.0829E-03
	Worst	-2.7533E-05	4.7685E-04	1.7487E-04	7.7127E-02	7.8241E-03	1.1940E-02
	Mean	-5.4515E-05	4.2021E-04	1.0731E-04	7.5350E-02	4.3046E-03	8.6810E-03
	Std	1.3840E-05	2.9248E-05	2.4787E-05	1.3818E-03	2.5328E-03	1.8284E-03
5e+5	Best	-8.6042E-05	1.0703E-04	1.0866E-07	2.6052E-02	-1.7980E-03	8.8631E-04
	Median	-8.5010E-05	1.9020E-04	4.4716E-07	2.8256E-02	-2.2479E-04	3.1352E-03
	Worst	-8.0692E-05	2.5153E-04	1.4871E-06	3.3207E-02	1.3283E-03	7.4165E-03
	Mean	-8.4712E-05	1.9330E-04	4.8313E-07	2.8693E-02	-2.6206E-04	3.6178E-03
	Std	1.3424E-06	3.4578E-05	3.6087E-07	1.8812E-03	9.1485E-04	1.8337E-03

Table 5-10: The results for R indicator on test functions 8-13 when M=5.

FES		8.S_DTLZ2	9.R_DTLZ2	10.S_DTLZ3	11.WFG1	12.WFG8	13.WFG9
5e+3	Best	1.3336E-04	2.1202E-04	1.9747E-04	5.7385E-02	1.4660E-02	8.4878E-03
	Median	1.6839E-04	2.3544E-04	2.5773E-04	5.9174E-02	2.1600E-02	1.1534E-02
	Worst	3.8511E-04	3.0566E-04	4.2791E-04	6.0612E-02	2.8726E-02	1.3687E-02
	Mean	1.9650E-04	2.3688E-04	2.7026E-04	5.9179E-02	2.1266E-02	1.1421E-02
	Std	6.0987E-05	1.9705E-05	5.5611E-05	8.1770E-04	3.8574E-03	1.4277E-03

Chapter 5: MOSaDE

5e+4	Best	1.8391E-05	1.0135E-04	1.6180E-05	5.0514E-02	9.1736E-03	5.7543E-03
	Median	3.5094E-05	1.2184E-04	3.4727E-05	5.5356E-02	1.9606E-02	7.4126E-03
	Worst	7.0910E-05	1.4215E-04	9.6151E-05	5.7685E-02	2.7911E-02	9.4715E-03
	Mean	3.7089E-05	1.2100E-04	4.2957E-05	5.5157E-02	1.8656E-02	7.6963E-03
	Std	1.2620E-05	1.1243E-05	2.1612E-05	1.6084E-03	4.1945E-03	1.0227E-03
5e+5	Best	5.7002E-06	5.0254E-05	3.5863E-06	1.1880E-02	1.7697E-03	5.7519E-03
	Median	1.4981E-05	7.8001E-05	1.0035E-05	1.8273E-02	8.5071E-03	7.2272E-03
	Worst	2.2887E-05	9.4545E-05	1.3672E-05	4.4155E-02	2.4882E-02	9.2790E-03
	Mean	1.5043E-05	7.7357E-05	8.9693E-06	1.9670E-02	9.4860E-03	7.2382E-03
	Std	4.6046E-06	1.0133E-05	3.3796E-06	6.7516E-03	6.1573E-03	8.0575E-04

Table 5-11: The results for Hypervolume indicator $I_{\bar{H}}$ on test functions 1-7.

FES		1.OKA2	2.SYMPART	3.S_ZDT1	4.S_ZDT2	5.S_ZDT4	6.R_ZDT4	7.S_ZDT6
5e+3	Best	4.0430E-01	5.1009E-03	1.1656E-01	1.8672E-01	1.3436E-01	4.8504E-02	2.8042E-01
	Media	4.1008E-01	1.1372E-02	1.4093E-01	2.0403E-01	1.8737E-01	6.7952E-02	2.9448E-01
	Worst	4.6110E-01	2.7516E-02	1.7098E-01	2.2825E-01	2.0729E-01	9.4807E-02	3.1227E-01
	Mean	4.1900E-01	1.2278E-02	1.4052E-01	2.0602E-01	1.8023E-01	6.9446E-02	2.9534E-01
	Std	1.7409E-02	5.5157E-03	1.2141E-02	1.0743E-02	1.9549E-02	1.2505E-02	6.7608E-03
5e+4	Best	4.0222E-01	1.8548E-04	1.9084E-02	4.7350E-02	9.7538E-03	-1.4131E-03	5.5264E-02
	Media	4.0654E-01	3.0014E-04	1.9316E-02	4.7788E-02	1.6809E-02	2.9314E-03	7.7371E-02
	Worst	4.1398E-01	1.0645E-03	1.9864E-02	4.8344E-02	2.2977E-02	7.6108E-03	1.0372E-01
	Mean	4.0670E-01	3.7957E-04	1.9306E-02	4.7798E-02	1.6463E-02	3.2291E-03	7.9428E-02
	Std	3.3921E-03	2.4546E-04	1.8761E-04	2.8443E-04	3.1222E-03	2.4322E-03	1.3164E-02
5e+5	Best	4.0055E-01	1.3898E-05	1.8360E-02	4.6615E-02	-1.3326E-03	-3.9027E-03	-3.7748E-03
	Media	4.0118E-01	3.7170E-05	1.8497E-02	4.6625E-02	-1.1611E-03	-2.6122E-03	-3.7739E-03
	Worst	4.0246E-01	1.1321E-04	1.8788E-02	4.6654E-02	-2.0865E-04	3.1454E-03	-3.7694E-03
	Mean	4.0119E-01	4.7963E-05	1.8510E-02	4.6627E-02	-1.0907E-03	-2.3664E-03	-3.7736E-03
	Std	4.3621E-04	2.7781E-05	9.7891E-05	8.3720E-06	2.9270E-04	1.4530E-03	1.1596E-06

Table 5-12: The results for Hypervolume indicator $I_{\bar{H}}$ on test functions 8-13 when M=3.

FES		8.S_DTLZ2	9.R_DTLZ2	10.S_DTLZ3	11.WFG1	12.WFG8	13.WFG9
5e+3	Best	1.3747E-03	4.4937E-02	1.0848E-02	4.3010E-01	1.3159E-01	1.0739E-01
	Median	2.0021E-03	5.4917E-02	1.8267E-02	4.4549E-01	1.6707E-01	1.4751E-01
	Worst	2.9621E-03	6.6822E-02	2.2099E-02	4.5433E-01	1.9934E-01	1.6949E-01
	Mean	2.0669E-03	5.4997E-02	1.7672E-02	4.4487E-01	1.6701E-01	1.4371E-01
	Std	4.3587E-04	4.6743E-03	3.0935E-03	6.5585E-03	1.8507E-02	1.8795E-02
5e+4	Best	-6.1722E-06	4.3891E-03	6.4275E-05	3.8431E-01	4.2362E-02	6.9557E-02
	Median	3.8128E-05	8.4381E-03	1.3546E-04	4.0102E-01	6.4037E-02	8.7440E-02
	Worst	2.1332E-04	1.3433E-02	3.7531E-04	4.0818E-01	7.4500E-02	1.1175E-01
	Mean	5.5237E-05	8.9137E-03	1.6626E-04	3.9824E-01	6.2723E-02	8.9336E-02
	Std	5.6052E-05	2.7359E-03	9.4887E-05	6.9752E-03	8.6806E-03	1.0332E-02
5e+5	Best	-2.4256E-05	5.1132E-04	-1.5005E-07	1.5343E-01	3.0677E-02	3.5270E-02
	Median	-2.1585E-05	1.5024E-03	-1.4993E-07	1.6597E-01	3.8616E-02	4.7204E-02
	Worst	-1.4835E-05	2.6268E-03	-1.3663E-07	1.9422E-01	4.3305E-02	7.2178E-02
	Mean	-2.0953E-05	1.5259E-03	-1.4816E-07	1.6826E-01	3.7447E-02	5.0744E-02
	Std	2.4065E-06	5.9954E-04	3.7259E-09	1.0301E-02	3.3994E-03	1.1031E-02

Table 5-13: The results for Hypervolume indicator $I_{\bar{H}}$ on test functions 8-13 when M=5.

FES		8.S_DTLZ2	9.R_DTLZ2	10.S_DTLZ3	11.WFG1	12.WFG8	13.WFG9
5e+3	Best	1.0261E-03	6.7402E-03	4.2992E-03	6.5477E-01	3.5961E-01	2.4853E-02
	Median	1.8525E-03	8.7590E-03	7.0782E-03	6.7364E-01	4.3100E-01	3.2674E-01
	Worst	6.5608E-03	1.2737E-02	1.0250E-02	6.8906E-01	4.8182E-01	4.0094E-01
	Mean	2.2131E-03	8.9669E-03	7.2009E-03	6.7337E-01	4.3299E-01	1.9572E-01
	Std	1.1297E-03	1.4052E-03	1.4751E-03	8.6932E-03	3.0002E-02	1.6573E-01
5e+4	Best	-5.5011E-04	9.1593E-04	7.4462E-06	5.8901E-01	2.0229E-01	2.0723E-02
	Median	-3.3791E-05	1.1354E-03	6.0642E-05	6.3783E-01	3.1912E-01	1.7747E-01
	Worst	6.8931E-05	1.6931E-03	6.9166E-04	6.5983E-01	4.1771E-01	2.7531E-01
	Mean	-1.9121E-04	1.1850E-03	1.3146E-04	6.3612E-01	3.1404E-01	1.3173E-01

	Std	2.4464E-04	1.9711E-04	1.7155E-04	1.5847E-02	5.2906E-02	1.0844E-01
5e+5	Best	-5.6794E-04	2.0609E-04	7.7695E-08	1.7538E-01	4.7156E-02	2.0723E-02
	Median	-3.9378E-05	4.1727E-04	1.4625E-06	2.5598E-01	1.3097E-01	1.4722E-01
	Worst	1.8010E-05	6.1831E-04	4.3906E-06	5.3614E-01	3.7756E-01	2.3139E-01
	Mean	-2.5991E-04	4.1870E-04	1.6230E-06	2.7218E-01	1.5836E-01	1.0625E-01
	Std	2.7681E-04	9.9036E-05	1.2085E-06	7.5849E-02	9.1277E-02	8.4222E-02

Table 5-14: The results for Covered sets CS for test function SYMPART.

FES	5e+3	5e+4	5e+5
Best	2	1	1
Median	1	1	1
Worst	1	1	1
Mean	1.04	1	1
Std	0.2	0	0

2. SYMPART

3. S_ZDT1

4. S_ZDT2

5. S_ZDT4

Figure 5-3: Pareto optimal front and 0%, 50%, 100% attainment surfaces after $5e+5$ FES on test functions 1-7.

8. S_DTLZ2

9. R_DTLZ2

12. WFG8

13. WFG9

Figure 5-4: 50% attainment surfaces after $5e+5$ FES on test function s 8-13($M=3$).

Figure 5-5: Pareto front after 5e+5 on test functions 12 and 13 (M=5)

5.5.2 Experimental results of OW-MOSaDE

Test problems:

We evaluate the performance of the proposed Objective-Wise MOSaDE algorithm (OW-MOSaDE) and the original MOSaDE on a set of 13 benchmark functions (UP1-UP13) chosen from [131], which include 7 two-objective test functions (UP1-UP7), 3 three-objective test functions (UP8-UP10) and 3 five-objective test functions (UP11-13).

Parameters Setting:

The population size is set at 50. The maximum external archive size is set at 100, 150 and 800 for 2, 3 and 5 objectives respectively, as specified in [131]. The maximum FES is set as 3e+5.

Performance Metric: IGD

Let P^* be a set of uniformly distributed points along the Pareto Front (PF) in the objective space. Let A be an approximate set to the PF, the average distance from P^* to A is defined as:

$$IGD(A, P^*) = \frac{\sum_{v \in P^*} d(v, A)}{|P^*|}$$

where $d(v, A)$ is the minimum Euclidean distance between v and the points in A . If $|P^*|$ is large enough to represent the PF very well, $IGD(A, P^*)$ could measure both the diversity and convergence of A in a sense. To have a low value of $IGD(A, P^*)$, The set A must be very close to the PF and cannot miss any part of the whole PF.

Results Achieved:

For all the test functions, we recorded the approximate set every generation in each run. According to these approximate sets, we calculated the performance metric Inverted Generational Distance (IGD). The smallest, largest, mean and standard deviation of the IGD values obtained for each test instance of the 30 runs are presented in Table 5-15 for OW-MOSaDE and the mean of the IGD values obtained by original MOSaDE is also shown in the table. Smaller IGD values indicate better results. For the OW-MOSaDE, the plot of the final approximation set with the smallest IGD value in the objective space for the test instances with 2 and 3 objectives are shown in Figure 5-6.

From the IGD values shown in Table 5-15, among all the 13 test problems, OW-MOSaDE performs better than the original MOSaDE on all the 13 benchmark test functions. It demonstrates that OW-MOSaDE improves upon the search ability of the original MOSaDE by using the new objective-wise learning strategy instead of the non-domination-wise learning strategy when solving these multi-objective test problems.

Table 5-15: The average IGD value of the 30 final approximation sets obtained for each test problem.

