Real-parameter optimization with particle swarm optimizer and differential evolution

Zhao, Shizheng

2011

Zhao, S. (2011). Real-parameter optimization with particle swarm optimizer and differential evolution. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/46547>

<https://doi.org/10.32657/10356/46547>

**REAL-PARAMETER OPTIMIZATION WITH
PARTICLE SWARM OPTIMIZER AND
DIFFERENTIAL EVOLUTION**

ZHAO SHIZHENG

SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING

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

2011

Acknowledgments

I would like to take this opportunity to express my gratitude to those who contributed to the successful completion of my postgraduate studies and this research thesis. I would thank them all, but there are some people who need special recognition.

First of all, I want to thank my supervisor, Associate Professor. Ponnuthurai Nagaratnam Suganthan. He has provided his careful guidance, stimulating suggestions, exact insights and profound knowledge in supervising my research work throughout my Ph.D study. Without his precise and appropriate instructions at every step during my research, I could not accomplish my postgraduate study smoothly.

Thank to Qu Boyang, Dr. Mallipeddi Rammohan, Yu Ling, Dr. Ashish Anand, Dr. Huang Ling and Dr. Liang Jing Jane who worked together with me. The valuable suggestions and help they provided enabled me complete these works.

I also want to express my thanks to Dr. Zhang Qingfu and Dr. Zhou Aimin. During the collaboration with them on several research works, they have provided valuable suggestions as well as shared precious experiences on those works.

I would like to thank Dr. S. Baskar and Dr. M. Willjuice Iruthayarajan, who were generous with their time and for the guidance on simulation of the PID control systems. I have benefited greatly from their vast technical expertise and insight.

Meanwhile, thanks to Dr S. Das and Siddharth Pal, as they have also worked together with us.

Thanks a lot to my dear parents and all my good friends.

Finally, thanks with lots of love to my wife, Pan Linna. She encouraged and supported me to complete the Ph.D study with her patience and love.

Contents

Acknowledgments	i
Contents	ii
Summary	vi
List of Figures	vii
List of Tables	ix
List of Abbreviations	xi
Chapter 1	1
Introduction	1
1.1 Motivations and Objectives	4
1.2 Major Contribution of the Thesis	7
1.3 Organization of the Thesis	8
Chapter 2	11
Literature Review	11
2.1 Particle Swarm Optimization (PSO)	15
2.1.1 Important Variants of PSO	19
2.1.1.1 Modifications of the Original PSO Parameters	19
2.1.1.2 The application of Topologies to the Canonical PSO	21
2.1.1.3 Hybrid PSO Algorithms	24
2.1.2 PSO in Variant Environments	27
2.1.3 Important Real world Applications of PSO	33
2.2 Differential Evolution (DE)	34
2.2.1 Important Variants of DE	39
2.2.1.1 Encoding Representation	40
2.2.1.2 Advanced Settings of control parameters	41
2.2.1.3 Diversified Mutation Strategies	43
2.2.1.4 Modified Recombination operations	46
2.2.1.5 Adaptation/self-adaptation techniques	50
2.2.1.6 Hybrid DE algorithms	54
2.2.2 Important Real world Applications of DE	55
Chapter 3	57
Diversity Enhanced Particle Swarm Optimizer (DivEnh-PSO & DivEnh-CLPSO)	57
3.1 Introduction	57
3.2 DivEnh-PSO and DivEnh-CLPSO	59
3.2.1 The original basic PSO and original CLPSO	59
3.2.2 Population Diversity Measures	62
3.2.3 DivEnh-PSO and DivEnh-CLPSO	63
3.3 Experimental Setup and Results	65
3.3.1 Test Functions Considered in the Experimental Study	65
3.3.2 Parameter Settings and Experimental Study	68
3.3.2.1 The basic parameter settings for PSO, CLPSO, DivEnh-PSO & DivEnh-CLPSO	68
3.3.2.2 The particular parameter settings of CLPSO	68
3.3.2.3 The particular parameter settings of DivEnh-PSO and DivEnh-CLPSO	69
3.3.3 Comparison among different PSO Variants	71
3.3.3.1 Experimental Results and Discussions among basic PSO and	

DivEnh-PSO	72
3.3.3.2 Experimental Results and Discussions among CLPSO and DivEnh-CLPSO	74
3.4 Conclusions.....	77
Chapter 4.....	79
Improved dynamic multi-swarm particle swarm optimizer with the Harmony search (DMS-PSO-HS)	79
4.1 Introduction.....	79
4.2 Improved Dynamic Multi-swarm Particle Swarm.....	81
Optimizer Hybridized with the Harmony Search (DMS-PSO-HS).	81
4.2.1 The original DMS-PSO algorithm	81
4.2.2 The original HS algorithm	83
4.2.3 Improved Dynamic Multi-swarm Particle Swarm Optimizer Hybridized with the Harmony Search (DMS-PSO-HS).	85
4.3 Experiments Results and Discussion	88
4.3.1 Test Functions Considered in the Experimental Study.....	88
4.3.2 Parameter Settings	90
4.3.3 Experimental Study.....	91
4.4 Conclusions.....	94
Chapter 5.....	97
Two-Lbests Based Multiobjective Particle Swarm Optimizer (2LB-MOPSO).....	97
5.1 Introduction.....	97
5.2 Multi-objective Particle Swarm Optimization (MOPSO)	100
5.3 Two lbests based Multi-objective Particle Swarm Optimizer (2LB-MOPSO).....	103
5.3.1 Non-domination Sorted External Archive	103
5.3.2 Two- <i>lbests</i> Based Multi-objective Particle Swarm Optimizer.....	104
5.4 Experimental Procedures	107
5.4.1 MO algorithm Variants and Benchmark Problems	108
5.4.2 Parameter Setting	109
5.4.3 Performance Measures.....	109
5.5 Discussions on Experimental Results	110
5.5.1 Parameter Sensitivity Studies	111
5.5.1.1 Parameter Sensitivity Study on <i>count</i>	111
5.5.1.2 Parameter Sensitivity Study on <i>n_bin</i>	112
5.5.2 Comparison with other algorithms in CEC 2007 Competition.....	113
5.5.3 Comparison among 2LB-MOPSO Variants.....	127
5.5.4 Comparison with SMM	128
5.6 Multi-objective Robust PID Controller Tuning using 2LB-MOPSO	130
5.6.1 Introduction on Robust PID Controller Tuning	130
5.6.2 Multi-objective Robust PID Controller	132
5.6.2.1 Robust PID Controller	133
5.6.2.2 Robust PID Controller Structure.....	135
5.6.3 Test Systems.....	136
5.6.3.1 Test System-I	136
5.6.3.2 Test System-II	137
5.6.4 Implementation of Multi-Objective PID Controller	138
5.6.4.1 Test System-I	139
5.6.4.2 Test System-II	139
5.6.5 Simulation Results	140
5.6.5.1 Results for Test System-I	141

5.6.5.2	Results for Test System-II.....	141
5.7	Multi-Objective Design of Monopulse Antenna using 2LB-MOPSO.....	147
5.7.1	Introduction on Design of Monopulse Antenna.....	147
5.7.2	Formulation of the Design Problem.....	149
5.7.3	Study of Optimal Pareto Fronts for Designing Monopulse Antenna	151
5.7.4	Design of Monopulse Antenna using 2LB-MOPSO.....	155
5.8	Conclusions.....	157
Chapter 6		161
	Multi-objective Particle Swarm Optimizer with Ensemble of External Archives	161
6.1	Introduction.....	161
6.2	MOPSOs with Diversity Preservation	162
6.3	MOPSO with Ensemble of Epsilon-non-domination Sorted External Archives ..	163
6.3.1	ε -Dominance	163
6.3.2	Non-domination Sorted External Archive	165
6.3.3	Two local bests (<i>Lbest</i>) Based MOPSO (2LB-MOPSO).....	167
6.3.4	2LB-MOPSO with Ensemble of External Archives	167
6.4	Experimental Results and Discussion.....	171
6.4.1	2LB-MOPSO Variants Considered in the Experimental Study	171
6.4.2	Performance Measures.....	171
6.4.3	Parameter Settings	171
6.4.4	Benchmark Problems and Portfolio Optimization.....	172
6.4.5	Discussions on Experimental Results	173
6.5	Conclusions.....	179
Chapter 7		181
	Decomposition Based Multiobjective Evolutionary Algorithm with an Ensemble of	
	Neighborhood Sizes	181
7.1	Introduction.....	181
7.2	Review on the MOEA/D.....	183
7.3	Ensemble of Neighborhood Sizes for MOEA/D (ENS-MOEA/D)	184
7.4	Experimental Setup and Results	187
7.5	Conclusions.....	191
Chapter 8		193
	Self-adaptive Differential Evolution with Modified Multi-trajectory Search for Large Scale	
	Optimization	193
8.1	Introduction.....	193
8.2	SaDE with MMTS	194
8.2.1	Self-adaptive DE (SaDE).....	195
8.2.2	JADE Mutation	196
8.2.3	MTS	196
8.2.4	SaDE-MMTS	197
8.3	Experimental Results and Discussion.....	201
8.3.1	Parameter Setting.....	202
8.3.2	Analysis of the Results	203
8.4	Conclusions.....	206
Chapter 9		209
	Conclusions and Future work	209
9.1	Conclusions.....	209
9.2	Future work.....	212
Author's Publications		215

Bibliography	219
Appendix A	249
Appendix B	251
Appendix C	257
Appendix D	267

Summary

Numerous real world problems can be formulated as optimization problems with various parameters to be optimized. Thus several optimization algorithms have been proposed to solve these problems. Particle Swarm Optimizer (PSO) and Differential Evolution (DE) are two relatively new optimization algorithms which have shown their strengths in the optimization world. Based on the investigation on both algorithms, this thesis presents a few improved variants of PSO and DE, which are applied to solve optimization problems with various complexities.

In order to solve complex multi-modal single objective optimization, diversity enhanced technique based on the selected past solutions is proposed to discourage premature convergence of the swarm in basic PSO and Comprehensive Learning PSO (CLPSO). Furthermore, a hybridized dynamic multi-swarm particle swarm optimizer (DMS-PSO) with the Harmony search (HS) is presented to avoid all particles getting trapped into inferior local optimal regions and to increase the diversity of the whole swarm by taking merits of the DMS-PSO and the HS.

A novel two local best (*lbests*) based multi-objective particle swarm optimizer (MOPSO) is illustrated to solve Multi-objective Optimization problems. The 2LB-MOPSO is applied to design multiobjective robust PID controllers of two MIMO systems, as well as a Multi-objective optimization of Monopulse Antennas system. Moreover, An ensemble of ε -dominance external archives with the MOPSO implementation as well as an ensemble of different neighborhood sizes of the subproblems integrated with recent Multiobjective Evolutionary Algorithm based on Decomposition (MOEA/D) are demonstrated to be well performing when solving multiobjective optimization (MO) problems with different characteristics.

Self-adaptive DE (SaDE) is enhanced by incorporating the JADE mutation strategy and hybridizing with modified multi-trajectory search (MMTS) algorithm (SaDE-MMTS) to solve the large scale continuous optimization problems.

List of Figures

Figure 1. 1	Illustration of global optimum and local optima	2
Figure 2. 1	The Flowchart of Original PSO Algorithm	18
Figure 2. 2	Flowchart of Canonical Differential Evolution Algorithm.....	35
Figure 2. 3	Domains of the different recombinant vectors generated using discrete, line and random intermediate recombination.....	49
Figure 2. 4	Change of the trial vectors generated through the discrete and random intermediate recombination due to rotation of the coordinate system.	49
Figure 3. 1	The Median Convergence Characteristics of 10-D and 30-D problems f9 and f18.....	74
Figure 4. 1	DMS-PSO's regrouping phase.....	82
Figure 4. 2	The flowchart of original DMS-PSO.....	83
Figure 4. 3	The flowchart of DMS-PSO-HS.....	87
Figure 5. 1	The flowchart of a typical MOPSO implementation	99
Figure 5. 2	An illustration of two lbests based search process with minimization objectives	106
Figure 5. 3	Ranking results of Parameter Sensitivity Investigation on <i>count</i> and <i>n_bins</i>	113
Figure 5. 4	Pareto fronts obtained by the 2LB-MOPSO and SMM.....	130
Figure 5. 5	Control system with plant perturbation and external disturbance	133
Figure 5. 6	Best Obtained Pareto fronts for Test System-I and II using 2LB-MOPSO and NSGA-II	144
Figure 5. 7	Step response of Test System-I with plant perturbation using IGA method	144
Figure 5. 8	Step response of Test System-I with plant perturbation using NSGA-II method.....	145
Figure 5. 9	Step response of Test System-I with plant perturbation using 2LB-MOPSO method.....	145
Figure 5. 10	The controller outputs (u_1 and u_2) using reported IGA method.....	145
Figure 5. 11	The controller outputs (u_1 and u_2) using 2LB-MOPSO method.....	146
Figure 5. 12	Response of Test System-II with plant perturbation and external disturbance using IGA method	146
Figure 5. 13	Response of Test System-II with plant perturbation and external disturbance using 2LB-MOPSO method	146
Figure 5. 14	Geometry of Subarrayed linear array.....	151
Figure 5. 15	Trade-off Curves for 20 element array (N=10)	153
Figure 5. 16	Trade-off Curve for 40 element array (N=20).....	154
Figure 5. 17	Trade-off Curve for fixed number of subarrays (P=8)	154
Figure 5. 18	Normalized patterns for 20 element array and 8 subarrays	157
Figure 6. 1	The ε -dominance concept (minimizing f1 and f2 objectives).	165
Figure 6. 2	An illustration of external archive updating process	166

Figure 6. 3	The flowchart of ε -Dominance 2LB-MOPSO Algorithm with Ensemble of External Archives.....	170
Figure 6. 4	Pareto fronts obtained by using Methods A-D (median run) on Test Problem 2 (S_ZDT4)	178
Figure 6. 5	Pareto fronts obtained by using Methods A-D (median run) on Test Problem 3 (R_ZDT4)	179
Figure 7. 1	Convergence graphs in term of mean of IGD obtained by two methods.	191

List of Tables

Table 2. 1	Significant Applications of PSO.....	33
Table 2. 2	Applications of DE	55
Table 3. 1	The Properties and Initialization Ranges of Test Problems.....	67
Table 3. 2	Mean objective function values with different replacement frequencies (RF).	70
Table 3. 3	Mean objective function values with different replacement ratios (RR)..	70
Table 3. 4	Mean objective function values with DM sizes.....	71
Table 3. 5	Results of 10-D and 30-D problems.	72
Table 3. 6	Results of 10-D and 30-D test problems.....	75
Table 4. 1	The Initialization of the Test Problems (Global optimum, search domains and initialization ranges).	89
Table 4. 2	Parameter Setting on the Sub-population.	90
Table 4. 3	Parameter Setting on the regrouping period.	91
Table 4. 4	Results of 10-D test problems.....	92
Table 4. 5	Results of 30-D test problems.....	93
Table 5. 1	The Pseudo-code of the 2LB-MOPSO	106
Table 5. 2	R_indicator of parameter Study on count of 2LB-MOPSO on 500k function evaluation.....	112
Table 5. 3	R_indicator of parameter Study on n_bins of 2LB-MOPSO on 500k function evaluation.....	112
Table 5. 4	The Rank of mean value for R & H indicators from 25 runs on test problems 1-7, M=2.....	115
Table 5. 5	The Rank of mean value for R & H indicators from 25 runs on test problems 8-13, M=3.....	119
Table 5. 6	The Rank of mean value for R & H indicators from 25 runs on test problems 14-19, M=5.....	122
Table 5. 7	Summation of ranks based on R & H indicators on all test problems.	126
Table 5. 8	R_indicator obtained by the variants of 2LB-MOPSO at 500k function evaluations	128
Table 5. 9	Comparison between SMM and 2LB-MOPSO	129
Table 5. 10	Best performance of various PID controllers in terms of $J = J_2 + J_\infty$ for Test System-I	143
Table 5. 11	Best performance of various PID controllers in terms of $J = J_2 + J_\infty$ for Test System-II.....	143
Table 5. 12	Statistical performance of 2LB-MOPSO and NSGA-II in terms of J for Test System-I and II.....	143
Table 5. 13	Hypervolume indicators H obtained by 2LB-MOPSO and NSGA-II for Test System-I and II.....	144
Table 5. 14	Optimal Compromise table for 20 Element Array.....	152
Table 5. 15	Subarray Configuration.....	156
Table 5. 16	Subarray Weights.....	156
Table 5. 17	Design Objectives Achieved.....	156

Table 6. 1	ε values used in different algorithms	165
Table 6. 2	The 2LB-MOPSO Algorithm	168
Table 6. 3	The results for R indicator after 5e+5 functions evaluations.....	174
Table 6. 4	The results for H indicator after 5e+5 functions evaluations.....	176
Table 6. 5	The results for H and R indicators after 5e+5 functions evaluations in Portfolio Optimization Application	178
Table 7. 1	Mean IGD obtained by two methods at two phases.	188
Table 7. 2	Results of parameter investigation of LP.....	189
Table 7. 3	Results of the variants of MOEA/D with different NS and ENS-MOEA/D on two-objective problems F1 to F7 (Original MOEA/D sets NS=60) ..	189
Table 7. 4	Results of the variants of MOEA/D with different NS and ENS-MOEA/D on three-objective problems F8 to F10 (Original MOEA/D sets NS=100)	189
Table 8. 1	The algorithm description of SaDE-MMTS	199
Table 8. 2	Experimental Results of 25 runs obtained by SaDE-MMTS on Problem 1- 10.....	204
Table 8. 3	Experimental Results of 25 runs obtained by SaDE-MMTS on Problem 11- 19.....	204
Table 8. 4	Comparison on the average error of 25 runs obtained by four algorithms	205
Table 8. 5	Matlab average running time on the 25 runs for the SaDE-MMTS	206
Table 8. 6	Comparison on the average error of 25 runs obtained by SaDE-MMTS and SaDE with JADE mutation on 4 functions.	206

List of Abbreviations

- CEC:** IEEE Congress on Evolutionary Computation
- EA:** Evolutionary Algorithm
- SI:** Swarm Intelligence algorithms
- EP:** Evolutionary Programming
- ES:** Evolution Strategies
- GA:** Genetic Algorithm
- PSO:** Particle Swarm Optimizer
- DE:** Differential Evolution
- DivEnh:** Diversity Enhanced
- CLPSO:** Comprehensive Learning Particle Swarm Optimizer
- DMS-PSO:** Dynamic Multi-Swarm Particle Swarm Optimizer
- HS:** Harmony search
- DMS-PSO-HS:** Dynamic Multi-Swarm Particle Swarm Optimizer Hybridized with the Harmony Search
- MO:** Multi-Objective Optimization
- 2LB-MOPSO :** Two-Lbest Based Multiobjective Particle Swarm Optimizer
- ENS-MOEA/D:** Decomposition Based Multiobjective Evolutionary Algorithm with an Ensemble of Neighborhood Sizes
- SaDE-MMTS:** Self-adaptive Differential Evolution with Modified Multi-trajectory Search for Large Scale Optimization

Chapter 1

Introduction

Many real world problems can be formulated as optimization problems with various parameters to be optimized. Classical optimization techniques only make use of mathematical differential calculus or other techniques for locating the optimal solution of convex minimization problems. In other words, they assume that the objective function is differentiable twice with respect to the design variables and that the derivatives are continuous in a convex set. Classical methods are known to be vulnerable when solving practical tasks commonly involving nonlinearities, large dimensionalities, non-differentiable functions, non-convexities, multiple optimas, multiple objectives, uncertainties in decision and problem parameters, large computational overheads, as well as problems with various other complexities.

Optimization can be mathematically defined as finding one or more solutions to minimize or maximize the values of one or more objectives. Generally a constrained optimization problem can be defined as follows:

$$\text{Minimize/Minimize } f_i(\mathbf{x}), i = 1, \dots, M, \mathbf{x} = [x_1, x_2, \dots, x_D] \quad (1.1)$$

$$\text{Subject to: } g_j(\mathbf{x}) \leq 0, j = 1, \dots, J$$

$$h_k(\mathbf{x}) = 0, k = 1, \dots, K$$

$$\mathbf{x} \in [\mathbf{X}_{\min}, \mathbf{X}_{\max}]^D$$

Because a maximization problem, $\max(f)$, can be transformed into a minimization problem, $\min(-f)$, in this thesis, the optimization generally means minimization. An optimization problem which involves only one objective function, is called single-objective optimization problem; while an optimization problem addressing more than one objective function is defined as a multi-objective optimization problem. Furthermore, when an optimization problem requires some of the parameters satisfying one or more constraints, it is known as constrained optimization problem. Otherwise, it is regarded as an unconstrained optimization problem or a bound-constrained optimization with $\mathbf{x} \in [\mathbf{X}_{\min}, \mathbf{X}_{\max}]^D$. A solution satisfying $f(\mathbf{x}^*) \leq f(\mathbf{x}), \forall \mathbf{x} \in S$ is called global optimum, where S denotes the whole search space. More than one global optimum can possibly be obtained for an optimization problem. Apart from the global optima, for a complex problem, there always exist solutions which satisfy $f_L(\mathbf{x}^*) \leq f_L(\mathbf{x}), \forall \mathbf{x} \in L$, where $L \subset S$. Figure 1.1 shows two local optimal and one global optimal solutions of $f(x)$. The problems that have more than one local optimal solution are called multi-modal optimization problems.

Figure 1. 1 Illustration of global optimum and local optima

Evolutionary computation uses iterative process, such as reproduction, mutation, recombination (crossover), selection, growth or development in a population. This population is then selected in a guided random search to achieve the desired end. Evolutionary Algorithms (EAs), inspired by Darwinian theory of evolution, are well-known

for their ability to handle complex optimization problems, which have become increasingly popular in design and optimization in the last six decades, with a constantly growing development of new algorithms and successful applications in fields as diverse as engineering, art, biology, economics, marketing, genetics, operations research, robotics, social sciences, physics, politics, chemistry, and so on. Despite this considerably large volume of research, new areas remain to be explored with sufficient depth.

For complex problems, EAs have higher chance of avoiding getting stuck into a local optimum and can achieve better and more stable results. Another important advantage of EAs is that they do not have to know any property of the problem, instead they work by their own internal rules. The evolution is executed through the comparison of the fitness values of the individuals. The output of a system or the results of experiments can be used as the comparison criteria of EAs. Thus, they can solve problems which do not have exact mathematical models and cannot be solved using the traditional methods [1]. This is very useful for complex or loosely defined problems. At the same time, EA eliminates the need to compute derivative information.

Generally, depending on the encoding schemes and evolutionary operators employed as well as the way in which the evolutionary operators are used, mainstream methods of evolutionary algorithms are classified as Genetic Algorithms (GAs), Genetic Programming (GP), Evolution Strategies (ES), Evolutionary Programming (EP), and Differential Evolution (DE).

In addition, there are also several swarm intelligence based methods such as Particle Swarm Optimization (PSO), Ant colony optimization (ACO), and so on. The expression of Swarm intelligence (SI) was introduced by Beni and Wang in 1980s [406], in the context of cellular robotic systems. SI is the collective behaviour of decentralized, self-organized systems, natural or artificial. The concept is employed in work on artificial intelligence. SI simulates the collective and social behavior of living creatures such as ants, bees or termites.

Although both EA and SI have their own distinguishing features, similar to EA, SI also has strong ability of avoiding getting stuck into a local optimum when solving complex search and optimization problems. SI can also achieve better and more stable results without the knowledge of any property of the problem.

DE and PSO are most popular in the nature-inspired optimization community and are being widely used for real-world applications due to their simplicity. They are investigated in this Thesis.

1.1 Motivations and Objectives

Particle Swarm Optimization (PSO) is a heuristic search technique proposed by Kennedy and Eberhart [2] who were inspired by the choreography of bird flocks, animal herds and fish schools. As one of the Swarm intelligence algorithms, PSO is a relatively new technique that has been empirically shown to perform well on many optimization problems. PSO is a population-based search algorithm and is initialized with a population of random solutions, called particles. Unlike evolutionary computation techniques, each particle in PSO is also associated with a velocity. Particles fly through the search space with velocities, which are dynamically adjusted according to their experience. However, a major problem associated with the PSO is premature convergence when solving highly multimodal problems, frequently resulting in local optimal solutions. Rapid information flow between particles in the standard PSO seems to be the reason for clustering of particles. Diversity declines rapidly, making it difficult for the basic PSO algorithm to escape from local optima when solving multimodal problems. Consequently, the decrease of diversity in search space leads to a total implosion and consequent fitness stagnation of the swarm. Hence, maintaining high diversity is crucial for preventing premature convergence of the PSO when solving multimodal optimization problems.

Motivated by the observations above, two improved PSO variants are presented first in this thesis. A Diversity Enhancement Approach is proposed to be applied into the basic PSO and Comprehensive Learning Particle Swarm Optimizer (CLPSO). This work presents a procedure using an external memory of selected past solutions to enhance the diversity of the swarm as well as to discourage premature convergence in the PSO and CLPSO. The proposed external memory is used to hold selected past solutions with good diversity and fitness values. Selected solutions from the external memory are periodically injected into the swarm to enhance the diversity.

In the PSO domain, there are two main variants: global PSO and local PSO. In the local version of the PSO, each particle's velocity is adjusted according to its personal best position *pbest* and the best position *lbest* achieved so far within its neighborhood. The global PSO learns from the personal best position *pbest* and the best position *gbest* achieved so far by the whole population. As a local version of PSO, the dynamic multi-swarm particle swarm optimizer (DMS-PSO) applies a multi swarm technique. It is improved by hybridizing with the Harmony search (DMS-PSO-HS). A new approach is presented to merge a harmony search into each sub-swarm of DMS-PSO after the generation of new solutions by DMS-PSO during every iteration. Combining the potentials of fast directional exploitation in the original PSO and the strong stochastic exploration from the original HS, the sub-swarms we applied here are dynamic and their individual sizes are small, which fit to the population of the harmony search. The whole DMS-PSO population is divided into a large number of sub-swarms, which are also individual HS populations, and these sub-swarms are regrouped frequently by using various regrouping schedules. Information is exchanged among the particles in the whole swarm.

As the second part of this thesis, three multi-objective optimization algorithms are proposed. The goal of the multi-objective optimization is to find a set of non-dominated solutions, namely Pareto Front (PF), alternatively namely Top Front from the non-dominated sorting. The global best (*gbest*) or local best (*lbest*) of every particle in state-of-the-art multi-objective particle swarm optimization (MOPSO) implementations is selected from the non-dominated solutions in the external archive. This approach emphasizes the elitism at the expense of diversity when the size of the current set of non-dominated solutions in the external archive is small. We propose that the *gbests* or *lbests* should be chosen from the top fronts in a non-domination sorted external archive of reasonably large size. In addition, a novel two local bests (*lbest*) based MOPSO (2LB-MOPSO) version is proposed to focus the search around small regions in the parameter space in the vicinity of the best existing fronts, unlike the current MOPSO variants in which the *pbest* and *gbest* (or *lbest*) of a particle can be located far apart in the parameter space, thereby potentially resulting in a chaotic search behavior. Furthermore, 2LB-MOPSO is applied to design multi-objective robust Proportional-Integral-Derivative (PID) controllers for two Multiple Input Multiple Output (MIMO) systems, as well as a Multi-objective Monopulse Antennas system.

EAs have attracted much attention recently as effective approaches for solving numerical optimization problems. However, the performances of most EAs are sensitive to the choice of their parameters. Choosing suitable parameter values is a problem-dependant task and generally requires time-consuming trial-and-error parameter tuning process to realize optimal performance. To overcome this difficulty, the ensemble approach of different parameters is applied.

EAs are naturally computer time-intensive, involving many randomised trials. Hence, it is important that an algorithm's parameters are given appropriate values to reduce the amount of wasteful computation. Ensemble methodology has been developed in various disciplines such as statistics, pattern recognition and machine learning [3],[4]. It integrates different methods available into a single method to perform the same task more flexibly so that the reliability can be improved. Benefited by this idea, an ensemble of ε parameter values and an ensemble of external archives are proposed to be applied into multi-objective particle swarm optimization algorithm (MOPSO) so that the difficulty of tuning the parameters' numerical values for every objective in every problem can be overcome. Moreover, an ensemble of different neighborhood sizes is proposed to employ in the Multiobjective Evolutionary Algorithm based on Decomposition (MOEA/D) to overcome the difficulties of tuning the numerical values of the neighborhood sizes for different problems over different evolution stages.

In order to solve a particular problem, since a single setting may not be apt throughout the process of evolution, different parameter settings as well as different search methods can be appropriate during different stages of the evolution. Based on this observation, adaptation is employed, where both trial vector generation strategies and their associated control parameter values in Self-adaptive DE (SaDE) as well as the initial step sizes in modified multi-trajectory search (MMTS) are gradually self-adapted by learning from their previous experiences in generating promising solutions. The success rates of both SaDE and the MMTS are determined and compared. Consequently, future function evaluations for both search algorithms are assigned proportionally to their recent past performance. This hybrid algorithm for large scale optimization is presented as the final part of this thesis.