Fun	OW-MOSaDE				Original-MOSaDE
	Smallest (IGD)	Largest (IGD)	Mean (IGD)	Std (IGD)	Mean (IGD)
1. UF1	0.0103	0.0152	0.0122	0.0012	0.0983
2. UF2	0.0056	0.0135	0.0081	0.0023	0.0607
3. UF3	0.0727	0.1512	0.1030	0.0190	0.3248
4. UF4	0.0483	0.0562	0.0513	0.0019	0.0977
5. UF5	0.3991	0.4766	0.4303	0.0174	0.6963
6. UF6	0.0926	0.2271	0.1918	0.0290	0.3640
7. UF7	0.0208	0.1576	0.0585	0.0291	0.1916
8. UF8	0.0800	0.1199	0.0945	0.0119	0.4019
9. UF9	0.0574	0.1621	0.0983	0.0244	0.3984
10. UF10	0.5777	0.9402	0.7430	0.0885	2.9313
11. UF11	0.3360	0.4724	0.3951	0.0384	0.5450
12. UF12	593.3990	812.4774	734.5680	65.0659	783.2456
13. UF13	2.8298	4.5500	3.2573	0.3521	3.6881

(1) UF1

(2) UF2

(3) UF3

(4) UF4

(5) UF5

(6) UF6

(7) UF7

(8) UF8

(9) UF9

(10) UF10

Figure 5-6: The plot of the final approximation set with the smallest IGD value in the objective space for each test instances UF1-UF13.

5.6 Conclusion

In this chapter, we extended the Self-adaptive Differential Evolution (SaDE) algorithm for solving optimization problems with multiple objectives. Taking the SaDE algorithm from single-objective optimization to multi-objective optimization, we have modified the selection criteria to evaluate promising or inferior solutions, and incorporated a secondary population (external archive) to store the set of non-dominated solutions. Our approach also adopted the concept of ϵ -dominance and harmonic mean distance to produce well-distributed non-dominated solutions. We evaluated the performance of the proposed approach on the test bed from the CEC2007 special session on Performance Assessment of Multi-Objective Optimization Algorithms.

Finally, we enhanced the multi-objective self-adaptive differential evolution algorithm with objective-wise learning of parameters and mutation strategies. We assigned an individual set of parameters and strategies to each objective for optimizing each objective. The effectiveness of the objective-wise learning approach was evaluated on the test benchmark functions from CEC2009 special session on Performance Assessment of Constrained / Bound Constrained Multi-Objective Optimization Algorithms. The experimental results illustrated that objective-wise learning approach improved the performance of original MOSaDE.

Chapter 6 Conclusion and Future Work

6.1 Conclusion

This thesis investigated the improvement of Differential Evolution (DE) for global optimization. Our conclusions can be summarized as follows:

We conducted a thorough literature survey on DE, going through its origin, fundamental rationale, improved DE variants and applications. To avoid the expensive computational costs spent on selecting the most appropriate trial vector generation strategy as well as its associated parameter values in DE implementation, we proposed the Self-adapted Differential Evolution (SaDE) algorithm. Both trial vector generation strategies and their associated parameters are gradually self-adapted by learning from previous experiences of generating promising solutions. Consequently, a more suitable strategy along with its parameter settings can be determined adaptively to suit different stages of evolution.

We evaluated the performance of SaDE over a suite of 26 bound constrained numerical optimization problems, and compared it with conventional DE and three other adaptive DE variants. It was concluded that SaDE was more effective in obtaining better quality solutions, more stable with relatively smaller standard deviations, and had higher success rates, which was also demonstrated by the overall performance comparisons. Furthermore, we investigated the self-adaptive characteristics of the CR value and trial vector generation strategies. The experimental results verified the promising performance of SaDE to handle problems with distinct properties such as uni-modality and multi-modality, and showed that the SaDE algorithm could evolve suitable strategies and parameter values during evolution progress. Finally, we studied the impact of the parameter, learning period LP , on the performance of SaDE. The sensitivity analysis of the parameter LP indicated that the convergence speed of SaDE algorithm has no significant change when the parameter LP values between 20 and 60.

We generalized the SaDE algorithm for handling optimization problem with constraints, without introducing any additional parameters. The performance of our

approach was evaluated on the testbed for Congress on Evolutionary Computation 2006 (CEC2006) special session on constrained real parameter optimization. Under these conditions, the SaDE algorithm demonstrated effectiveness and robustness.

We analyzed the limitations of Zitzler-Deb-Thiele (ZDT) and Deb-Thiele-Laumanns-Zitzler (DTLZ) test suites: 1) the global optimum has the same parameter values for different variables/dimensions; 2) the global optimum lies in the center of the search range; 3) the global optimum lies on the bounds. Moreover, all these test problems are separable. To overcome these limitations, we shifted or rotated the original ZDT and DTLZ problem to obtain S_ZDT and S_DTLZ test suites.

We extended the SaDE algorithm from single-objective optimization domain to multi-objective optimization domain, by modifying the selection criteria to evaluate promising or inferior solutions, and added a secondary population (external archive) to store the set of non-dominated solutions. Our approach also adopted the concept of ϵ -dominance and harmonic mean distance to produce well-distributed non-dominated solutions. We evaluated the performance of the proposed approach on the test bed for CEC2007 special session on Performance Assessment of Multi-Objective Optimization Algorithms.

Finally, we enhanced the Multi-Objective Self-adaptive Differential Evolution (MOSaDE) algorithm with objective-wise learning of parameters and mutation strategies. We assigned an individual set of parameters and strategies to each objective for optimizing each objective value. The effectiveness of the objective-wise learning approach was evaluated on the test benchmark functions from CEC2009 special session on Performance Assessment of Constrained / Bound Constrained Multi-Objective Optimization Algorithms. The experimental results illustrated that objective-wise learning approach improved the performance of the original MOSaDE.

6.2 Recommendations of future research

Based on the research work conducted in this thesis, research directions that deserve future investigation are recommended as follows:

- Investigate the strategy pool

Generally speaking, a good candidate pool should be restrictive so that the unfavorable influences of less effective strategies can be suppressed. Moreover, a set of effective strategies contained in a good candidate pool should have diverse characteristics, that is, the used strategies should demonstrate distinct capabilities when dealing with a specific problem at different stages of evolution. The theoretical study on the choice of the optimal pool size and the selection of strategies used in the pool are attractive research issues and deserve further investigation.

- Self-adaption of the population size of DE

The population size is also a crucial parameter of DE. Adapting the population size of SaDE so as to render it as non-user defined parameter is greatly appreciated. In literature, various adaptive population size schemes have been proposed for genetic algorithms such as Genetic Algorithms with Varying Population Size (GAVaPS) [133] which adapted population size based on the concept of age and lifetime of an individual. Schlierkamp-Voosen and Muhlenbein [134] proposed an adaptive scheme which simulates the population changes of species which compete for the same resource. Well adapted species increase in size while species that adapted poorly decrease. These ideas can be extended into SaDE to self-adapt the population size.

- Apply OW-MOSaDE to solve financial optimization problems

Financial portfolio optimization is a challenging problem with two competing objectives, to minimize the risk of the portfolio while at the same time, maximizing the return of the portfolio. Moreover, portfolio managers often face real-world constraints, which are typically non-linear. Hence, conventional optimization techniques, such as quadratic programming, cannot be used. Heuristic search methods can be an attractive alternative. Figure 6-1 and Figure 6-2 are examples of the Pareto fronts for unconstrained portfolio optimization and constrained portfolio optimization problems respectively. We plan to apply OW-MOSaDE to solve these problems in future work.

Figure 6-1 Pareto Front of an example of unconstrained portfolio optimization

Figure 6-2 Pareto Front of an example of constrained portfolio optimization

- Employ the self-adaptive scheme to other population-based algorithm

The proposed self-adaptive schemes are general enough and can be utilized to adapt the parameter or strategy of other population-based algorithms such as Genetic Algorithms (GA) or Particle Swarm Optimization (PSO).

List Of Publications

JOURNAL

- 1 V. L. Huang, P. N. Suganthan and J. J. Liang, "Comprehensive learning particle Swarm optimizer for solving multiobjective optimization problems," *Int. J of Int. Systems*, Vol. 21(2), pp. 209-226, 2006.
- 2 A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. on Evolutionary Computations*, DOI: 10.1109/TEVC.2008.927706, 2009.
- 3 J. J. Liang, C. C. Chan, P. N. Suganthan and V. L. Huang, "Wavelength detection in FBG sensor network using tree search dynamic multi-swarm particle swarm optimizer," *IEEE Photonics Technol. Letters*, vol. 18(12), pp.1305-1307, June 2006.
- 4 J. J. Liang, C. C. Chan, V. L. Huang and P. N. Suganthan, "Improving the performance of a FBG sensor network using a novel dynamic multi-swarm particle swarm optimizer," *OAM Rapid-Comm.*, vol. 1(8), pp. 373-378, Aug 2007.

CONFERENCE

- 1 V. L. Huang, A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for constrained real-parameter optimization," *2006 IEEE Congress on Evolutionary Computation (CEC'06)*, Vancouver, BC, Canada, July 2006.
- 2 V. L. Huang, A. K. Qin, P. N. Suganthan and M. F. Tasgetiren, "Multi-objective optimization based on self-adaptive differential evolution Algorithm," *2007 IEEE Congress on Evolutionary Computation (CEC'07)*, Singapore, September 2007.
- 3 J. J. Liang, C. C. Chan, V. L. Huang and P. N. Suganthan, "Improving the performance of a FBG sensor network using a novel dynamic multi-swarm

- particle swarm optimizer,” *SPIE Symposium on Optics East*, Boston, Massachusetts USA, October 2005.
- 4 V. L. Huang, S. Z. Zhao, R. Mallipeddi and P. N. Suganthan, “Multiobjective Self-adaptive Differential Evolution algorithm with objective-wise learning strategies,” *2009 IEEE Congress on Evolutionary Computation (CEC'09)*, Trondheim, Norway, May 2009.

TECHNICAL REPORTS

- 1 V. L. Huang, P. N. Suganthan, A. K. Qin, and S. Baskar, “Multiobjective differential evolution with external archive and harmonic distance-based diversity measure”, *Technical Report*, Nanyang Technological University, Singapore, 2005.
- 2 V. L. Huang, A. K. Qin, K. Deb, E. Zitzler, P. N. Suganthan, J. J. Liang, M. Preuss and S. Huband, “Problem definitions for performance assessment of multi-objective optimization algorithms,” Special Session on Performance Assessment of Multi-Objective Optimization Algorithms, *Technical Report*, Nanyang Technological University, Singapore, 2007.

APPENDIX

A. Definitions of the CEC'06 Test Functions

g01

Minimize

$$f(\mathbf{x}) = 5 \sum_{i=1}^4 x_i - 5 \sum_{i=1}^4 x_i^2 - \sum_{i=5}^{13} x_i$$

subject to:

$$g_1(\mathbf{x}) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0$$

$$g_2(\mathbf{x}) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0$$

$$g_3(\mathbf{x}) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0$$

$$g_4(\mathbf{x}) = -8x_1 + x_{10} \leq 0$$

$$g_5(\mathbf{x}) = -8x_2 + x_{11} \leq 0$$

$$g_6(\mathbf{x}) = -8x_3 + x_{12} \leq 0$$

$$g_7(\mathbf{x}) = -2x_4 - x_5 + x_{10} \leq 0$$

$$g_8(\mathbf{x}) = -2x_6 - x_7 + x_{11} \leq 0$$

$$g_9(\mathbf{x}) = -2x_8 - x_9 + x_{12} \leq 0$$

where the bounds are $0 \leq x_i \leq 100$ ($i = 10, 11, 12$) and $0 \leq x_{13} \leq 1$. The gloval minimum is at $\mathbf{x}^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$ where six constraints are active ($g_1, g_2, g_3, g_7, g_8, g_9$) and $f(\mathbf{x}^*) = -15$.

g02

Minimize:

$$f(\mathbf{x}) = - \left| \frac{\sum_{i=1}^D \cos^4(x_i) - 2 \prod_{i=1}^D \cos^2(x_i)}{\sqrt{\sum_{i=1}^D i x_i^2}} \right|$$

subject to:

$$g_1(\mathbf{x}) = 0.75 - \prod_{i=1}^D x_i \leq 0$$

$$g_2(\mathbf{x}) = \sum_{i=1}^D x_i - 7.5D \leq 0$$

where $D=20$ and $0 \leq x_i \leq 10$ ($i = 1, \dots, D$). The global minimum $\mathbf{x}^* = (3.16246061572185, 3.12833142812967, 3.09479212988791, 3.06145059523469, 3.02792915885555, 2.99382606701730, 2.95866871765285, 2.92184227312450, 0.49482511456933, 0.48835711005490, 0.48231642711865, 0.47664475092742, 0.47129550835493, 0.46623099264167, 0.46142004984199, 0.45683664767217, 0.45245876903267, 0.44826762241853, 0.44424700958760, 0.44038285956317)$, $f(\mathbf{x}^*) = -0.80361910412559$.

g03

Minimize

$$f(\mathbf{x}) = -(\sqrt{D})^D \prod_{i=1}^D x_i \quad (-1)$$

subject to:

$$h_1(\mathbf{x}) = \sum_{i=1}^D x_i^2 - 1 = 0$$

where $D = 10$ and $0 \leq x_i \leq 1$ ($i = 1, \dots, D$). The global minimum is at $\mathbf{x}^* = (0.31624357647283069, 0.316243577414338339, 0.316243578012345927, 0.316243575664017895, 0.316243578205526066, 0.31624357738855069, 0.316243575472949512, 0.316243577164883938, 0.316243578155920302, 0.316243576147374916)$, where $f(\mathbf{x}^*) = -1.00050010001000$.

g04

Minimize:

$$f(\mathbf{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141 \quad (-2)$$

subject to:

$$g_1(\mathbf{x}) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0$$

$$g_2(\mathbf{x}) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0$$

$$g_3(\mathbf{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 0$$

$$g_4(\mathbf{x}) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0$$

$$g_5(\mathbf{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0$$

$$g_6(\mathbf{x}) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0$$

where $76 \leq x_1 \leq 102$, $33 \leq x_2 \leq 45$ and $27 \leq x_i \leq 45$ ($i = 3, 4, 5$). The optimum solution is $\mathbf{x}^* = (78, 33, 29.9952560256815985, 45, 36.7758129057882073)$, where $f(\mathbf{x}^*) = -3.066553867178332e+004$.

g05:

Minimize:

$$f(\mathbf{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.00002/3)x_2^3 \quad (-3)$$

subject to:

$$g_1(\mathbf{x}) = -x_4 + x_3 - 0.55 \leq 0$$

$$g_2(\mathbf{x}) = -x_3 + x_4 - 0.55 \leq 0$$

$$h_3(\mathbf{x}) = 1000 \sin(-x_3 - 0.25) + 1000 \sin(-x_4 - 0.25) + 894.8 - x_1 = 0$$

$$h_4(\mathbf{x}) = 1000 \sin(x_3 - 0.25) + 1000 \sin(x_4 - 0.25) + 894.8 - x_2 = 0$$

$$h_5(\mathbf{x}) = 1000 \sin(x_4 - 0.25) + 1000 \sin(x_4 - x_3 - 0.25) + 1294.8 = 0$$

where $0 \leq x_1 \leq 1200$, $0 \leq x_2 \leq 1200$, $-0.55 \leq x_3 \leq 0.55$ and $-0.55 \leq x_4 \leq 0.55$. The best known solution $\mathbf{x}^* = ((679.945148297028709, 1026.06697600004691, 0.118876369094410433, -0.39623348521517826))$ where $f(\mathbf{x}^*) = 5126.4967140071$.

g06:

Minimize:

APPENDIX

$$f(\mathbf{x}) = (x_1 - 10)^3 + (x_2 - 20)^3 \quad (-4)$$

subject to:

$$g_1(\mathbf{x}) = -(x_1 - 10)^3 - (x_2 - 20)^3 + 100 \leq 0$$

$$g_2(\mathbf{x}) = (x_1 - 6)^3 - (x_2 - 5)^3 + 82.81 \leq 0$$

where $13 \leq x_1 \leq 100$ and $0 \leq x_2 \leq 100$. The optimum solution is $\mathbf{x}^* = (14.09500000000000064, 0.8429607892154795668)$, where $f(\mathbf{x}^*) = -6961.81387558015$. Both constraints are active.

g07:

Minimize:

$$f(\mathbf{x}) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45 \quad (-5)$$

subject to:

$$g_1(\mathbf{x}) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0$$

$$g_2(\mathbf{x}) = -10x_1 - 8x_2 - 17x_7 + 10x_7 + 2x_8 \leq 0$$

$$g_3(\mathbf{x}) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0$$

$$g_4(\mathbf{x}) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0$$

$$g_5(\mathbf{x}) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0$$

$$g_6(\mathbf{x}) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0$$

$$g_7(\mathbf{x}) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0$$

$$g_8(\mathbf{x}) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0$$

where $-10 \leq x_i \leq 10$ ($i = 1, \dots, 10$). The optimum solution is $\mathbf{x}^* = (2.17199634142692, 2.3636830416034, 8.77392573913157, 5.09598443745173, 0.990654756560493, 1.43057392853463, 1.32164415364306, 9.82872576524495, 8.2800915887356, 8.3759266477347)$ where $f(\mathbf{x}^*) = 24.30620906818$. (The provided results may suffer from rounding errors which may cause slight infeasibility some times in the best given solutions). Six constraints are active ($g_1, g_2, g_3, g_4, g_5, g_6$).

g08

Minimize:

$$f(\mathbf{x}) = -\frac{\sin^3(2\pi x_1) \sin(2\pi x_2)}{x_1^3(x_1 + x_2)} \quad (-6)$$

subject to:

$$g_1(\mathbf{x}) = x_1^2 - x_2 + 1 \leq 0$$

$$g_2(\mathbf{x}) = 1 - x_1 + (x_2 - 4)^2 \leq 0$$

where $0 \leq x_1 \leq 10$ and $0 \leq x_2 \leq 10$. The optimum is located at $\mathbf{x}^* = (1.22797135260752599, 4.24537336612274885)$ where $f(\mathbf{x}^*) = -0.0958250414180359$.

g09

Minimize:

APPENDIX

$$f(\mathbf{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7 \quad (-7)$$

subject to:

$$g_1(\mathbf{x}) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0$$

$$g_2(\mathbf{x}) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0$$

$$g_3(\mathbf{x}) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0$$

$$g_4(\mathbf{x}) = 4x_1^2 + x_2^2 - x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0$$

where $-10 \leq x_i \leq 10$ ($i = 1, \dots, 7$). The optimum solution is $\mathbf{x}^* = (2.33049935147405174, 1.95137236847114592, -0.477541399510615805, 4.36572624923625874, -0.624486959100388983, 1.03813099410962173, 1.5942266780671519)$ where $f(\mathbf{x}^*) = 680.630057374402$. Two constraints are active (g_1 and g_4).

g10

Minimize:

$$f(\mathbf{x}) = x_1 + x_2 + x_3 \quad (-8)$$

subject to:

$$g_1(\mathbf{x}) = -1 + 0.0025(x_4 + x_6) \leq 0$$

$$g_2(\mathbf{x}) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0$$

$$g_3(\mathbf{x}) = -1 + 0.01(x_8 - x_5) \leq 0$$

$$g_4(\mathbf{x}) = -x_1x_6 + 833.33252x_4 + 100x_1 - 83333.333 \leq 0$$

$$g_5(\mathbf{x}) = -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leq 0$$

$$g_6(\mathbf{x}) = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leq 0$$

where $100 \leq x_1 \leq 10000$, $1000 \leq x_i \leq 10000$ ($i = 2, 3$) and $10 \leq x_i \leq 1000$ ($i = 4, \dots, 8$). The optimum solution is $\mathbf{x}^* = (579.306685017979589, 1359.97067807935605, 5109.97065743133317, 182.01769963061534, 295.601173702746792, 217.982300369384632, 286.41652592786852, 395.601173702746735)$, where $f(\mathbf{x}^*) = 7049.24802052867$. All constraints are active.

g11

Minimize:

$$f(\mathbf{x}) = x_1^2 + (x_2 - 1)^2 \quad (-9)$$

subject to:

$$h(\mathbf{x}) = x_2 - x_1^2 = 0$$

where $-1 \leq x_1 \leq 1$ and $-1 \leq x_2 \leq 1$. The optimum solution is $\mathbf{x}^* = (-0.707036070037170616, 0.500000004333606807)$ where $f(\mathbf{x}^*) = 0.7499$.

g12

Minimize:

$$f(\mathbf{x}) = -(100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2)/100 \quad (-10)$$

APPENDIX

subject to:

$$g(\mathbf{x}) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0$$

where $0 \leq x_i \leq 10$ ($i = 1, 2, 3$) and $p, q, r = 1, 2, \dots, 9$. The feasible region of the search space consists of 9^3 disjointed spheres. A point (x_1, x_2, x_3) is feasible if and only if there exist p, q, r such that the above inequality holds. The optimum is located at $\mathbf{x}^* = (5, 5, 5)$ where $f(\mathbf{x}^*) = -1$.

g13

Minimize:

$$f(\mathbf{x}) = e^{x_1 x_2 x_3 x_4 x_5} \quad (-11)$$

subject to:

$$h_1(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$$

$$h_2(\mathbf{x}) = x_2 x_3 - 5 x_4 x_5 = 0$$

$$h_3(\mathbf{x}) = x_1^3 + x_2^3 + 1 = 0$$

where $-2.3 \leq x_i \leq 2.3$ ($i = 1, 2$) and $-3.2 \leq x_i \leq 3.2$ ($i = 3, 4, 5$). The optimum solution is $\mathbf{x}^* = (-1.71714224003, 1.59572124049468, 1.8272502406271, -0.763659881912867, -0.76365986736498)$ where $f(\mathbf{x}^*) = 0.053941514041898$.

g14

Minimize:

$$f(\mathbf{x}) = \sum_{i=1}^{10} x_i (c_i + \ln(x_i / \sum_{j=1}^{10} x_j)) \quad (-12)$$

subject to:

$$h_1(\mathbf{x}) = x_1 + 2x_2 + 2x_3 + x_6 + x_{10} - 2 = 0$$

$$h_2(\mathbf{x}) = x_4 + 2x_5 + x_6 + x_7 - 1 = 0$$

$$h_3(\mathbf{x}) = x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 = 0$$

where the bounds are $0 < x_i \leq 10$ ($i = 1, \dots, 10$), and

$$c_1 = -6.089, c_2 = -17.164, c_3 = -34.054, c_4 = -5.914, c_5 = -24.721, c_6 = -14.986,$$

$$c_7 = -24.1, c_8 = -10.708, c_9 = -26.662, c_{10} = -22.179. \text{ The best solution is at}$$

$$\mathbf{x}^* = (0.0406684113216282, 0.147721240492452, 0.783205732104114,$$

$$0.00141433931889084, 0.485293636780388, 0.000693183051556082,$$

$$0.0274052040687766, 0.0179509660214818, 0.0373268186859717,$$

$$0.0968844604336845) \text{ where } f(\mathbf{x}^*) = -47.7648884594915.$$

g15

Minimize:

$$f(\mathbf{x}) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1 x_2 - x_1 x_3 \quad (-13)$$

subject to:

$$h_1(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2 - 25 = 0$$

$$h_2(\mathbf{x}) = 8x_1 + 14x_2 + 7x_3 - 56 = 0$$

where the bounds are $0 \leq x_i \leq 10$ ($i=1,2,3$). The best known solution is at $\mathbf{x}^* = (3.51212812611795133, 0.216987510429556135, 3.55217854929179921)$ where $f(\mathbf{x}^*) = 961.715022289961$.

g16

Minimize:

$$f(\mathbf{x}) = 0.000117y_{14} + 0.1365 + 0.00002358y_{13} + 0.000001502y_{16} + 0.0321y_{12} \\ + 0.004324y_5 + 0.0001\frac{c_{15}}{c_{16}} + 37.48\frac{y_2}{c_{12}} - 0.0000005843y_{17} \quad (-14)$$

subject to:

$$g_1(\mathbf{x}) = \frac{0.28}{0.72}y_5 - y_4 \leq 0$$

$$g_2(\mathbf{x}) = x_3 - 1.5x_2 \leq 0$$

$$g_3(\mathbf{x}) = 3496\frac{y_2}{c_{12}} - 1.5x_2 \leq 0$$

$$g_4(\mathbf{x}) = 110.6 + y_1 - \frac{62212}{c_{17}} \leq 0$$

$$g_5(\mathbf{x}) = 213.1 - y_1 \leq 0$$

$$g_6(\mathbf{x}) = y_1 - 405.23 \leq 0$$

$$g_7(\mathbf{x}) = 17.505 - y_2 \leq 0$$

$$g_8(\mathbf{x}) = y_2 - 1053.6667 \leq 0$$

$$g_9(\mathbf{x}) = 11.275 - y_3 \leq 0$$

$$g_{10}(\mathbf{x}) = y_3 - 35.03 \leq 0$$

$$g_{11}(\mathbf{x}) = 214.228 - y_4 \leq 0$$

$$g_{12}(\mathbf{x}) = y_4 - 665.585 \leq 0$$

$$g_{13}(\mathbf{x}) = 7.458 - y_5 \leq 0$$

$$g_{14}(\mathbf{x}) = y_5 - 584.463 \leq 0$$

$$g_{15}(\mathbf{x}) = 0.961 - y_6 \leq 0$$

$$g_{16}(\mathbf{x}) = y_6 - 265.916 \leq 0$$

$$g_{17}(\mathbf{x}) = 1.612 - y_7 \leq 0$$

$$g_{18}(\mathbf{x}) = y_7 - 7.046 \leq 0$$

$$g_{19}(\mathbf{x}) = 0.146 - y_8 \leq 0$$

$$g_{20}(\mathbf{x}) = y_8 - 0.222 \leq 0$$

$$g_{21}(\mathbf{x}) = 107.99 - y_9 \leq 0$$

$$g_{22}(\mathbf{x}) = y_9 - 273.366 \leq 0$$

$$g_{23}(\mathbf{x}) = 922.693 - y_{10} \leq 0$$

$$g_{24}(\mathbf{x}) = y_{10} - 1286.105 \leq 0$$

$$g_{25}(\mathbf{x}) = 922.832 - y_{11} \leq 0$$

APPENDIX

$$g_{26}(\mathbf{x}) = y_{11} - 1444.046 \leq 0$$

$$g_{27}(\mathbf{x}) = 18.766 - y_{12} \leq 0$$

$$g_{28}(\mathbf{x}) = y_{12} - 537.141 \leq 0$$

$$g_{29}(\mathbf{x}) = 1072.163 - y_{13} \leq 0$$

$$g_{30}(\mathbf{x}) = y_{13} - 3247.039 \leq 0$$

$$g_{31}(\mathbf{x}) = 8961.448 - y_{14} \leq 0$$

$$g_{32}(\mathbf{x}) = y_{14} - 26844.086 \leq 0$$

$$g_{33}(\mathbf{x}) = 0.063 - y_{15} \leq 0$$

$$g_{34}(\mathbf{x}) = y_{15} - 0.386 \leq 0$$

$$g_{35}(\mathbf{x}) = 71084.33 - y_{16} \leq 0$$

$$g_{36}(\mathbf{x}) = y_{16} - 140000 \leq 0$$

$$g_{37}(\mathbf{x}) = 2802713 - y_{17} \leq 0$$

$$g_{38}(\mathbf{x}) = y_{17} - 12146108 \leq 0$$

where:

$$y_1 = x_2 + x_3 + 41.6$$

$$c_1 = 0.024x_4 - 4.62$$

$$y_2 = \frac{12.5}{c_1} + 12$$

$$c_2 = 0.0003535x_1^2 + 0.5311x_1 + 0.08705y_2x_1$$

$$c_3 = 0.052x_1 + 78 + 0.02377y_2x_1$$

$$y_3 = \frac{c_2}{c_3}$$

$$y_4 = 19y_3$$

$$c_4 = 0.04782(x_1 - y_3) + \frac{0.1956(x_1 - y_3)^2}{x_2} + 0.6376y_4 + 1.594y_3$$

$$c_5 = 100x_2$$

$$c_6 = x_1 - y_3 - y_4$$

$$y_5 = c_6c_7$$

$$y_6 = x_1 - y_5 - y_4 - y_3$$

$$y_7 = \frac{c_8}{y_1}$$

$$y_8 = \frac{c_8}{3798}$$

$$c_9 = y_7 - \frac{0.0663y_7}{y_8} - 0.3153$$

$$y_9 = \frac{96.82}{c_9} + 0.321y_1$$

$$y_{10} = 1.29y_5 + 1.258y_4 + 2.29y_3 + 1.71y_6$$

$$y_{11} = 1.71x_1 - 0.452y_4 + 0.58y_3$$

$$c_{10} = \frac{12.3}{752.3}$$

$$c_{11} = (1.75y_2)(0.995x_1)$$

$$c_{12} = 0.995y_{10} + 1998$$

$$y_{12} = c_{10}x_1 + \frac{c_{11}}{c_{12}}$$

$$y_{13} = c_{12} - 1.75y_2$$

$$y_{14} = 3623 + 64.4x_2 + 58.4x_3 + \frac{146312}{y_9 + x_5}$$

$$c_{13} = 0.995y_{10} + 60.8x_2 + 48x_4 - 0.1121y_{14} - 5095$$

$$y_{15} = \frac{y_{13}}{c_{13}}$$

$$y_{16} = 148000 - 331000y_{15} + 40y_{13} - 61y_{15}y_{13}$$

$$c_{14} = 2324y_{10} - 28740000y_2$$

$$y_{17} = 14130000 - 1328y_{10} - 531y_{11} + \frac{c_{14}}{c_{12}}$$

$$c_{15} = \frac{y_{13}}{y_{15}} - \frac{y_{13}}{0.52}$$

$$c_{16} = 1.104 - 0.72y_{15}$$

$$c_{17} = y_9 + x_5$$

and where the bounds are $704.4148 \leq x_1 \leq 906.3855$, $68.6 \leq x_2 \leq 288.88$,

$0 \leq x_3 \leq 134.75$, $193 \leq x_4 \leq 287.0966$ and $25 \leq x_5 \leq 84.1988$. The best known solution is at $\mathbf{x}^* = (705.174537070090537, 68.5999999999999943, 102.899999999999991, 282.324931593660324, 37.5841164258054832)$ where $f(\mathbf{x}^*) = -1.90515525853479$.

g17

Minimize:

$$f(\mathbf{x}) = f(x_1) + f(x_2) \quad (-15)$$

where

$$f_1(x_1) = \begin{cases} 30x_1 & 0 \leq x_1 < 300 \\ 31x_1 & 300 \leq x_1 < 400 \end{cases}$$

$$f_2(x_2) = \begin{cases} 28x_2 & 0 \leq x_2 < 100 \\ 29x_2 & 100 \leq x_2 < 200 \end{cases}$$

subject to

$$h_1(\mathbf{x}) = -x_1 + 300 - \frac{x_3x_4}{131.078} \cos(1.48477 - x_6) + \frac{0.90798x_4^2}{131.078} \cos(1.47588)$$

$$h_2(\mathbf{x}) = -x_2 - \frac{x_3x_4}{131.078} \cos(1.48477 + x_6) + \frac{0.90798x_4^2}{131.078} \cos(1.47588)$$

$$h_3(\mathbf{x}) = -x_5 - \frac{x_3 x_4}{131.078} \sin(1.48477 + x_6) + \frac{0.90798 x_4^2}{131.078} \sin(1.47588)$$

$$h_4(\mathbf{x}) = 200 - \frac{x_3 x_4}{131.078} \sin(1.48477 + x_6) + \frac{0.90798 x_3^2}{131.078} \sin(1.47588)$$

where the bounds are
 $0 \leq x_1 \leq 400, 0 \leq x_2 \leq 1000, 340 \leq x_3 \leq 420, 340 \leq x_4 \leq 420, -1000 \leq x_5 \leq 1000$ and
 $0 \leq x_6 \leq 0.5236$. The best known solution is at $\mathbf{x}^* = (201.784467214523659,$
 $99.9999999999999005, 383.071034852773266, 420, -10.9076584514292652,$
 $0.0731482312084287128)$ where $f(\mathbf{x}^*) = 8853.53967480648$.

g18

Minimize:

$$f(\mathbf{x}) = -0.5(x_1 x_4 - x_2 x_3 + x_3 x_9 - x_5 x_9 + x_5 x_8 - x_6 x_7) \quad (-16)$$

subject to

$$g_1(\mathbf{x}) = x_3^2 + x_4^2 - 1 \leq 0$$

$$g_2(\mathbf{x}) = x_9^2 - 1 \leq 0$$

$$g_3(\mathbf{x}) = x_5^2 + x_6^2 - 1 \leq 0$$

$$g_4(\mathbf{x}) = x_1^2 + (x_2 - x_9)^2 - 1 \leq 0$$

$$g_5(\mathbf{x}) = (x_1 - x_5)^2 + (x_2 - x_6)^2 - 1 \leq 0$$

$$g_6(\mathbf{x}) = (x_1 - x_7)^2 + (x_2 - x_8)^2 - 1 \leq 0$$

$$g_7(\mathbf{x}) = (x_3 - x_5)^2 + (x_4 - x_6)^2 - 1 \leq 0$$

$$g_8(\mathbf{x}) = (x_3 - x_7)^2 + (x_4 - x_8)^2 - 1 \leq 0$$

$$g_9(\mathbf{x}) = x_7^2 + (x_8 - x_9)^2 - 1 \leq 0$$

$$g_{10}(\mathbf{x}) = x_2 x_3 - x_1 x_4 \leq 0$$

$$g_{11}(\mathbf{x}) = -x_3 x_9 \leq 0$$

$$g_{12}(\mathbf{x}) = x_5 x_9 \leq 0$$

$$g_{13}(\mathbf{x}) = x_6 x_7 - x_5 x_8 \leq 0$$

where the bounds are $-10 \leq x_i \leq 10$ ($i = 1, \dots, 8$) and $0 \leq x_9 \leq 20$. The best known solution is at $\mathbf{x}^* = (-0.657776192427943163,$
 $-0.153418773482438542,$
 $0.323413871675240938, -0.946257611651304398, -0.657776194376798906,$
 $-0.753213434632691414, 0.323413874123576972, -0.346462947962331735,$
 $0.59979466285217542)$ where $f(\mathbf{x}^*) = -0.866025403784439$.

g19

Minimize:

$$f(\mathbf{x}) = \sum_{j=1}^5 \sum_{i=1}^5 c_{ij} x_{(10+i)} x_{(10+j)} + 2 \sum_{j=1}^5 d_j x_{(10+j)}^3 - \sum_{i=1}^{10} b_i x_i \quad (-17)$$

subject to:

$$g_j(\mathbf{x}) = -2 \sum_{i=1}^5 c_{ij} x_{(10+i)} - 3d_j x_{(10+j)}^2 - e_j + \sum_{i=1}^{10} a_{ij} x_i \leq 0, j = 1, \dots, 5$$

where $\mathbf{b} = [-40, -2, -0.25, -4, -4, -1, -40, -60, 5, 1]$ and the remaining data is on Table B-1. The bounds are $0 \leq x_i \leq 10$ ($i = 1, \dots, 15$). The best known solution is at $\mathbf{x}^* = (1.66991341326291344\text{e-}17, 3.95378229282456509\text{e-}16, 3.94599045143233784, 1.06036597479721211\text{e-}16, 3.2831773458454161, 9.99999999999999822, 1.12829414671605333\text{e-}17, 1.2026194599794709\text{e-}17, 2.50706276000769697\text{e-}15, 2.24624122987970677\text{e-}15, 0.370764847417013987, 0.278456024942955571, 0.523838487672241171, 0.388620152510322781, 0.298156764974678579)$ where $f(\mathbf{x}^*) = 32.6555929502463$.

Table A-1 Data set for test problem g19

j	1	2	3	4	5
e_j	-15	-27	-36	-18	-12
c_{1j}	30	-20	-10	32	-10
c_{2j}	-20	39	-6	-31	32
c_{3j}	-10	-6	10	-6	-10
c_{4j}	32	-31	-6	39	-20
c_{5j}	-10	32	-10	-20	30
d_j	4	8	10	6	2
a_{1j}	-16	2	0	1	0
a_{2j}	0	-2	0	0.4	2
a_{3j}	-3.5	0	2	0	0
a_{4j}	0	-2	0	-4	-1
a_{5j}	0	-9	-2	1	-2.8
a_{6j}	2	0	-4	0	0
a_{7j}	-1	-1	-1	-1	-1
a_{8j}	-1	-2	-3	-2	-1
a_{9j}	1	2	3	4	5
a_{10j}	1	1	1	1	1

g20

Minimize:

$$f(\mathbf{x}) = \sum_{i=1}^{24} a_i x_i$$

(-18)

subject to:

$$g_i(\mathbf{x}) = \frac{(x_i + x_{(i+12)})}{\sum_{j=1}^{24} x_j + e_i} \leq 0, i = 1, 2, 3$$

$$g_i(\mathbf{x}) = \frac{(x_{(i+3)} + x_{(i+15)})}{\sum_{j=1}^{24} x_j + e_i} \leq 0, i = 4, 5, 6$$

$$h_i(\mathbf{x}) = \frac{x_{(i+12)}}{b_{(i+12)} \sum_{j=13}^{24} \frac{x_j}{b_j}} - \frac{c_i x_i}{40b_i \sum_{j=1}^{12} \frac{x_j}{b_j}} = 0, i = 1, \dots, 12$$

$$h_{13}(\mathbf{x}) = \sum_{i=1}^{24} x_i - 1 = 0$$

$$h_{14}(\mathbf{x}) = \sum_{i=1}^{12} \frac{x_i}{d_i} + k \sum_{i=13}^{24} \frac{x_i}{b_i} - 1.671 = 0$$

where $k = 0.7302 * 530 * 14.7 / 40$ and the data set is detailed on Table B-2. The bounds are $0 \leq x_i \leq 10$ ($i = 1, \dots, 24$). The best known solution is at $\mathbf{x}^* = (1.28582343498528086e-18, 4.83460302526130664e-34, 0, 0, 6.30459929660781851e-18, 7.57192526201145068e-34, 5.03350698372840437e-34, 9.28268079616618064e-34, 0, 1.76723384525547359e-17, 3.55686101822965701e-34, 2.99413850083471346e-34, 0.158143376337580827, 2.29601774161699833e-19, 1.06106938611042947e-18, 1.31968344319506391e-18, 0.530902525044209539, 0, 2.89148310257773535e-18, 3.34892126180666159e-18, 0, 0.310999974151577319, 5.41244666317833561e-05, 4.84993165246959553e-16)$. This solution is a little infeasible and no feasible solution is found so far. This problem can have feasible solutions after deleting the first 6 inequality constraints.

g21

Minimize:

$$f(\mathbf{x}) = x_1 \quad (-19)$$

subject to:

$$g_1(\mathbf{x}) = -x_1 + 35x_2^{0.6} + 35x_3^{0.6} \leq 0$$

$$h_1(\mathbf{x}) = -300x_3 + 7500x_5 - 7500x_6 - 25x_4x_5 + 25x_4x_6 + x_3x_4 = 0$$

$$h_2(\mathbf{x}) = 100x_2 + 155.365x_4 + 2500x_7 - x_2x_4 - 25x_4x_7 - 15536.5 = 0$$

$$h_3(\mathbf{x}) = -x_5 + \ln(-x_4 + 900) = 0$$

$$h_4(\mathbf{x}) = -x_6 + \ln(x_4 + 300) = 0$$

$$h_5(\mathbf{x}) = -x_7 + \ln(-2x_4 + 700) = 0$$

where the bounds are $0 \leq x_1 \leq 1000$, $0 \leq x_2, x_3 \leq 40$, $100 \leq x_4 \leq 300$, $6.3 \leq x_5 \leq 6.7$, $5.9 \leq x_6 \leq 6.4$ and $4.5 \leq x_7 \leq 6.25$. The best known solution is at $\mathbf{x}^* = (193.724510070034967, 5.56944131553368433e-27, 17.3191887294084914, 100.047897801386839, 6.68445185362377892, 5.99168428444264833, 6.21451648886070451)$ where $f(\mathbf{x}^*) = 193.724510070035$.