1.2 Major Contribution of the Thesis

The major contributions of this thesis are classified into three categories according to the aforementioned three parts:

A. *Improved single objective PSO algorithms*

- A diversified external memory of selected past solutions is proposed to enhance the diversity of the swarm in the basic PSO and the CLPSO. The external memory holds selected past solutions with good diversity and fitness values. To discourage premature convergence of the swarm, selected members from the external memory are periodically injected into the swarm by replacing the worst current positions and their *pbests* without extra function evaluations.
- A hybridized dynamic multi-swarm particle swarm optimizer (DMS-PSO) with the Harmony search (DMS-PSO-HS) is presented. The DMS-PSO-HS attempts to combine the merits of the PSO and the HS, to avoid having all particles getting trapped in inferior local optimal regions. To increase the diversity of the whole swarm, the dynamic multiple swarms is also applied. The DMS-PSO-HS enables the particles to have more diverse exemplars to learn from after the swarms' frequent regrouping, and the harmonies are allowed to search in a larger potential space among different sub-populations.

B. *Multi-objective optimization algorithms*

- A novel two local best (*lbests*) based MOPSO is proposed. The two *lbests* are chosen from a non-domination sorted archive of reasonably large size, and are located close to each other in the vicinity of the best front sections. The 2LB-MOPSO is applied to design multi-objective robust PID controllers for two MIMO systems, as well as a Multi-objective Monopulse Antennas system.
- The multi-objective particle swarm optimization implementation with an ensemble of ε -dominance external archives is demonstrated to be suitable when solving problems with different characteristics, as well as when solving a single problem over different stages of evolution. This approach is applied to solve the portfolio

optimization problem, which is the process of changing the set of financial instruments held to meet various criteria.

- For effective performance of MOEA/D neighborhood size parameter has to be tuned. An ensemble of different neighborhood sizes with self-adaptation is proposed (ENS-MOEA/D) to overcome the difficulties of tuning the numerical values of this parameter for different problems over different evolution stages.

C. large scale optimization algorithms

- Self-adaptive DE is hybridized with modified multi-trajectory search (MMTS) algorithm (SaDE-MMTS) to solve large scale optimization problems. The adaptation is employed on the trial vector generation strategies and their associated control parameter values within the SaDE. The parallel adaptation is also applied with the search step sizes within MMTS. Overall, the function evaluations for both search algorithms namely SaDE and MMTS are assigned proportionally to their recent past performance with adaptation.

1.3 Organization of the Thesis

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

Chapter 2 briefly introduces the Evolutionary Algorithms. A detailed literature review of the Evolutionary Algorithms, Particle Swarm Optimizer and Differential Evolution used in the Thesis is presented.

Chapter 3 presents the details of the proposed Diversity Enhanced basic Particle Swarm Optimizer (DivEnh-PSO) and Diversity Enhanced Comprehensive Learning Particle Swarm Optimizer (DivEnh-CLPSO). Experiments are conducted on multimodal and composition test problems with and without coordinate rotations. Improved performances of DivEnh-PSO and DivEnh-CLPSO over standard PSO and CLPSO are shown by using statistical tests.

Chapter 4 proposes hybridized dynamic multi-swarm particle swarm optimizer (DMS-PSO) with the Harmony search (DMS-PSO-HS). The statistically improved performance on the

strongly multimodal and composition test functions is demonstrated when comparing the DMS-PSO-HS with the original DMS-PSO and the original HS.

Chapter 5 proposes that the *gbests* or *lbests* in MOPSO should be chosen from the top fronts in a non-domination sorted external archive of reasonably large size. In addition, a novel two local bests (*lbest*) based MOPSO (2LB-MOPSO) version is proposed. Comparative evaluation using 19 multi-objectives benchmark functions and 11 state-of-the-art multi-objective evolutionary algorithms overall ranks the 2LB-MOPSO as the best, while two state-of-the-art MOPSO algorithms are ranked the worst among other multi-objective evolutionary algorithms. Two real world design problems are solved by applying the 2LB-MOPSO.

Chapter 6 illustrates an ensemble of ε -dominance external archives which can be employed in multi-objective Evolutionary Algorithms. The experimental results indicate the superior performance of the ensemble of external archives with different ε values over implementations with only one archive. Furthermore, the proposed approach is applied to solve the optimal asset allocation problem in portfolio optimization.

Chapter 7 demonstrates an ensemble of different neighborhood sizes belonging to the subproblems of Multiobjective Evolutionary Algorithm based on Decomposition (MOEA/D) with self-adaptation (ENS-MOEA/D). The trial-and-error procedure of tuning suitable numerical values for neighborhood sizes when solving different problems and over different stages of evolution can be avoided.

Chapter 8 shows how Self-adaptive DE is enhanced by incorporating with modified multi-trajectory search algorithm (SaDE-MMTS) for solving large scale continuous optimization problems. Competitive results on 19 numerical large scale optimization problems in the Special Issue of Soft Computing are presented.

Chapter 9 summarizes the conclusions and future work.

Chapter 2

Literature Review

Chapter 2 introduces the definitions of evolutionary algorithms (EAs) and Swarm Intelligence algorithms (SIs). Particle Swarm Optimizer (PSO) and Differential Evolution (DE) are reviewed in detail in this chapter.

The Evolutionary Algorithms (EAs) are a class of stochastic search algorithms applicable to a wide range of problems in learning and optimization. EAs use some mechanisms inspired by biological evolution: reproduction, mutation, recombination (crossover), and selection. They are guided stochastic search methods mimicking the metaphor of natural biological evolution.

Different from the classical optimization techniques, EAs start the optimization with a population of potential solutions instead of a single point. At each generation, each individual of the population is evaluated. Through recombination and mutation, new offspring are generated. The individuals with higher fitness values have higher chance to have offspring. Then the offspring are evaluated, and the individuals are selected for next generation from the old individuals and offspring. This process drives the population towards regions of the space from which good solutions have already been seen. Through

survival of the fittest, the population gradually converges to the range with higher fitness values. With the recombination and mutation operators, the individuals have a larger potential search space, thus, the adverse influence of initialization is less when compared with local search algorithms.

Mutation and recombination are two of the most important components in EAs which determine the success or failure of the algorithm. They alter the genetic composition of the individual solutions in the population, enabling them to explore new regions of the search space. The operators associated with parameters of any EA must be chosen carefully in accordance with the selected representation and proper selection scheme to achieve good performance.

As another main operator in EA, selection, is primarily used to emphasize better solutions in a population. Selection does not create any new solution but tries to maintain relatively good solutions in future generation through the process of evolution. Selection pressure in EAs can be applied commonly at two different stages of evolution: *parent selection* to decide which vectors participate in recombination (eg. Genetic Algorithm (GA), Genetic Programming (GP) and so on), as well as *survivor selection* to form new population from the combined parents and new offspring of the current generation (eg. Evolutionary Programming (EP), Evolution Strategies (ES), DE and so on).

Swarm Intelligence (SI) is another group of nature-inspired algorithms for multi-agent search and optimization. Swarm Intelligence algorithms (SIs) are typically made up of a population of simple agents interacting locally with one another and with their environment. The agents follow very simple rules. Interactions between such agents lead to the emergence of "intelligent" global behavior. Natural examples of SI include ant colonies, bird flocking, animal herding, bacterial growth, and fish schooling. The efforts to mimic such behaviors through computer simulation finally resulted into the fascinating field of SI.

SI mainly depends on the dynamics of the virtual agents while EA involves the change in distribution of the trial solutions over the fitness landscape following mutation, recombination, and natural selection principles from Darwinian Theory. However, the two nature-inspired optimization families can very well complement each-other and already there has been a good trend of hybridizing the algorithms from both.

EAs and SIs deal with a population of individuals where each individual represents an instantiation of the solution to the underlying problem. Different algorithms use different representation schemes to encode the solutions in the search space. GA normally uses binary string representation to represent decision variables, while ES and EP commonly use a combination of real-valued decision variables and a set of strategy parameters as a solution vector. Real valued encoding is more natural and permits a greater degree of precision. Michalewicz [5] experimentally demonstrated that real-valued representation allowed better convergence than binary encoding while ensuring better balance between exploration in the initial stages and exploitation in the final stages. Most DE and PSO variations employ the real-parameter encoding.

Different EAs and SIs have been defined based on the representation or encoding schemes, search operators, and selection schemes employed as well as the order in which the search and selection operators are performed. GA, GP, EP and ES are considered as the important contemporaneous nature-inspired optimization algorithms during the early stage of the EA history, while DE, PSO and ACO are gaining more attention recently.

Genetic Algorithms (GAs), was originally proposed by Holland in 1962 [6], [7]. Each individual is referred to as a chromosome which is represented as a bit string of fixed length in classical GAs. The operator which plays an important role in GA is the recombination. Mutation acts as an assisting operator. GA uses the sexual reproduction where two parents are combined to form an offspring. GAs usually apply selection operator first to select good solutions and then apply the recombination and mutation operators on these good solutions to create a hopefully better set of solutions. In contrast to GA, ES and EP prefer using the recombination and mutation operator first to create a set of solutions and then use selection operator to choose a good set of solutions. In addition, the selections of the crossover and mutation probabilities in GAs are rather arbitrary and they are not adapted during evolution. Genetic programming (GP) is a specialization of GA where each individual is a computer program [407],[408].

Evolutionary Programming (EP) was developed by Fogel [13],[14] in the context of evolving finite state-machines to be used in the prediction of time series and relied on mutation operator. EP can be considered as a population-based variant of generate-and-test algorithms [15],[18].

Evolution Strategies (ES), was devised by Rechenberg and Schwefel [8],[12]. Different from GAs, the main operator in ES is mutation.

EP [13][14] and ES [8][12] work with continuous parameters. Unlike GAs, they can be easily extended to include discrete and mixed parameters. EP simulates evolution at the phenotypic level emphasizing behavioral evolution rather than genetic evolution. This is one of the major reasons why EP does not use any crossover operator and relies on asexual reproduction (only one parent used in the creation of an offspring). In another words, only mutation is applied in EP. In ES, the strategy control parameters are also incorporated in the representation of the individuals and evolved along with individuals in the population. The evolution process in ES takes place at the individual level and it can be either sexual or panmictic (several parents are used in creation of an offspring). Unlike GA and EP, ES emphasizes both recombination and mutation, the latter being a dominant operator.

Differential Evolution (DE) was originally proposed by Storn and Price [19]. DE is a floating-point encoded, simple yet powerful population-based stochastic search technique, which is also an efficient and effective global optimizer in the continuous search domain. Unlike traditional EAs, DE perturbs the current-generation population members with scaled differences of randomly selected and distinct population members. Therefore, no separate probability distribution has to be used, which makes the scheme self-organizing [23].

Particle Swarm Optimization (PSO) was originally proposed by Kennedy and Eberhart [2]. PSO is one of the Swarm Intelligence algorithms, which was first designed to simulate birds seeking food. PSO is also a population-based search algorithm and is initialized with a population of random solutions, called particles. Unlike the aforementioned evolutionary computation techniques, no recombination and mutation is involved in classical PSO, while each particle in PSO is associated with a velocity. Particles fly through the search space with velocities, which are dynamically adjusted according to their experience.

Ant Colony Optimization algorithm (ACO) is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. Initially proposed by Marco Dorigo in 1992 in his PhD thesis [409][410], the original idea comes from observing the exploitation of food resources among ants, in which ants' individually limited cognitive abilities have collectively been able to find the shortest path

between a food source and the nest. This algorithm is a member of swarm intelligence (SI) methods. The original idea has since been diversified to solve a wider class of numerical problems, and as a result, several problems have emerged, drawing on various aspects of the behavior of ants.

A detailed review of PSO and DE algorithms, which are used in our work, is presented in the following sections.

2.1 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO), a stochastic search and optimization algorithm based on Swarm intelligences, has been investigated for over fifteen years. PSO is a relatively new technique that has been empirically shown to perform well on many optimization problems. It was inspired by the choreography of bird flocks, animal herds and fish schools. Different from the other EAs which emphasize crossover, mutation or asexual reproduction, PSO simulates evolution by the social communication among the particle swarm rather than genetic evolution.

Social influence and social learning enable a person to maintain cognitive consistency. People solve problems by talking with other people about them, and as they interact, their beliefs, attitudes, and behaviors change. The changes could typically be depicted as the individuals moving toward one another in a socio-cognitive space. The particle swarm simulates this kind of social optimization. A problem is given, and some way to evaluate a proposed solution to it exists in the form of a fitness function. A communication structure or social network is also defined, assigning neighbors for each individual to interact with. Then a population of individuals defined as random guesses of the problem solutions is initialized. These individuals are candidate solutions. They are also known as the particles, hence the name particle swarm. An iterative process to improve these candidate solutions is set in motion. The particles iteratively evaluate the fitness of the candidate solutions and remember the location where they had their best successes. The individual's best solution is called the particle best or the local best. Each particle makes this information available to its neighbors. It is also able to see where its neighbors have success. Movements through the search space are guided by these successes, with the population usually converging, by the end of a trial,

on a problem solution. Hence the particles have a tendency to fly towards the better search area. The relative simplicity of the PSO as well as few parameters to adjust make it easy to implement.

The Canonical PSO algorithm is developed based on simplified social model simulation. The PSO was first designed to simulate birds seeking food. A bird would find food through social cooperation with other birds around it (within its neighborhood). It was then expanded to multidimensional search. The topological rather than Euclidean neighborhood was utilized [2], [25]. The original PSO algorithm is described as below:

$$V_i(d) = V_i(d) + c_1 \times rand1() \times (pbest_i(d) - X_i(d)) + c_2 \times rand2() \times (gbest(d) - X_i(d)) \quad (2-1a)$$

$$X_i(d) = X_i(d) + V_i(d) \quad (2-1b)$$

where c_1 and c_2 in the equation 2-1a are the acceleration constants, which represent the weightings of stochastic acceleration terms that pull each particle toward $pbest$ and $gbest$ positions; $rand1()$ and $rand2()$ are two random functions in the range; $X_i = (x_{i1}, x_{i2}, \dots, x_{iD})$ represents the position of the i th particle; $pbest_i = (pbest_{i1}, pbest_{i2}, \dots, pbest_{iD})$ represents the best previous position (the position giving the best fitness value) of the i th particle; $gbest = (gbest_1, gbest_2, \dots, gbest_D)$ represents the best previous position of the population; $V_i = (v_{i1}, v_{i2}, \dots, v_{iD})$ represents the rate of the position change (velocity) for particle i .

Equation (2-1) is the equation describing the flying trajectory of particles in the swarm population. Equation (2-1a) describes how the velocity is dynamically updated and Equation (2-1b) is the position updating of the “flying” particles. Equation (2-1a) consists of three parts. The first part is the momentum part, which indicates that the velocity changes from current velocity instead of altering abruptly. The second part is the “cognitive” part, which represents private thinking of the particle itself, learning from its own flying experience. The third part is the “social” part, which shows the collaboration among particles of the swarm; learning from flying experience of the group [26]. In Equation (2-1a), if the sum of the three parts on the right side exceeds a constant value specified by the user, then the velocity on that dimension is assigned to be $\pm V_{max}$, i.e. the particles’ velocity on each dimension is

clamped to a maximum velocity $V_{\max}$. $V_{\max}$ is an important parameter, and originally is the only parameter required to be adjusted by users. Large $V_{\max}$ allows particles to have the potential to fly far past good solution areas, while a small $V_{\max}$ causes particles to have the tendency to be trapped into local optima, and thus unable to fly into better solution areas. Usually a fixed constant value is used as the $V_{\max}$, but a well-designed dynamically changing $V_{\max}$ might improve the PSO's performance [27]. The original procedure for implementing PSO can be illustrated in Figure 2.1.

Like some other EAs, PSO algorithms are also population based search algorithms with random initialization, and there are interactions among population members. In PSO, each particle flies through the solution space, and has the ability to remember its previous best position, surviving from generation to generation [28]. Furthermore, compared with the evolutionary algorithms such as evolutionary programming, the original version of PSO is faster in initial convergence while slower in the refining stage [29].

$pops$: population size

g : generation counter from 1 to max_gen

$X_i(d)$: i^{th} particle's d^{th} dimension's value

Max_gen : maximum generations

p : particle's id counter from 1 to $pops$

$c_1 = c_2 = 2$ d : dimension

Figure 2. 1 The Flowchart of the Original PSO Algorithm

2.1.1 Important Variants of PSO

The performance of PSO depends mostly on its parameter setting and the configuration of social communication style among the swarm. Hence, to improve the performance various variants of PSO have been proposed in the way of:

1. Modifications of the Original PSO Parameters.
2. The application of Topologies to the Canonical PSO.
3. Hybrid basic PSO with other evolution techniques.

2.1.1.1 Modifications of the Original PSO Parameters

Velocity changes of PSO consist of three parts: the “momentum” part, the “social” part and the “cognitive” part. The balance among these parts determines the balance of the global and local search ability, i.e. the performance of PSO.

A. Inertia Weight

The first new parameter added into the original PSO algorithms is the inertia weight [26]. The dynamics equation of PSO with inertia weight is modified to be:

$$V_i(d) = \omega \times V_i(d) + c_1 \times rand1() \times (pbest_i(d) - X_i(d)) + c_2 \times rand2() \times (gbest(d) - X_i(d)) \quad (2-2a)$$

$$X_i(d) = X_i(d) + V_i(d) \quad (2-2b)$$

Equation (2-2) is the same as the Equation (2-1) except for a new parameter, inertia weight ω . The inertia weight is introduced to balance between the global and local search abilities. The large inertia weight facilitates global search while the small inertia weight emphasizes local search. The introduction of the inertia weight also eliminates the requirement of carefully setting the maximum velocity V_{max} each time the PSO algorithm

is used. The V max can be simply set to the value of the dynamic range of each variable and the PSO algorithm still performs well enough if not better.

B. Constriction Coefficient

Another parameter called constriction coefficient is introduced with the hope that it can insure a PSO to converge [30], [31]. A simplified method of incorporating it appears in Equation (2-3), where k is a function of c_1 and c_2 as seen in equation (2-4).

$$V_i(d) = k \times [V_i(d) + c_1 \times rand1() \times (pbest_i(d) - X_i(d)) + c_2 \times rand2() \times (gbest(d) - X_i(d))] \quad (2-3a)$$

$$X_i(d) = X_i(d) + V_i(d) \quad (2-3b)$$

$$\text{with } k = \frac{2}{2 - \varphi - \sqrt{\varphi^2 - 4\varphi}} \text{ where } \varphi = c_1 + c_2, \varphi > 4 \quad (2-4)$$

Mathematically, Equation (2-2) and (2-3) are equivalent by setting inertia weight ω to be k , and c_1 and c_2 to meet the condition $\varphi = c_1 + c_2$, where $\varphi > 4$. The PSO algorithm with the constriction factor can be considered as a special case of PSO with inertia weight where the three parameters are connected as in Equation (2-4) [32].

When Clerc's constriction method is used, φ is commonly set to 4.1 (i.e. $c_1 = c_2 = 2.05$) and the constant multiplier k is approximately 0.729. This is equivalent to the basic PSO with inertia weight $\omega \approx 0.729$, and $c_1 = c_2 = 1.49445$. In both PSO versions, although two sets of c_1 and c_2 are assigned with different values, they have the equivalent effect. Since the search process of a PSO algorithm is nonlinear and complicated, a PSO with well-selected parameter set can have good performance. However, much better performance could be obtained if a dynamical parameter is well designed. Intuitively, the PSO should favor global search at the beginning of PSO running while local search ability is more important at the end of PSO running. Shi and Eberhart [33], [34] first introduced a linearly decreasing inertia weight to the PSO, and then certain fuzzy systems are further designed to nonlinearly change the inertia weight. The fuzzy systems have some measurements of the PSO performance as the input and the new inertia weight as the fuzzy system output. Furthermore, inertia weight

with a random component $[0.5 + (rand() / 2.0)]$ rather than a time decreasing setting is utilized. This produces a random number varying between 0.5 and 1.0, with a mean of 0.75.

C. Adaptation Coefficient

In [35], Parsopoulos *et al.* proposed Unified Particle Swarm Optimization (UPSO). Then in [36], the performance of UPSO was investigated under different schemes for the determination and adaptation of the unification factor, which is the main parameter of the method, controlling its exploration and exploitation properties. Numerous experiments were conducted along with statistical tests to yield useful conclusions regarding the effect of the parameter on the algorithm's performance as well as the efficient adaptation schemes.

In [37], Ratnaweera *et al.* introduced time varying acceleration coefficients into PSO in addition to the time varying inertia weight. Furthermore, a PSO called "Self-Organizing Hierarchical Particle Swarm Optimizer" was proposed, in which only the "social" part and the "cognitive" part were kept in the algorithm as in [38], while the "momentum" part was only used for reinitializing particles when the particles had stagnated in the search space, i.e. inertia weight was set to be 0, except at the time of re-initialization. In [27], a linearly decreasing V_{max} was introduced as mentioned before.

Recently, stochastic approximation theory was used to analyze the dynamics of the PSO [39]. The authors proposed a decreasing coefficient that was reduced to zero as the number of iterations increased, and a stochastic velocity with fixed expectation to enhance the exploratory mode of the swarm. While a decreasing coefficient facilitated the particles to spread around the problem hyperspace at the beginning of the search, the stochastic velocity term provided additional exploration ability, thus helping the particles to escape from local minima.

2.1.1.2 The application of Topologies to the Canonical PSO

There are two main versions of PSO: global version and local version of PSO. In the global version of PSO, each particle flies through the search space with a velocity that is dynamically adjusted according to the particle's personal best performance achieved so far and the best performance achieved so far by all the particles. While in the local version of PSO, each particle's velocity is adjusted according to its personal best and the best

performance achieved so far within its neighborhood. The global version of PSO also can be considered as a local version of PSO with each particle's neighborhood being the whole population. It has been suggested that the global version of PSO converges fast, but with the potential to converge to a local minimum, while the local version of PSO might have more chances to find better solutions slowly [40][41]. This is because in the global version of PSO, all particles in the swarm learn from the gbest even if the current gbest is far from the global optimum. In the local paradigm, particles have information only of their own and their nearest array neighbors' bests, rather than that of the entire group. Local PSO searches a larger part of the search space and is more resistant to local optimum. However, in addition to its slow convergence, local optimum traps in a complex search environment with numerous local solutions still pose serious problems. Since then, a lot of researchers have worked on improving its performance by designing or implementing different types of neighborhood structures in PSOs.

A. Static topologies

Kennedy [41] first claimed that PSO with small neighborhoods might perform better on complex problems while PSO with large neighborhood would perform better for simple problems. Kennedy and Medes [42] tested PSOs with regular shaped neighborhoods, such as the global version, the local version, the pyramid structure, the star structure, the "small" structure, and the von Neumann, as well as PSOs with randomly generated neighborhoods.

In the standard version of PSO, the effective sources of influence are in fact only two: self and best neighbor. Information from the remaining neighbors is unused. Mendes revised the way particles interact with their neighbors [43],[44]. Whereas in the traditional algorithm each particle is affected by its own previous performance and the single best success found in its neighborhood, in Mendes' fully informed particle swarm (FIPS), the particle was affected by all its neighbors, sometimes with no influence from its own previous success. In [47], Mendes and Kennedy finally introduced a fully informed particle swarm optimization algorithm based on φ coefficient analysis. In this new algorithm, all the neighbors of a particle were involved in calculating the next movement instead of using the previous best positions in the original particle swarm optimization algorithm. The influence of each particle to its neighbors was weighted based on its fitness value and the neighborhood size. FIPS can be depicted as follows:

$$\begin{cases} \vec{V}_i \leftarrow \chi \left(\vec{V}_i + \frac{1}{K_i} \sum_{n=1}^{K_i} \vec{U}(0, \phi) \otimes (\vec{P}_{nbr_n} - \vec{X}_i) \right) \\ \vec{X}_i \leftarrow \vec{X}_i + \vec{V}_i \end{cases} \quad (2-5)$$

where K_i is the number of neighbors for particle i , and nbr_n is particle i 's n^{th} neighbor. It can be seen that this is the same as the traditional particle swarm if only the self and neighborhood best in a $K_i = 2$ model are considered. With appropriate parameters, FIPS appears to find better solutions in fewer iterations than the canonical algorithm, but it is much more dependent on the population topology.

B. Dynamic topologies

Suganthan [45] applied a combined version of PSO where a local version PSO was first run followed by a global version of PSO. Furthermore in the local version of PSO, the neighborhood of each particle was dynamically adjusted. The distances among particles were calculated and were then used as reference parameters to form new neighborhoods against a predefined criterion.

In [46], Hu and Eberhart proposed a dynamic neighborhood concept for their multi-objective optimization problems using PSO. The neighborhood of each particle was dynamically adjusted. The m closest particles in the performance space were selected to be its new neighborhood in each generation. The performance space was the space occupied each coordinate of which was the variable representing the performance value of each objective optimization problem.

To reduce the local minimum traps when the search environment is complex with numerous local solutions, CLPSO [48] used a novel learning strategy whereby all other particles' historical best information was used to update a particle's velocity. This strategy enabled the diversity of the swarm to be preserved to discourage premature convergence.

Modified local best PSO (MLPSO) [49], differed in a main aspect from other local PSO variants: neighborhoods do not have any overlap. Thus, if particles of a neighborhood trapped into an inferior local optimum, particles of other neighborhoods would not be

influenced. On the other hand, each local optimum could attract particles of one neighborhood maximally. Therefore, the chance of finding global optimum by swarm would be increased.

Each neighborhood structure has its strengths and weaknesses. One works better than others in a few kinds of problems, but worse on some other problems with different characteristics. When using PSO to solve problem, not only does the problem need to be specified, but the neighborhood structure of the PSO utilized should also be carefully chosen.

2.1.1.3 Hybrid PSO Algorithms

Many researchers tried different ways to merge or combine the PSO with the other techniques, especially the other nature-inspired optimization techniques to improve PSO's performance.

A. Combined PSO with Evolutionary Operators

Evolutionary operators like selection, crossover and mutation have been applied into the PSO. By applying selection operation in PSO, the particles with the best performance are copied into the next generation. Therefore, PSO can always keep the best performing into the next generation. By applying crossover operation, information can be swapped between two individuals to generate the ability to “fly” to the new search area such as that in evolutionary programming and genetic algorithm. Among the three evolutionary operators, the mutation operators are the most commonly applied evolutionary operators in PSO. The purpose of applying mutation to PSO is to increase the diversity of the population and grant the ability to escape the local minima [37],[50],[51],[52],[53]. One approach is to mutate parameters such as the acceleration constants, the position of the neighborhood best, as well as the inertia weight [37].

Another approach is to prevent particles from moving too close to each other, so that the diversity could be maintained and therefore particles could escape from being trapped into local minima. [50],[53],[59],[60] introduced self adaptation capabilities to the swarm by modifying the concept of a particle to include, not only the objective parameters, but also a set of strategic parameters (inertia and acceleration constants, simply called weights). The

general Evolutionary Particle Swarm algorithm (EPSO) scheme can be summarized as follows [50],[59],[60].

- Replication: Each particle is replicated multiple times.
- Mutation: Each particle has its weights mutated.
- Reproduction: Each mutated particle generates an offspring according to the particle movement rule.
- Evaluation: Each offspring has a fitness value.
- Selection: Stochastic tournament is carried out in order to select the best particle which survives to the next generation.

In [51],[52], collision-avoiding mechanisms were designed to prevent particles from colliding with each other and therefore increase the diversity of the population.

B. Hybrid PSO with Evolutionary Algorithms

In addition to incorporating evolutionary operations into PSO, different approaches to combine PSO with the evolutionary algorithms have been reported. Robinson et. al. [54] obtained better results by applying PSO first, followed by applying Genetic Algorithm (GA) in their profiled corrugated horn antenna optimization problem. In [55], a hybrid PSO (HPSO) with a feasibility-based rule was proposed to solve constrained optimization problems. In contrast to the penalty function method, the rule required no additional parameters and could guide the swarm to the feasible region quickly. In addition, to avoiding the premature convergence, simulated annealing (SA) was applied to the best solution of the swarm to help the algorithm escape from local optimal solutions.

GA-PSO [88] combined the advantages of swarm intelligence and a natural selection mechanism, such as GA, in order to increase the number of frequently evaluated agents, while decreasing the number of rarely evaluated agents at each iteration step. Therefore, not only was it possible to successively change the current searching area by considering *pbest* and *gbest* values, but also to jump from one area to another by the selection mechanism, which results in acceleration in the convergence speed of the whole algorithm. The GA-PSO algorithm basically employs a major aspect of the classical GA approach, which is the capability of “breeding.” However, some authors had also analyzed the inclusion of

mutation or a simple replacement of the best fitted value as a means of improvement to the standard PSO formulation [56], [57].

Angeline [58] proposed adding the tournament selection method used in evolutionary programming (EP) for this purpose. In this approach, the update formulas remained the same as in the original PSO algorithm; however, the particles were selected as follows.

- The fitness value of each particle was compared with other particles and a point was scored for each particle with the worse fitness value. The population was sorted based on this score.
- The current positions and velocities of the best half of the swarm replaced the positions and velocities of the worst half.
- The individual best of each particle of the swarm (best and worst half) remained unmodified. Therefore, at each iteration step, half of the individuals were moved to positions of the search space that were closer to the optimal solution than their previous positions while keeping their personal best points.

The difference between this method and the original particle swarm is that the exploitative search mechanism is emphasized. This should help the optimum to be found more consistently than the original particle swarm.

Differential Evolution (DE) was proposed to improve the performance of the PSO algorithm. It was applied to the particle's best position to eliminate the particles falling into local minima (DEPSO) [61]-[63]. The DEPSO method alternates the original PSO algorithm and the DE operator. The DE mutation operator was defined over the particle's best positions with a trial point. In addition, DE was used to find the optimal parameters (inertia and acceleration constants) for the canonical PSO (composite PSO) [64].