Table A-2 Data set for test problem g20

i	a_i	b_i	c_i	d_i	e_i
1	0.0693	44.094	123.7	31.244	0.1
2	0.0577	58.12	31.7	36.12	0.3
3	0.05	58.12	45.7	34.784	0.4
4	0.2	137.4	14.7	92.7	0.3
5	0.26	120.9	84.7	82.7	0.6
6	0.55	170.9	27.7	91.6	0.3
7	0.06	62.501	49.7	56.708	
8	0.1	84.94	7.1	82.7	
9	0.12	133.425	2.1	80.8	
10	0.18	82.507	17.7	64.517	
11	0.1	46.07	0.85	49.4	
12	0.09	60.097	0.64	49.1	
13	0.0693	44.094			
14	0.0577	58.12			
15	0.05	58.12			
16	0.2	137.4			
17	0.26	120.9			
18	0.55	170.9			
19	0.06	62.501			
20	0.1	84.94			
21	0.12	133.425			
22	0.18	82.507			
23	0.1	46.07			
24	0.09	60.097			

g22

Minimize:

$$f(\mathbf{x}) = x_1$$

(-20)

subject to

$$g_1(\mathbf{x}) = -x_1 + x_2^{0.6} + x_3^{0.6} + x_4^{0.6} \leq 0$$

$$h_1(\mathbf{x}) = x_5 - 100000x_8 + 1 \times 10^7 = 0$$

$$h_2(\mathbf{x}) = x_6 + 100000x_8 - 100000x_9 = 0$$

$$h_3(\mathbf{x}) = x_7 + 100000x_9 - 5 \times 10^7 = 0$$

$$h_4(\mathbf{x}) = x_5 + 100000x_{10} - 3.3 \times 10^7 = 0$$

$$h_5(\mathbf{x}) = x_6 + 100000x_{11} - 4.4 \times 10^7 = 0$$

$$h_6(\mathbf{x}) = x_7 + 100000x_{12} - 6.6 \times 10^7 = 0$$

$$h_7(\mathbf{x}) = x_5 - 120x_2x_{13} = 0$$

$$h_8(\mathbf{x}) = x_6 - 80x_3x_{14} = 0$$

$$h_9(\mathbf{x}) = x_7 - 40x_4x_{15} = 0$$

$$h_{10}(\mathbf{x}) = x_8 - x_{11} + x_{16} = 0$$

APPENDIX

$$h_{11}(\mathbf{x}) = x_9 - x_{12} + x_{17} = 0$$

$$h_{12}(\mathbf{x}) = -x_{18} + \ln(x_{10} - 100) = 0$$

$$h_{13}(\mathbf{x}) = -x_{19} + \ln(-x_8 + 300) = 0$$

$$h_{14}(\mathbf{x}) = -x_{20} + \ln(x_{16}) = 0$$

$$h_{15}(\mathbf{x}) = -x_{21} + \ln(-x_9 + 400) = 0$$

$$h_{16}(\mathbf{x}) = -x_{22} + \ln(x_{17}) = 0$$

$$h_{17}(\mathbf{x}) = -x_8 - x_{10} + x_{13}x_{18} - x_{13}x_{19} + 400 = 0$$

$$h_{18}(\mathbf{x}) = x_8 - x_9 - x_{11} + x_{14}x_{20} - x_{14}x_{21} + 400 = 0$$

$$h_{19}(\mathbf{x}) = x_9 - x_{12} - 4.60517x_{15} + x_{15}x_{22} + 100 = 0$$

where the bounds $0 \leq x_1 \leq 20000$, $0 \leq x_2, x_3, x_4 \leq 1 \times 10^6$, $0 \leq x_5, x_6, x_7 \leq 4 \times 10^7$, $100 \leq x_8 \leq 299.99$, $100 \leq x_9 \leq 399.99$, $100.01 \leq x_{10} \leq 300$, $100 \leq x_{11} \leq 400$, $100 \leq x_{12} \leq 600$, $0 \leq x_{13}, x_{14}, x_{15} \leq 500$, $0.01 \leq x_{16} \leq 300$, $0.01 \leq x_{17} \leq 400$, $-4.7 \leq x_{18}, x_{19}, x_{20}, x_{21}, x_{22} \leq 6.25$. The best known solutions is at $\mathbf{x}^* = (236.430975504001054, 135.82847151732463, 204.818152544824585, 6446.54654059436416, 3007540.83940215595, 4074188.65771341929, 32918270.5028952882, 130.075408394314167, 170.817294970528621, 299.924591605478554, 399.258113423595205, 330.817294971142758, 184.51831230897065, 248.64670239647424, 127.658546694545862, 269.182627528746707, 160.000016724090955, 5.29788288102680571, 5.13529735903945728, 5.59531526444068827, 5.43444479314453499, 5.07517453535834395)$ where $f(\mathbf{x}^*) = 236.430975504001$.

g23

Minimize:

$$f(\mathbf{x}) = -9x_5 - 15x_8 + 6x_1 + 16x_2 + 10(x_6 + x_7)$$

(-21)

subject to:

$$g_1(\mathbf{x}) = x_9x_3 + 0.02x_6 - 0.025x_5 \leq 0$$

$$g_2(\mathbf{x}) = x_9x_4 + 0.02x_7 - 0.015x_8 \leq 0$$

$$h_1(\mathbf{x}) = x_1 + x_2 - x_3 - x_4 = 0$$

$$h_2(\mathbf{x}) = 0.03x_1 + 0.01x_2 - x_9(x_3 + x_4) = 0$$

$$h_3(\mathbf{x}) = x_3 + x_6 - x_5 = 0$$

$$h_4(\mathbf{x}) = x_4 + x_7 - x_8 = 0$$

where the bounds are $0 \leq x_1, x_2, x_6 \leq 300$, $0 \leq x_3, x_5, x_7 \leq 100$, $0 \leq x_4, x_8 \leq 200$ and $0.01 \leq x_9 \leq 0.03$. The best known solution is at $\mathbf{x}^* = (0.00510000000000259465, 99.99470000000000514, 9.01920162996045897e-18, 99.99990000000000535, 0.0001000000000027086086, 2.75700683389584542e-14, 99.999999999999574, 2000.0100000100000100008)$, where $f(\mathbf{x}^*) = -400.055099999999584$.

g24

Minimize:

APPENDIX

$$f(\mathbf{x}) = -x_1 - x_2$$

(-22)

subject to:

$$g_1(\mathbf{x}) = -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \leq 0$$

$$g_2(\mathbf{x}) = -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 36 \leq 0$$

where the bounds are $0 \leq x_1 \leq 3$ and $0 \leq x_2 \leq 4$. The feasible global minimum is at $\mathbf{x}^* = (2.32952019747762, 3.17849307411774)$ where $f(\mathbf{x}^*) = -5.50801327159536$. This problem has a feasible region consisting on two disconnected sub-regions.

Table A-3 Details of the 24 test problems

Function		<i>n</i>	Type of <i>f</i>	ρ	<i>LI</i>	<i>NI</i>	<i>LE</i>	<i>NE</i>	<i>a</i>
G1	Min	13	quadratic	0.0111%	9	0	0	0	6
G2	Max	20	nonlinear	99.8474%	0	2	0	0	1
G3	Max	10	polynomial	0.0000%	0	0	0	1	1
G4	Min	5	quadratic	52.1230%	0	6	0	0	2
G5	Min	4	cubic	0.0000%	2	0	0	3	3
G6	Min	2	cubic	0.0066%	0	2	0	0	2
G7	Min	10	quadratic	0.0003%	3	5	0	0	6
G8	Max	2	nonlinear	0.8560%	0	2	0	0	0
G9	Min	7	polynomial	0.5121%	0	4	0	0	2
G10	Min	8	linear	0.0010%	3	3	0	0	6
G11	Min	2	quadratic	0.0000%	0	0	0	1	1
G12	Max	3	quadratic	4.7713%	0	1	0	0	0
G13	Min	5	nonlinear	0.0000%	0	0	0	3	3
G14	Min	10	nonlinear	0.0000%	0	0	3	0	3
G15	Min	3	quadratic	0.0000%	0	0	1	1	2
G16	Max	5	nonlinear	0.0204%	4	34	0	0	4
G17	Min	6	nonlinear	0.0000%	0	0	0	4	4
G18	Max	9	quadratic	0.0000%	0	12	0	0	4
G19	Max	15	nonlinear	33.4761%	0	5	0	0	-
G20	Min	24	linear	0.0000%	0	6	2	12	-
G21	Min	7	linear	0.0000%	0	1	0	5	6
G22	Min	22	linear	0.0000%	0	1	8	11	-
G23	Min	9	linear	0.0000%	0	2	3	1	-
G24	Min	2	linear	79.6556%	0	2	0	0	2

* *n* is the number of decision variables, $\rho=|F|/|S|$ is the estimated ratio between the feasible region and the search space, *LI* is the number of linear inequality constraints, *NI* the number of nonlinear inequality constraints, *LE* is the number of linear equality constraints and *NE* is the number of nonlinear equality constraints. *a* is the number of active constraints on the optima. $\varepsilon = 0.0001$ for *NE*.

Table A-4 $f(x^*)$ and the bounds for the 24 problems

Function	n	$f(x^*)$	bounds
G1	13	-15	$0 \leq x_i \leq 1$ ($i=1, \dots, 9$), $0 \leq x_i \leq 100$ ($i=10, 11, 12$), $0 \leq x_{13} \leq 1$
G2	20	0.803619	$0 \leq x_i \leq 10$ ($i=1, \dots, 20$)
G3	10	1	$0 \leq x_i \leq 1$ ($i=1, \dots, 10$)
G4	5	-30665.539	$78 \leq x_1 \leq 102$, $33 \leq x_2 \leq 45$, $27 \leq x_i \leq 45$ ($i=3, 4, 5$)
G5	4	5126.4981	$0 \leq x_i \leq 1200$ ($i=1, 2$), $-0.55 \leq x_i \leq 0.55$ ($i=3, 4$)
G6	2	-6961.81381	$13 \leq x_1 \leq 100$, $0 \leq x_2 \leq 100$
G7	10	24.3062091	$-10 \leq x_i \leq 10$ ($i=1, \dots, 10$)
G8	2	0.095825	$0 \leq x_i \leq 10$ ($i=1, 2$)
G9	7	680.6300573	$-10 \leq x_i \leq 10$ ($i=1, \dots, 7$)
G10	8	7049.3307	$100 \leq x_i \leq 10000$, $1000 \leq x_i \leq 10000$ ($i=2, 3$), $0 \leq x_i \leq 1$ ($i=4, \dots, 8$)
G11	2	0.75	$-1 \leq x_i \leq 1$ ($i=1, 2$)
G12	3	1	$0 \leq x_i \leq 10$ ($i=1, 2, 3$)
G13	5	0.0539498	$-2.3 \leq x_i \leq 2.3$ ($i=1, 2$), $-3.2 \leq x_i \leq 3.2$ ($i=3, 4, 5$)
G14	10	-47.7644	$0 \leq x_i \leq 10$ ($i=1, \dots, 10$)
G15	3	955.0351	$0 \leq x_i \leq 10$ ($i=1, \dots, 3$)
G16	5	1.9146	$704.4148 \leq x_1 \leq 906.3855$, $68.6 \leq x_2 \leq 288.88$, $0 \leq x_3 \leq 134.75$, $193 \leq x_4 \leq 287.0966$, $25 \leq x_5 \leq 84.1988$
G17	6	8877	$0 \leq x_i \leq 400$, $0 \leq x_2 \leq 1000$, $340 \leq x_3 \leq 420$, $340 \leq x_4 \leq 420$, $-1000 \leq x_5 \leq 1000$, $0 \leq x_6 \leq 0.5236$
G18	9	1	$-10 \leq x_i \leq 10$ ($i=1, \dots, 8$), $0 \leq x_9 \leq 20$
G19	15	-	$0 \leq x_i \leq 10$ ($i=1, \dots, 15$)
G20	24	-	$0 \leq x_i \leq 10$ ($i=1, \dots, 24$)
G21	7	193.7783	$0 \leq x_i \leq 1000$, $0 \leq x_2 \leq 40$, $0 \leq x_3 \leq 40$, $100 \leq x_4 \leq 300$, $6.3 \leq x_5 \leq 6.7$, $5.9 \leq x_6 \leq 6.4$, $4.5 \leq x_7 \leq 6.25$
G22	22	-	$0 \leq x_i \leq 20000$, $0 \leq x_i \leq 1 \times 10^6$ ($i=2, 3, 4$), $0 \leq x_i \leq 4 \times 10^7$ ($i=5, 6, 7$), $100 \leq x_8 \leq 299.99$, $100 \leq x_9 \leq 399.99$, $100.01 \leq x_{10} \leq 300$, $100 \leq x_{11} \leq 400$, $100 \leq x_{12} \leq 600$, $0 \leq x_i \leq 500$ ($i=13, 14, 15$), $0.01 \leq x_{16} \leq 300$, $0.01 \leq x_{17} \leq 400$, $-4.7 \leq x_i \leq 6.25$ ($i=18, \dots, 22$)
G23	9	-	$0 \leq x_i \leq 300$ ($i=1, 2, 6$), $0 \leq x_i \leq 100$ ($i=3, 5, 7$), $0 \leq x_i \leq 200$ ($i=4, 8$), $0.01 \leq x_9 \leq 0.03$
G24	2	-5.5080	$0 \leq x_1 \leq 3$, $0 \leq x_2 \leq 4$