C. Hybrid PSO with Non-Evolutionary techniques

Also, non-evolutionary techniques have been incorporated into PSO. In [65], a Cooperative Particle Swarm Optimizer (CPSO) was implemented. The CPSO employed cooperative behavior to significantly improve the performance of the original PSO algorithm through using multiple swarms to optimize different components of the solution vector cooperatively. They split the solution vectors into smaller vectors to partition the search

space. The hybrid NM-PSO algorithm [66] was based on the Nelder-Mead (NM) simplex search method and particle swarm optimization (PSO) for unconstrained optimization. The standard particle swarm optimizers were improved by incorporating a hybridization strategy which was a modification of both the Nelder-Mead simplex search method. Compared to other cooperative PSO (CPSO), this approach could produce faster and more accurate convergence.

According to the discrete characteristic of flow-shop scheduling problem (FSSP), a novel particle swarm optimization (NPSO) algorithm was presented and successfully applied to permutation flow-shop scheduling to minimize makespan [67]. Comparing the NPSO with standard GA, the NPSO was clearly more effective than standard GA for FSSP to minimize makespan. Inspired by the classical PSO method and quantum mechanics theories, the works by Coelho [68] presented a novel Quantum-behaved PSO (QPSO) [69], [70] using chaotic mutation operator. The application of chaotic sequences based on chaotic Zaslavskii map instead of random sequences in QPSO was a powerful strategy to diversify the QPSO population and to improve the QPSO's performance in preventing premature convergence to local minima.

By including the popular genetic algorithm operators such as cross-over and root mean square (RMS) variants into particle swarm optimization (PSO) algorithm the convergence was made faster [71]. Two different PSO algorithms were considered in Arumugam's work [71]: the first one was the conventional PSO (cPSO) and the second was the global-local best-values-based PSO (GLbest-PSO). The GLbest-PSO included global-local best inertia weight (GLbestIW) with global-local best acceleration coefficient (GLbestAC), whereas the cPSO had a time varying inertia weight (TVIW), and either the time-varying acceleration coefficient (TVAC) or the fixed AC (FAC). The results clearly demonstrated that the GLbest-PSO with the cross-over operator was a promising optimization technique. Similar conclusions could be made for the GLbest-PSO with RMS variants as well.

2.1.2 PSO in Variant Environments

Since PSO is simple in concept with few parameters to adjust and is easy to implement, many modifications and adjustments have been made to the basic PSO algorithm over the past decade in many optimization environments. In general, all categories of optimization problems in which evolutionary application techniques are good at are also solvable by PSO. Some PSOs have resulted in improved general performance in a few optimization environments, and some have improved performance on particular kinds of problems. Unfortunately, there are too numerous PSO variants and specializations for us to describe, and we had to leave out some streams in particle swarm research.

A. Binary/Discrete optimization

The original PSO is designed for the real-valued problems. Some researchers extended the real-value version of PSO to binary/discrete space by modifying the updating equations to make them fit in the binary/discrete space.

Kennedy and Eberhart [72] calculated the velocity using the same equation as the position updating equation and a logistic function $s(v) = 1/(1 + \exp(-v))$ to determine the state of x_i^d . If a randomly generated number within $[0,1]$ was less than $s(v_i^d)$, then x_i^d was set to be 1. Otherwise it was set to be 0. Cervantes *et al.* [73] introduced a repulsive force to favor particle competition in the binary PSO above and employed it in two different classification systems - Pittsburgh and the Michigan approaches. Kennedy and Spears [89] compared this binary particle swarm to several kinds of GAs using Spears' multimodal random problem generator. This paradigm allowed the creation of random binary problems with some specified characteristics, e.g., number of local optima, dimensions, and so on. In that study, the binary particle swarm was the only algorithm that found the global optimum on every single trial, regardless of problem features. It also progressed faster than GAs with crossover, mutation, or both, on all problems except the simplest ones, with low dimension and a small number of local optima. The mutation-only GA was slightly faster in those cases.

Mohan and Al-Kazemi [94] suggested several ways that the particle swarm could be implemented on binary spaces. One version, which he called the "regulated discrete particle swarm," performed very well on a suite of test problems. Agrafiotis and Cedeno [74] treated position updating equation as probabilities and used the roulette wheel to determine whether

the new corresponding bit was 1 or 0 in the next generation. Based on the original binary version of PSO (BPSO) introduced by Kennedy in [72], Lee et al. proposed a modified binary version of PSO (BPSO) algorithm, which escaped from a local optimum by employing a bit change mutation. In Pamparä et al. [90], instead of directly encoding bit strings in the particles, each particle stored the small number of coefficients of a trigonometric model (angle modulation) which was then run to generate bit strings.

Applying BPSO on Gene expression data [75], feature selection was implemented using an improved binary particle swarm optimization (IBPSO), and a K -NN served as an evaluator for the feature selection obtained by IBPSO. The procedure of the proposed method was the following: initially, the position of each particle was represented in binary string form and was randomly generated; the bit value {0} and {1} represented a non-selected and selected feature, respectively. Ye *et al.* [76] proposed a new approach based upon binary particle swarm optimization algorithm (BPSO) to find solutions for hard knapsack problems. The standard PSO iteration equations were modified to operate in discrete space. Furthermore, a heuristic operator based on the total-value greedy algorithm was employed into the BPSO approach to deal with constraints. Afshinmanesh *et al.* [77] proposed a novel binary PSO method based on the theory of immunity in biology.

Extending PSO to more complex combinatorial search spaces is also of great interest. The difficulty lies in that notions of velocity and direction have no natural extensions for TSP tours, permutations, schedules, and so on. Nonetheless, progress had been made in [91], [92], [93].

B. Constrained Optimization

An important issue in the constrained optimization is how to handle the constraints. Many constraint-handling techniques have been proposed in the nature-inspired optimization algorithms literatures. By combining those constraint-handling methods with PSO, different PSO variants for constrained optimization have been proposed. Parsopoulos and Vrahatis [78] employed the penalty function and converted the constrained optimization problems into a non-constrained optimization problem. Hu and Eberhart preserved feasible solutions and repaired the infeasible solutions in [79].

For solving constrained large-scale nonlinear non-convex optimization problems, the paper in [80] described an adaptive strategy for tuning the parameters of the PSO method by analyzing the dynamics of PSO. This adaptive tuning strategy was based on the analysis of the dynamics of the average velocity of particles with successful search processes. In another correspondence, an approach based on co-evolutionary particle swarm optimization to solve constrained optimization problems formulated as min-max problems is presented in [81]. A Gaussian probability distribution was used to generate the accelerating coefficients of PSO. Two populations of PSO using Gaussian distribution were used on the optimization algorithm that was tested on a suite of well-known benchmark constrained optimization problems. The Particle Evolutionary Swarm Optimization (PESO) algorithm was introduced for solving single objective constrained optimization problems [82]. PESO algorithm proposed two perturbation operators: “c-perturbation” and “m-perturbation”. The goal of these operators was to prevent premature convergence and the poor diversity issues observed in Particle Swarm Optimization (PSO) implementations.

C. Multimodal Search

Unimodal optimization techniques assume that a single solution exists in a problem's search space. In multimodal search domains, multiple equally acceptable solutions exist. Brits *et al.* proposed a niching particle swarm optimizer, in which multiple sub-swarms were grown from an initial particle swarm by monitoring the fitness of individual particles in [83], [84]. For locating all the global minima of multi-modal functions, Multi-Species Particle Swarm Optimizer (MSPSO) [85] extended the original PSO by dividing the particle swarm spatially into a multiple cluster called a species in a multi-dimensional search space. Each species explored a different area of the search space and tried to find out the global or local optima of that area. Multi-grouped particle swarm optimization (MGPSO), a new algorithm for the multimodal function optimization was proposed in [86]. It showed a straightforward convergence compared to conventional hybrid type approaches as a basic PSO. Moreover, the MGPSO had a unique advantage that one could search for N superior peaks of a multimodal function when the number of groups is N . In [87], PSACO (particle swarm ant colony optimization) algorithm for highly non-convex optimization problems was tested on several benchmark functions in literature. A hybrid method combining two heuristic optimization techniques, genetic algorithms (GA) and particle swarm optimization (PSO), for the global optimization of multimodal functions was presented in [88]. Denoted as GA-

PSO, this hybrid technique incorporated concepts from GA and PSO and created individuals in a new generation not only by crossover and mutation operations as found in GA but also by mechanisms of PSO.

D. Dynamic problems

Dynamic problems are challenging for PSO. These are typically modeled by fitness functions which change over time, rendering particle memory obsolete [95].

Parsopoulos and Vrahatis [96] showed that the particle swarm could track slowly moving optima without any changes at all. Eberhart and Shi [97] made a slight adjustment to the inertia by randomizing the inertia weight between 0.5 and 1.0. The idea was that when tracking a single dynamic optimum, it could not be predicted whether exploration (a larger inertia weight) or exploitation (a smaller inertia weight) would be better at any given time.

However, in many cases more specialized changes are required to handle dynamic problems. The problem can be split into two: how to detect change and how to respond. Carlisle and Dozier, Carlisle and Dozier [98], [99] occasionally re-evaluated the previous best of a single particle. Change was indicated by a different function value upon re-evaluation. They tried two responses to change. In the first, the current position was set to be the previous best and in the second, and more successful response, previous bests were compared with current positions, and memory was updated accordingly. Hu and Eberhart [100] used a similar scheme for change detection but, as a response, randomized the entire swarm in the search space.

As an alternative to these strategies, a level of diversity can be maintained throughout the run. Parsopoulos and Vrahatis [101] used repulsion to keep particles away from detected optima. Blackwell and Bentley [51] introduced charged particles into the swarm. These particles mutually repelled and orbited a converging nucleus of “neutral” particles, in analogy with the picture of an atom. Charged swarms could detect optimum shifts within their orbit and thus were able to track quite severe changes. Diversity can also be maintained with a grid-like local neighborhood [102] and using hierarchical structures [103].

Dynamic multi-modal landscapes are especially challenging for PSO [104]. The strategies mentioned above, restart and diversity enhancement, are less successful in situations where function peaks can vary in height as well as location because a small change in peak height might entail a large change in the position of the global optimum.

In these cases, multi-population approaches have proven to be beneficial. The idea behind multi-swarm models is to position swarms on different peaks so that, should a suboptimal peak become optimal, a swarm will be ready to immediately begin re-optimizing. Parrot and Li [105] adjusted the size and number of swarms dynamically by ordering the particles into “species” in a technique called clearing. Blackwell and Branke [104], borrowing from the atomic analogy, invoked an exclusion principle to prevent swarms from competing on the same peak and an anti-convergence measure to maintain diversity of the multi-swarm as a whole. Recently, a self-adapting multi-swarm had been derived [106]. The multi-swarm with exclusion has been favorably compared, on the moving peaks problem, to the hierarchical swarm, PSO re-initialization and a state-of-the-art dynamic-optimization evolutionary algorithm known as self-organizing scouts.

E. Noisy functions

Noisy fitness functions are important since they are often encountered in real-world problems. In these problems, unlike dynamic problems where the fitness function changes over time, the fitness function remains the same. However, its evaluation is noisy. Therefore, if a PSO explores the same position more than once, the fitness values associated with each evaluation may differ. Parsopoulos and Vrahatis [96] studied the behavior of the PSO when Gaussian distributed random noise was added to the fitness function and random rotations of the search space were performed. Experimental results indicated that the PSO remained effective in the presence of noise. In some cases, noise even helped the PSO avoid being trapped in local optima.

Pugh et al. [107] compared the PSO to a noise-resistant variant where the main PSO loop was modified so that multiple evaluations of the same candidate solution were aggregated to better assess the actual fitness of this particular solution. The comparison considered several numerical problems with added noise, as well as unsupervised learning of obstacle

avoidance using one or more robots. The noise-resistant PSO showed considerably better performance than the original.

F. Multi-objective Optimization

Different from the single-objective optimization, in the multi-objective optimization, there exists more than one objective. The goal of the multi-objective optimization is to find a set of non-dominated solutions, namely Pareto Front (PF). Some researcher convert multi-objective problem to a single objective optimization problem. A more detailed review will be presented in following chapter 6.

2.1.3 Important Real world Applications of PSO

Due to PSO's simplicity in concept, few parameters to adjust and ease of implementation, it has been commonly applied to solve real world optimization problems in extensive areas. A few significant applications of PSO are listed in the Table 2.1.

Table 2. 1 Significant Applications of PSO

Area	Application and reference
Electrical Power Systems	Economic dispatch [108][109][110][126] Reactive Power Control and Power Losses Reduction [111][112][113] Optimal Power Flow (OPF) [114][115][116] Power System Controller Design [117][118][119]
Electric Machinery	Stator fault estimation [121] Cogging torque reduction of transverse flux permanent motor [122] Electromagnetics [125] Antenna array design [234]
Control Systems and Robotics	Controller design and tuning [127],[135] Robot systems [133]

Assignment and Scheduling	Task assignment [128] Task scheduling [129], [130]
Pattern Recognition and Image Processing	Image registration [131] Pattern Recognition [137]
Artificial neural networks	Artificial neural networks training and parameters tuning [120][123][124][132]
Signal processing	Digital filter design [235][236]
Others	Management problem [134] Manufacturing [136]

2.2 Differential Evolution (DE)

Differential evolution grew out of the attempts to use genetic annealing [138] 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 well the Chebyshev problem. Price changed bit-string encoding into floating point and logical operations into arithmetic. Thus, the differential mutation operator was discovered. 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.

The canonical Differential Evolution (DE) algorithm can be illustrated with a flowchart shown in Figure 2.2. Four main stages of canonical DE are summarized as follows.

Figure 2. 2 Flowchart of Canonical Differential Evolution Algorithm

A. Initialization

Similar to other population-based EAs, Differential Evolution (DE) aims at executing a parallel direct search method utilizing NP D -dimensional parameter vectors, which represent the candidate solutions (also called individuals) towards the global optima at G generation, i.e. $\mathbf{X}_{i,G} = \{x_{i,G}^1, \dots, x_{i,G}^D\}$, $i = 1, \dots, NP$. G represents the generation count, and NP is the population size. The uniform randomization of the initial population tries to cover the search space constrained by the prescribed minimum and maximum parameter bounds, $\mathbf{X}_{\min} = \{x_{\min}^1, \dots, x_{\min}^D\}$ and $\mathbf{X}_{\max} = \{x_{\max}^1, \dots, x_{\max}^D\}$, as much as possible. For example, the initial value of the j^{th} parameter of the i^{th} individual at generation $G = 0$ is given by:

$$x_{i,0}^j = x_{\min}^j + rand^j(0,1) \times (x_{\max}^j - x_{\min}^j) \quad j = 1, 2, \dots, D \quad (2.6)$$

where $rand^j(0,1)$ represents a uniformly distributed random variable within the range $[0,1)$, i.e. $0 \leq rand^j(0,1) < 1$ [24]. The superscript j indicates that different random value is generated for each parameter.

B. Mutation operation

After initialization, DE employs the mutation operation to produce a mutant vector $\mathbf{V}_{i,G}$ corresponding to each individual $\mathbf{X}_{i,G}$, the so-called target vector, in the current population. For each target vector $\mathbf{X}_{i,G}$ at the 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 via certain mutation strategy. The most frequently used mutation strategies are summarized as below:

$$\text{“DE/best/1” [19]:} \quad \mathbf{V}_{i,G} = \mathbf{X}_{best,G} + F \times (\mathbf{X}_{r_1^i,G} - \mathbf{X}_{r_2^i,G}) \quad (2.7)$$

$$\text{“DE/best/2” [19]:} \quad \mathbf{V}_{i,G} = \mathbf{X}_{best,G} + F \times (\mathbf{X}_{r_1^i,G} - \mathbf{X}_{r_2^i,G}) + F \times (\mathbf{X}_{r_3^i,G} - \mathbf{X}_{r_4^i,G}) \quad (2.8)$$

$$\text{“DE/rand/1” [19]:} \quad \mathbf{V}_{i,G} = \mathbf{X}_{r_1^i,G} + F \times (\mathbf{X}_{r_2^i,G} - \mathbf{X}_{r_3^i,G}) \quad (2.9)$$

$$\text{"DE/rand/2"} [139]: \quad V_{i,G} = X_{r_1^i,G} + F \times (X_{r_2^i,G} - X_{r_3^i,G}) + F \times (X_{r_4^i,G} - X_{r_5^i,G}) \quad (2.10)$$

$$\left. \begin{array}{l} \text{"DE/rand-to-best/1"} [19] \\ \text{"DE/target-to-best/1"} [24] \end{array} \right\} V_{i,G} = X_{i,G} + K \times (X_{best,G} - X_{i,G}) + F \times (X_{r_1^i,G} - X_{r_2^i,G}) \quad (2.11)$$

$$\left. \begin{array}{l} \text{"DE/rand-to-best/2"} [139] \\ \text{"DE/target-to-best/2"} [19] \end{array} \right\} V_{i,G} = X_{i,G} + K \times (X_{best,G} - X_{i,G}) + F \times (X_{r_1^i,G} - X_{r_2^i,G} + X_{r_3^i,G} - X_{r_4^i,G}) \quad (2.12)$$

$$\text{"DE/current-to-rand/1"} [140] \quad V_{i,G} = X_{i,G} + K \times (X_{r_1^i,G} - X_{i,G}) + F \times (X_{r_2^i,G} - X_{r_3^i,G}) \quad (2.13)$$

The indices $r_1^i, r_2^i, r_3^i, r_4^i, r_5^i$ are mutually exclusive integers randomly generated separately for each mutant vector within the range $[1, NP]$, which are also different from the index i . $\mathbf{X}_{best,G}$ is the best individual vector in terms of the fitness value in the population at generation G . The scale factor $F \in (0, 1+)$ [19] is a positive real number that controls the rate of population evolution. K is randomly chosen within the range $[0, 1]$.

In DE, as evolution progresses, the difference vectors tend to adapt to the natural scaling of the problem. For example, if the range of the search space varies hugely among the dimensions, the whole population becomes compact in one variable with small search domain and remains widely dispersed in another due to the self-adaptive nature of DE. This automatic adaptation significantly improves the convergence of the algorithm. This kind of problems is generally encountered in real-world applications and may be difficult to solve using standard probability distribution.

C. Crossover operation

After the mutation, 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: $\mathbf{U}_{i,G} = \{u_{i,G}^1, u_{i,G}^2, \dots, u_{i,G}^D\}$. The crossover operation speeds up the convergence by a constant factor [24]. DE generally uses two types of crossover operators [24] - *binomial* (or uniform) and *exponential* (or two-point modulo).

In the basic version, DE employs the binomial crossover defined as follows [19]:

$$u_{i,G}^j = \begin{cases} v_{i,G}^j & \text{if } (rand^j(0,1) \leq CR) \text{ or } (j = j_{rand}) \\ x_{i,G}^j & \text{otherwise} \end{cases} \quad j = 1, 2, \dots, D \quad (2.14)$$

In equation (2.14), the crossover probability $CR \in [0, 1]$ is a user-specified constant that controls the fraction of parameter values that are copied to the trial vector from the mutant vector. j_{rand} is a randomly chosen integer in the range $[1, D]$ to make sure that at least one parameter is copied from the mutant vector. 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 $\mathbf{X}_{i,G}$. The condition $j = j_{rand}$ is introduced to ensure that the trial vector $\mathbf{U}_{i,G}$ will differ from its corresponding target vector $\mathbf{X}_{i,G}$. *Binomial* crossover operator is uniform, i.e. regardless of the location of each parameter in the mutant vector, it has the same probability, CR , to inherit its value to the trial vector. For this reason, uniform crossover does not exhibit a representational bias.

In exponential crossover [141], an integer $n \in [1, D]$ is chosen randomly at first, acting as a starting point in the target vector, from where the crossover segment of the mutant vector starts. $L \in [1, D]$ denotes the number of parameters that are contributed by the mutant vector to the target vector. The integer L is drawn from $[1, D]$ depending on the crossover probability (CR), according to the following pseudo-code:

```

L = 0;
WHILE (rand(0, 1) ≤ CR and L < D);
    L = L + 1;
END
    
```

After choosing n and L , the trial vector is obtained as:

$$u_{i,G}^j = \begin{cases} v_{i,G}^j & \text{for } j = \langle n \rangle_D, \langle n+1 \rangle_D, \dots, \langle n+L-1 \rangle_D \\ x_{i,G}^j & \text{for all other } j \in [1, D] \end{cases} \quad (2.15)$$

where the angular brackets $\langle \rangle_D$ denote a modulo function with modulus D .

In exponential crossover operator, the parameters of trial vector $\mathbf{U}_{i,G}$ are inherited from the corresponding mutant vector $\mathbf{V}_{i,G}$ starting from a randomly chosen parameter index (n) till the first time $\text{rand}(0,1) > CR$. The remaining parameters of the trial vector $\mathbf{U}_{i,G}$ are copied from the corresponding target vector $\mathbf{X}_{i,G}$.

D. Selection operation

Following the crossover, the boundary checking is conducted. If the values of some parameters of a newly generated trial vector exceed the corresponding upper and lower bounds, they are reinitialized randomly and uniformly within the pre-specified range. Then the objective function values of all trial vectors are evaluated. After that, a selection operation is performed. For instance, in minimization problem, 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 than or equal objective function value to the corresponding target vector, the target vector will be replaced by the trial vector and the trial vector the population of the next generation. Otherwise, the target vector remains in the population for the next generation. The selection operation can be expressed as follows:

$$X_{i,G+1} = \begin{cases} U_{i,G} & \text{if } f(U_{i,G}) \leq f(X_{i,G}) \\ X_{i,G} & \text{otherwise} \end{cases} \quad (2.16)$$

The above 3 steps are repeated generation after generation until a specific termination criterion (e.g. reaching the maximum number of function evaluations set) is satisfied.

The algorithmic description of the DE can be summarized as:

Initialization

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

Optimization Loop (minimization)

WHILE stopping criterion is not satisfied

FOR $i = 1$ to NP

Mutation

Generate a mutated vector $V_{i,G} = \{v_{i,G}^1, v_{i,G}^2, \dots, v_{i,G}^D\}$ corresponding to the target vector $X_{i,G}$ via one of the equations (2.7 – 2.13).

Crossover

Generate a trial vector $U_{i,G} = \{u_{i,G}^1, u_{i,G}^2, \dots, u_{i,G}^D\}$ for each target vector $X_{i,G}$ through binomial crossover (equation (2.14)) or exponential crossover (equation (2.15))

Selection

Evaluate the trial vector $U_{i,G}$

IF $f(U_{i,G}) \leq f(X_{i,G})$,

THEN $X_{i,G+1} = U_{i,G}$, $f(X_{i,G+1}) \leq f(U_{i,G})$

IF $f(U_{i,G}) < f(X_{best,G})$,

THEN $X_{best,G} = U_{i,G}$, $f(X_{best,G}) \leq f(U_{i,G})$

END IF

ELSE $X_{i,G+1} = X_{i,G}$, $f(X_{i,G+1}) \leq f(X_{i,G})$

END IF

END FOR

Increment the generation count $G = G + 1$

END WHILE

2.2.1 Important Variants of DE

In the recent fifteen years, DE has been attracting the attention of many researchers from diverse fields of Science and Engineering. This has resulted in different variants of the basic DE algorithm. Some of these variants are application specific while others are generalized

for global numerical optimization. The DE variants that exist in literature can be categorized based on:

- 1) Encoding representations.
- 2) Advanced settings of control parameters.
- 3) Diversified mutation strategies.
- 4) Modified Recombination operations.
- 5) Adaptation/self-adaptation techniques.
- 6) Hybrid DE algorithms

2.2.1.1 Encoding Representation

DE was devised for real parameter optimization originally, and has been modified to tackle binary and discrete optimization problems as well. In order to handle integer variables, Lampinen and Zelinka recommended truncating the parameter values for objective function evaluation while the population of DE works with floating-point values [142]. The truncation changed the effective objective function landscape from DE's point of view by introducing flat areas to the fitness landscape. However, DE was able to move across the flat areas introduced by truncation through its self-adaptive reproduction scheme. A discrete variable problem was optimized indirectly. The discrete set of available values was arranged in an ascending sequence and indexed. DE optimized the integer indices given to each discrete value. However, for objective function evaluation the actual discrete value pointed by the index was used and the population of DE worked with floating-point values. Due to indexing, this approach could even handle an unevenly distributed sequence of discrete values.

Based on the *angle modulation* (AM) technique derived from tele-communications, a new DE variant (AMDE) was proposed by Pampara *et al.* [143]. MADE was able to operate in binary spaces without deviating from the basic search mechanism of the classical DE. In MADE, a trigonometric function was employed as a bit string generator. The trigonometric

generating function used in the *angle modulation* technique was a composite sinusoidal function, which was given as:

$$g(x) = \sin(2\pi(x - a) \times b \times \cos A) + d \quad (2.17)$$

where $A = 2\pi \times c \times (x - a)$ and x was a single element from a set of evenly separated intervals, determined by the required number of bits that need to be generated. The DE part was used to evolve the coefficients to the trigonometric function (a, b, c, d), thereby allowing a mapping from continuous-space to binary-space. Thus, instead of evolving the higher-dimensional binary solution directly, angle modulation together with DE was used to convert the complex problem into a 4-dimensional continuous-valued problem.

2.2.1.2 Advanced Settings of control parameters

Being fascinated by the simplicity and smaller number of parameters to tune, DE has attracted much attention recently. However, the performance of DE is more sensitive to the selection of the control parameter values when the problem is complex [144]. Inappropriate choice of parameters may lead to premature convergence, stagnation or wastage of computational resources [24], [144]-[147]. Initially it was thought that the control parameters of DE are not difficult to choose [19], [148]. But due to the complex interaction of control parameters with the DE's performance on hard optimization problems, choosing appropriate control parameters becomes highly demanding [149].

A. Population Size (NP)

Initially $NP = 10D$ was considered as a good choice for DE to obtain a global optima [150]. However, to balance the speed (number of function evaluations) and reliability (ability to find global optimum), different ranges of NP values such as $5D$ to $10D$ [19], $3D$ to $8D$ [144] and $2D$ to $40D$ [23] were suggested.

The larger the population size, the higher is the probability of finding a global optima for multi-modal problems. But, a larger population implies a slower convergence rate requiring a larger number of function evaluations. Therefore, smaller population sizes are required to speed up the convergence in separable and uni-modal functions, while larger population sizes are required to avoid premature convergence in parameter-linked multi-modal

functions. In [24], it was stated that $NP=5D \times CR$ was usually a good low-end default setting, while for highly multi-modal, parameter dependant functions, NP might need to be $10D$ or higher. However, NP should be larger than a critical value depending on the mutation strategy used [144]. For example, NP must be at least 4 for DE/rand/1/bin and 5 for DE/best/2/bin respectively to ensure that DE has enough mutually different vectors.

B. Crossover Rate (CR)

The crossover rate controls which and how many components of an individual in the current population are mutated apart from its pair of target vector [152]. The crossover rate CR is a probability of mixing between trial and target vectors with a range of $0 \leq CR \leq 1$. A large CR speeds up convergence [150], [19], [144]. It was recommended that $CR = 0.1$ was a good initial choice while $CR = 0.9$ or 1.0 could be tried to increase the convergence speed [19]. In [144], a good choice for CR was set to be between 0.3 and 0.9. When solving separable problems, CR from the range (0, 0.2) was the best choice while it was better to choose CR from the range (0.9,1.0) for multi-modal and parameter dependent problems [24], [23]. Based on these observations, CR with non-continuous ranges was also considered [153]. If the number of trial solutions reduced dramatically leading to stagnation when $CR = 1.0$ was chosen, $CR = 0.9$ or 0.99 could be used instead [146], [23].

C. Scaling Factor (F)

Scaling factor F is usually chosen in the range [0.5, 1] [150]. The upper limit of $F = 1.2$ is empirically determined. Optimization problems requiring F greater than 1.2 to solve efficiently have not been encountered. In [19], it was said that values of F smaller than 0.4 and greater than 1.0 were occasionally effective, as long as F was strictly greater than zero. A larger F increases the probability of escaping from a local optimum [144], [23]. However, $F > 1$ slows down the convergence speed, making it difficult to converge as the perturbation is larger than the distance between two members. $F \leq 1$ is usually faster and reliable. F must be above a certain critical value to avoid premature convergence to a sub-optimal solution, but if F becomes too large, the number of function evaluations to find the optimum grows very quickly [144], [23]. The critical value of F is given as [154]

$$F_{crit} = \sqrt{\frac{(1 - CR/2)}{NP}} \quad (2.18)$$

In [19], [144], it was suggested that $F = 0.6$ or 0.5 would be a good initial choice. While in [23], it was mentioned that $F = 0.9$ would be a good initial choice. Typical values of F are between 0.4 to 0.95 according to [23].

Zaharie [155] theoretically proved that DE could converge to global optimum with a long time limit if F could be transformed into a Gaussian random variable. Despite the theoretical proof, it was proved that transforming F into a Gaussian random variable did not significantly enhance DE's performance [156]. In addition, Price *et al.* [24] demonstrated that unless the variance of the randomizing distribution was very small, DE would suffer a significant performance loss on highly conditioned non-separable functions [23]. Similar to CR, $F = 1.0$ is not recommended since it reduces the number of potential trial solutions and may lead to stagnation.

Even though the aforementioned guidelines are useful for choosing the individual parameters of DE to some extent, the performance of DE is more sensitive to the combination of the associated parameters. For a mutation strategy, a particular value of CR makes the parameter F sensitive while some other values of CR make the same F robust [149]. It was recommended in [21] to use the mutation strategy DE/current-to-rand/1/bin and parameter setting $NP = 20D$, $K = 0.5$, $F = 0.8$. If the DE converges prematurely, one should increase the value of NP or F , or randomly choose K within the range $[0, 1]$. If none of the above configuration works, one may try the strategy DE/rand/1/bin with a small CR value.

From the investigations above, it can be observed that various conflicting conclusions had been drawn regarding the manual parameter tuning of DE, which lack sufficient justifications. To avoid the tuning of parameters by trial-and-error procedure, various techniques have been developed. The scale factor F can be linearly reduced from a maximum to a minimum value with increasing generation count [145], or randomly varied in the range $(0.5, 1)$ or a uniform distribution between 0.5 and 1.5 (with mean of 1) can be employed [157]. Several variants of DE [139], [158]-[162] had been proposed based on the different adaptive and self-adaptive parameter control mechanisms used, and are discussed briefly in the following sections.

2.2.1.3 Diversified Mutation Strategies

DE/rand/1/either-or mutation operator proposed by Price *et al.* [24] provides a simple way to implement the dual axis search in the K - F plane (K indicating the combination coefficient of the arithmetic crossover and F being the scale factor). By adopting this mutation strategy, the trial vectors that are pure mutants occur with a probability p_F and those that are pure recombinants occur with a probability $(1 - p_F)$ as shown:

$$U_{i,G} = \begin{cases} X_{r_1,G} + F \times (X_{r_2,G} - X_{r_3,G}) & \text{if } \text{rand}^j(0,1) < p_F \\ X_{r_0,G} + K \times (X_{r_1} + X_{r_2} - 2 \times X_{r_0}) & \text{otherwise} \end{cases} \quad (2.19)$$

Price *et al.* recommended $K = 0.5 \cdot (F + 1)$ as a good choice for the parameter K for a given F . The scheme provided efficient solutions for functions that were best minimized by either mutation-only ($p_F = 1$) or recombination only ($p_F = 0$), as well as generic functions that could be solved by randomly interleaving both operations ($0 < p_F < 1$).

Fan and Lampinen [163] proposed trigonometric mutation operator as a new local search operation that could be embedded into DE to get a better trade-off between the convergence rate and the robustness. The trigonometric mutation scheme can be expressed as:

$$V_{i,G+1} = \frac{(X_{r_1} + X_{r_2} + X_{r_3})}{3} + (p_2 - p_1) \times (X_{r_1} - X_{r_2}) + (p_3 - p_2) \times (X_{r_2} - X_{r_3}) + (p_1 - p_3) \times (X_{r_3} - X_{r_1}) \quad (2.20)$$

The indices $r_1, r_2, r_3 \in [1, NP]$ and $r_1 \neq r_2 \neq r_3 \neq i$. The coefficients were formed according to the following equations:

$$p' = |f(X_{r_1})| + |f(X_{r_2})| + |f(X_{r_3})| \quad (2.21)$$

$$p_1 = \frac{|f(X_{r_1})|}{p'} \quad (2.22)$$

$$p_2 = \frac{|f(X_{r_2})|}{p'} \quad (2.23)$$

$$p_3 = \frac{|f(X_{r_3})|}{p'} \quad (2.24)$$

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

By using the objective value of the individuals in the mutation operation, the perturbations could be biased towards the points with lowest objective values. This made the trigonometric mutation operator greedy and useful for local search. The scheme proposed in [163] used trigonometric mutation with a probability of defined as Γ and the mutation scheme of DE/rand/1 with a probability of $(1 - \Gamma)$.

To improve the performance of DE by providing a proper trade-off between exploration and exploitation, Das *et al.* [141] proposed two kinds of neighborhood models: a *local neighborhood model* and a *global neighborhood model*. In *local neighborhood model* each vector was mutated using the best position found so far in a small neighborhood around it and *not* in the entire population. The global neighborhood model took into account the global best vector $X_{best,G}$ of the entire population at current generation G for mutating a population member. For a DE population $P_G = [X_{1,G}, X_{2,G}, \dots, X_{NP,G}]$ at generation G , the vector indices were sorted randomly (as obtained during initialization) in order to preserve the diversity of each neighborhood. For every vector $X_{i,G}$, the mutant vector $V_{i,G}$ was formed as:

$$V_{i,G} = w \cdot g_{i,G} + (1 - w) \cdot L_{i,G} \quad (2.25)$$

$L_{i,G}$ and $g_{i,G}$ are local and global donor vectors respectively and $w \in (0,1)$ is a scalar weight. A neighborhood of radius k (where k was a non-zero integer from 0 to $(NP-1)/2$ as the neighborhood size must be smaller than the population size i.e. $2k+1 \leq NP$), consisting of vectors $X_{i-k,G}, \dots, X_{i,G}, \dots, X_{i+k,G}$ was defined. The vectors were organized on a ring topology with respect to their indices, such that vectors $X_{NP,G}$ and $X_{2,G}$ were the two immediate neighbors of vector $X_{1,G}$. For each member of the population a local donor vector was created by employing the best (fittest) vector in the neighborhood of that member and any two other vectors chosen from the same neighborhood. The model may be expressed as:

$$L_{i,G} = X_{i,G} + \alpha \cdot (X_{n_best,G} - X_{i,G}) + \beta \cdot (X_{p,G} - X_{q,G}) \quad (2.26)$$

where the subscript n_best_i indicates the best vector in the neighborhood of $X_{i,G}$ and $p, q \in [i-k, i+k]$ with $p \neq q \neq i$. Similarly the global donor vector was created as:

$$g_{i,G} = X_{i,G} + \alpha \cdot (X_{g_best,G} - X_{i,G}) + \beta \cdot (X_{r_1,G} - X_{r_2,G}) \quad (2.27)$$

where the subscript g_best indicates the best vector in the entire population at iteration G and $r_1, r_2 \in [1, NP]$ with $r_1 \neq r_2 \neq i$. α and β are the scaling factors.

Clearly if $w = 1$ and in addition $\alpha = \beta = F$, the donor vector generation scheme in equation (2.27) reduced to that of DE/target-to-best/1. Hence the latter may be considered as a special case of this more general strategy involving both global and local neighborhood of each vector synergistically.

Maulik and Saha [164] proposed a modified DE algorithm for fuzzy clustering application. The proposed algorithm was different from classical DE algorithm in the process of mutation. In the proposed scheme, a modified mutation strategy was used with a probability of α and the mutation scheme of DE/rand/1 with a probability of $(1 - \alpha)$. The modified mutation strategy was carried out with three vectors; the local best ($LBest$) and the global best ($GBest$) which were adaptive in nature, and a randomly selected vector. The crossover and selection were same as in original DE. The mutation operation could be represented as:

$$V_{i,G+1} = \begin{cases} X_{GBest,G} + \alpha \cdot (X_{LBest,G} - X_{r_3,G}) & \text{if } rand(0,1) \leq \alpha \\ X_{r_1,G} + F \cdot (X_{r_2,G} - X_{r_3,G}) & \text{otherwise} \end{cases} \quad (2.28)$$

where $\alpha = 1/(1+\exp(-1/\text{generation}))$, The α value decreased from a maximum of 1.0 in the initial generation to a minimum of 0.5 in the final generation. This results in a higher probability of using modified mutation in the initial generation and a lower probability in the later generations of the evolution. In the initial generation, when modified mutation was used, $LBest$ had more contribution for evolving the mutant vector than in the later stages of evolution process. As the contribution of $LBest$ for the mutant vector decreased with generations, the contribution of $Gbest$ increased. This allowed the trial vector to be pushed to global optima quickly.

2.2.1.4 Modified Recombination operations

Discrete recombination, also known as *crossover*, is an operation in which trial vector parameters are copied randomly from selected vectors (target and mutant vectors). Since it only copies information, crossover can be applied to binary, real-valued or even symbolic data. In contrast, *continuous* or *arithmetic recombination* expresses trial vectors as linear combinations of mutant and target vectors, so it is inapplicable to symbolic data and inappropriate for binary variables. *Continuous* or *arithmetic recombination* can be further classified into *line recombination* and *intermediate recombination*.

The line recombination between two vectors $X_{r_1,G}$ and $X_{r_2,G}$ can be represented as:

$$W_{i,G} = X_{r_1,G} + k_i \cdot (X_{r_1,G} - X_{r_2,G}) \quad (2.29)$$

The *coefficient of combination* k_i can either be a constant or a uniform random variable between 0 and 1 sampled anew for each vector.

The intermediate recombination between two vectors $X_{r_1,G}$ and $X_{r_2,G}$ for the j -th component can be represented as:

$$w_{i,j,G} = x_{r_1,j,G} + k_j \cdot (x_{r_1,j,G} - x_{r_2,j,G}) \quad (2.30)$$

The *coefficient of combination* k_j can either be a constant or a uniform random variable between 0 and 1 sampled anew for each component of the vectors.

Figure 2.3 schematically shows the regions searched by discrete, line and arithmetic recombination between mutant vector $V_{i,G}$ and the target vector $X_{i,G}$ when the coefficient of combination is a uniformly distributed random number between 0 and 1. The two recombinant vectors occupy the opposite corners of a hypercube whose remaining corners are the trial vectors $U'_{i,G}$ and $U''_{i,G}$ created by discrete recombination. Line recombination searches along the axis connecting the recombinant vectors, while the intermediate recombination explores the entire D -dimensional volume contained within the hypercube. As can be perceived from Figure 2.4, neither the discrete nor the intermediate recombination is rotationally invariant processes. If the coordinate system rotates by an angle, the corners of the hypercube are relocated, which in turn redefines the area searched by the intermediate

recombination. On the other hand, the line recombination is rotationally invariant. $U_{i,G}^{R/}$ and $U_{i,G}^{R//}$ indicate the new trial vectors due to discrete recombination in rotated coordinate system.

Since a good global optimizer needs to be rotationally invariant, to overcome the limitation of the discrete recombination usually employed in DE, a new trial vector generation strategy “DE/current-to-rand/1” was proposed in [151], which replaced the binomial crossover operator with the rotationally invariant arithmetic line recombination operator to generate the trial vector $U_{i,G}$ by linearly combining the target vector $X_{i,G}$ and the corresponding donor vector $V_{i,G}$ as follows:

$$U_{i,G} = X_{i,G} + k_i \cdot (V_{i,G} - X_{i,G}) \quad (2.31)$$

After incorporating equation (2.9) in equation (2.31) we have:

$$U_{i,G} = X_{i,G} + k_i \cdot (X_{r_1,G} + F \cdot (X_{r_2,G} - X_{r_3,G}) - X_{i,G}) \quad (2.32)$$

which further simplifies to:

$$U_{i,G} = X_{i,G} + k_i \cdot (X_{r_1,G} - X_{i,G}) + F' \cdot (X_{r_2,G} - X_{r_3,G}) \quad (2.33)$$

where k_i is the combination coefficient, which has been experimentally shown [24], [151] to be effective when it is chosen with a uniform random distribution from [0, 1] and $F' = k_i \cdot F$ is a new scaling constant.

Figure 2. 3 Domains of the different recombinant vectors generated using discrete, line and random intermediate recombination.

Figure 2. 4 Change of the trial vectors generated through the discrete and random intermediate recombination due to rotation of the coordinate system.

2.2.1.5 Adaptation/self-adaptation techniques

DE is sensitive to parameter settings while solving some of the objective functions [165]. The performance of DE may be degraded by low convergence speed or premature convergence which are due to improper selection of control parameters, parameters being kept fixed through the whole process without response to the evolving population's information, the environment in which the DE operated being variable, as well as lack of knowledge from the search space [159]. To avoid manual tuning of parameters and to obtain an optimal performance, several variants of DE based on the adaptation and self-adaptation of mutation strategies and control parameters have been proposed recently.

From the observations above, Fuzzy Adaptive Differential Evolution (FADE) was proposed to adapt the control parameters F and CR based on fuzzy logic controllers, whose inputs were the relative function values and individuals of successive generations, to adapt the parameters for the mutation and crossover operation [159]. FADE chose the initial control parameters freely and adjusted the control parameters on-line to dynamically adapt to changing situations. FADE outperformed the conventional DE on higher dimensional problems. A parameter adaptation of DE (ADE) based on controlling the population diversity and a multi-population approach was proposed [146], and was later extended to form an adaptive Pareto DE algorithm for multi-objective optimization [166]. Self-adapted crossover rate of DE for multi-objective optimization problems was presented, based on encoding the crossover rate into each individual, while other parameters were simultaneously evolved [158]. The scale factor F was generated using a Gaussian distribution $N(0,1)$ for each individual. Omran *et al.* [160] introduced a self-adaptation scheme (SDE) in which CR was generated randomly for each individual using a normal distribution $N(0.5,0.15)$, while scale factor F was adapted analogously to the adaptation of crossover rate CR in [158]. In addition to control parameters F and CR , based on Abbas's self-adaptive Pareto DE, Teo proposed DE with self-adapting populations (DESAP) [161]. In DESAP, the population size (NP) was automatically adapted from initialization to the completion of the evolution process. DESAP was proposed with 2 encoding methodologies: the *absolute encoding methodology* and *relative encoding methodology*. In both methodologies, a self-adaptive strategy was used to control the crossover and mutation rates.

In [167], a fitness-based adaptation on scale factor F was proposed. A system with two evolving populations had been implemented. The crossover rate CR was set equal to 0.5 from an empirical study. Unlike CR , the value of F was adaptively updated at each generation by the following scheme:

$$F = \begin{cases} \max \left\{ l_{\min}, 1 - \left| \frac{f_{\max}}{f_{\min}} \right| \right\} & \text{if } \left| \frac{f_{\max}}{f_{\min}} \right| < 1 \\ \max \left\{ l_{\min}, 1 - \left| \frac{f_{\min}}{f_{\max}} \right| \right\} & \text{otherwise,} \end{cases} \quad (2.34)$$

where $l_{\min} = 0.4$ was the lower bound of f , $f_{\min}$ and $f_{\max}$ were the minimum and maximum objective function values over the individuals of the populations obtained in a generation.

A self-adaptation scheme (jDE [168] or SADE) was proposed [168], in which control parameters F and CR were encoded into the individuals and adjusted by introducing two new parameters τ_1 and τ_2 . The appropriate values of these encoded control parameters led to better individuals, which were more likely to survive and to produce offspring, i.e. to propagate these better parameter values. Initially, F and CR values of each individual were assigned to 0.5 and 0.9 respectively. The new control parameters for the next generation were computed as follows:

$$F_{i,G+1} = \begin{cases} F_l + rand_1 \times F_u & \text{if } rand_2 < \tau_1 \\ F_{i,G} & \text{otherwise} \end{cases} \quad (2.35)$$

$$CR_{i,G+1} = \begin{cases} rand_3 & \text{if } rand_4 < \tau_2 \\ CR_{i,G} & \text{otherwise} \end{cases} \quad (2.36)$$

where $rand_j, j \in \{1,2,3,4\}$ were uniform random values in the interval $[0, 1]$, τ_1 and τ_2 interpreted as the probabilities of adjusting the factors F and CR . Brest *et al.* [168] used $\tau_1 = \tau_2 = 0.1$. As $F_l = 0.1$ and $F_u = 0.9$, the new F took a value from $[0.1, 0.9]$ while the new CR took a value from $[0, 1]$. As the $F_{i,G+1}$ and $CR_{i,G+1}$ values were obtained before the mutation was performed, they influenced the mutation, crossover, and selection operations of the new vector $X_{i,G+1}$. However, it remained unclear whether the scheme would maintain

its superiority if not a fixed number of evaluations but a fixed value-to-reach (VTR) had been chosen as a goal.

DE can encompass a number of trial vector generation strategies, each of which may be effective over certain problems but performing poorly over the others, or good during some search stages but failing in others. Qin *et al.* [139] proposed a self-adaptive DE algorithm (SaDE) in which the mutation strategies and the respective control parameter were self-adapted based on their previous experiences of generating promising solutions. Consequently, it was possible to determine a more suitable generation strategy along with its control parameter settings adaptively to match different phases of the search process / evolution. The scale factor, F , was randomly generated with a mean and standard deviation of 0.5 and 0.3 respectively. In SaDE, four effective trial vector generation strategies namely the DE/rand/1/bin, DE/rand-to-best/2/bin, DE/rand/2/bin and finally DE/current-to-rand/1 were chosen to constitute a strategy candidate pool. The first three DE-variants were equipped with binomial type crossover while the last one used arithmetic recombination. In the SaDE algorithm, for each target vector in the current population, one trial vector generation strategy was selected from the candidate pool according to the probability learned from its success rate in generating improved solutions (that can survive to the next generation) within a certain number of previous generations, called the *Learning Period* (LP). The selected strategy was 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 were summed to 1. These probabilities were initially equal ($1/K$ for K total number of strategies in the pool) and were then gradually adapted during evolution, based on the *Success* and *Failure Rates* over the previous LP generations. The adaptation of the probabilities took place in such a fashion that, the larger the success rate was for the k -th strategy in the pool within the previous LP generations, the larger the probability of applying it was to generate trial vectors at the current generation.

In JADE [169], a new mutation strategy “DE/current-to- p best”, a generalization of the classic “DE/current-to-best”, with *optional external archive* was proposed. The reliability of the algorithm was further improved by the adaptive parameter control. The “DE/current-to- p best”, unlike its classical version, utilized not only the information of the best solution but also the information of other good solutions.

$$V_{i,G} = X_{i,G} + F_{i,G} \cdot (X_{best,G}^P - X_{i,G}) + F_{i,G} \cdot (X_{r_1,G}^j - X_{r_2,G}^j) \quad (2.37)$$

where $X_{i,G}$ and $X_{best,G}^p$ were selected from P_G (the current population), while $X_{r_2,G}$ was randomly chosen from the union, $P_G \cup A$ (for the archive case) or P_G (for none-archive case). A was a archive of recently explored inferior solutions compared to the current population, which provided additional information about the promising progress direction. $X_{best,G}^p$ was randomly chosen as one of the top $100p\%$ individuals in P_G with $p \in (0,1]$. In spite of its greedy property, the proposed mutation strategy was able to diversify the population so that problems such as premature convergence could be alleviated.

$F_{i,G}$ was the mutation factor of $X_{i,G}$, generated independently at each generation G , according to a Cauchy distribution with location parameter μ_F and scale parameter 0.1

$$F_{i,G} = \text{randc}_i(\mu_F, 0.1) \quad (2.38)$$

It was then truncated to be 1.0 if $F_i \geq 1.0$ or regenerated if $F_i \leq 0$. S_F was denoted as the set of all successful mutation factors in generation G . The location parameter μ_F of the Cauchy distribution was initialized to be 0.5 and then updated at the end of each generation as

$$\mu_F = (1-c) \cdot \mu_F + c \cdot \text{mean}_L(S_F) \quad (2.39)$$

where $\text{mean}_L(.)$ was the Leher mean defined as follows,

$$\text{mean}_L = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F} \quad (2.40)$$

$CR_{i,G}$ was the crossover probability of $X_{i,G}$, generated independently at each generation G , according to a normal distribution of mean μ_{CR} and standard deviation 0.1

$$CR_{i,G} = \text{randn}_i(\mu_{CR}, 0.1) \quad (2.41)$$

It was then truncated to $[0, 1]$. S_{CR} was denoted as the set of all successful crossover probabilities in generation G . The mean μ_{CR} was initialized to be 0.5 and then updated at the end of each generation as

$$\mu_{CR} = (1-c) \cdot \mu_{CR} + c \cdot \text{mean}_A(S_{CR}) \quad (2.42)$$

where $\text{mean}_A(.)$ was the usual arithmetic mean.

The performance of JADE was insensitive to the setting of the new parameters c that controlled the rate of parameter adaptation and p that determined the greediness of the mutation strategy. JADE usually performed better with $1/c \in [5, 20]$ and $p \in [5\%, 20\%]$, that is to say, the life span of μ_{CR} and μ_F values ranged from 5 to 20 generations and top 5-20% high quality solutions were considered for the mutation.

2.2.1.6 Hybrid DE algorithms

Over the past few years, DE has been successfully hybridized with optimization algorithms, by combining the best features of DE and other algorithms, to form a new algorithm that is expected to outperform its ancestors over application-specific or general benchmark problems. Significant hybrid DE algorithms are summarised here:

A. DE hybrid with local search techniques

- Neighborhood Search (NS) [170]
- Adaptive Local Search [171]
- Self-adaptive local search [172]
- crossover-based local search (XLS) with adaptive length [173]

B. DE hybrid with other global optimization algorithms

- PSO (Particle Swarm Optimization) [61]-[64], [174]-[178]
- SA (Simulated annealing) [179]
- ACS (Ant Colony Systems) [180], [182]
- AIS (Artificial Immune Systems) [183], [184]
- BFOA (Bacterial Foraging Optimization Algorithm) [185]

C. DE integrated into Memetic Algorithms (MAs), one of the recent growing areas of research in evolutionary computation, denoting a synergy of evolutionary or any

population-based approach with separate individual learning or local improvement procedures [186], [187].

2.2.2 Important Real world Applications of DE

Fascinated by the simplicity and efficient property, DE has been applied to various fields of engineering and science. A few significant applications are listed in Table 2.2.

Table 2. 2 Applications of DE

Area	Application and reference
Electrical Power Systems	Economic dispatch [188]-[191] Optimal power flow and Power system planning [192]-[196] Power system planning, generation expansion planning [197]-[200]
Electromagnetism and microwave Engineering	Electromagnetic inverse scattering [201] Design of circular waveguide mode converters [202] Antenna array design [203], [204]
Control Systems and Robotics	System identification [205][206] Optimal control [207] Controller design and tuning [208][209] Robot motion planning and navigation [210], [211]
Pattern Recognition and Image Processing	Data clustering [212] Pixel clustering and region-based image segmentation [164][212] Feature extraction [213] Image registration and enhancement [214]
Artificial neural networks (ANN)	Training of feed-forward ANNs [215] Training of wavelet neural networks (WNNs) [216] Training of B-Spline neural networks [217]
Signal Processing	Digital filter design [218]-[221]
Bioinformatics	Gene regulatory networks [222][223] Micro-array data analysis [224] Protein folding [225]

Chemical Engineering	Chemical process synthesis and design [226] Phase equilibrium and phase study [227] Parameter estimation of chemical process [228]
Others	Layout synthesis for MEMS [229] Engineering design [230] Manufacturing Optimization [231] Optoelectronics [232]

Chapter 3

Diversity Enhanced Particle Swarm Optimizer (DivEnh-PSO & DivEnh-CLPSO)

3.1 Introduction

The particle swarm optimization (PSO) algorithm is a population based search strategy, which has exhibited good performance in solving real-parameter optimization problems. However, a major problem associated with the PSO is premature convergence when solving strongly multimodal problems frequently resulting in locally optimal solutions. Rapid information flow between particles in the standard PSO seems to be the reason for clustering of particles. Diversity reduces rapidly, making it difficult for the basic PSO algorithm to escape from local optima when solving multimodal problems. Consequently, the decrease of diversity in search space leads to a total implosion and consequent fitness stagnation of the

swarm. Hence, maintaining high diversity is crucial for preventing premature convergence of the PSO when solving multimodal optimization problems.

As we mentioned in the previous section 2.1.1.2, there are global and local PSO variants. In this chapter, the Diversity Enhanced basic Particle Swarm Optimizer (DivEnh-PSO) and Diversity Enhanced Comprehensive Learning Particle Swarm Optimizer (DivEnh-CLPSO) use the global version of the PSO to retain its faster convergence while enhancing its diversity by using an external archive of high quality and diverse past solutions. The usage of external memory containing past solutions enables us to enhance diversity using highly fit solutions without requiring additional function evaluations.

Although maintaining high diversity and obtaining fast convergence is partially contradicting, it makes perfectly good sense to try to improve the diversity in an attempt to avoid convergence to a locally optimal solution. Several attempts have been made on this subject with EAs and SIs [237], [238], [239], [240], [241], [242]. In an effort to handle this tradeoff in PSO models, some notable diversity increasing approaches have been proposed [243], [244], [245]. The Spatial Extension PSO (SEPSO) involves endowing each particle with a radius and then causing particles to bounce back from one another [243]. A related approach, called Attractive-Repulsive PSO (ARPSO), measures the global diversity of the swarm and triggers modes of global attraction or repulsion when it crosses predefined thresholds [245]. Though effective when well-tuned, finding good problem specific parameters for these methods is non-trivial. Benefiting from SEPSO, its behavior has been improved by adapting its radius and bounce parameters in response to collisions in Adaptive Diversity PSO in [244], which is simple to implement and effective if tuned correctly. However, these approaches require additional function evaluations after reversing the velocity to create post-bounce positions. In [240], a replacement strategy based on the contribution of diversity of an offspring to the population is proposed to address the population diversity versus selective pressure issue in the replacement phase of the steady-state GA. The objective is to replace an individual of the population, with a poor fitness value and with a lower contribution to diversity, by a better offspring. The replacement phase is conducted immediately after offspring are generated every generation. A probabilistic update rule and a sampling procedure based on incremental learning are proposed in [242]. This method is based on the concept of opposition to generate new

solutions to enhance the diversity within a given population. Based on the observation that solutions which differ only in a few variable values will contribute less to the overall diversity than solutions which differ in many values, a probability mutation rule is applied by randomly forcing changes in several parameter values of an existing solution. No memory is used in all the above methods, while in our proposed work, an external memory is incorporated and updated by using the solutions generated every iteration based on both their diversity and fitness. Furthermore, the diversity of the population is enhanced frequently by replacing the weak population individuals by selected diverse memory members with good fitness.

Based upon the Basic PSO version [26] as well as Comprehensive Learning Particle Swarm Optimizer (CLPSO) model presented in [48], in this chapter, a novel diversity enhanced PSO (DivEnh-PSO) and CLPSO (DivEnh-CLPSO) are developed by using an external Diversity Memory (DM) to enhance the diversity of the whole swarm without requiring any additional function evaluations. The objective of this external memory is different from that used in multi-objective evolutionary algorithms (MOEAs) and dynamic problem optimization. In MOEAs [246], [247], [248], the external archive is used as a repository of current non-dominated solutions. In dynamic optimization, the external memory is used to store good past solutions and to detect changes. The DM used in our DivEnh-CLPSO holds selected past solutions with good diversity as well as good fitness values. It is found to be an effective method, yet conceptually it is simple as well as being very easy to implement

3.2 DivEnh-PSO and DivEnh-CLPSO

In this section, starting with the review of the basic PSO [26] and Comprehensive Learning Particle Swarm Optimizer (CLPSO) [48], we will present how we maintain the diversity memory using the past solutions found by the PSO and CLPSO, as well as the way we inject the selected members from the diversity memory into the swarm periodically during the CLPSO search process to enhance the diversity.

3.2.1 The original basic PSO and original CLPSO

In this section 2.1.1.1, the basic PSO [26] has been introduced. For each particle, the velocity updating and position updating schemes are given in equation (2-2). The main procedure of this PSO version is presented in Figure 2.1 in Chapter 2.

Although there are numerous variants of the PSO in the past years, premature convergence when solving multimodal problems is still the main deficiency. As a variant of the particle swarm optimizers, the Comprehensive Learning Particle Swarm Optimizer (CLPSO) was proposed to solve single objective optimization problems with bound constraints [48]. CLPSO uses a learning strategy whereby all other particles' historical best information is used to update a particle's velocity. This strategy enables the diversity of the swarm to be preserved to discourage premature convergence.

In the original PSO, when updating the velocity, the following equation is used:

$$V_i^d \leftarrow rand1_i^d \times (pbest_i^d - X_i^d) + rand2_i^d \times (gbest^d - X_i^d) \quad (3.1)$$

In this equation, each particle learns from its **pbest** and **gbest** simultaneously. Restricting the social learning aspect to only the **gbest** makes the original PSO converge fast. However, because all particles in the swarm learn from the **gbest** even if the current **gbest** is far from the global optimum, particles may easily be attracted to the **gbest** region and get trapped in a local optimum if the search space is complex with numerous local solutions. The fitness value of a particle is possibly determined by all dimensions, and a particle which has discovered the values corresponding to the global optimum in some dimensions may have a low fitness value because of the poor solutions in the other dimensions. Based on the above analysis, a new learning strategy was designed in CLPSO. In this new learning strategy, the following velocity updating equation was used:

$$V_i^d \leftarrow w \times V_i^d + c \times rand_{fi(d)}^d \times (pbest_{fi(d)}^d - X_i^d) \quad (3.2)$$

where $f_i = [f_i(1), f_i(2), \dots, f_i(D)]$ defines which particle's **pbest** particle i should follow. $pbest_{fi(d)}^d$ can be the d^{th} dimension of any other particle's **pbest** or its own **pbest**, and the decision depends on probability Pc , referred to as the learning probability, which can take different values for different particles. For each dimension of particle i , a random number is generated. If this random number is larger than Pc_i , this dimension will learn from

its own *pbest*, otherwise it will learn from another particle's *pbest*. CLPSO employs a tournament selection procedure with size two when the particle's dimension learns from another particle's *pbest*.

All these $pbest_{f_i}$ contribute to the generation of new positions in the search space using the information derived from different particles' historical best positions. To ensure that a particle learns from good exemplars and to minimize the time wasted on poor directions, CLPSO allows a particle to learn from the exemplars until the particle ceases improving for a certain number of iterations called the refreshing gap m after which a different f_i for the particle is re-generated. The original CLPSO is summarized as follows:

INITIALIZE

Initialize particle positions, associated velocities, f_i , assign pbest and gbest of the population $pops$, set $fitcount=pops$.