B. Comparison of Results on the 2006 CEC Benchmark Function Set

The list of algorithms

- DE Karin Zielinski & Rainer Laur (Constrained Single-...)[136]
- DMS-PSO J. J. Liang & P. N. Suganthan (Dynamic Multi-Swarm...) [137]
- ϵ _DE Tetsuyuki Takahama & Setsuko Sakai (Constrained ...) [138]
- GDE Saku Kukkonen & Jouni Lampinen (Constrained ...) [139]
- jDE-2 Janez Brest & Viljem Zumer (Self-Adaptive ...) [140]
- MDE Efrén Mezura-Montes, *et al* (Modified Differential....) [141]
- MPDE M. Fatih Tasgetiren & P. N. Suganthan (A Multi-...)[142]
- PCX Ankur Sinha, *et al* (A Population-Based, Parent) [143]
- PESO+ Angel E. Munoz-Žavala *et al* (PESO+ for...) [144]
- SaDE V. L. Huang *et al* (Selt-adaptive Differential...) [66]

The empirical distribution of normalized success performance is shown in Figure B-1. Test problems g20 and g22 are excluded from this figure since no algorithm was successful on these two problems. FEs is Success Performance. FEs_best is FEs of the best algorithm on the respective function.

Figure B-1 Empirical Distribution of Normalized Success Performance

Table B-1 Feasible rate, success rate of all algorithms

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

APPENDIX

Table B-2 Success Performance FEs divided by FES_{best} (g01-g08)

Test Problem	g01	g02	g03	g04	g05	g06	g07	g08
FES_{best}	25115	96222	24861	15281	21306	5202	26578	918
DE	1.3304	1.4017	-	1.0461	5.0256	1.3731	3.5290	1.1830
DMS	1.3272	1.8201	1.0289	1.6625	1.3790	5.3126	1.0000	4.4928
e_DE	2.3615	1.5571	3.5963	1.7156	4.5729	1.4189	2.7957	1.2407
GDE	1.6133	1.5543	143.8877	1.0000	9.0821	1.2501	4.6654	1.6002
jDE-2	2.0062	1.5163	-	2.6653	20.9724	5.6686	4.8064	3.5251
MDE	3.0011	1.0000	1.8096	2.7198	1.0000	1.0000	7.3069	1.0000
MPDE	1.7292	3.1694	1.0000	1.3666	10.1600	2.0327	2.1597	1.6498
PCX	2.1981	1.3292	1.4053	2.0279	4.4478	6.5015	4.4067	3.0784
PESO+	4.0427	4.2905	18.1268	5.2271	21.2267	10.8627	13.8191	6.6710
SaDE	1.0000	1.9107	12.0254	1.6430	3.4263	2.4118	1.0398	1.4412

Table B-3 Success Performance FEs divided by FES_{best} (g09-g16)

Test Problem	g09	g10	g11	g12	g13	g14	g15	g16
FES_{best}	16152	25520	3000	1308	21732	25220	10458	8730
DE	1.5976	4.6715	4.4600	3.9021	1.5976	2.7052	5.5429	1.3278
DMS	1.8237	1.0000	4.8750	4.1356	1.8237	1.0000	2.7634	6.1260
e_DE	1.4315	4.1236	5.4733	3.1529	1.4315	4.4980	8.0528	1.4875
GDE	1.8716	3.2368	2.8200	2.4075	1.8716	9.1247	7.1605	1.5148
jDE-2	3.4001	5.7269	17.9760	4.8593	3.4001	3.8797	23.0812	3.6306
MDE	1.0000	6.4326	1.0000	1.0000	1.0000	11.5639	1.0000	1.0000
MPDE	1.3029	1.9055	7.7854	3.2401	1.3029	1.6937	19.1408	1.4963
PCX	2.8806	3.4886	12.8960	6.8502	2.8806	2.3488	4.4880	3.4817
PESO+	6.0391	110.8383	150.0333	6.1835	6.0391	-	43.0388	5.6174
SaDE	1.3278	1.7307	8.3703	1.9694	1.3278	1.7843	2.5818	1.7123

Table B-4 Success Performance FEs divided by FES_{best} (g17-g23)

Test Problem	g17	g18	g19	g21	g22	g23
FES_{best}	26364	28261	21830	38217	-	129550
DE	50.3891	2.8151	8.1186	4.2571	-	-
DMS	-	1.1741	1.0000	3.6722	-	1.6251
e_DE	3.7498	2.0931	16.3239	3.5362	-	1.5497
GDE	81.4890	16.9874	10.5489	15.1615	-	8.2081
jDE-2	426.0602	3.6963	9.1548	3.3103	-	2.7592
MDE	1.0000	3.6617	-	2.9455	-	2.7821
MPDE	27.7422	1.5585	5.4180	5.4703	-	1.6261
PCX	5.1627	2.4779	5.9403	1.0000	-	1.2900
PESO+	-	8.2431	-	-	-	-
SaDE	474.1314	1.0000	2.3896	4.2958	-	1.0000

APPENDIX

These 10 algorithms are ranked respectively according to five metrics *i.e.*, $\overline{f(\bar{x})}$, feasible rate, success rate, success performance and empirical distribution as shown in Table B-5.

For each metric, we first rank the 10 algorithms on 23 test problems (g1-g19 and g21-g24) so that each algorithm has 23 rankings. Then, we rank the sum of these 23 rankings to get the rank order for each algorithm *i.e.*, $Rank(f) = Rank(\sum_{i=1}^{19} Rank(f_i) + \sum_{i=21}^{24} Rank(f_i))$, g20 is not considered in the ranking since no algorithm ever found a feasible solution on it. The final rank is obtained according to the mean value of the ranks on the five metrics.

Table B-5 Rank results

Algorithms	$\overline{f(\bar{x})}$	Feasible Rate	Success Rate	Success Performance	Empirical Distribution	Final Rank
DE	7	4	8	6	9	7
DMS-PSO	3	1	3	3	6	2
e_DE	1	1	1	4	2	1
GDE	10	10	9	8	5	9
jDE-2	8	4	7	9	8	8
MDE	5	4	4	1	7	5
MPDE	4	9	5	5	4	6
PCX	2	4	2	7	1	2
PESO+	9	8	10	10	10	10
SaDE	6	1	6	1	3	4

Table B-6 Final rank

Rank	Algorithms
1st	e_DE
2nd	DMS-PSO, PCX
3rd	SaDE
5th	MDE
6th	MPDE
7th	DE
8th	jDE-2
9th	GDE
10th	PESO+

BIBLIOGRAPHY

- [1] J. H. Holland, *Adaptation in natural and artificial systems*, The University of Michigan Press, Ann Arbor, MI, 1975.
- [2] L. J. Fogel. "Autonomous automata," *Industrial Research*, 4:14-19, 1962.
- [3] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence through Simulated Evolution*, Wiley, New York, 1966.
- [4] D. B. Fogel, "An evolutionary approach to the traveling salesman problem," *Biological Cybernetics*, Vol. 60, 1988, pp. 139-144.
- [5] H. -P. Schwefel, *Evolution and Optimization Seeking*. John Wiley & Sons, New York, 1995.
- [6] T. Back, U. Hammel, and H.-P. Schwefel, "Evolutionary computation: Comments on the history and current state." *IEEE Transactions on Evolutionary Computation* 1(1), 1997, pp. 3–17.
- [7] M. Pelikan, *Hierarchical Bayesian Optimization Algorithm: Toward a New Generation of Evolutionary Algorithms*, of *Studies in Fuzziness and Soft Computing Vol 170*, Springer, 2005.
- [8] C. Lima, F. Lobo, and Z. Michalewicz (Editors), *Parameter Setting in Evolutionary Algorithms*, *Springer Series Studies in Computational Intelligence*, Springer-Verlag, Berlin, 2007.
- [9] K. Price, "Genetic annealing." *Dr. Dobb's Journal*, pp. 127 - 132, October 1994.
- [10] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," *IEEE Congress on Evolutionary Computation (CEC 2005)* Edinburgh, Scotland, IEEE Press, Sept. 2005, pp. 1785-1791.
- [11] A. K. Qin, V. L. Huang and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. on Evolutionary Computations*, DOI: 10.1109/TEVC.2008.927706, 2009.

BIBLIOGRAPHY

-
- [12] A. Iorio and X. Li, "Solving rotated multi-objective optimization problems using differential evolution," *Australian Conference on Artificial Intelligence 2004*, Cairns, Australia, December, 2004, pp. 861-872.
 - [13] D. Zaharie, "Control of population diversity and adaptation in differential evolution algorithms," *Proc. of Mendel 2003, 9th International Conference on Soft Computing*, R. Matousek, P. Osmera (Eds.), Brno, Czech Republic, June 2003, pp. 41-46.
 - [14] D. Zaharie and D. Petcu, "Adaptive pareto differential evolution and its parallelization," *Proc. of 5th International Conference on Parallel Processing and Applied Mathematics*, Czestochowa, Poland, Sept. 2003, pp. 261-268.
 - [15] D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos and M. N. Vrahatis, "Parallel differential evolution," *IEEE Congress on Evolutionary Computation 2004 (CEC 2004)*, Portland, OR, Jun 2004, IEEE Press, Vol. 2, pp. 2023 – 2029.
 - [16] F. Xue, "Multi-objective differential evolution and its application to enterprise planning," *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA'03)* Vol. 3, IEEE Press, 2003, pp. 3535-3541.
 - [17] F. Xue, A. C. Sanderson, P. P. Bonissone and R. J. Graves, "Fuzzy logic controlled multi-objective differential evolution," *The 14th IEEE International Conference on Fuzzy Systems*, 2005, pp. 720-725.
 - [18] K. E. Parsopoulos, D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos and M. N. Vrahatis, "Vector evaluated differential evolution for multiobjective optimization," *Proceedings of the 2004 Congress on Evolutionary Computation (CEC2004)*, Portland, OR, 2004.
 - [19] S. Kukkonen and J. Lampinen "Comparison of generalized differential evolution algorithm to other multi-objective evolutionary algorithms," *Proceedings of the 4th European Congress on Computational Methods in Applied Sciences and Engineering*, Jyvaskyla, Finland.
 - [20] S. Kukkonen and J. Lampinen, "An extension of Generalized Differential Evolution for multi-objective optimization with constraints," *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, Birmingham, England, Sept 2004, pp. 752–761.

BIBLIOGRAPHY

-
- [21] S. Kukkonen and J. Lampinen, "GDE3: The third evolution step of Generalized Differential Evolution," *Proceedings of the 2005 Congress on Evolutionary Computation (CEC 2005)*, Edinburgh, Scotland, Sept 2005, pp. 443–450.
 - [22] H. A. Abbass, "The self-adaptive pareto differential evolution algorithm," *Proc. of the 2002 Congress on Evolutionary Computation (2002)*, Honolulu, HI, USA, May 2002, pp. 831-836.
 - [23] E. Zitzler and L. Thiele, "Multi-objective evolutionary algorithms: a comparative case study and the strength Pareto approach," *IEEE Trans. on Evolutionary Computation*, 3(4): 1999, pp. 257-271.
 - [24] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," *Swiss Federal Institute of Technology, Lausanne, Switzerland*, Tech. Rep. TIK-Rep, 103. (2001).
 - [25] C. A. C Coello, G. T. Pulido and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Trans. on Evolutionary Computation*, 8(3): 256-279. 2004.
 - [26] J. D. Knowles and D. W. Corne, "Approximating the non-dominated front using the Pareto archive evolutionary strategy," *Evolutionary Computation*, 8(2): 149-172. 2000.
 - [27] K. Deb and S. Tiwari, "Omni-Optimizer: A procedure for single and multi-objective optimization," in *Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization (EMO-2005)*. Guanajuato, Mexico. Lecture Notes on Computer Science 3410, 2005, pp. 41-65.
 - [28] T. Okabe, Y. Jin, M. Olhofer, and B. Sendhoff. "On Test Functions for Evolutionary Multi-objective Optimization," *Parallel Problem Solving from Nature (PPSN VIII)*, 2004, pp.792 - 802.
 - [29] E. Zitzler, K. Deb, and L. Thiele, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," *Evolution Computation*, Vol. 8, No. 2, Summer 2000, pp. 173–195.
 - [30] K. Deb, L. Thiele, M. Laumanns and E. Zitzler. "Scalable Multi-Objective Optimization Test Problems," *CEC 2002*, IEEE Press, 2002, pp. 825 - 830.
 - [31] J. Knowles and L. Thiele and E. Zitzler, "A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers", *TIK-Report No.214*, Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland, Feb 2006.