REPEAT

FOR EACH PARTICLE $i=i+1, i < pops$

$$w = w_0 - \frac{(w_0 - w_1) \times fitcount}{Max_FES}$$

IF $flag_i \geq m$, **THEN**

For each dimension $d=d+1, d < D$

$$V_i^d = \omega \times V_i^d + c \times rand_i^d \times (pbest_{f_i(d)}^d - X_i^d)$$

$$V_i^d = \min(V_{\max}^d, \max(-V_{\max}^d, V_i^d))$$

$$X_i^d = X_i^d + V_i^d$$

$$X_i \in [X_{\min}, X_{\max}]$$

Evaluate the new position X_i , $fitount=fitount+1$;

Update the *pbest* and *gbest* if better solution generated.

Update the $flag_i$.

UNTIL $fitcount \geq Max_FES$.

In the above algorithm, “ $pops$ ” is the population size; “ i ” is the particle’s id counter from 1 to $pops$; “ d ” is dimension; X_i^d is i^{th} particle’s d^{th} dimension’s value; w is inertia weight; m is the refreshing gap; $fitcount$ is the current number of fitness evaluation from 1 to Max_FES ; Max_FES is the maximum fitness evaluations; $flag_i$ is the number of iterations the i^{th} particle has not improved its own $pbest$.

3.2.2 Population Diversity Measures

In PSO and CLPSO, the population POP, consists of i ($i \in pops$) individuals, can be denoted in a matrix form or in a set form, depending on appropriateness. The matrix form of the population is [249]:

$$\text{POP} = \begin{bmatrix} X_1^1 & X_1^2 & \cdots & X_1^D \\ X_2^1 & X_2^2 & \cdots & X_2^D \\ \vdots & \vdots & \ddots & \vdots \\ X_{pops}^1 & X_{pops}^2 & \cdots & X_{pops}^D \end{bmatrix} \in \mathbb{R}^{pops \times D} \quad (3.3)$$

$$X_i^d \in \mathbb{R}, i = 1, 2, \dots, pops; d = 1, 2, \dots, D$$

Here, rows represent individuals in $\mathbb{R}^D$. A set form of the population is:

$$\text{POP} = \{X_1, X_2, \dots, X_{pops}\}, \quad (3.4)$$

where $X_i = [X_i^1 \ X_i^2 \ \dots \ X_i^D]$ denotes i -th particle within the observed population ($i=1, 2, \dots, pops$).

There are two most frequently used diversity measurement methods. One is based on summing and averaging of all mutual Euclidian distances between individuals [250], [251], [252]:

$$D_1(\text{POP}) = \frac{2}{\text{pops}(\text{pops}-1)} \sum_{i=1}^{\text{pops}-1} \sum_{j=i+1}^{\text{pops}} \|X_i - X_j\|. \quad (3.5)$$

The other one is based on summing (averaging) Euclidian distances of all points to the average point [253], [254], [255]:

$$D_2(\text{POP}) = \frac{1}{\text{pops}} \sum_{i=1}^{\text{pops}} \|X_i - \bar{X}\|, \quad \bar{X} = \frac{1}{\text{pops}} \sum_{i=1}^{\text{pops}} X_i. \quad (3.6)$$

In the DivEnh-CLPSO, we use the diversity measure D_1 as the criterion when updating the diversity memory. The details on how DM members are selected to be injected into the swarm are presented in the next subsection.

3.2.3 DivEnh-PSO and DivEnh-CLPSO

Particles' velocity updating and position updating in the DivEnh-PSO and DivEnh-CLPSO are the same as in the original PSO and CLPSO reviewed in subsection 3.2.1. The two main modules incorporated into the PSO and CLPSO are the Diversity Memory Updating (DMU), executed every iteration and the Population Diversity Enhancement (PDE) done periodically.

After population initialization, all population members are stored in an archive named Diversity Memory (DM). During every iteration in the first 80% of the function evaluations, new positions are stored in the DM. When the size of the DM exceeds a predefined threshold, the diversity measure based on Euclidean distance explained in subsection 2.2 is used to compute the diversity of the combined new positions and memory members. Here all distances between all pairs of members are calculated. For each member, all distances to all other members are summed up. Based on the resulting summation values, they are sorted in descending order. Those points with smallest summation of distances will be discarded from the memory as they are clustered with other solutions. In this way, the memory is truncated to the predefined memory size to keep the most diverse members in the DM.

During the first 80% of function evaluations, a certain percentage of the population members with worst fitness values as well as their *pbests* are periodically replaced by candidates selected from the DM based on their high diversity and fitness values. During the early search stage, the members from the DM with the largest distances with respect to the current *gbest* are selected to replace 20% of the worst population members and their *pbests*. As the number of iterations increases, the number of high diversity members linearly increases to

the whole of the DM and out of them, based on the fitness value, the best candidates are chosen to replace the worst 20% of the population members and their *pbests*. Hence, this approach can initially consider primarily the diversity of population and subsequently convergence when performing the Population Diversity Enhancement (PDE) operation. In the last 20% of the function evaluations, we use the original PSO or CLPSO without the PDE. The DivEnh-PSO or DivEnh-CLPSO algorithm can be summarized as follows:

INITIALIZE

Initialize particle positions, associated velocities, assign *pbest* and *gbest* of the population *pops*, store all the population into DM, set *fitcount* = *pops*.

REPEAT

FOR EACH PARTICLE $i=i+1, i < pops$

Original PSO or CLPSO particle velocity updating and position updating

IF $fitcount < 80\% \times Max_FES$. **THEN**

Store the new position into DM. Check the diversity of the combining the new positions and memory members, truncate the DM if it exceeds the pre-defined threshold.

IF $mod(fitcount, RF) = 0$. **THEN**

Replace the *RR* of population members with worst fitness value and their *pbests* by the selected diversity memory members

IF $fitcount > 80\% \times Max_FES$. **THEN**

Original PSO or CLPSO without the PDE.

UNTIL $fitcount \geq Max_FES$.

where “*pops*” is the population size; *RR* is the replacement ratio; *RF* is the Replace Frequency of the PDE; *fitcount* is the current number of fitness evaluation from 1 to *Max_FES*; *Max_FES* is the maximum fitness evaluations.

3.3 Experimental Setup and Results

3.3.1 Test Functions Considered in the Experimental Study

Since our main objective is to evaluate the performance of the DivEnh-PSO and DivEnh-CLPSO on hard optimization problems, we choose several multimodal benchmark problems [256] with 10 and 30 dimensions. According to their properties, these problems are divided into three groups: 8 *unrotated multimodal problems*, 8 *rotated multimodal problems* and 2 *composition problems*. The properties and the expressions of these problems are presented as following and detailed function definitions are list in Appendix B.

Group A: Unrotated Multimodal Problems:

In this group, there are eight multimodal test functions. i.e. *Ackley’s function* (f_1), *Griewank’s function* (f_2), *Weierstrass function* (f_3), *Rastrigin’s function* (f_4), *Non-continuous Rastrigin’s function* (f_5), *Schwefel’s function* (f_6), *Michalewicz’s function* (f_7), *Sum of powers function* (f_8)

Ackley’s function has one narrow global optimum basin and many minor local optima. It is probably the easiest problem among the six as its local optima are neither deep nor wide.

Griewank’s function has a $\prod_{i=1}^D \cos(\frac{x_i}{\sqrt{i}})$ component causing linkages among dimensions. An

interesting phenomenon of Griewank’s function is that it is more difficult for lower dimensions than higher dimensions [257]. The Weierstrass function is continuous but differentiable only on a set of points. Rastrigin’s function is a complex multimodal problem with a large number of local optima. When attempting to solve Rastrigin’s function, algorithms may easily fall into a local optimum. Hence, an algorithm capable of maintaining a larger diversity is likely to yield better results. Non-continuous Rastrigin’s function is constructed based on the Rastrigin’s function and it has the same number of local optima as

the continuous Rastrigin's function. The complexity of Schwefel's function is due to its deep local optima being far from the global optimum. It will be hard to find the global optimum, if many particles fall into one of the deep local optima.

Group B: Rotated Multimodal Problems:

In Group A, some functions are separable and they can be solved by using D one-dimensional searches where D is the dimensionality of the problem. Hence, in Group B, we have the corresponding eight rotated multimodal problems. i.e. *Rotated Ackley's function* (f_9), *Rotated Griewank's function* (f_{10}), *Rotated Weierstrass function* (f_{11}), *Rotated Rastrigin's function* (f_{12}), *Rotated Non-continuous Rastrigin's function* (f_{13}), *Rotated Schwefel's function* (f_{14}), *Rotated Michalewicz's function* (f_{15}), *Rotated Sum of powers function* (f_{16}).

In rotated Schwefel's function, in order to keep the global optimum in the search domain after rotation, noting that the original global optimum of Schwefel's function is at $[420.96, 420.96, \dots, 420.96]$, $\mathbf{y}' = \mathbf{M} * (\mathbf{x} - 420.96)$ and $\mathbf{y} = \mathbf{y}' + 420.96$ are used instead of $\mathbf{y} = \mathbf{M} * \mathbf{x}$. Since Schwefel's function has better solutions out of the search domain $[-500, 500]^D$, when $|y_i| > 500$, $z_i = 0.001(|y_i| - 500)^2$, z_i is set in portion to the square distance between y_i and the bound.

Group C: Composition Problems

Composition functions are constructed using some basic benchmark problems to obtain more challenging problems with a randomly located global optimum and several randomly located deep local optima. The Gaussian function is used to combine the simple benchmark functions and blur the function's structures. The composition functions are asymmetrical multimodal problems, with different properties in different search regions. The details on how to construct this class of functions and six composition functions are presented in [258]. Two of the six composition functions defined in [258] are used here. Parameters settings for the following two composition functions:

17) Composition function 1 (CF1) in [258]:

The f_9 (CF1) are composed using ten sphere functions. The global optimum is easy to find once the global basin is found.

18) Composition function 2 (CF5) in [258]:

The f_{10} (CF5) is composed using ten different benchmark functions: two rotated Rastrigin's functions, two rotated Weierstrass functions, two rotated Griewank's functions, two rotated Ackley's functions and two sphere functions. The CF5 is more complex than CF1 since even after the global basin is found, the global optimum is not easy to locate.

The global optimum $\mathbf{x}^*$, the corresponding fitness value $f(\mathbf{x}^*)$, the search domains $[\mathbf{x}_{\min}, \mathbf{x}_{\max}]$ and the initialization range of each function are given in Table 3.1. Biased initializations are used for the functions whose global optimum is at the centre of the search domain.

Table 3. 1 The Properties and Initialization Ranges of Test Problems

f	$\mathbf{x}^*$	$f(\mathbf{x}^*)$	Search Domain	Initialization Range
f_1	[0,0,...,0]	0	$[-32.768, 32.768]^D$	$[-32.768, 16]^D$
f_2	[0,0,...,0]	0	$[-600, 600]^D$	$[600, 200]^D$
f_3	[0,0,...,0]	0	$[-0.5, 0.5]^D$	$[-0.5, 0.2]^D$
f_4	[0,0,...,0]	0	$[-5.12, 5.12]^D$	$[-5.12, 2]^D$
f_5	[0,0,...,0]	0	$[-5.12, 5.12]^D$	$[-5.12, -2]^D$
f_6	[420.96, 420.96,...420.96]	0	$[-500, 500]^D$	$[-500, 500]^D$
f_7	[0,0,...,0]	-9.66015	$[0, \pi]^D$	$[0, \pi]^D$
f_8	[0,0,...,0]	0	$[-1, 1]^D$	$[-1, 1]^D$
f_9	[0,0,...,0]	0	$[-32.768, 32.768]^D$	$[-32.768, 16]^D$
f_{10}	[0,0,...,0]	0	$[-600, 600]^D$	$[600, 200]^D$
f_{11}	[0,0,...,0]	0	$[-0.5, 0.5]^D$	$[-0.5, 0.2]^D$
f_{12}	[0,0,...,0]	0	$[-5.12, 5.12]^D$	$[-5.12, 2]^D$
f_{13}	[0,0,...,0]	0	$[-5.12, 5.12]^D$	$[-5.12, 2]^D$

f_{14}	[420.96, 420.96,...420.96]	0	$[-500, 500]^D$	$[-500, 500]^D$
f_{15}	[0,0,...,0]	-9.66015	$[0, \pi]^D$	$[0, \pi]^D$
f_{16}	[0,0,...,0]	0	$[-1, 1]^D$	$[-1, 1]^D$
f_{17}	Predefined rand number distributed in the search domain	0	$[-500, 500]^D$	$[-5, 5]^D$
f_{18}	Predefined rand number distributed in the search domain	0	$[-500, 500]^D$	$[-5, 5]^D$

3.3.2 Parameter Settings and Experimental Study

The computer configurations are Windows XP, Pentium(R) 4 3.00GHz CPU, 2 GB RAM running Matlab 7.1.

3.3.2.1 The basic parameter settings for PSO, CLPSO, DivEnh-PSO & DivEnh-CLPSO

The parameters of the PSO and CLPSO are set as in their original works [26], [48]: inertia weight $\omega=0.9$ and it linearly reduces to 0.4, $c_1=c_2=2$ and population size as 30. $V_{\max}$ is set to 50% of the search domain. For each problem, the DivEnh-CLPSO and the CLPSO without diversity enhancement are run 30 times. The maximum number of function evaluations Max_FEs is set at 100,000 for 10-D, and 200,000 for 30-D.

3.3.2.2 The particular parameter settings of CLPSO

For those particular parameters of CLPSO, are also referred to its original work [48] for original CLPSO & DivEnh-CLPSO.

➤ Learning Probability P_c

As in the original CLPSO, P_c values are set using the following equation so that each particle has a different P_c value. Therefore, particles have different levels of exploration and exploitation ability in the population and are able to solve diverse problems.

$$Pc_i = 0.05 + 0.45 \times (\exp(\frac{10(i-1)}{ps-1}) - 1) / (\exp(10) - 1) \quad (3.7)$$

➤ Search Bounds

In CLPSO, the particles are constrained within the search domain as follows: Fitness value of a particle is computed and its *pbest* and *gbest* are updated only if the particle is within the search domain. Since all *pbests* are within the range, all particles will eventually return to the search domain.

➤ Refreshing Gap m

As an important parameter in the CLPSO, the impact of this refreshing gap parameter m has been carefully investigated by experiments and statistically analysed. The best value for m was found to be 7 [48] and also used in our investigations for all test functions.

3.3.2.3 The particular parameter settings of DivEnh-PSO and DivEnh-CLPSO

Regarding to the DivEnh-CLPSO, the same parameter settings for the learning probability Pc and the refreshing gap m as in the CLPSO are used. For the tuning of parameters that were introduced in the DivEnh-PSO and DivEnh-CLPSO, i.e. Replacement frequency, Replacement Ratio, and Diversity Memory Size, a set of experiments on six selected test problems are performed. Those problems are: f_1 Ackley's function, f_2 Griewanks's function, f_4 Rastrigin's function, f_9 Rotated Ackley's function, f_{10} Rotated Griewanks's function, f_{12} Rotated Rastrigin's function. In this chapter, for the simplicity, for each test function, only the DivEnh-CLPSO with different parameter settings is run 30 times. The Mean value of the results over the 30 runs are presented in Table 3.2 with other parameters set as *the Replacement Ratio of PDE (RR) = 0.2 and the diversity memory size = 100 × number of dimensions*; Table 3.3 with other parameters set as *the Replace frequency RF of PDE = 100 × pops and the diversity memory size = 100 × number of dimensions*; Table 3.4 with *Replace frequency RF of PDE = 100 × pops and the Replacement Ratio of PDE (RR) = 0.2*.

➤ Replacement frequency (RF)

During the first 80% of function evaluations, the Population Diversity Enhancement (PDE) operation is periodically performed. The frequency of PDE should not be very high since we should allow the swarm to perform global exploration between every PDE operation. According to the parameter tuning investigations, the replacement frequency is not that sensitive since the final results do not vary much with respect to the settings of RF . The experimental results in Table 3.2 show that the replacement frequency can be set at $RF = 100 \times pops$, which means the PDE operation is performed every 100 iterations. The other two parameters are set as: *the Replacement Ratio of PDE (RR) = 0.2; the diversity memory size = 100 × number of dimensions*.

Table 3. 2 Mean objective function values with different replacement frequencies (RF).

RF	$50 \times pops$	$75 \times pops$	$100 \times pops$	$125 \times pops$	$150 \times pops$
f_1	1.7302e-13	3.3147e-13	5.7436e-14	2.9227e-13	2.0961e-13
f_2	4.9330e-07	4.9195e-07	3.6996e-07	3.9992e-07	4.4478e-07
f_4	0	0	0	0	0
f_9	1.8657e-12	2.3073e-13	5.3391e-14	1.6229e-13	2.5099e-13
f_{10}	3.0039e-02	2.7721e-02	2.4690e-02	2.6531e-02	2.9628e-02
f_{12}	3.2407e-00	3.6602e-00	3.7977e-00	4.1975e-00	4.8364e-00

➤ Replacement Ratio (RR)

When the PDE operation is periodically performed, the Replacement Ratio of PDE (RR) is also an important parameter affecting the quality of the final results. If numerous current *pbests* and their positions are replaced in every PDE operation, the convergence of the swarm will be adversely affected in the subsequent generations, while insufficient replacement will not be able to retain diversity of the swarm. Hence, there is a need to tune the replacement ratio (RR) parameter. The experimental results in Table 3.3 show that the $RR=0.2$ is an appropriate choice. The other two parameters are set as: *the Replace frequency RF of PDE = 100 × pops; the diversity memory size = 100 × number of dimensions*.

Table 3. 3 Mean objective function values with different replacement ratios (RR).

RR	0.1	0.15	0.2	0.3	0.4	0.8
f_1	1.6698e-13	1.2198e-13	5.7436e-14	9.3107e-10	4.8025e-10	2.6755e-08

f_2	2.4665e-04	2.4680e-04	3.6996e-07	3.1879e-04	3.1255e-03	4.9184e-02
f_4	0	0	0	2.9606e-16	5.1900e-11	2.1958e-06
f_9	3.4563e-13	1.4566e-13	5.3391e-14	8.3253e-07	2.5554e-10	8.9900e-08
f_{10}	2.7939e-02	2.6880e-02	2.4690e-02	3.3476e-02	3.9117e-02	5.1564e-01
f_{12}	3.7093e-00	3.9538e-00	3.7977e-00	4.3296e-00	4.2537e-00	1.1610e+01

➤ Diversity Memory Size

In order to store adequate past solutions with good diversity and good fitness, the size of the DM should be fixed at an appropriate value. The size of the memory will directly impact on the computation time. To determine an appropriate size for the archived members in the DM, the following investigation is conducted. The other two parameters are set as: *the Replace frequency RF of PDE* = $100 \times pops$; *the Replacement Ratio of PDE (RR)* = 0.2. According to our experimental results presented in Table 3.4, once enough solutions are stored in the DM, the performance is almost stable. However, the computation time for a single run increased to more than 15 minutes for the DivEnh-CLPSO with memory size larger than $200 \times number\ of\ dimensions$. The other three cases in Table 3.4 require less than 5 minutes. Hence, we choose *the diversity memory size* = $100 \times number\ of\ dimensions$ as the compromise setting.

Table 3. 4 Mean objective function values with DM sizes.

<i>Memory Size</i>	$50 \times number\ of\ dimensions$	$100 \times number\ of\ dimensions$	$150 \times number\ of\ dimensions$	$200 \times number\ of\ dimensions$
f_1	9.6415e-13	5.7436e-14	1.0028e-13	6.1224e-14
f_2	5.9202e-05	3.6996e-07	3.6425e-07	1.9989e-07
f_4	0	0	0	0
f_9	2.1913e-11	5.3391e-14	5.2702e-14	5.2313e-14
f_{10}	2.4621e-02	2.4690e-02	2.5945e-02	2.4450e-02
f_{12}	4.0001e-00	3.7977e-00	3.7589e-00	2.8622e-00

3.3.3 Comparison among different PSO Variants

3.3.3.1 Experimental Results and Discussions among basic PSO and DivEnh-PSO

Table 3.5 presents the best values, median values, worst values, means and STDs of the 30 runs of the two algorithms, DivEnh-PSO and the basic PSO without diversity enhancement on the ten test problems out of the above Table 3.1 with $D=10$ and $D=30$. The best mean values achieved by the two algorithms for each test function are shown in bold. In order to determine whether the results obtained by DivEnh-PSO are statistically different from the results generated by the PSO without diversity enhancement, the h values presented in the last row of Table 3.5 are the results of t -tests as well as Wilcoxon tests. (The details of t -test as well as Wilcoxon tests are listed in Appendix A). An h value of 1 indicates that the performance of the DivEnh-PSO is statistically superior with 95% certainty, whereas h value of 0 implies that the performances are not statistically different.

From Table 3.5, we can easily find that the DivEnh-PSO always achieved better results on all three multimodal groups. According to the results of t -tests as well as Wilcoxon tests, DivEnh-PSO significantly improves the results on almost all the test problems especially on problems f_5 , f_9 and f_{10} . On the two composition functions with randomly distributed local and global optima, DivEnh-PSO performs much better.

As the DivEnh-PSO achieves the better results on most multimodal problems in Groups B and C, especially on Group C - compositional multimodal problems, this implies that the DivEnh-PSO is more effective in solving hard multimodal problems than the PSO without diversity enhancement which loses diversity rapidly when solving hard multimodal problems. However, the DivEnh-PSO uses the diversity memory to keep past diverse solutions and periodically inject these past diverse solutions into population and as their *pbests*. Therefore, the DivEnh-PSO does not easily get trapped by an inferior local optimal region. Especially on more complex asymmetrical landscapes in Group C, DivEnh-PSO obviously surpasses the PSO without diversity enhancement. Hence, the DivEnh-PSO explores a larger search space. Because of this, the DivEnh-PSO performs comparably to or better than the basic PSO on all of the ten multimodal problems.

Table 3. 5 Results of 10-D and 30-D problems.

Function	10D f_4		10D f_5		10D f_6		10D f_9	
Method	PSO	DivEnh-PSO	PSO	DivEnh-PSO	PSO	DivEnh-PSO	PSO	DivEnh-PSO
<i>Best</i>	0	0	1.1843e+02	9.0949e-13	0	0	3.5527e-15	3.5527e-15
<i>Median</i>	1.0000e+00	0	2.3687e+02	2.3687e+02	0	0	7.1054e-15	7.1054e-15
<i>Worst</i>	5.0000e+00	4.0000e+00	5.9219e+02	5.9219e+02	3.5527e-15	0	1.1551e+00	1.1054e-14
<i>Mean</i>	9.6674e-01	4.6670e-01	2.6451e+02	2.1583e+02	1.1842e-16	0	3.8544e-02	8.7120e-15
<i>Std</i>	1.2172e+00	1.0016e+00	1.4483e+02	1.2261e+02	6.4863e-16	0	2.1094e-01	5.7050e-15
<i>h</i>	1/1		1/1		1/1		1/1	
Function	10D f_{10}		10D f_{12}		10D f_{13}		10D f_{14}	
Method	PSO	DivEnh-PSO	PSO	DivEnh-PSO	PSO	DivEnh-PSO	PSO	DivEnh-PSO
<i>Best</i>	2.7066e-02	2.4268e-02	2.9849e+00	1.9899e+00	0	0	0	0
<i>Median</i>	1.1073e-01	1.0309e-01	8.3526e+00	9.9496e+00	3.7577e+02	2.3841e+02	1.0300e+00	5.5440e-01
<i>Worst</i>	2.4095e-01	2.4917e-01	2.6864e+01	1.5942e+01	1.2041e+03	8.4892e+02	2.0139e+00	2.6613e+00
<i>Mean</i>	1.2195e-01	1.1659e-01	1.0834e+01	9.8857e+00	4.3811e+02	2.9261e+02	8.3778e-01	5.4480e-01
<i>Std</i>	5.6097e-02	8.7377e-02	5.9050e+00	3.7880e+00	3.6611e+02	2.3350e+02	7.4144e-01	4.0100e-01
<i>h</i>	0/0		1/1		1/1		1/1	
Function	10D f_{17}		10D f_{18}		30D f_4		30D f_5	
Method	PSO	DivEnh-PSO	PSO	DivEnh-PSO	PSO	DivEnh-PSO	PSO	DivEnh-PSO
<i>Best</i>	0	0	2.3433	1.3460	4.0000	4.0000	7.1063e+02	6.9089e+02
<i>Median</i>	2.0000e+02	1.0000e+02	1.0000e+02	1.1042e+01	2.7000e+01	2.1000e+01	1.1844e+03	1.0659e+03
<i>Worst</i>	4.0000e+02	4.0000e+02	5.0910e+02	5.0336e+02	5.8000e+01	5.3000e+01	1.8950e+03	2.0135e+03
<i>Mean</i>	1.5000e+02	9.0666e+01	1.4114e+02	7.7588e+01	2.9734e+01	2.4684e+01	1.2153e+03	1.1837e+03
<i>Std</i>	1.0748e+02	8.1427e+01	1.9245e+02	1.2488e+02	1.7973e+01	1.3141e+01	3.0421e+02	3.0741e+02
<i>h</i>	1/1		1/1		1/1		0/0	
Function	30D f_6		30D f_9		30D f_{10}		30D f_{12}	
Method	PSO	DivEnh-PSO	PSO	DivEnh-PSO	PSO	DivEnh-PSO	PSO	DivEnh-PSO
<i>Best</i>	0	0	3.1974e-14	2.1316e-14	0	0	5.9698e+01	5.1737e+01
<i>Median</i>	1.4121e-04	1.1297e-04	1.6462e+00	1.6462e+00	7.3960e-03	2.3451e-03	8.3576e+01	8.0591e+01
<i>Worst</i>	1.5010e+00	1.0031e+00	2.8141e+00	2.6605e+00	4.4278e-02	4.3912e-02	1.3034e+02	1.2337e+02
<i>Mean</i>	1.2274e-01	9.8270e-02	1.7138e+00	1.6962e+00	1.7748e-02	1.0870e-02	8.3742e+01	7.3675e+01
<i>Std</i>	3.8102e-01	3.6720e-01	9.3454e-01	7.7740e-01	1.5334e-02	1.0205e-02	1.7578e+01	1.1727e+01
<i>h</i>	1/1		0/0		1/1		1/1	
Function	30D f_{13}		30D f_{14}		30D f_{17}		30D f_{18}	
Method	PSO	DivEnh-PSO	PSO	DivEnh-PSO	PSO	DivEnh-PSO	PSO	DivEnh-PSO
<i>Best</i>	9.8700e+02	5.7245e+02	3.6566e+00	3.4277e+00	0	0	1.0717e+01	1.0282e+01
<i>Median</i>	1.8935e+03	2.4143e+03	8.4604e+00	7.4482e+00	2.2351e-30	1.3148e-31	2.2941e+01	1.1428e+01
<i>Worst</i>	3.8290e+03	3.9314e+03	1.1982e+01	1.1802e+01	2.0000e+02	2.0000e+02	8.0000e+02	5.0000e+02
<i>Mean</i>	2.6758e+03	2.4504e+03	8.2279e+00	7.7530e+00	3.0000e+01	9.0000e+00	1.7708e+02	9.9640e+01
<i>Std</i>	7.3606e+02	7.0156e+02	2.3936e+00	2.0252e+00	6.5126e+01	4.1217e+01	2.1805e+02	1.0944e+02
<i>h</i>	1/1		1/1		1/1		1/1	

Figure 3.2 shows the best fitness values of the median run obtained for test problems f_9 and f_{18} on 10-D and 30-D instances. We can observe that the PSO without diversity enhancement can achieve faster convergence speed during the early stages, while DivEnh-PSO converges slowly due to the PDE procedure. However, the DivEnh-PSO can perform better during the late stages.

Figure 3. 1 The Median Convergence Characteristics of 10-D and 30-D problems f_9 and f_{18}

3.3.3.2 Experimental Results and Discussions among CLPSO and DivEnh-CLPSO

Table 3.6 presents the means and standard deviations of the 30 runs of the two algorithms, DivEnh-CLPSO and the same CLPSO without diversity enhancement on the above 18 test problems with $D=10$ and $D=30$. The better mean values achieved by the two algorithms for each test function are shown in bold. In order to determine whether the results obtained by DivEnh-CLPSO are statistically different from the results generated by the CLPSO without the diversity enhancement, the h values obtained by the t -tests as well as Wilcoxon tests are presented in Table 3.6. An h value of 1 indicates that the performance of the DivEnh-CLPSO is statistically superior with 95% certainty, whereas an h value of 0 implies that the

performances are not statistically different. An h value of -1 indicates that the performance of the DivEnh-CLPSO is statistically worse than the original CLPSO with 95% certainty.

Table 3. 6 Results of 10-D and 30-D test problems.

(A1—CLPSO, A2—DivEnh-CLPSO.)

Function	f_1 -10D		f_2 -10D		f_3 -10D		f_4 -10D	
Method	A1	A2	A1	A2	A1	A2	A1	A2
<i>Mean</i>	6.1507e-14	5.7436e-14	1.3069e-05	3.6996e-07	0	0	0	0
<i>Std</i>	5.6417e-15	3.8641e-15	9.2844e-07	4.5241e-08	0	0	0	0
h	1/1		1/1		0/0		0/0	
Function	f_5 -10D		f_6 -10D		f_7 -10D		f_8 -10D	
Method	A1	A2	A1	A2	A1	A2	A1	A2
<i>Mean</i>	0	0	0	0	-8.2680e-00	-8.7653e-00	2.5644e-12	6.8995e-13
<i>Std</i>	0	0	0	0	6.4555e-04	5.0021e-05	3.5654e-13	6.2320e-14
h	0/0		0/0		1/1		1/1	
Function	f_9 -10D		f_{10} -10D		f_{11} -10D		f_{12} -10D	
Method	A1	A2	A1	A2	A1	A2	A1	A2
<i>Mean</i>	5.9895e-14	5.3391e-14	2.6128e-02	2.4690e-02	1.7672e-01	3.1427e-02	3.9290e-00	3.7977e-00
<i>Std</i>	6.2004e-15	7.4451e-15	3.5664e-03	3.1448e-03	1.0581e-01	2.2012e-02	8.6214e-01	5.6578e-01
h	1/1		1/1		1/1		1/1	
Function	f_{13} -10D		f_{14} -10D		f_{15} -10D		f_{16} -10D	
Method	A1	A2	A1	A2	A1	A2	A1	A2
<i>Mean</i>	3.7602e-00	3.2013e-00	3.7410e+02	2.7528e+02	-7.3716e-00	-7.4783e-00	1.9600e-08	7.5677e-09
<i>Std</i>	4.4518e-01	1.8214e-01	2.2199e+01	9.5004e+00	4.2508e-03	5.9514e-04	2.1548e-09	1.1475e-09
h	1/1		1/1		1/1		1/1	
Function	f_{17} -10D		f_{18} -10D		f_1 -30D		f_2 -30D	
Method	A1	A2	A1	A2	A1	A2	A1	A2
<i>Mean</i>	1.5805e-00	4.1951e-02	4.6972e-00	3.3333e-00	5.5934e-14	5.6147e-14	7.1005e-10	6.1424e-10
<i>Std</i>	2.3515e-01	8.2514e-03	1.5959e-00	9.5151e-02	2.3514e-15	1.5148e-15	2.5156e-11	3.8144e-11
h	1/1		1/1		0/0		1/1	
Function	f_3 -30D		f_4 -30D		f_5 -30D		f_6 -30D	
Method	A1	A2	A1	A2	A1	A2	A1	A2
<i>Mean</i>	3.4689e-07	3.1550e-07	4.8590e-10	3.0399e-10	8.6002e-10	8.4328e-10	4.5859e+01	2.3688e+01
<i>Std</i>	9.5145e-09	1.0251e-08	3.4648e-11	2.4841e-11	8.2641e-11	7.2315e-11	1.2516e+00	2.1515e+00
h	1/1		1/1		0/0		1/1	
Function	f_7 -30D		f_8 -30D		f_9 -30D		f_{10} -30D	
Method	A1	A2	A1	A2	A1	A2	A1	A2
<i>Mean</i>	-1.2345e+01	-1.1827e+01	8.3775e-13	8.2904e-13	2.1358e-04	3.6418e-07	1.2296e-06	1.0146e-06
<i>Std</i>	6.1251e-03	5.1564e-04	9.1844e-15	8.4998e-15	5.2915e-05	5.1545e-08	2.1644e-07	9.1645e-08
h	1/1		0/0		1/1		1/1	
Function	f_{11} -30D		f_{12} -30D		f_{13} -30D		f_{14} -30D	
Method	A1	A2	A1	A2	A1	A2	A1	A2
<i>Mean</i>	3.5040e-00	1.2554e-00	3.7608e+01	3.1452e+01	3.3568e+01	3.3255e+01	2.2702e+03	2.1078e+03
<i>Std</i>	8.1651e-01	9.5147e-01	7.2515e-01	1.2231e+00	1.1548e+00	1.8445e+00	5.1546e+02	2.1519e+02
h	1/1		1/1		0/0		1/1	

Function	f_{15} -30D		f_{16} -30D		f_{17} -30D		f_{18} -30D	
Method	A1	A2	A1	A2	A1	A2	A1	A2
<i>Mean</i>	-1.9072e+01	-1.9052e+01	2.3344e-08	4.7907e-09	7.5020e-01	1.0454e-01	1.8824e+01	1.6367e+01
<i>Std</i>	5.1546e-02	1.1546e-02	3.1644e-09	5.1646e-10	1.1564e-01	6.3214e-02	5.1646e+00	1.6546e+00
<i>h</i>	0/0		1/1		1/1		1/1	

From Table 3.6, we can easily find that the DivEnh-CLPSO achieved better results on all three multimodal groups. According to the results of t -tests as well as Wilcoxon tests, DivEnh-CLPSO significantly improves the results on most test problems. Out of the 18 test problems, DivEnh-CLPSO performs much better on the most difficult problems such as the two composition functions with randomly distributed local and global optima.

As the DivEnh-CLPSO achieves better results on most multimodal problems in Groups B and C, this clearly demonstrates that the DivEnh-CLPSO is more effective in solving hard multimodal problems than the original CLPSO without the diversity enhancement. CLPSO can lose diversity rapidly once the most positions of the swarm converge into a locally optimal region, since CLPSO's comprehensive learning strategy learns from all particles' historical best information to update a particle's velocity. However, the DivEnh-CLPSO uses the diversity memory to store past diverse solutions with good fitness values and to periodically inject these past diverse solutions to replace some worst particles in the swarm as their new positions and *pbests*. Therefore, the DivEnh-CLPSO does not easily get trapped by an inferior local optimal region. Especially on complex asymmetrical landscapes in Group C, DivEnh-CLPSO obviously surpasses the CLPSO without diversity enhancement. Hence, the DivEnh-CLPSO explores a larger search space. Because of this, the DivEnh-CLPSO performs comparably to or better than the original CLPSO on all of the multimodal problems.

By analyzing the results of the PSO, DivEnh-PSO, CLPSO and DivEnh-CLPSO on 10- D and 30- D problems, one may conclude that the DivEnh-PSO and DivEnh-CLPSO performs the same correspondingly as or slightly better than the original PSO and CLPSO on some relatively easier multimodal problems in Group A. According to the "No Free Lunch" theorem [259], "any elevated performance over one class of problems is offset by performance over another class". There is a cost for better performance on complex multimodal problems and the cost is the slow convergence on simple multimodal problems.

This issue will be more obvious on unimodal problems. Therefore, we may not expect the best performance on all classes of problems, as the proposed DivEnh-PSO and DivEnh-CLPSO focuses on improving the PSO and CLPSO's performance on multimodal problems.

3.4 Conclusions

This Chapter presents two diversity enhanced particle swarm optimizer (DivEnh-PSO and DivEnh-CLPSO) employing an external memory to store the previous solutions with good diversity in the parameter space. By periodically replacing the worst current positions and their *pbests* by diversity memory members, the DivEnh-PSO and DivEnh-CLPSO attempt to avoid all particles getting trapped in an inferior locally optimal region and increase the diversity of the whole swarm. The DivEnh-PSO and DivEnh-CLPSO enables the particles to have more diverse exemplars to learn from and a larger potential space to fly. From the analysis of experimental results, we observe that this new diversity memory enables the DivEnh-PSO and DivEnh-CLPSO to use the information captured in past solutions more effectively to generate more better quality solutions frequently when compared to the same PSO and CLPSO without diversity enhancement. Based on the results of these algorithms on the test problems belonging to three classes, we can conclude that DivEnh-PSO and DivEnh-CLPSO significantly improves the performance upon PSO and CLPSO without diversity enhancement on most multimodal problems irrespective of if they are rotated or not. When solving real world problems, we frequently do not know the shape of the fitness landscape. Hence, it is advisable to use an algorithm that performs well on all kinds of problems no matter whether they are multimodal or unimodal problems, even though the diversity enhancement version is not the best choice for solving unimodal problems. Another attractive property of the configuration of diversity enhancement is that it does not introduce further complex operations to the original PSO and CLPSO. The only differences from the original PSO and CLPSO are the diversity memory and population replacement. The diversity enhancement procedure is also simple and easy to implement with all nature-inspired optimization algorithms. In addition, the population diversity enhancement (PDE) procedure does not require additional function evaluations.

Chapter 4

Improved dynamic multi-swarm particle swarm optimizer with the Harmony search (DMS-PSO-HS)

4.1 Introduction

As mentioned in the Introduction, in the PSO domain, there are two main variants: global PSO and local PSO. In the local version of the PSO, each particle's velocity is adjusted according to its personal best position $pbest$ and the best position $lbest$ achieved so far within its neighborhood. The global PSO learns from the personal best position $pbest$ and the best position $gbest$ achieved so far by the whole population. The velocity update of the local PSO is:

$$V_i^d = w \times V_i^d + c_1 \times rand1_i^d \times (pbest_i^d - X_i^d) + c_2 \times rand2_i^d \times (lbest_i^d - x_i^d) \quad (4.1)$$

where $lbest = (lbest_i^1, lbest_i^2, \dots, lbest_i^D)$ is the best position achieved within i^{th} particle's neighborhood.

Focusing on improving the local variants of the PSO, different neighborhood structures were proposed and discussed [41], [260], [261], [262], [263], [264], [265], [266]. Except these local PSO variants, some variants that use multi-swarm [267], subpopulation [268] can also be regarded as the local PSO variants if we view the sub-groups as special neighborhood structures. In the existing local versions of PSO with different neighborhood structures and the multi-swarm PSOs, the swarms are predefined or dynamically adjusted according to the distance. Hence, the freedom of sub-swarms is limited. In [269], a dynamic multi-swarm particle swarm optimizer (DMS-PSO) was proposed whose neighborhood topology is dynamic and randomized. DMS-PSO gives a better performance on multimodal problems than some other PSO variants, but the local search performance is not satisfactory.

Harmony search (HS) algorithm [270] conceptualized a behavioral phenomenon of music players in improvisation process, where each player continues to polish its tune in order to produce better harmony in a natural musical performance processes. Originated in an analogy equating music improvisation to engineering optimization, the engineers seek for a global solution as determined by an objective function, just like the musicians seek to find musically pleasing harmony as determined by an aesthetic. In music improvisation, each player sounds any pitch within the possible range, together making one harmony vector. If all the pitches make a good solution, the experience is stored in each variable's memory, and the possibility to make a good solution is also increased next time.

Recently, some researchers have improved the HS algorithm by introducing the particle-swarm concept to continuous-value optimization problems such as benchmark mathematical functions [271] with good results. The particle-swarm harmony search (PSHS), is presented in [272]. The particle-swarm concept is introduced to the original discrete-value HS algorithm for the first time, and to apply the modified HS algorithm to the water-network-design problem.

In this chapter, we improved this DMS-PSO by combining original harmony search (HS) algorithm, and test this DMS-PSO-HS on difficult multimodal and composition test functions. In order to compare fairly, the original HS and the original DMS-PSO are included in the comparison.

4.2 Improved Dynamic Multi-swarm Particle Swarm Optimizer Hybridized with the Harmony Search (DMS-PSO-HS).

In this section, we will sequent introduce the original dynamic multi-swarm particle swarm optimizer (DMS-PSO) and the original Harmony Search (HS), finally, we will explain how we combine the two approaches to form the proposed Dynamic Multi-Swarm Particle Swarm Optimizer with Harmony based sub-swarm search (DMS-PSO-HS) [274].

4.2.1 The original DMS-PSO algorithm

The dynamic multi-swarm particle swarm optimizer was constructed based on the local version of PSO with a new neighborhood topology [269]. Many existing nature-inspired optimization algorithms require larger populations, while PSO needs a comparatively smaller population size. A population with three to five particles can achieve satisfactory results for simple problems. According to many reported results on the local version of PSO [41], [260], PSO with small neighborhoods performs better on complex problems. Hence, in order to slow down convergence speed and to increase diversity to achieve better results on multimodal problems, in the DMS-PSO, small neighborhoods are used. The population is divided into small sized swarms. Each sub-swarm uses its own members to search for better regions in the search space.

Since the small sized swarms are searching using their own best historical information, they are easy to converge to a local optimum because of PSO's speedy convergence behavior. Further, unlike a co-evolutionary PSO, we allow maximum information exchange among the particles to enhance the diversity of the particles. Hence, a randomized regrouping schedule is introduced to make the particles have a dynamically changing neighborhood structures. Every R generations, the population is regrouped randomly and starts searching using a new configuration of small swarms. Here R is called regrouping period. In this way, the information obtained by each swarm is exchanged among the swarms. Simultaneously the diversity of the population is increased. The new neighborhood structure has more freedom when compared with the classical neighborhood structure. In this algorithm, in order to

constrain the particles within the range, the fitness value of a particle is calculated and updated the corresponding *pbest* only if the particle is in the range. Since all exemplars are within the range, the particle will eventually return to the search domain.

For example, suppose we have three swarms with three particles in each swarm. First, the nine particles are divided into three swarms randomly. Then the three swarms use their own particles to search for better solutions. In this period, they may converge to near a local optimum. Then the whole population is regrouped into new swarms. The new swarms begin their search. This process is continued until a stop criterion is satisfied. With the randomly regrouping schedule, particles from different swarms are grouped in a new configuration so that each small swarms search space is enlarged and better solutions are possible to be found by the new small swarms. This regrouping procedure is shown in Figure 4.1, and the flowchart of the original DMS-PSO is given in Figure 4.2.

Figure 4. 1 DMS-PSO's regrouping phase

Figure 4. 2 The flowchart of original DMS-PSO

4.2.2 The original HS algorithm

The original HS algorithm [270], [271], [272] is based on natural musical performance processes that occur when a musician searches for a better state of harmony, such as during jazz improvisation. The optimization operators of HS algorithm are specified as: the harmony memory (HM), which stores the feasible vectors which are all in the feasible space, as show in the following equation (4.2). The harmony memory size (HMS , number of solution vectors in harmony memory) determines how many vectors the HM stores. The harmony memory considering rate ($HMCR$), and the pitch adjusting rate (PAR). A new vector is generated by selecting the components of different vectors randomly in the harmony memory.

$$HM = \left[\begin{array}{cccc|c} x_1^1 & x_2^1 & \dots & x_D^1 & fitness(x^1) \\ x_1^2 & x_2^2 & \dots & x_D^2 & fitness(x^2) \\ \vdots & \vdots & \dots & \vdots & \vdots \\ x_1^{HMS} & x_2^{HMS} & \dots & x_D^{HMS} & fitness(x^{HMS}) \end{array} \right] \quad (4.2)$$

When a musician improvises one pitch, usually he (or she) follows any one of three rules: (1) playing any one pitch from his (or her) memory, (2) playing an adjacent pitch of one pitch from his (or her) memory, and (3) playing totally random pitch from the possible sound range. Similarly, when each decision variable chooses one value in the HS algorithm, it apply the above three rules as follows in the whole HS procedure. And if a new harmony is better than existing worst harmony in the HM, the new harmony is included in the HM and the worst harmony is excluded from the HM. This procedure is repeated until fantastic harmony is found.

Step1. Initialize the problem and algorithm parameters.

Step2. Initialize the harmony memory (HM).

Step3. Improve a new harmony from the HM.

If a random number < HMCR

Choosing any one value from HS memory (defined as memory considerations);

If a random number < (PAR)

Choosing an adjacent value of one value from the HS memory by using the arbitrary distance bandwidth bw (defined as pitch adjustments);

Endif

Else Choosing totally random value from the possible value range (defined as randomization).

Endif.

Step4. Update the HM.

Step5. Repeat Steps 3 and 4 until the termination criterion is satisfied.

4.2.3 Improved Dynamic Multi-swarm Particle Swarm Optimizer Hybridized with the Harmony Search (DMS-PSO-HS).

In order to achieve better results on multi-modal problems, DMS-PSO [269] is designed to make the particles have a large diversity with sacrificing of convergence speed. Even after the global region is found, the particles will not converge very fast in order to avoid premature convergence. How to maintain the diversity and get the good result at the same time is a problem. Thus, in DMS-PSO, a HS phase is added into each sub-swarm as a further search phase after the new PSO solutions generated during the every iteration. Combining with the HS schedule, the DMS-PSO is modified to DMS-PSO-HS. A new harmony will be obtained from the temporary HM formed by the current *pbests* in each sub-swarm. We calculate the Euclidean distance to all the *pbests* in the corresponding sub-swarm for the new harmony and replace the nearest one with the new harmony if the new harmony has better fitness value. Except the HS phase in each sub-swarm, everything is same as the original DMS-PSO. In this way, the strong exploration abilities of the original PSO and the original HS can be fully exerted in each swarm searching, while the dynamic exchanging of information among the swarms can also enhance the capability of exploitation and exploration in each sub-searching region. Exchanging of information among the dissimilar swarms promotes collaboration among sub-searching regions, which advances the information exchange among the diverse regions, and thus it speeds up the convergence of the whole population; while exchanging of information among the similar swarms

encourages combination of similar solutions and is good for local search in a local neighborhood, which maintains the diversity for the whole population. However, in some cases, exchanging of information among the dissimilar swarms can also benefit on diversity recovery; while exchanging of information among the similar swarms is also able to facilitate the convergence. For instance, during the evolution, some sub-swarms may get trapped in locally optimal regions. In order to force those sub-swarms escape from premature convergence, exchanging of information among dissimilar swarms is required for the exploration. On the other hand, if the global optima area is already found, exchanging of information among the similar swarms will be favorable for local exploitation. The Flowchart of the proposed DMS-PSO-HS is given in Figure 4.3.

Figure 4. 3 The flowchart of DMS-PSO-HS

4.3 Experiments Results and Discussion

4.3.1 Test Functions Considered in the Experimental Study

In order to make comprehensive comparison on DMS-PSO-HS with two original algorithms, DMS-PSO and HS, 16 diverse benchmark problems are tested here [256], most of them are strongly multimodal and composition test functions. All functions are tested on 10 and 30 dimensions. According to their properties, these problems are divided into four groups: 2 *unimodal problems*, 6 *unrotated multimodal problems*, 6 *rotated multimodal problems* and 2 *composition problems*. The properties and the expressions of these problems are presented as following and detailed formulas are list in Appendix B.

Group A: Unimodal and Simple Multi-modal Problems:

Sphere function (f_1) and *Rosenbrock's function* (f_2) are included in this group. The first problem is the sphere function and is easy to solve. The second problem is the Rosenbrock function. It can be treated as a multimodal problem. It has a narrow valley from the perceived local optima to the global optimum. In the experiments below, we find that the algorithms which perform well on sphere function also perform well on Rosenbrock function.

Group B: Unrotated Multimodal Problems:

In this group, there are six multimodal test functions, which are already introduced in previous section 3.3.1. They are *Ackley's function* (f_3), *Griewanks's function* (f_4), *Weierstrass function* (f_5), *Rastrigin's function* (f_6), *Non-continuous Rastrigin's function* (f_7), *Schwefel's function* (f_8).

Group C: Rotated Multimodal Problems:

In Group B, some functions are separable and they can be solved by using D one-dimensional searches where D is the dimensionality of the problem. Hence, in Group C, we have the corresponding six rotated multimodal problems, which are also introduced in previous section 3.3.1. i.e. *Rotated Ackley's function* (f_9), *Rotated Griewanks's function* (f_{10}), *Rotated Weierstrass function* (f_{11}), *Rotated Rastrigin's function* (f_{12}), *Rotated Non-continuous Rastrigin's function* (f_{13}), *Rotated Schwefel's function* (f_{14}).

Group D: Composition Problems

Two composition functions included in this section are also same as the previous section 3.3.1, which are *composition function 1 (CF1) (f_{15})* and *composition function 2 (CF5) (f_{16})*.

The global optimum $\mathbf{x}^*$, the corresponding fitness value $f(\mathbf{x}^*)$, the search domains $[\mathbf{X}_{\min}, \mathbf{X}_{\max}]$ and the initialization range of each function are given in Table 4.1. Biased initializations are used for the functions whose global optimum is at the centre of the search domain.

Table 4. 1 The Initialization of the Test Problems (Global optimum, search domains and initialization ranges).

f	$\mathbf{x}^*$	$f(\mathbf{x}^*)$	Search domain	Initialization Range
f_1	$[0,0,\dots,0]$	0	$[-100, 100]^D$	$[-100, 50]^D$
f_2	$[1,1,\dots,1]$	0	$[-2.048, 2.048]^D$	$[-2.048, 2.048]^D$
f_3	$[0,0,\dots,0]$	0	$[-32.768, 32.768]^D$	$[-32.768, 16]^D$
f_4	$[0,0,\dots,0]$	0	$[-600, 600]^D$	$[600, 200]^D$
f_5	$[0,0,\dots,0]$	0	$[-0.5, 0.5]^D$	$[-0.5, 0.2]^D$
f_6	$[0,0,\dots,0]$	0	$[-5.12, 5.12]^D$	$[-5.12, 2]^D$
f_7	$[0,0,\dots,0]$	0	$[-5.12, 5.12]^D$	$[-5.12, 2]^D$
f_8	$[420.96, 420.96,\dots,420.96]$	0	$[-500, 500]^D$	$[-500, 500]^D$
f_9	$[0,0,\dots,0]$	0	$[-32.768, 32.768]^D$	$[-32.768, 16]^D$
f_{10}	$[0,0,\dots,0]$	0	$[-600, 600]^D$	$[600, 200]^D$
f_{11}	$[0,0,\dots,0]$	0	$[-0.5, 0.5]^D$	$[-0.5, 0.2]^D$
f_{12}	$[0,0,\dots,0]$	0	$[-5.12, 5.12]^D$	$[-5.12, 2]^D$
f_{13}	$[0,0,\dots,0]$	0	$[-5.12, 5.12]^D$	$[-5.12, 2]^D$
f_{14}	$[420.96, 420.96,\dots,420.96]$	0	$[-500, 500]^D$	$[-500, 500]^D$
f_{15}	Predefined rand number distributed in the search domain	0	$[-500, 500]^D$	$[-5, 5]^D$
f_{16}	Predefined rand number distributed in the search domain	0	$[-500, 500]^D$	$[-5, 5]^D$

4.3.2 Parameter Settings

As the general PSO variants [2][273][269], $\omega = 0.729$, $c_1 = c_2 = 1.49445$, $R=10$. $V_{\max}$ restricts particles' velocities, where $V_{\max}$ is equal to 20% of the search domain. To solve these problems, the number of sub-swarms is set at 10 which is also the same setting as in the original DMS-PSO [269]. The other parameters need to be tuned, six selected test functions are used to investigate the impact of them. They are 10 dimensions test functions: f_3 Ackley's function, f_4 Griewank's function, f_5 Weierstrass function, f_9 Rotated Ackley's function, f_{10} Rotated Griewank's function, f_{11} Rotated Weierstrass function. Experiments were conducted on the six test functions with 10 Dimensions, the mean value of 30 runs are given below.

Sub-population SP

For the sub-population of each sub-swarm, the results of investigation on a few selected test problems are shown in the following Table 4.2. In this table, the mean values of six problems with different parameter settings are given. Based on the comparison of those results, the optimal setting is each sub-swarm has 5 particles, which is also the general setting for the HS population size. Hence, in DMS-PSO-HS the population size is 50.

Table 4. 2 Parameter Setting on the Sub-population.

<i>SP</i>	3	4	5	6	7
f_3	2.7844e-15	2.9425e-15	2.5847e-15	2.8451e-15	5.2118e-15
f_4	2.8958e-03	2.8417e-03	1.2389e-03	3.8145e-03	1.6541e-02
f_5	0	0	0	0	0
f_9	2.9606e-15	3.2518e-15	3.3159e-15	5.8110e-15	6.8887e-15
f_{10}	4.1842e-02	3.9254e-02	1.9966e-02	7.1814e-02	9.4799e-02
f_{11}	5.9698e-00	0	0	3.0029e-00	4.5151e-00

Regrouping Period R

For the regrouping period R , it should not be very small, since we need give enough iterations for the each new sub-swarm search, while it also should not too large to waste the

evaluations when the sun-swarm could not go further search. The following Table 4.3 gives the results on the tuning of R . Base on it, it is easy to make a decision of $R=5$.

Table 4. 3 Parameter Setting on the regrouping period.

R	3	4	5	6	7
f_3	3.1974e-15	3.1974e-15	2.5847e-15	3.3159e-15	3.3159e-15
f_4	1.2145e-02	1.1412e-02	1.2389e-03	1.3376e-02	1.3702e-02
f_5	0	0	0	0	0
f_9	3.3159e-15	3.1974e-15	3.3159e-15	3.1974e-15	3.1974e-15
f_{10}	2.2805e-02	2.1011e-02	1.9966e-02	1.8709e-02	2.1990e-02
f_{11}	4.0333e-00	4.0957e-00	0	4.0000e+00	3.6000e-00

For the part of HS in DMS-PSO-HS and the original HS, the effect of the two main parameters of HS (i.e. $HMCR$ and HMS) was investigated by many researcher [272][273]. Base on those observations, a large value for $HMCR$ (e.g. larger than 0.95) generally uses except for the case of problems with very low dimensionality where a small value of $HMCR$ is recommended. A small value for HMS seems to be a good choice for general cases. A dynamically adjusting PAR is also recommended. Hence, in this chapter, the parameters of the HS are setting as: $HMCR=0.98$, $bw_{min}= 0.0001$, $bw_{max}= 0.05 \times (UB-LB)$, $PAR_{min}=0.01$, $PAR_{max}=0.99$, the PAR and the bandwidth bw are updating as the following definitions, where the $PAR(t)$ is the pitch adjusting rate for generation t , PAR_{min} is the minimum adjusting rate, PAR_{max} is the maximum adjusting rate and t is the generation number; $bw(t)$ is the bandwidth for generation t , bw_{min} is the minimum bandwidth and bw_{max} is the maximum bandwidth, Max_gen is total number of generations.

$$PAR(t) = PAR_{min} + \frac{(PAR_{max} - PAR_{min})}{Max_gen} \times t \quad (4.3)$$

$$bw(t) = bw_{max} e^{\frac{\ln(\frac{bw_{min}}{bw_{max}})}{Max_gen} \times t} \quad (4.4)$$

4.3.3 Experimental Study

For each problem, the DMS-PSO-HS, the DMS-PSO and the HS are run 30 times. The maximum function evaluations Max_FEs are set at 100,000 for 10-D, and 200,000 for 30-D. The computer system configuration: **System:** Windows XP (SP1), **CPU:** Pentium(R) 4 3.00GHz, **RAM:** 2 G, **Language:** Matlab 7.1. For each function, we present the Mean for the 30 runs in Table 4.4 and 4.5. The better mean values achieved by the three algorithms for each test function are shown in bold. In order to determine whether the results obtained by DMS-PSO-HS are statistically different from the results generated by the DMS-PSO and HS, the h values presented in the last rows of Tables 4.4 and 4.5 is the results of t-tests as well as Wilcoxon tests. An h value of 1 indicates that the performance of the DMS-PSO-HS is statistically superior with 95% certainty, whereas h value of 0 implies that the performances are not statistically different.

Table 4. 4 Results of 10-D test problems.

<i>Func</i> <i>Algorithms</i>	Group A	Group A	Group B	Group B
	f_1	f_2	f_3	f_4
<i>DMS-PSO</i>	5.6941e-100	1.4986e-00	8.9514e-14	1.6456e-03
<i>HS</i>	2.5709e-09	3.0398e-00	1.8476e-05	6.5027e-02
<i>DMS-PSO-HS</i>	1.9541e-102	9.4626e-02	2.5847e-15	1.2389e-03
h	1/1	1/1	1/1	1/1
<i>Func</i> <i>Algorithms</i>	Group B	Group B	Group B	Group B
	f_5	f_6	f_7	f_8
<i>DMS-PSO</i>	0	0	0	1.7384e+02
<i>HS</i>	2.8941e-01	1.0918e-09	1.4175e-09	9.4118e-09
<i>DMS-PSO-HS</i>	0	0	0	0
h	0/0	0/0	0/0	1/1
<i>Func</i> <i>Algorithms</i>	Group C	Group C	Group C	Group C
	f_9	f_{10}	f_{11}	f_{12}
<i>DMS-PSO</i>	3.3159e-15	1.9966e-02	0	3.2289e-00
<i>HS</i>	9.4486e-01	1.5422e-01	3.4432e-00	2.0198e+01
<i>DMS-PSO-HS</i>	3.3190e-15	1.9038e-02	0	3.0512e-00
h	0/0	1/1	0/0	1/1
<i>Func</i> <i>Algorithms</i>	Group C	Group C	Group D	Group D
	f_{13}	f_{14}	f_{15}	f_{16}
<i>DMS-PSO</i>	4.3225e-00	3.6628e+02	1.3338e+01	1.6485e+01
<i>HS</i>	1.2533e+01	1.0648e+03	1.7333e+02	2.5391e+02
<i>DMS-PSO-HS</i>	4.1333e-00	3.5394e+02	6.6667e-00	1.0518e+01

h	1/1	1/1	1/1	1/1
-----	-----	-----	-----	-----

Table 4. 5 Results of 30-D test problems.