BIBLIOGRAPHY

-
- [32] S. Huband, P. Hingston, L. Barone, L. While, "A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit", *IEEE Transactions on Evolutionary Computation*, Vol.10 (5), Oct. 2006, pp. 477-506.
 - [33] G. Rudolph, B. Naujoks and M. Preuss, "Capabilities of EMOA to Detect and Preserve Equivalent Pareto Subsets". *Evolutionary Multi-Criterion Optimization (EMO 2007)*, Vol.4403, Springer, Berlin. 2007, pp. 36-50.
 - [34] H. A. Abbass, R. Sarker and C. Newton, "PDE: A Pareto-frontier differential evolution approach for multiobjective optimization problems," in *Proc. of IEEE Congress on Evolutionary Computation*, 2001, pp. 971-978.
 - [35] H.-Y. Fan and J. Lampinen, "A trigonometric mutation operation to differential evolution," *J. Global Optim.*, Vol. 27, No. 1, 2003, pp. 105-129.
 - [36] H.-Y. Fan and J. Lampinen, "A directed mutation operation for the differential evolution algorithm," *International Journal of Industrial Engineering: Theory, Applications and Practice*, 1(10), 2003, pp. 6-15.
 - [37] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," In J. J. Grefenstette (Ed.), *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, Pittsburgh, PA, 1985, pp. 93-100.
 - [38] J. Ilonen, J.-K. Kamarainen and J. Lampinen, "Differential evolution training algorithm for feed-forward neural networks," *Neural Processing Letters*, Vol.7, No.1, 2003, pp. 93-105.
 - [39] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing*, Vol. 9 No. 6, April 2005, pp. 448-462.
 - [40] J. Rönkkönen, S. Kukkonen, and K. Price, "Real-parameter optimization with differential evolution," *2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, Edinburgh, Scotland, IEEE Press, Sept. 2005, pp. 506-513.
 - [41] J. Sun, Q. Zhang, and E. P. Tsang, "DE/EDE: A new evolutionary algorithm for global optimisation," *Information Sciences*, Vol. 169, 2005, pp. 249-262.
 - [42] J. Teo, "Exploring dynamic self-adaptive populations in differential evolution," *Soft Computing*, Vol.10, No.8, 2006, pp.637-686.
 - [43] J. E. Baker, "Reducing bias and inefficiency in the selection algorithm," *Proc. of the 2nd Intl. Conference on Genetic Algorithms*, Cambridge, MA, 1987, pp. 14-21.

BIBLIOGRAPHY

-
- [44] J. J. Liang, P. N. Suganthan and K. Deb, "Novel composition test functions for numerical global optimization," in *IEEE Swarm Intelligence Symposium 2005*, Pasadena, California, June 2005, pp. 68-75.
 - [45] J. Vesterstrøm and R. Thomson, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Proc. IEEE Congress on Evolutionary Computation (CEC 2004)*, IEEE Press, Portland, OR, Jun 2004, pp. 1980-1987.
 - [46] 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," *IEEE Transactions on Evolutionary Computation*, Vol. 10, Issue: 6, Dec. 2006, pp. 646–657.
 - [47] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, Chichester, U.K., Wiley, 2001.
 - [48] K. Price, "An Introduction to Differential Evolution". Corne, D., Dorigo, M., and Glover, F. (eds.), *New Ideas in Optimization*, McGraw-Hill, London, U.K., 1999, pp. 79-108.
 - [49] K. Price, "Differential evolution vs. the functions of the 2nd ICEO," *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC97)*, Cat. No.97TH8283, Indianapolis, IN., 13-16 April 1997, pp. 153 - 157.
 - [50] K. Price, R. Storn, and J. Lampinen, *Differential Evolution - A Practical Approach to Global Optimization*, Springer, Berlin, 2005.
 - [51] M. M. Ali and A. Torn, "Population set-based global optimization algorithms: Some modifications and numerical Studies," *Computers & Operations Research*, Vol. 31, No.10, 2004, pp. 1703-1725.
 - [52] M. G. H. Omran, A. Salmsan and A. P. Engelbrecht, "Self-adaptive differential evolution," in *Computational Intelligence And Security, PT 1, Proceedings Lecture Notes In Artificial Intelligence*, 2005, pp. 192-199.
 - [53] K. Price, "Differential Evolution vs the functions of the 2nd ICEO," *Proceedings of 1997 IEEE International Conference on Evolutionary Computation*, 1997, pp. 153-157.
 - [54] R. Storn and K. Price, "Differential evolution-A simple and efficient heuristic for global optimization over continuous Spaces," *Journal of Global Optimization*, Vol. 11, 1997, pp. 341-359.

BIBLIOGRAPHY

-
- [55] R. Storn, "On the usage of differential evolution for function optimization," *Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS 1996)*, Berkeley, pp. 519-523.
 - [56] R. Storn, "Differential evolution design of an IIR-filter," *Proc. of IEEE Int. Conference on Evolutionary Computation ICEC'96*, Nagoya, Japan, May 1996. pp. 268-273.
 - [57] R. Joshi and A. C. Sanderson, "Minimal representation multisensor fusion using differential evolution," *IEEE Trans. Systems, Man, and Cybernetics*, Part A, Vol. 29, No. 1, 1999, pp. 63-76.
 - [58] R. Storn and K. Price, "Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces," *Technical Report TR-95-012, ICSI*, <http://http.icsi.berkeley.edu/~storn/litera.html>, 1995.
 - [59] R. Gämperle, S. D. Müller and P. Koumoutsakos, "A parameter study for differential evolution," in A. Grmela, N. E. Mastorakis, editors, *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, Interlaken, Switzerland, WSEAS Press, 2002, pp. 293-298.
 - [60] S. Das, A. Konar, and U. K. Chakraborty, "Two improved differential evolution schemes for faster global search," *ACM-SIGEVO Proceedings of Genetic and Evolutionary Computation Conference (GECCO-2005)*, Washington DC, pp. 991-998.
 - [61] S. Das, A. Konar, and U. K. Chakraborty, "Improving particle swarm optimization with differentially perturbed velocity," *2005 Genetic and Evolutionary Computation Conference (GECCO 2005)*, Washington DC, June 2005, pp. 177-184.
 - [62] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition based differential evolution algorithms," *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, 2006, pp. 7363-7370.
 - [63] —, "Opposition-based differential evolution for optimization of noisy problems," *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, 2006, pp. 6756-6763.
 - [64] T. Rogalsky, R.W. Derksen, and S. Kocabiyik, "Differential evolution in aerodynamic optimization," *Proc. of 46th Annual Conf. of Canadian Aeronautics and Space Institute*, Montreal, Quebec, May 1999, pp. 29-36.

BIBLIOGRAPHY

-
- [65] U. K. Chakraborty, S. Das and A. Konar, "Differential evolution with local neighborhood," in *Proceedings of Congress on Evolutionary Computation (CEC 2006)*, Vancouver, BC, Canada, IEEE Press, pp. 2042-2049.
 - [66] V. L. Huang, A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for constrained real-parameter optimization," in *2006 IEEE Congress on Evolutionary Computation (CEC 2006)*, Vancouver, BC, Canada, July 2006, pp. 17-24.
 - [67] V. L. Huang, P. N. Suganthan, A. K. Qin and S. Baskar, "Multiobjective differential evolution with external archive and harmonic distance-based diversity measure," *Technical Report*, Nanyang Technological University, 2005.
 - [68] V. L. Huang, A. K. Qin, K. Deb, E. Zitzler, P. N. Suganthan, J. J. Liang, M. Preuss and S. Huband, "Problem definitions for performance assessment of multi-objective optimization algorithms," *Special Session on Performance Assessment of Multi-Objective Optimization Algorithms, Technical Report*, Nanyang Technological University, Singapore, 2007.
 - [69] W. J. Zhang and X. F. Xie, "DEPSO: hybrid particle swarm with differential evolution operator," *IEEE International Conference on Systems, Man & Cybernetics (SMCC)*, Washington DC, USA, 2003, pp. 3816-3821.
 - [70] Y.-J. Shi, H.-F. Teng, and Z.-Q. Li, "Cooperative Co-evolutionary differential evolution for function optimization," *Proc. 1st Int. Conf. Advances in Natural Comput.*, Changsha, China, Aug. 27-29, 2005, pp. 1080-1088.
 - [71] J. A. Rumpler and F. W. Moore, "Automatic selection of subpopulation and minimal spanning distances for improved numerical optimization," *Proceedings of the IEEE Congress on Evolutionary Computation (CEC2001)*, Seoul, South Korea, 2001.
 - [72] Y. J. Shi, H. F. Teng, and Z. Q. Li, "Cooperative co-evolutionary differential evolution for function optimization," *Advances in Natural Computation, Pt 2, Proceedings, vol. 3611, Lecture Notes in Computer Science*, L. Wang, K. Chen, and Y. S. Ong, Eds., 2005, pp. 1080-1088.
 - [73] D. Zaharie, "A multi-population differential evolution algorithm for multi-modal optimization," *Mendel' 04 - 10th International Conference on Soft Computing*, R. Matousek and P. Osmera, Eds., Brno, Czech Republic, 16-18 June 2004, pp. 17-22,

BIBLIOGRAPHY

-
- [74] Z. Hendershot, "A differential evolution algorithm for automatically discovering multiple global optima in multidimensional, discontinuous spaces," *Proceedings of MAICS 2004, Fifteenth Midwest Artificial Intelligence and Cognitive Sciences Conference*, 2004, pp. 92-97.
 - [75] R. Thomsen, "Multimodal optimization using crowding-based differential evolution," *Proceedings of the 2004 Congress on Evolutionary Computation*, Vol. 2, 2004, pp. 1382 - 1389.
 - [76] X. Li, "Efficient differential evolution using speciation for multimodal function optimization," *Proceeding of Genetic and Evolutionary Computation Conference 2005 (GECCO'05)*, eds. Beyer, Hans-Georg *et al.*, Washington DC, USA, 25-29 June, pp. 873-880.
 - [77] M. M. Ali, C. Khompatraporn, and Z. B. Zabinsky, "A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems," *J. Global Optim.*, Vol. 31, 2005, pp. 635-672.
 - [78] J. Sun, Q. Zhang, and E. Tsang, "DE/EDA: A new evolutionary algorithm for global optimization," *Inf. Sci.*, Vol. 169, 2004, pp. 249-262.
 - [79] S. Das, A. Konar, and U. K. Chakraborty, "Improving particle swarm optimization with differentially perturbed velocity," *Proceedings of the 2005 Genetic and Evolutionary Computation Conference (GECCO 2005)*, Washington DC, USA, June 2005, pp. 177-184.
 - [80] N. Noman and H. Iba, "Enhancing differential evolution performance with local search for high dimensional function optimization," *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation (GECCO 2005)*, Washington DC, USA, June 2005, pp. 967-974.
 - [81] N. Noman and H. Iba, "A new generation alternation model for differential evolution," in *Proceedings of the 2006 Genetic and Evolutionary Computation Conference (GECCO'2006)*, Maarten Keijzer *et al.* (Eds.), July 2006, pp. 1265-1272.
 - [82] H.-Y. Fan and J. Lampinen, "A directed mutation operation for the differential evolution algorithm," *International Journal of Industrial Engineering: Theory, Applications and Practice*, 1(10):6 - 15, 2003.
 - [83] K. Price and J. I. Ronkkonen, "Comparing the uni-modal scaling performance of global and local selection in a mutation-only differential evolution algorithm,"

BIBLIOGRAPHY

- 2006 *IEEE Congress on Evolutionary Computation (CEC 2006)*, Vancouver, BC, Canada, July 2006, pp. 2019-2026.
- [84] S. Rahnamayan, H. R. Tizhoosh, and M. M. Salama, "Opposition-Based Differential Evolution," to appear in *IEEE Transactions on Evolutionary Computation* 2007.
- [85] C.-Y. Lee and X. Yao, "Evolutionary programming using mutations based on the levy probability distribution," *IEEE Transaction on Evolutionary Computation* 8(1):1-13.2004.
- [86] H. Bersini, M. Dorigo, S. Langerman, G. Seront and L. Gambardella, "Results of the first international contest on evolutionary optimization (1st ICEO)," *Proceedings of the 1996 international conference on evolutionary computation*, IEEE press, Nagoya, Japan, May 20-22 1996.
- [87] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, Vol. 3(2), July 1999, pp. 82 – 102.
- [88] K. Chellapilla, "Combining mutation operators in evolutionary programming," *IEEE Transactions on Evolutionary Computation*, 2:91-96.1998.
- [89] X. Yao and Y. Liu, "Fast evolution strategies," *Angeline P.J., Reynolds R. J., McDonnell J. R., Eberhart R.(eds) Evolutionary programming VI*. Springer, Berlin, 1997, pp. 151-161.
- [90] F. Van den Bergh and A. P. Englebrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation* 8(3): 225-236.2004.
- [91] P. J. Angeline, "Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences," *Proceedings of the 7th International Conference on Evolutionary Programming VII*. Springer-Verlag, London, UK, 1998, pp. 601–610.
- [92] Y.-W. Leung, and Y. Wang, "An orthogonal genetic algorithm with quantization for global numerical optimization," *IEEE Transactions on Evolutionary Computation* 5(1), 2001, pp. 41–53.
- [93] J.-T. Tsai, T.-K. Liu, and J.-H. Chou, "Hybrid Taguchi-genetic algorithm for global numerical optimization," *IEEE Transactions on Evolutionary Computation* 8(4), 365–377. 2004.