$\begin{matrix} \text{Func} \\ \text{Algorithms} \end{matrix}$	Group A	Group A	Group B	Group B
	f_1	f_2	f_3	f_4
<i>DMS-PSO</i>	1.2658e-66	1.9828e+01	6.5178e-14	0
<i>HS</i>	9.6429e-08	5.1963e+01	7.7705e-05	2.8834e-02
<i>DMS-PSO-HS</i>	3.6514e-78	1.7023e+01	3.2145e-15	0
h	1/1	1/1	1/1	0/1
$\begin{matrix} \text{Func} \\ \text{Algorithms} \end{matrix}$	Group B	Group B	Group B	Group B
	f_5	f_6	f_7	f_8
<i>DMS-PSO</i>	0	1.4014e+01	1.8754e+01	1.7282e+03
<i>HS</i>	7.6028e-01	5.4821e-08	5.1302e-08	3.0365e-01
<i>DMS-PSO-HS</i>	0	0	0	9.9496e-02
h	0/0	1/1	1/1	1/1
$\begin{matrix} \text{Func} \\ \text{Algorithms} \end{matrix}$	Group C	Group C	Group C	Group C
	f_9	f_{10}	f_{11}	f_{12}
<i>DMS-PSO</i>	4.8554e-15	4.8307e-04	1.1058e-02	2.8101e+01
<i>HS</i>	1.3461e-00	2.3159e-02	1.1471e+01	7.9255e+01
<i>DMS-PSO-HS</i>	4.3817e-15	4.3147e-04	1.1086e-02	1.9269e+01
h	1/1	1/1	0/0	1/1
$\begin{matrix} \text{Func} \\ \text{Algorithms} \end{matrix}$	Group C	Group C	Group D	Group D
	f_{13}	f_{14}	f_{15}	f_{16}
<i>DMS-PSO</i>	3.2051e+01	3.2048e+03	1.3333e+01	1.9840e+01
<i>HS</i>	5.1067e+01	3.1524e+03	1.5667e+02	1.5861e+02
<i>DMS-PSO-HS</i>	1.7967e+01	1.9558e+03	2.4542e-31	4.4920e-00
h	1/1	1/1	1/1	1/1

From Table 4.4 and 4.5, we can easily find that the DMS-PSO-HS always achieved better results on all four multimodal groups. According to the results of t -tests as well as Wilcoxon tests, DMS-PSO-HS significantly improves the results on almost all the test problems especially on the difficult multimodal problems in Group B, C and D. On the two composition functions with randomly distributed local and global optima, DMS-PSO-HS performs much better on both cases of 10 and 30 dimensions.

For the first function in Group A, as well as the relative easier multimodal functions f_3, f_6 and f_7 with 10 dimensions in Group B, all the three approaches can solve them if we apply the $1e-4$ success criterion, while the HS performs worst among them. For the more difficult test problems f_2, f_4, f_5, f_8 in Group B with 10 dimensions, HS is almost getting the worst performance except the f_8 , HS performs better than the original DMS-PSO on this test function, while DMS-PSO-HS still performs superior on all the two groups. According to the results of the rotated functions with 10 dimensions in Group C, the DMS-PSO-HS performs much better than the original HS. Since the original HS is hard to solve this kind of difficult test problems, the proposed Hybrid algorithm takes the advantage of the DMS-PSO's specialty, it keeps the best performance among them, which are slightly better than the original DMS-PSO. For the two most difficult test problems in Group D, the DMS-PSO-HS is leading the performance in an obvious range. For the 30 dimensions test functions, we could observe the performance of the original DMS-PSO is falling down rapidly on functions f_6, f_7 and f_8 in Group B when comparing with lower dimensions cases. However, the original HS performs stably on these functions even the dimensions of the test functions increasing from 10 to 30, which are much better than the original DMS-PSO. As the hybrid algorithm of them, the DMS-PSO-HS can still get the statistically superior performance on most functions, especially on f_6, f_7, f_8, f_{15} and f_{16} . In hence, the DMS-PSO-HS can fully exert its comprehensive search ability benefited from the two original approaches when solving the complicate test functions.

By analyzing the results on 10- D and 30- D problems, one may conclude that the DMS-PSO-HS benefits from both the DMS-PSO and the HS algorithms, combines the fast convergent speed as well as the strong exploration ability, assembling with dynamic multiple swarms structure with diversity preserving characteristic, for all kinds of functions used in this four groups, it performs significantly better than the other two original approaches. Furthermore, most results of those functions can be markedly improved by the proposed DMS-PSO-HS. The DMS-PSO-HS can perform stably among scaling the dimension of the test functions, which obviously surpass the other two original algorithms when rotating and compositing the difficult multimodal test functions.

4.4 Conclusions

This chapter proposed a hybridized dynamic multi-swarm particle swarm optimizer (DMS-PSO) with the Harmony search (DMS-PSO-HS). Under the configuration of original DMS-PSO, we periodically generate the new harmonies base on the current *pbests* in each sub-swarm after the PSO's positions have be updated, then replace the worst *pbest* by the new harmony with better fitness. The DMS-PSO-HS attempts to take merits of the PSO and the HS, to avoid all particles getting trapped into inferior local optimal regions and increase the diversity of the whole swarm, the dynamic multiple swarms is also applied. The DMS-PSO-HS enables the particles to have more diverse exemplars to learn from after we frequently regroup the swarms and allow the harmonies to search in a larger potential space among different sub-populations. From the analysis of experimental results, we observe that this proposed DMS-PSO-HS make good use of the information in previous solutions in every sub-swarm more effectively to generate better quality solutions frequently when compared to the original DMS-PSO and the original HS. Based on the results of the three approaches on the 16 benchmark test problems belonging to four classes, we can conclude that DMS-PSO-HS significantly improves the performance upon two original approaches on most multimodal and unimodal problems. When solving real world problems, we are unable to restrict a fixed number of dimensions depend on the approach's speciality, and we also do not frequently know the shape of the fitness landscape and the complexity of the modeling functions. Hence, it is advisable to use an algorithm that performs well on all kinds of problems. In addition, the configuration of DMS-PSO-HS does not introduce further complex operations beyond the original DMS-PSO and HS, contrarily, the DMS-PSO-HS eliminate the parameters in the original HS, which normally need to be adjusted according the property of the test problems, such as the bandwidth bw . The DMS-PSO-HS is simple and easy to implement with fast speed. In our future work, we will investigate more simple and stable parameter settings in the DMS-PSO-HS. And we will also investigate a better variant to combine PSO and HS, to solve both continuous and discrete problems efficiently.

Chapter 5

Two-Lbests Based Multiobjective Particle Swarm Optimizer (2LB-MOPSO)

5.1 Introduction

In PSO, as introduced previously, the c_1 and c_2 in the velocity update equation in Figure 5.1 are the acceleration constants, which represent the weighting of stochastic acceleration terms that pull each particle toward $pbest$ and $gbest$ positions. $rand1()$ and $rand2()$ are two uniform random numbers between 0 and 1. $X_i = (x_{i1}, x_{i2}, \dots, x_{iD})$ represents the position of the i^{th} particle; $pbest_i = (pbest_{i1}, pbest_{i2}, \dots, pbest_{iD})$ represents the best previous position of the i^{th}

particle with the best fitness value; $gbest = (gbest_1, gbest_2, \dots, gbest_D)$ represents the best position obtained by the swarm so far; $V_i = (v_{i1}, v_{i2}, \dots, v_{iD})$ represents the rate of the positional change (velocity) of particle i . The relative simplicity of the PSO and the fact that it is a population-based technique has made it a natural candidate to be extended for solving multi-objective optimization problems.

In this chapter, it is pointed out that all MOPSO implementations select the global best ($gbest$) or local best ($lbest$) from only the non-dominated solutions in the external archive in the current iteration [275][292]. This approach emphasizes the elitism at the expense of diversity with possible adverse consequences due to the potential loss of diversity if the size of the non-dominated solutions is very small for several consecutive iterations. This chapter proposes that the $gbests$ or $lbests$ should be selected from the top fronts in a non-domination sorted external archive of reasonably large size. A novel two local bests ($lbests$) based MOPSO version (2LB-MOPSO) is also proposed to effectively focus the search around a small region in the parameter space in the vicinity of the best existing fronts whereas in the current MOPSO variants, the $pbest$ and $gbest$ (or $lbest$) of a particle can be far apart in the parameter space due to the random nature of their selection process thereby potentially resulting in a chaotic search process.

Figure 5. 1 The flowchart of a typical MOPSO implementation

5.2 Multi-objective Particle Swarm Optimization (MOPSO)

Several optimization problems have many objectives conflicting with each other. As there is no single solution for these problems, our aim is to find Pareto optimal trade-off solutions representing the best possible compromises among all objectives. A multi-objective optimization problem (MOP) with m conflicting objectives can be defined as [293]:

$$\begin{aligned}
 &\text{Minimize/Maximize} && \mathbf{y} = f(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_m(\mathbf{x})), \\
 &\text{subject to:} && g_j(\mathbf{x}) \leq 0, j = 1, \dots, J \\
 &&& h_k(\mathbf{x}) = 0, k = 1, \dots, K
 \end{aligned} \tag{5.1}$$

$\mathbf{x}$ is the decision vector, $x \in \Omega$, where Ω is the *decision (variable) space*, normally $\Omega = [X_{\min}, X_{\max}]$. $\mathbf{y}$ is the objective vector. Different from the single objective optimization, there are two spaces to be considered. One is the decision space denoted as $\mathbf{x}$ and the other is the objective space denoted as $\mathbf{y}$. Some basic concepts with respect to multi-objective optimization are defined below:

Definition 1: Pareto Dominance - For any two-objective 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 2: Non-dominated set - Among a set of solutions $\mathbf{P}$, the non-dominated set of solutions $\mathbf{P}'$ are those that are not dominated by any member of the set $\mathbf{P}$.

Definition 3: Pareto Optimality - When the set $\mathbf{P}$ is the entire feasible search space, the resulting non-dominated set $\mathbf{P}'$ is called the Pareto-optimal solution set.

In every iteration, all solutions are sorted based on non-domination concept into different fronts. The solutions belonging to the first front (front 1) are not dominated by any solution in the current iteration. The solutions belonging to the second front are not dominated by any solution in the current iteration except solutions belonging to front 1. After sorting all

solutions into fronts, each solution is assigned with a rank value as the front number to which the solution belongs.

Development of multi-objective nature-inspired optimization algorithms has attracted much interest and several techniques have been proposed [293], [299]. While these algorithms were developed to achieve fast convergence to the Pareto-optimal front with good distribution of solutions along the front, each algorithm employs a combination of techniques to achieve these goals. The main advantage of EAs and SIs in solving multi-objective optimization problems is their ability to find multiple Pareto-optimal solutions in one single run. As particle swarm optimizers (PSO) also have this ability, several multi-objective particle swarm optimizers (MOPSOs) [275], [292] have been proposed.

When solving single-objective optimization problems, the guides (*i.e.* $pbest$, $lbest$, $gbest$) used by each particle to update its position are completely determined once a neighborhood topology is chosen. However, when solving multi-objective optimization problems, due to the existence of numerous non-dominated solutions, each particle can have a set of guides from which one or more can be selected in order to update its position. All MOPSO implementations maintain an external archive to hold the current set of non-dominated solutions. In order to hold the best solutions of a fixed size in the external archive, the non-domination sorting combined with either the crowding distance sorting [295] or clustering [278] or binning [289] or gridding [279] or niching methods [284], [288] or dynamic neighborhood strategy [281] or dominated tree [300] are applied to the external archive in MOPSO variants. The non-dominated solutions contained in the external archive are used as $lbest$ or $gbest$ guides when the positions of the particles in the swarm are updated. Figure 5.1 shows a typical MOPSO implementation. The external archive only stores the non-dominated front 1 solutions in all current MOPSO implementations. Since the size of the repository is usually limited to the finally required approximation solution set size, whenever the front 1 solution set size exceeds the finally required approximation solution set size, an appropriate strategy [278], [300], [279], [281], [284], [288], [289] is used to truncate the archive size by giving priority to solutions located in less populated areas of objective space over those in highly populated regions.

Branke and Mostaghim [276] allowed each particle to memorize all non-dominated personal best particles it has encountered. If the updated personal best position did not dominate the old one, they kept both in the personal archive. Alvarez-Benitez *et al.* [277] proposed

methods based exclusively on dominance for selecting guides from a non-dominated archive. Mostaghim *et al.* [285] introduced the Sigma method for finding the best local guides for each particle of the population from the non-dominated set. In [283], Ireland *et al.* compared a new principal “Centroid” method with the Sigma method for selecting the guide particles from front 1 solutions only. Jiang *et al.* introduced a particle angle division method for finding the global best particle from the current set of non-dominated solutions for each particle in the population [301].

Instead of a single comparison between a particle’s personal best and its new position, NSPSO [284] compared all particles’ personal bests and their new positions of the entire swarm, to propel the swarm towards the Pareto-optimal front by using the non-domination sorting concept and two parameter-free niching methods. In TV-MOPSO [302], a new diversity parameter has been used to ensure sufficient diversity amongst the solutions of the non-dominated front, while retaining at the same time the convergence to the Pareto-optimal front.

Fieldsend and Singh [300] used an unconstrained elite external archive in which a special data structure called “dominated tree” is adopted to store the non-dominated individuals found along the search process. Hao *et al.* [279] described an MOPSO algorithm with enhanced ε -dominance used to update the archive which also included only the non-dominated solutions. Abido [275] presented a two-level of non-dominated solutions approach to MOPSO using more than one external archive. All external archives store only non-dominated individuals and all dominated solutions are deleted from the external archives. To avoid the effort associated with choosing control parameter settings, Zielinski [292] presented an adaptive approach for parameter setting of an MOPSO. Multi-objective comprehensive learning particle swarm optimizer (MOCLPSO) [282], which extended the CLPSO [303], has also demonstrated competitive performance against several MOEAs [278], [295], [296]. All these MOPSO implementations select guides from external archives having only the non-dominated solutions in the current iteration.

The Vector Evaluated Particle Swarm Optimization (VEPSO) algorithm [287], [304] has been inspired by the concept of the Vector Evaluated Genetic Algorithm (VEGA) [305]. In VEPSO [287], [304], there are M swarms corresponding to each of the M objectives. Each swarm optimizes just one objective very much like the single objective PSO, except the *gbest* of one swarm is selected from the *gbests* of other swarms. However, once a solution

has the best objective value with respect to one objective, that solution must belong to the current non-dominated solution set. Hence, VEPSO also selects the *gbests* from non-dominated solutions only. Further, instead of considering all non-dominated solutions as the potential *gbests*, VEPSO considers only the end points of the current non-dominated solutions.

In [306], the individuals of a swarm update their flight directions through their neighboring leaders selected from the non-dominated solutions in the population. When the number of front 1 solutions is not larger than half of the population size, they assign individuals with less than or equal to the average rank of the whole population to be the set of leaders (SOL); when the number of front 1 solutions is larger than half of the population size, the SOL is selected only from front1 solutions. This algorithm without an external archive is referred to as SMM [306] and included in comparison with our proposed algorithm in the later experiment section. It should be observed that the SMM may be regarded as a hybrid MOPSO because, in SMM, leaders and new positions are combined, non-domination sorted and truncated to the size of population. The top half of this truncated population is selected as the SOL and not evaluated in the next generation, the other half acquire information from the SOL and move to new positions through a simple generational operator. In all MOPSOs, all particles continuously fly through the search space without a selection process and non-dominated solutions are captured in the external archive.

5.3 Two lbests based Multi-objective Particle Swarm Optimizer (2LB-MOPSO)

In this section, first the non-domination sorted external archive is briefly presented. Subsequently, our novel two-*lbests* based MOPSO with non-domination sorted external archive is presented [328], [234].

5.3.1 Non-domination Sorted External Archive

An external archive is commonly used to store the non-dominated solutions obtained in the search process by an MOPSO. Since the size of the non-dominated set can be very large and the complexity of updating the archive increases with the archive size, the size of the archive

is usually restricted to a pre-specified size which is usually the same as the finally required approximation solution set size. To truncate the external archive in our MOPSO implementation, the procedure used in the NSGA-II [295] to truncate NSGA-II's population is employed.

In our MOPSO implementation, the initialized archive includes all initialized solutions at iteration 1. In every iteration, all new positions $Q(t)$ generated in iteration t is combined with the members in the archive $P(t)$ to obtain the mixed-temporary external archive. The non-domination sorting is applied to this mixed-temporary archive to obtain the sorted archive $R(t)$. During this process, all the sorted solutions retain two indicators namely, the front rank F_R and crowding distance value D_N [295]. The sorted solutions with the lowest front rank [295] will be included in the archive $P(t+1)$ first. When the size of the archive hits the permitted maximum size of archive $|P(t+1)|$ ($=|P(t)|$), the crowding distance [293] is applied to select the required number of members to be included in $P(t+1)$ from the lowest front that still remains unselected in archive $R(t)$.

5.3.2 Two-*lbests* Based Multi-objective Particle Swarm Optimizer

We observe that in most MOPSO algorithms, particles fly in directions in the search space determined by *pbest* and *gbest* positions. In reality, these positions can be far apart from each other in the parameter space. Therefore, guiding the directions of acceleration of particles by using two far apart *pbest* and *gbest* positions may not be very effective when solving hard problems. Therefore, this chapter proposes two *lbests* based MOPSO variant. In this variant, each objective function range in the external archive is divided into a number of bins (n_bins). The two *lbests* are chosen from the external archive members located in two neighboring bins, so that they are near each other in the parameter space. In order to select the first *lbest* for a particle, an objective is first randomly selected followed by a random selection of a non-empty bin of the chosen objective. Within this bin, the archived member with the lowest front number and among these with the highest crowding distance is selected as the first *lbest*. The second *lbest* is selected from solutions in the neighborhood non empty bins with the lowest front number and the smallest Euclidean Distance in the parameter space to the first *lbest*. As each particle is guided by two *lbests* from a neighborhood in parameter and objective spaces with the smallest front number, the velocity updating of each

particle will be in the direction between the positions of two *lbests* and oriented to improve upon the current non-dominated solutions as shown in Figure 5.2.

Given that the first *lbest* of every particle is chosen somewhat randomly, it will not be effective if every particle is assigned with a new pair of *lbests* which come from the different pair of bins in every iteration, because the flight of each particle will be almost random in this case. Therefore, after assigning a pair of *lbests* to a particle, the number of iterations the particle fails to contribute a solution to the archive $P(t)$ is counted. If the *count* exceeds a pre-specified threshold, the particle is reassigned with another pair of *lbests*. During the initialization stage and when *count* is larger than the pre-specified threshold during the iterative optimization stage, the first *lbest* for a particle is chosen randomly by selecting an objective and one bin of the objective. The second *lbest* is chosen from the neighborhood of the first *lbest* in the parameter space. Otherwise, when *count* is less than or equal to the pre-specified threshold during the iterative optimization stage, two *lbests* are chosen from the same assignment of the objective and the bin as used in the last iteration. The particle will accelerate potentially in a direction between the two *lbests*. Hence, this particle may explore the region of the two *lbests*. The Figure 5.2 illustrates both of the two cases during the iterative search.

In Figure 5.2, a two-objective minimization problem with **Objective1** and **Objective2** is shown. Each objective range in the archive $P(t)$ is equally divided. $P(t)$ has twelve archived solutions in two fronts, among them, archived solutions (**AS**) 1 to 6 are non-dominated front 1 solutions, the other six are front 2 solutions.

The four particles namely **P1** to **P4** fly in directions guided by their corresponding two *lbests*. For example, assuming that the *count* for **P4** exceeds the pre-specified threshold in the current iteration, when selecting the two *lbests* for the particle **P4**, it is have to randomly select one objective and one bin for its first *lbest*. Here, objective 1 and bin 5 are assumed to be selected for the first *lbest* of **P4**. Among the two candidates **AS1** and **AS3** with the lowest front number in bin5, the **AS1** is chosen as the first *lbest* as it has the larger crowding distance. The **AS3** is selected to be the second *lbest* of **P4** as it is nearest to the **AS1** in the neighborhood in the parameter space as well as the objective space with the lowest front number. Hence, the **P4** will accelerate in direction **D4** in the next iteration. Assuming the first *lbest* of **P2** is assigned to objective 1 and bin3 in the previous iteration, and the *count* for **P2** does not exceed the pre-specified threshold in the current iteration. **AS4** is selected as the

first *lbest* because it has the lowest front member in bin3. From the neighboring solutions with the lowest front members in each bin, *AS9* with the smallest parameter space Euclidean Distance to *AS4* will be chosen as the second *lbest*. Hence, *P2* will accelerate in *D2* guided by the *AS4* and *AS9* as the first *lbest* and second *lbest* respectively. Assuming that the *counts* for *P1* and *P3* also do not exceed the pre-specified threshold, following the same rules, *AS1* and *AS3* can be chosen as the first *lbest* and second *lbest* for *P1* as well as *AS6* and *AS5* can be chosen as the first *lbest* and second *lbest* for *P3*. Table 5.1 presents the pseudo-code of the 2LB-MOPSO.

Figure 5. 2 An illustration of two lbests based search process with minimization objectives

Table 5. 1 The Pseudo-code of the 2LB-MOPSO

Initialize the swarm: Initialize NP number of particles randomly and uniformly in the D -dimensional search space. Evaluate the fitness values of all particles. Set all current positions to be $P(0)$, the external archive.

Select $lbests$ from the $P(0)$ external archive using the steps within “If $count(j)>5$ ” below.

Optimization Loop:

```

For  $i=1$  to  $Feval\_max$                                 // iteration loop
For  $j=1$  to  $NP$                                           // iteration loop for updating velocity, position of each
particle
    If  $count(j)>5$                                        // updating  $lbests$  (more details in subsection 5.3.2)
        For particle  $j$ , the first  $lbest$  is selected by (a) randomly selecting an objective, (b)
        randomly choosing a bin of the selected objective, (c) choosing the solution with the
        lowest front number and largest crowding distance in the chosen bin. The second  $lbest$ 
        is selected from the neighborhood of the first  $lbest$  in the parameter space.
    Else
        For particle  $j$ , update the two  $lbests$  in the same bins used in the last iteration. The first
         $lbest$  should have the lowest front number and the largest crowding distance in the bin.
        The second  $lbest$  is selected from the neighborhood of the first  $lbest$  in the parameter
        space.
    Endif
     $V(j) = \omega \times (V(j) + c_1 \times rand() \times (lbest(j) - particle(j)) + c_2 \times rand() \times (lbest(j+NP) - particle(j)))$  // updating velocity
     $V_i(d) = (\min(V_{max}(d), V_i(d)) \& \max(-V_{max}(d), V_i(d)))$  // Limit the velocity for each dimension
     $X_i(d) = X_i(d) + V_i(d)$  // updating position of each particle
    If any dimension exceeds the search space
        Set the corresponding bound value to be the dimension value of  $particle(j)$ ;
    Endif
    Evaluate the fitness values of  $particle(j)$ 
     $feval\_count = feval\_count + 1$ 
EndFor  $j$ 
Do non-dominated sorting on combined  $P(t)$  and new particles  $Q(t)$  to obtain external
archive  $P(t+1)$ .
Update  $count$  of each particle ( $count$  is the number of iterations the particle fails to
contribute a solution to the external archive as explained in subsection 5.3.2).
Stop if a stop criterion is satisfied (here is when  $feval\_count$  reached the  $Feval\_max$ ).
EndFor  $i$ 

```

5.4 Experimental Procedures

5.4.1 MO algorithm Variants and Benchmark Problems

We make use of 19 multi-objective problems [307]-[310], [299]. Among these problems [309], the first seven are two-objective problems, the next six are three-objective problems and the last six are five-objective problems. All the detailed definitions of these problems are list in Appendix C. Our proposed 2LB-MOPSO is compared against two MOPSO variants and seven MOEA variants. Eight of these MOEA variants were from the CEC07 Special Session and Competition on “Performance Assessment on Multi-objective Optimization Algorithms” [309] on the multi-objective problems. The algorithms used in our experimental comparisons are:

1. Proposed 2LB-MOPSO: Two *Lbests* based MOPSO. Our 2LB-MOPSO was coded in Matlab language. Population size: $NP = 50$, $\omega = 0.729$; $c_1 = c_2 = 2.05$; $V_{\max} = 0.25(X_{\max} - X_{\min})$; $n_bin = 10$ $V = (2 \times rand(NP, 1) - 1) \times (0.25 \times (X_{\max} - X_{\min}))$.
2. MOCLPSO (Multi-objective Comprehensive Learning Particle Swarm Optimizer) [282]: The same parameter settings as in the original MOCLPSO paper were used. Population size: $N=50$. Inertia weight updating, for $k=1$ to $maxgen$ $\omega(k) = \frac{(\omega_0 - \omega_1) \times (\max gen - k)}{\max gen} + \omega_1$ and $\omega_0 = 0.9, \omega_1 = 0.2$. Acceleration rate, $c1=c2=2.05$. Velocity definition, $V_{\max} = 0.25 \times (X_{\max} - X_{\min})$. Learning probability $P_c = 0.1$, elitism probability $P_m = 0.4$. The Original Matlab codes of MOCLPSO [282] were used. As MOCLPSO outperforms the MOPSO, we choose MOCLPSO [282] and exclude MOPSO [278] from the comparisons.
3. MO_PSO: In this work, adaptive control parameter setting is employed with a multi-objective particle swarm optimization algorithm [292].
4. NSGA2_SBX: In this work, NSGA-II with SBX crossover is employed. The sequential quadratic programming (SQP) method is used as a local search procedure [311].
5. NSGA2_PCX: A hybrid multi-objective optimization approach using NSGA-II and PCX crossover is employed. The sequential quadratic programming (SQP) is used as the local search procedure [312].
6. GDE3: In this work, the generalized differential evolution 3 is applied to solve the multi-objective optimization problems [313].

7. MO_DE: A multi-objective differential evolution algorithm with an adaptive approach for setting control parameters is used [314].
8. MOSaDE: This multi-objective optimization algorithm uses self-adaptive differential evolution algorithm which automatically adapts the trial vector generation strategies and their associated parameters according to their previous experience of generating promising or inferior individuals [315].
9. DEMOwSA: This multi-objective differential evolution algorithm employs self adaptation strategies to adapt F and CR parameters of DE [316].
10. MTS: Multiple trajectory search for multi-objective optimization (MTS) uses multiple agents to search the solution space concurrently. Each agent performs an iterated local search using one of four candidate local search methods [317].

5.4.2 Parameter Setting

In order to be comparable to the CEC07 “Performance Assessment on Multi-objective Optimization Algorithms”, for the other two MOPSOs (2LB-MOPSO and MOCLPSO [282]) which are not included in the CEC07 Special Session and Competition, the maximum of function evaluations (MAX_FES) with 25 independent runs is set at 500,000 and the results are recorded three steps after 5000, 50,000 and 500,000 function evaluations. The archive sizes are set as 100 for two-objective problems, 150 for three-objective problems and 800 for five-objective problems. The maximum number of non-dominated solutions used to calculate the performance metrics is the same as the archive size for all the algorithms. All the above parameter settings are exactly the same as in the other algorithm variants from the CEC07 Special Session and Competition.

5.4.3 Performance Measures

In order to compare the performances of the above algorithms, quantitative performance metrics are needed. There are two goals in a multi-objective optimization: 1) convergence to the true Pareto-optimal front and 2) distribution of approximated solutions. The following two metrics were used and the metrics were computed using normalized objective values. The same performance measures were used in the CEC07 “Performance Assessment on Multi-objective Optimization Algorithms” [309].

1) **R indicator** (I_{R2}) [318] :

$$I_{R2} = \frac{\sum_{\lambda \in \Lambda} u^*(\lambda, A) - u^*(\lambda, R)}{|\Lambda|} \quad (5.2)$$

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., $u^* = \max_{z \in A} u_\lambda(z)$. The augmented Tchebycheff function was used as the utility function. Λ is the full set of λ_i , $\lambda_i > 0$. The R indicator shows how much the approximation set A has to be translated/scaled so that it covers the reference set R .

2) **Hypervolume difference to a reference set** ($I_{\bar{H}}$) [318]: The hypervolume indicator I_H measures the hypervolume of the objective space that is weakly dominated by an approximation set A , and is to be maximized. Here, the hypervolume difference to a reference set R is used and referred 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 . The hypervolume indicator possesses the following two properties. On the one hand, it is sensitive to any type of improvements, i.e., whenever an approximation set A dominates another approximation set B , then the measure yields a strictly better quality value for the former than for the latter set. On the other hand, the hypervolume measure guarantees that any approximation set A that achieves the maximally possible quality value for a particular problem contains all Pareto-optimal objective vectors.

The R indicator and the hypervolume indicator capture both convergence to the Pareto front and diversity of the approximated set of solutions.

5.5 Discussions on Experimental Results

We conducted a series of simulation experiments. The simulation results of tuning parameters *count* and *n_bin* are reported in subsection 5.5.1. In subsection 5.5.2, the main experimental results obtained by using our proposed 2LB-MOPSO on all the 19 test problems are presented and compared with the results of the CEC 2007 competition. The

competition considered the speed of convergence and the quality of the final solutions by recording the results after 5000, 50,000 and 500,000 function evaluations. Different algorithms are compared by ranking the mean results. In order to perform the comparison, the results as reported in the corresponding publications of algorithms 3 to 10 in the CEC 2007 competition are copied. The associated parameter values of these eight algorithms are presented in the respective publications. The parameter settings related to the multi-objective optimization problems are presented in previous subsection 5.4.2 and these are the same for all 10 algorithms. M indicates the number of objectives in the test problem in each comparison. Additional experiments were conducted in subsection 5.5.3 to compare the performance when the two *lbests* are selected from a) a non-domination sorted external archive of a reasonably large size, b) only non-dominated (front 1) solutions. Experiments were also conducted to show the benefit of selecting *gbest* (or *lbest*) from the neighborhood in both parameter space and objective space and the advantage of retaining the *gbest* or *lbest* for every particle in the same neighborhood for a few iterations (*i.e. count*). Finally, in subsection 5.5.4, SMM [304] and 2LB-MOPSO are compared by testing on the same test problems as used in the SMM. In Tables 5.2 to 5.9, the best results and the best ranks in each comparison category are highlighted in bold.