BIBLIOGRAPHY

-
- [94] K.-H. Han and J.-H. Kim, "Quantum-inspired evolutionary algorithms with a new termination criterion, He gate, and two phase scheme," *IEEE Transactions on Evolutionary Computation* 8(2), 2004, pp. 156–169.
 - [95] Z. Tu and Y. Lu, "A robust stochastic genetic algorithm for global numerical optimization," *IEEE Transactions on Evolutionary Computation* 8(5), 2004, pp. 456–470.
 - [96] V. Feoktistov, *Differential Evolution: In Search of Solutions*, Springer, Berlin, 2006.
 - [97] V. Feoktistov and S. Janaqi, "Generalization of the strategies in differential evolution," *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS 2004)*, Santa Fe, NM, IEEE Computer Society, New York, 2004, pp. 165.
 - [98] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. C. Coello, "A comparative study of differential evolution variants for global optimization," in Maarten Keijzer *et al.* (editors), *2006 Genetic and Evolutionary Computation Conference (GECCO 2006)*, ACM Press, Seattle, Washington, July 2006, pp 485-492.
 - [99] V. P. Plagianakos and M. N. Vrahatis, "Training Neural Networks with 3-bit Integer Weights," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 99)*, Orlando, USA, 13-17 July 1999, Vol. 1, Morgan Kaufmann Publishers, San Francisco, CA, pp. 910–915.
 - [100] B. V. Babu and K. K. N. Sastry, "Estimation of heat transfer parameters in a trickle-bed reactor using differential evolution and orthogonal collocation," *Computers & Chemical Engineering* 23(3): 327–339, 28 Feb 1999.
 - [101] P. Thomas and D. Vernon. "Image registration by differential evolution," *Proceedings of the First Irish Machine Vision and Image Processing Conference IMVIP-97*, Magee College, University of Ulster, 1997, pages 221 - 225.
 - [102] M. G. H. Omran, A. P. Engelbrecht, and A. Salman. "Differential Evolution methods for unsupervised image classification," *The 2005 IEEE Congress on Evolutionary Computation*, 2005.
 - [103] G. D. Magoulas, V. P. Plagianakos, and M. N. Vrahatis, "Hybrid methods using evolutionary algorithms for on-line training," *Proceedings of IJCNN'01, International Joint Conference on Neural Networks*, Vol. 3, Washington, DC, 15 - 19 July 2001, pp. 2218 - 2223.

BIBLIOGRAPHY

- [104] J. G. Marin-Blázquez, Q. Shen, and A. Tuson. "Tuning fuzzy membership functions with neighbourhood search techniques: A comparative study," *Proceedings of the 3rd IEEE International Conference on Intelligent Engineering Systems*, November 1999, pp. 337 - 342.
- [105] Y.-C. Lin, K.-S. Hwang, and F.-S. Wang, "Plant scheduling and planning using mixed-integer hybrid differential evolution with multiplier updating," *Proceedings of the 2000 Congress on Evolutionary Computation*, Vol. 1, Piscataway, NJ, IEEE, 2000, pp. 593 - 600.
- [106] K. Rządca and F. Seredynski, "Heterogeneous multiprocessor scheduling with differential evolution," *The 2005 IEEE Congress on Evolutionary Computation*, 2005.
- [107] F.-S. Wang and J.-P. Chiou, "Differential evolution for dynamic optimization of differential-algebraic systems," *Proceedings of the IEEE International Conference on Evolutionary Computation - ICEC'97*, Indianapolis, IN. Piscataway, NJ, IEEE Press, 1997, pp. 531 - 536.
- [108] F. Cheong and R. Lai. "Designing a hierarchical fuzzy logic controller using differential evolution," *Proceedings of 1999 IEEE International Fuzzy Systems Conference, FUZZ-IEEE'99*, Vol. 1, Seoul, Korea, IEEE, Piscataway, NJ, 1999, pp. 277 - 282.
- [109] C. S. Chang and D.Y. Xu. "Differential evolution based tuning of fuzzy automatic train operation for mass rapid transit system," *IEE Proceedings on Electric Power Applications*, Vol. 147, May 2000, pp. 206 - 212.
- [110] T. Rogalsky, S. Kocabiyik, and R.W. Derksen. "Differential evolution in aerodynamic optimization," *Canadian Aeronautics and Space Journal*, 46(4): 183 - 190, December 2000.
- [111] D. K. Tasoulis, V. P. Plagianakos, and M. N. Vrahatis, "Differential evolution algorithms for finding predictive gene subsets in microarray data," *Artificial Intelligence Applications and Innovations*, vol. 204, *International Federation for Information Processing*, I. Maglogiannis, K. Karpouzis, and M. Bramer, Eds., 2006, pp. 484-491.
- [112] H. K. Kim, J. K. Chong, K. Y. Park, and D. A. Lowther, "Differential evolution strategy for constrained global optimization and application to practical engineering problems," *IEEE Transactions on Magnetics*, Vol. 43, 2007, pp. 1565-1568.

BIBLIOGRAPHY

- [113] N. G. Pavlidis, M. N. Vrahatis, and P. Mossay, "Existence and computation of short-run equilibria in economic geography," *Applied Mathematics and Computation*, Vol. 184, 2007, pp. 93-103.
- [114] N. Noman and H. Iba, "On the reconstruction of gene regulatory networks from noisy expression profiles," *2006 IEEE Congress on Evolutionary Computation*, Vols 1-6, 2006, pp. 2528-2535.
- [115] R. Mendes, I. Rocha, E. C. Ferreira, and M. Rocha, "A comparison of algorithms for the optimization of fermentation processes," *2006 IEEE Congress on Evolutionary Computation*, Vols 1-6, 2006, pp. 2003-2010.
- [116] J. P. Chiou and F. S. Wang, "A hybrid method of differential evolution with application to optimal control problems of a bioprocess system," *1998 IEEE International Conference on Evolutionary Computation - Proceedings*, 1998, pp. 627-632.
- [117] A. Tuson and P. Ross, "Adapting operator settings in genetic algorithms," *Evolutionary Computation*, Vol. 6, No. 2, 1998, pp. 161-184.
- [118] J. Gomez, D. Dasgupta and F. Gonzalez, "Using adaptive operators in genetic search," in *Proc. of the Genetic and Evolutionary Computation Conference 2003 (GECCO '03)*, Chicago, Illinois, USA, July 2003, pp. 1580-1581.
- [119] B. R. Julstrom, "What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm," *Proc. of the 6th International Conference on Genetic Algorithms*, Pittsburgh, PA, USA, July 15-19, 1995, pp. 81-87.
- [120] P. J. Angeline, "Adaptive and self-adaptive evolutionary computation," *Computational Intelligence: A Dynamic System Perspective*, M. Palaniswami, Y. Attikiouzel, R. J. Marks, D. Fogel, and T. Fukuda, Eds. New York: IEEE Press, 1995, pp. 152-161.
- [121] J. E. Smith and T. C. Fogarty. "Operator and parameter adaptation in genetic algorithms," *Soft Computing*, Vol.1, No. 2, June 1997, pp. 81-87.
- [122] A. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, IEEE, Vol. 3, No.2, 1999, pp.124-141.
- [123] N. Hansen, "Compilation of results on the 2005 CEC benchmark function set," May 4, 2006.
http://www.ntu.edu.sg/home/epnsugan/index_files/CEC05/compareresults.pdf

BIBLIOGRAPHY

-
- [124] J. J. Liang, T. P. Runarsson, E. Mezura-montes, M. Clerc, P.N.Suganthan, C. A. C. Coello, and K.Deb, "Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization," *Technical Report*, 2005. <http://www.ntu.edu.sg/home/EPNSugan>
 - [125] Z. Michalewicz and M. Schoenauer, "Evolutionary Algorithms for Constrained Parameter Optimization Problems," *Evolutionary Computation*, 4(1) 1–32, 1996.
 - [126] K. Deb. "An efficient constraint handling method for genetic algorithms", *Computer Methods in Applied Mechanics and Engineering*, 186(2/4): 311–338, 2000.
 - [127] J. Lampinen. "A constraint handling approach for the differential evolution algorithm," *Proceedings of the Congress on Evolutionary Computation 2002 (CEC'2002)*, vol 2, pages 1468–1473, Piscataway, New Jersey, May 2002.
 - [128] R. Landa-Becerra and C. A. C. Coello. "Optimization with constraints using a cultured differential evolution Approach," *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2005)*, Vol. 1, New York, June 2005, ACM Press, Washington DC, pp 27-34.
 - [129] C. S. Chang and D. Y. Xu, "Differential evolution of fuzzy automatic train operation for mass rapid transit system," *IEEE Proc. of Electric Power Applications* Vol. 147, No. 3, 2000, pp. 206-212.
 - [130] N. K. Madavan, "Multiobjective optimization using a pareto differential evolution approach," *Proceedings of the 2002 Congress on Evolutionary Computation (CEC'2002)* Vol. 2, IEEE Press, 2002, pp.1145-1150.
 - [131] Q. F. Zhang, A. Zhou, S. Z. Zhao, P. N. Suganthan, W. D. Liu and S. Tiwari, "Multiobjective optimization test instances for the CEC2009 special session and competition," *Special Session on Performance Assessment of Multi-Objective Optimization Algorithms*, Technical Report, University of Essex, Colchester, UK and Nanyang Technological University, Singapore, 2008.
 - [132] F. G. Lobo and C. F. Lima, "A review of adaptive population sizing schemes in genetic algorithms." *Proceedings of the 2005 workshops on Genetic and evolutionary computation*, June 25-26, 2005, Washington, D.C.
 - [133] J. Arabas, Z. Michalewicz and J. Mulawka, "GAVaPS – a genetic algorithm with varying population Size," *Proceedings of the 1st IEEE conference on evolutionary computation*, IEEE Press, Piscataway, NJ, 1994, pp 73–78.

BIBLIOGRAPHY

- [134] D. Schlierkamp-Voosen and H. Muhlenbein, "Strategy adaptation by competing subpopulations," *Davidor Y, Schwefel H-P, Manner R Eds. Proceedings of the 3rd conference of parallel problem solving from nature, Lecture notes in computer science*, Springer, Berlin, Heidelberg, New York, 1994, pp 199-208.
- [135] H. Markowitz, *Portfolio selection: efficient diversification of investments*, New York: Wiley, 1959.
- [136] K. Zielinski and R. Laur, "Constrained Single-Objective Optimization Using Differential Evolution," *2006 IEEE Congress on Evolutionary Computation (CEC 2006)*, Vancouver, BC, Canada, July 2006.
- [137] J. J. Liang and P. N. Suganthan, "Dynamic Multi-Swarm Particle Swarm Optimizer with a Novel Constraint-Handling Mechanism", *2006 IEEE Congress on Evolutionary Computation (CEC 2006)*, Vancouver, BC, Canada, July 2006.
- [138] T. Takahama, and S. Sakai, "Constrained Optimization by the ε Constrained Differential Evolution with Gradient-Based Mutation and Feasible Elites," *2006 IEEE Congress on Evolutionary Computation (CEC 2006)*, Vancouver, BC, Canada, July 2006.
- [139] S. Kukkonen and J. Lampinen, "Constrained Real-Parameter Optimization with Generalized Differential Evolution," *2006 IEEE Congress on Evolutionary Computation (CEC 2006)*, Vancouver, BC, Canada, July 2006.
- [140] J. Brest, V. Zumer, and M. S. Maucec, "Self-Adaptive Differential Evolution Algorithm in Constrained Real-Parameter Optimization", *2006 IEEE Congress on Evolutionary Computation (CEC 2006)*, Vancouver, BC, Canada, July 2006.
- [141] E. Mezura-Montes, J. Vel'azquez-Reyes and C. A. Coello Coello, "Modified Differential Evolution for Constrained Optimization," *2006 IEEE Congress on Evolutionary Computation (CEC 2006)*, Vancouver, BC, Canada, July 2006.
- [142] M. Fatih Tasgetiren, and P. N. Suganthan, "A Multi-Populated Differential Evolution Algorithm for Solving Constrained Optimization Problems," *2006 IEEE Congress on Evolutionary Computation (CEC 2006)*, Vancouver, BC, Canada, July 2006.
- [143] Kalyanmoy Deb, Ankur Sinha and S. Aravind, "A Population-Based, Parent Centric Procedure for Constrained Real-Parameter Optimization," *2006 IEEE Congress on Evolutionary Computation (CEC 2006)*, Vancouver, BC, Canada, July 2006.

BIBLIOGRAPHY

- [144] A. E. Muñoz-Zavala, A. Hernández-Aguirre, E. R. Villa-Diharce and S. Botello-Rionda, "PESO+ for Constrained Optimization," *2006 IEEE Congress on Evolutionary Computation (CEC 2006)*, Vancouver, BC, Canada, July 2006