5.5.1 Parameter Sensitivity Studies

In 2LB-MOPSO, there are *count* and *n_bins* parameters defined in Section 5.3.2 to be tuned. In this section, out of these 19 multi-objectives problems, 6 problems are used to investigate the impact of the parameters. The 6 problems are No. 2 SYMPART (two-objectives), No. 7 S_ZDT6 (two-objectives), No. 9 R_DTLZ2 (three-objectives), No. 10 S_DTLZ3 (three-objectives), No. 18 WFG8 (five-objectives) and No. 19 WFG9 (five-objectives). The 2lb-MOPSO is run for a maximum of 500,000 function evaluations (FES) with 25 independent runs. The results are recorded after 5000, 50,000 and 500,000 function evaluations. For each of these test problems, the mean value of the R and H indicator achieved by the 25 runs are ranked among all different parameter values used.

5.5.1.1 Parameter Sensitivity Study on *count*

As explained in section 5.3.2, the *count* value is used as a pre-specified threshold to decide when to reassign a particle to another pair of *lbests* if the particle fails to generate a solution

to enter the external archive consecutively for *count* number of iterations. Experiments were conducted with *count* set to 3, 4, 5, 6 and 7 with 25 independent runs and results are recorded after 5000, 50,000 and 500,000 function evaluations. Mean values of *R* and *H* indicators obtained with different *count* on all 6 test functions are ranked. Two indicators on three searching steps sum up to 6 sets of comparisons. In order to save space, only one of them is shown in Table 5.2. In addition, Figure 5.3 shows the summation of all the rankings of *R* and *H* indicators on three searching steps of each *count* value on all 6 test functions. It can be observed that *count* can influence the results. But, the results are not highly sensitive as the values are approximately similar as can be seen from Table 5.2. For the six test functions, better results were obtained when *count* is around 5 as shown in Figure 5.3. Hence, in our proposed 2LB-MOPSO, the *count* is set at 5 for all test problems.

Table 5. 2 *R*_indicator of parameter Study on *count* of 2LB-MOPSO on 500k function evaluation

<i>FES</i>	<i>count</i>	2.SYMPART	7.S_ZDT6	9.R_DTLZ2	10.S_DTLZ3	18.WFG8	19.WFG9	Rank
5e+5	3	2.5709e-06(5)	5.5397e-04(5)	2.4273e-04(5)	1.2059e-05(4)	-5.3581e-03(5)	8.0232e-04(2)	26
	4	2.1907e-06(4)	4.0070e-04(4)	2.2154e-04(4)	1.1977e-05(3)	-5.5447e-03(4)	1.0624e-03(4)	23
	5	1.3453e-06(3)	3.5901e-04(3)	1.8229e-04(1)	1.1170e-05(1)	-5.7425e-03(3)	7.4545e-04(1)	12
	6	9.0367e-07(1)	2.9691e-04(2)	1.9760e-04(2)	1.1204e-05(2)	-6.2084e-03(1)	8.7012e-04(3)	11
	7	9.6147e-07(2)	2.8393e-04(1)	2.1844e-04(3)	1.4117e-05(5)	-6.0251e-03(2)	1.2411e-03(5)	18

5.5.1.2 Parameter Sensitivity Study on *n_bin*

The number of bins has to be assigned with an appropriate numerical value to coarsely partition each objective for the selection of the *lbests*. Experiments were conducted with the *n_bins* set to 8, 10, 12, 15 and 20 with 25 independent runs. The results are recorded after 5000, 50,000 and 500,000 function evaluations. From Figure 5.3 and Table 5.3, it can be observed that *n_bins* also influences the results, whereas they are not very sensitive as the differences among the values are even smaller than those in *count* investigation in subsection 5.5.1.1. For the 6 test problems, better results were obtained when the *n_bins* is 10. Hence, in our proposed 2LB-MOPSO, the *n_bins* is set at 10 for all test problems.

Table 5. 3 *R*_indicator of parameter Study on *n_bins* of 2LB-MOPSO on 500k function evaluation

<i>FES</i>	<i>n_bins</i>	2.SYMPART	7.S_ZDT6	9.R_DTLZ2	10.S_DTLZ3	18.WFG8	19.WFG9	Rank
------------	---------------	-----------	----------	-----------	------------	---------	---------	------

5e+5	8	1.3878e-06(5)	2.8428e-04(1)	2.3503e-04(5)	1.4967e-05(2)	-4.4631e-03(5)	9.5691e-04(2)	20
	10	1.3453e-06(4)	3.5901e-04(2)	1.8229e-04(1)	1.1170e-05(1)	-5.7425e-03(1)	7.4545e-04(1)	10
	12	1.2506e-06(2)	3.8362e-04(3)	1.9750e-04(2)	1.7923e-05(3)	-5.7410e-03(2)	1.1002e-03(5)	17
	15	1.3317e-06(3)	4.1344e-04(4)	2.0156e-04(3)	1.8445e-05(5)	-5.0425e-03(3)	9.8550e-04(3)	21
	20	1.1305e-06(1)	4.2775e-04(5)	2.1909e-04(4)	1.8279e-05(4)	-4.8530e-03(4)	1.0535e-03(4)	22

Figure 5.3 Ranking results of Parameter Sensitivity Investigation on *count* and *n_bins*

5.5.2 Comparison with other algorithms in CEC 2007 Competition

Tables 5.4-5.6 rank the mean values of R and H indicators among all algorithms on the 19 test problems. According to CEC 2007 Competition, the results at three points (5000, 50,000 and 500,000 function evaluations) are considered in the comparison. The ranks of 5000 and 50,000 function evaluations in Tables 5.4-5.6 show 2LB-MOPSO not only performs with fast convergence speed, but also keeps the good diversity of the solutions which is main shortcoming of most MOPSO variants. From the ranks of 500,000 function evaluations the Tables 5.4-5.6, it can be observed that the 2LB-MOPSO can almost approximate the Pareto optimal front on most of these 19 test problems except one difficult problem S_ZDT4. It can also be observed that the proposed 2LB-MOPSO overall achieves the best solution quality over all other algorithms. In Table 5.4-5.6, the rank for each problem is in () behind the R & H indicators and the summation is in the last column. Function Evaluations=5k, 50k and 500k.

Table 5.7 shows all ranks of indicators for every algorithm on two, three and five-objectives test problems. Out of the 9 cases, obviously, our proposed 2LB-MOPSO can achieve three first rank positions, four second rank position and two third rank position on both R indicator and H indicator among all algorithms on these 19 test problems. According to the ranks in Table 5.7, except the proposed 2LB-MOPSO, there is no other algorithm to be within the top 3 always. In addition, the other two recent MOPSO variants (MOCLPSO and MOPSO) almost perform the worst when compared to the other algorithms. Therefore, our original contributions have made the 2LB-MOPSO to attain the best ranking among all the algorithms while the two other state-of-the-art MOPSO variants were the worst among all the algorithms in the CEC 2007 competition. NSGA2_SBX and GDE3 were declared the joint winners. In Table 5.7 within “Overall Summation” column, NSGA2_SBX and GDE3 have overall aggregate rankings of 6 and 8 respectively, while the 2LB-MOPSO has an overall aggregate ranking of 4. In Table 5.7, for instance, the total ‘22’ for 2LB-MOPSO at FEs=5000 and M=2 is obtained by adding ‘11’ (A1, R and FEs=5000) and ‘11’ (A1, H and FEs=5000) in Table 5.4 in the last column. The ‘Overall Summation’ of 114 (M=2) for the 2LB-MOPSO is obtained by adding 22 (M=2, FEs=5000), 40 (M=2, FEs=50000) and 52 (M=2, FEs=500000). In (), the ranks vertically across different algorithms are shown.

Table 5. 4 The Rank of mean value for R & H indicators from 25 runs on test problems 1-7, $M=2$.

		1.OKA2	2.SYMPART	3.S_ZDT1	4.S_ZDT2	5.S_ZDT4	6.R_ZDT4	7.S_ZDT6	Rank
A1	R	-6.4554e-04 (1)	4.9388e-04 (1)	1.7667e-02 (2)	4.7653e-02 (1)	4.1537e-02 (2)	5.0475e-03 (2)	1.0736e-01 (2)	11
		-1.0650e-03 (1)	3.4465e-05 (3)	2.3668e-04 (4)	3.4675e-0 4(3)	1.1837e-02 (6)	3.2212e-03 (5)	8.1862e-03 (1)	23
		-1.0650e-03 (2)	1.3453e-06 (2)	3.0226e-06 (4)	7.8573e-06 (3)	1.0226e-02 (8)	2.1107e-03 (6)	3.5901e-04 (4)	29
	H	-3.2098e-04 (1)	1.3135e-03 (1)	7.2586e-02 (2)	1.1146e-01 (2)	1.2228e-01 (2)	1.4668e-02 (2)	2.5416e-01 (1)	11
		-1.1483e-03 (1)	8.0142e-05 (2)	2.2239e-03 (3)	1.7708e-03 (2)	3.4343e-02 (4)	1.1174e-02 (4)	1.8110e-02 (1)	17
		-1.2335e-03 (1)	4.1756e-06 (2)	1.3747e-04 (1)	6.1166e-04 (1)	2.9547e-02 (8)	6.3524e-03 (6)	6.9635e-04 (4)	23
A2	R	7.2815e-03 (5)	2.7460e-02 (8)	7.9928e-02 (10)	1.3115e-01 (9)	9.9320e-02 (10)	3.2415e-02 (10)	1.4418e-01 (9)	61
		6.5664e-03 (7)	1.5460e-02 (10)	6.5683e-02 (10)	9.1457e-02 (9)	8.8464e-02 (10)	2.5377e-02 (10)	1.3315e-01 (10)	66
		4.7119e-03 (7)	7.1294e-03 (10)	4.8761e-02 (10)	5.5555e-02 (9)	7.4249e-02 (10)	1.8585e-02 (10)	9.7862e-02 (9)	65
	H	1.3478e-02 (4)	6.5283e-02 (8)	2.5863e-01 (9)	3.4083e-01 (9)	3.0399e-01 (10)	1.0241e-01 (10)	3.6678e-01 (9)	59
		1.2612 e-02 (4)	3.7587e-0 2(8)	2.0720e-01 (8)	2.0826e-01 (6)	2.6926e-01 (10)	7.8247e-02 (9)	3.3486e-01 (7)	52
		9.3511e-03 (6)	1.7680e-02 (10)	1.5672e-01 (10)	9.5113e-02 (9)	2.2381e-01 (10)	5.7637e-02 (10)	2.3861e-01 (9)	64
	R	3.0915e-02 (9)	7.3858e-04 (3)	6.4726e-02 (8)	1.1642e-01 (7)	6.8431e-02 (5)	1.7595e-02 (4)	1.3767e-01 (7)	43

A3		2.7946e-02 (9)	7.2217e-05 (6)	1.9317e-02 (9)	9.6410e-02 (10)	4.1046e-02 (9)	8.1416e-03 (8)	1.2176e-01 (9)	60
		2.5289e-02 (9)	1.6377e-07 (1)	4.6135e-03 (9)	6.5010e-02 (10)	2.4968e-02 (9)	6.5601e-03 (8)	1.0326e-01 (10)	56
	H	6.4591e-02 (8)	2.1470e-03 (3)	1.8442e-01 (7)	2.9063e-01 (7)	2.0558e-01 (5)	5.2014e-02 (4)	3.4784e-01 (7)	41
		5.5488e-02 (7)	2.0980e-04 (3)	6.2726e-02 (5)	2.2567e-01 (7)	1.2170e-01 (6)	2.4289e-02 (6)	3.0257e-01 (6)	46
		5.0571e-02 (9)	4.0187e-07 (1)	1.4403e-02 (8)	1.2916e-01 (10)	7.3732e-02 (9)	1.9731e-02 (8)	2.5218e-01 (10)	55
A4	R	8.095e-04 (3)	1.543e-0 3(4)	1.349e-02 (1)	4.809e-02 (2)	4.164e-02 (3)	1.034e-02 (3)	1.111e-02 (1)	17
		-1.064e-03 (2)	1.449e-04 (8)	1.143e-04 (2)	4.377e-05 (2)	8.775e-04 (2)	2.947e-03 (4)	2.522e-02 (4)	24
		-1.065e-03 (3)	2.024e-05 (6)	2.092e-07 (2)	-2.061e-04 (2)	3.036e-08 (2)	2.126e-03 (7)	1.191e-02 (6)	28
	H	1.968e-03 (3)	4.473e-03 (4)	4.983e-02 (1)	9.915e-02 (1)	1.244e-01 (3)	3.192e-02 (3)	2.741e-01 (3)	18
		-1.047e-03 (3)	4.291e-04 (5)	7.811e-04 (2)	-1.156e-02 (1)	2.391e-03 (1)	9.266e-03 (3)	5.677e-02 (3)	18
		-1.047e-03 (3)	6.302e-05 (7)	3.104e-04 (5)	1.211e-02 (5)	1.507e-06 (2)	6.521e-03 (7)	2.618e-02 (6)	35
A5	R	9.064e-03 (6)	5.142e-04 (2)	7.929e-02 (9)	1.321e-01 (10)	8.586e-02 (9)	3.475e-03 (1)	1.446e-01 (10)	47
		4.922e-03 (5)	3.478e-05 (4)	4.710e-03 (8)	8.911e-03 (5)	1.142e-02 (5)	1.035e-03 (3)	6.071e-02 (8)	38
		-1.644e-03 (1)	6.401e-05 (8)	7.982e-04 (7)	9.618e-04 (4)	5.350e-06 (3)	8.636e-04 (4)	1.859e-02 (8)	35
	H	1.640e-02 (5)	1.499e-03 (2)	2.676e-01 (10)	3.569e-01 (10)	2.674e-01 (9)	1.147e-02 (1)	3.690e-01 (10)	47

		1.640e-02 (5)	1.499e-03 (6)	2.676e-01 (10)	3.569e-01 (10)	2.674e-01 (9)	1.147e-02 (5)	3.690e-01 (10)	55
		8.563e-03 (4)	1.927e-04 (8)	1.693e-03 (6)	1.532e-03 (2)	3.475e-05 (4)	2.965e-03 (4)	4.132e-02 (8)	36
A6	R	1.6276e-04 (2)	2.8114e-02 (9)	4.4531e-02 (5)	9.8161e-02 (5)	7.5424e-02 (6)	1.7979e-02 (5)	1.3425e-01 (6)	38
		-9.8540e-04 (3)	2.4138e-05 (1)	1.1611e-04 (3)	3.3835e-03 (4)	6.9527e-03 (4)	8.6134e-04 (1)	1.9906e-02 (3)	19
		-1.0583e-03 (4)	1.5629e-06 (3)	4.5328e-06 (5)	3.2126e-03 (5)	1.1464e-05 (4)	6.7173e-04 (3)	-1.0788e-06 (2)	26
	H	5.7619e-04 (2)	7.9939e-02 (9)	1.5566e-01 (5)	2.3674e-01 (5)	2.2904e-01 (7)	5.5075e-02 (5)	3.3957e-01 (6)	39
		-1.1430e-03 (2)	7.2177e-05 (1)	5.7459e-04 (1)	4.4286e-03 (3)	2.0748e-02 (3)	3.0337e-03 (1)	4.4900e-02 (2)	13
		-1.2271e-03 (2)	4.6477e-06 (3)	1.7484e-04 (3)	4.0254e-03 (4)	2.7290e-05 (3)	2.2690e-03 (3)	-2.3067e-04 (2)	20
A7	R	1.8533e-02 (7)	1.8554e-02 (7)	4.9050e-02 (6)	1.0030e-01 (6)	7.5535e-02 (7)	2.9114e-02 (9)	1.3414e-01 (5)	47
		5.6759e-03 (6)	5.7227e-05 (5)	6.2174e-04 (6)	5.2728e-02 (8)	1.3448e-02 (7)	6.6259e-03 (7)	1.0831e-02 (2)	41
		3.1177e-03 (6)	3.9920e-06 (4)	1.4460e-05 (6)	1.7833e-02 (8)	6.2459e-04 (6)	1.2570e-03 (5)	2.1275e-03 (5)	40
	H	6.3505e-02 (7)	5.2979e-02 (7)	1.6672e-01 (6)	2.4166e-01 (6)	2.2825e-01 (6)	8.9092e-02 (9)	3.3804e-01 (5)	46
		6.3505e-02 (8)	5.2979e-02 (9)	1.6672e-01 (7)	2.4166e-01 (8)	2.2825e-01 (7)	8.9092e-02 (10)	3.3804e-01 (8)	57
		9.2920e-03 (5)	1.1653e-05 (4)	2.0342e-04 (4)	2.1203e-02 (7)	1.8529e-03(6)	3.6475e-03 (5)	4.5873e-03 (5)	38
	R	1.2813e-01 (10)	4.2596e-03 (5)	2.5002e-02 (4)	5.5486e-02 (4)	5.8496e-02 (4)	2.1604e-02 (6)	1.1388e-01 (4)	37

A8		1.2234e-01 (10)	1.3952e-04 (7)	-5.0790e-03 (1)	-7.6630e-03 (1)	5.3110e-03 (3)	8.9153e-04 (2)	3.0298e-02 (5)	29
		1.2181e-01 (10)	2.5315e-05 (7)	-5.1914e-03 (1)	-8.2653e-03 (1)	-1.5774e-04 (1)	-7.1405e-04 (1)	-8.9649e-06 (1)	22
	H	4.1900e-01 (10)	1.2278e-02 (5)	1.4052e-01 (4)	2.0602e-01 (4)	1.8023e-01 (4)	6.9446e-02 (7)	2.9534e-01 (4)	38
		4.0670e-01 (10)	3.7957e-04 (4)	1.9306e-02 (4)	4.7798e-02 (4)	1.6463e-02 (2)	3.2291e-03 (2)	7.9428e-02 (4)	30
		4.0119e-01 (10)	4.7963e-05 (6)	1.8510e-02 (9)	4.6627e-02 (8)	-1.0907e-03 (1)	-2.3664e-03 (1)	-3.7736e-03 (1)	36
A9	R	1.0738e-03 (4)	3.2867e-02 (10)	5.9281e-02 (7)	1.2142e-01 (8)	8.5307e-02 (8)	2.5097e-02 (8)	1.4213e-01 (8)	53
		-2.6206e-04 (4)	2.5823e-05 (2)	2.4294e-04 (5)	1.5529e-02 (7)	3.0420e-02 (8)	3.2849e-03 (6)	3.0595e-02 (6)	38
		-8.0507e-04 (5)	4.1873e-06 (5)	2.2716e-06 (3)	1.4419e-02 (7)	3.0107e-03 (7)	3.6550e-04 (2)	-1.0152e-06 (3)	32
	H	2.8984e-02 (6)	9.3160e-02 (10)	2.1654e-01 (8)	3.1091e-01 (8)	2.5908e-01 (8)	7.6820e-02 (8)	3.6117e-01 (8)	56
		2.8984e-02 (6)	9.3160e-02 (10)	2.1654e-01 (9)	3.1091e-01 (9)	2.5908e-01 (8)	7.6820e-02 (8)	3.6117e-01 (9)	59
		1.7877e-02 (8)	1.2558e-05 (5)	1.4549e-04 (2)	1.7334e-02 (6)	9.3111e-03 (7)	1.0772e-03 (2)	-2.3035e-04 (3)	33
A10	R	2.3370e-02 (8)	6.0811e-03 (6)	2.0240e-02 (3)	5.2017e-02 (3)	3.0741e-02 (1)	2.1980e-02 (7)	1.0802e-01 (3)	31
		1.1673e-02 (8)	4.8844e-04 (9)	1.5633e-03 (7)	1.3735e-02 (6)	7.9382e-04 (1)	1.0574e-02 (9)	3.7430e-02 (7)	47
		6.5349e-03 (8)	3.9505e-04 (9)	1.0036e-03 (8)	3.3760e-03 (6)	1.2095e-04 (5)	6.6462e-03 (9)	1.3763e-02 (7)	52

	H	7.2530e-02 (9)	1.7546e-02 (6)	7.4565e-02 (3)	1.1336e-01 (3)	9.2170e-02 (1)	6.8130e-02 (6)	2.6939e-01 (2)	30
		7.2530e-02 (9)	1.7546e-02 (7)	7.4565e-02 (6)	1.1336e-01 (5)	9.2170e-02 (5)	6.8130e-02 (7)	2.6939e-01 (5)	44
		1.5385e-02 (7)	1.1601e-03 (9)	2.1198e-03 (7)	2.5849e-03 (3)	3.6566e-04 (5)	2.0422e-02 (9)	3.0606e-02 (7)	47

Algorithms: A1- 2LB-MOPSO, A2- MOCLPSO, A3- MO_PSO, A4- NSGA2_SBX, A5- NSGA2_PCX, A6- GDE3, A7- MO_DE, A8- MOSaDE, A9- DEMOWSA, A10- MTS.

Table 5. 5 The Rank of mean value for R & H indicators from 25 runs on test problems 8-13, $M=3$.

		8.S_DTLZ2	9.R_DTLZ2	10.S_DTLZ3	11.WFG1	12.WFG8	13.WFG9	Rank
A1	R	6.4504e-05 (1)	3.7011e-04 (1)	2.2861e-04 (1)	5.5102e-02 (1)	-1.4884e-02 (1)	-5.5996e-03 (2)	7
		1.6448e-05 (2)	1.0550e-04 (1)	6.5243e-05 (4)	5.2558e-02 (2)	-2.1083e-02 (3)	-9.7943e-03 (1)	13
		3.9674e-06 (3)	1.8229e-04 (4)	1.1170e-05 (4)	4.9602e-02 (6)	-2.3942e-02 (4)	-1.2453e-02 (1)	22
	H	7.0403e-04 (1)	7.8954e-03 (2)	5.9740e-03 (3)	2.8313e-01 (1)	-9.1201e-02 (2)	-2.4181e-02 (2)	11
		2.0920e-04 (4)	1.1042e-03 (3)	1.2920e-04 (4)	2.7285e-01 (1)	-1.3193e-01 (4)	-5.7042e-02 (1)	17
		1.0874e-04 (3)	9.0754e-04 (4)	1.8420e-04 (6)	2.5689e-01 (4)	-1.7761e-01 (2)	-6.7444e-02 (1)	20
A2	R	3.2129e-04 (8)	5.7171e-04 (7)	5.2451e-04 (7)	5.6116e-02 (3)	-6.4360e-03 (5)	3.3347e-03 (8)	38
		1.4121e-04 (8)	4.4859e-04 (9)	3.7022e-04 (8)	5.5901e-02 (6)	-9.7074e-03 (7)	-2.6475e-03 (8)	46
		9.9834e-05 (8)	3.6883e-04 (9)	2.7350e-04 (9)	5.5707e-02 (9)	-1.3097e-02 (7)	-6.3649e-03 (8)	50
	H	6.1530e-03 (9)	2.4016e-02 (7)	1.9667e-02 (8)	2.8808e-01 (3)	-5.0195e-02 (6)	2.6447e-02 (8)	41
		2.4779e-03	1.4653e-02	1.0548e-02	2.8673e-01	-7.1037e-02	-1.2323e-02	46

		(8)	(9)	(8)	(5)	(8)	(8)	
		1.7129e-03	9.0704e-03	5.2842e-03	2.8555e-01	-8.6727e-02	-3.3698e-02	50
		(8)	(9)	(9)	(8)	(8)	(8)	
A3	R	2.7326e-04	3.9968e-04	9.0004e-04	8.1564e-02	-3.0218e-03	2.2012e-03	41
		(7)	(2)	(10)	(9)	(7)	(6)	
		1.0398e-04	3.2456e-04	6.1022e-04	7.6854e-02	-1.3084e-02	-6.7830e-03	46
		(7)	(6)	(10)	(10)	(6)	(7)	
	H	9.8484e-05	3.0694e-04	5.2334e-04	7.4146e-02	-1.6847e-02	-1.0381e-02	44
		(7)	(8)	(10)	(10)	(6)	(3)	
A4	R	6.528e-05	4.522e-04	2.320e-04	7.457e-02	3.667e-03	-5.182e-03	27
		(2)	(5)	(2)	(6)	(9)	(3)	
		9.901e-05	1.422e-04	1.775e-05	5.364e-02	-7.643e-03	-9.790e-03	24
		(6)	(3)	(1)	(4)	(8)	(2)	
	H	9.541e-05	2.137e-05	6.402e-08	2.254e-02	-1.238e-02	-1.021e-02	22
		(6)	(1)	(1)	(2)	(8)	(4)	
A5	R	8.817e-04	1.058e-02	3.565e-03	6.603e-01	-2.238e-01	-2.242e-02	22
		(2)	(4)	(2)	(10)	(1)	(3)	
		1.116e-03	6.018e-04	2.800e-06	5.360e-01	-3.405e-01	-4.615e-02	25
	H	(7)	(2)	(1)	(10)	(1)	(4)	
		1.294e-03	7.885e-04	1.475e-09	4.231e-01	-3.991e-01	-5.158e-02	28
		(7)	(2)	(2)	(10)	(1)	(6)	
A5	R	5.972e-04	5.268e-04	4.487e-04	5.666e-02	-9.257e-03	-5.136e-03	32
		(10)	(6)	(4)	(4)	(4)	(4)	
		3.128e-04	1.769e-04	8.456e-05	5.648e-02	-1.738e-02	-6.983e-03	36
	H	(9)	(4)	(6)	(7)	(4)	(6)	
		1.010e-04	3.688e-05	3.028e-05	5.211e-02	-2.700e-02	-9.422e-03	34
		(9)	(3)	(7)	(7)	(4)	(5)	
A5	H	2.404e-02	2.206e-02	1.254e-02	2.919e-01	-5.312e-02	-8.135e-03	36
		(10)	(6)	(6)	(4)	(5)	(5)	
A5		4.069e-03	1.600e-03	2.395e-04	2.898e-01	-1.040e-01	-2.692e-02	37
		(9)	(4)	(6)	(6)	(5)	(7)	

		1.790e-03 (9)	3.998e-05 (3)	1.425e-03 (7)	2.694e-01 (6)	-1.579e-01 (5)	-4.815e-02 (7)	37
A6	R	2.0985e-04 (6)	4.3003e-04 (4)	4.8803e-04 (6)	8.0179e-02 (8)	-9.9919e-03 (3)	-3.3098e-03 (5)	32
		5.0877e-05 (4)	1.2273e-04 (2)	1.7951e-05 (2)	5.5117e-02 (5)	-2.5956e-02 (1)	-8.9123e-03 (3)	17
		6.0540e-06 (2)	2.1897e-05 (2)	3.4950e-07 (2)	1.6775e-03 (1)	-2.8504e-02 (1)	-9.2222e-03 (6)	14
	H	1.8318e-03 (5)	1.0864e-02 (5)	7.4275e-03 (4)	4.0540e-01 (7)	-7.1250e-02 (4)	-1.9533e-02 (4)	29
		1.1763e-04 (3)	5.5892e-04 (1)	2.9691e-06 (2)	2.8495e-01 (4)	-1.6072e-01 (2)	-5.6463e-02 (2)	14
		-6.8285e-06 (2)	8.5274e-06 (1)	2.2984e-09 (3)	5.5177e-03 (1)	-1.7461e-01 (3)	-5.9748e-02 (3)	13
A7	R	1.5777e-04 (4)	6.0730e-04 (8)	5.7379e-04 (8)	7.7326e-02 (7)	-4.8448e-03 (6)	2.6118e-03 (7)	40
		6.0514e-05 (5)	3.3387e-04 (7)	3.8328e-04 (9)	5.7269e-02 (8)	-1.4491e-02 (5)	-7.0635e-03 (5)	39
		6.6419e-05 (5)	2.7129e-04 (7)	1.8975e-04 (8)	5.4244e-02 (8)	-2.3331e-02 (5)	-1.1979e-02 (2)	35
	H	1.6057e-03 (3)	2.8615e-02 (9)	2.5215e-02 (9)	3.9009e-01 (6)	-3.9283e-02 (7)	2.1503e-02 (7)	41
		2.7736e-04 (5)	8.6156e-03 (7)	1.6016e-02 (9)	2.9321e-01 (7)	-8.8583e-02 (6)	-3.7428e-02 (6)	40
		2.9954e-04 (5)	4.2257e-03 (7)	2.9344e-03 (8)	2.7777e-01 (7)	-1.2888e-01 (6)	-6.7370e-02 (2)	35
A8	R	1.0316e-04 (3)	7.4496e-04 (9)	6.0347e-04 (9)	8.6235e-02 (10)	1.7493e-02 (10)	1.4920e-02 (10)	51
		-5.4515e-05 (1)	4.2021e-04 (8)	1.0731e-04 (7)	7.5350e-02 (9)	4.3046e-03 (10)	8.6810e-03 (10)	45
		-8.4712e-05 (1)	1.9330e-04 (6)	4.8313e-07 (3)	2.8693e-02 (3)	-2.6206e-04 (10)	3.6178e-03 (10)	33
	H	2.0669e-03 (6)	5.4997e-02 (10)	1.7672e-02 (7)	4.4487e-01 (9)	1.6701e-01 (10)	1.4371e-01 (10)	52
		5.5237e-05 (1)	8.9137e-03 (8)	1.6626e-04 (5)	3.9824e-01 (9)	6.2723e-02 (10)	8.9336e-02 (10)	43
		-2.0953e-05	1.5259e-03	-1.4816e-07	1.6826e-01	3.7447e-02	5.0744e-02	30

		(1)	(6)	(1)	(2)	(10)	(10)	
A9	R	1.8734e-04 (5)	2.2245e-03 (10)	3.3716e-04 (3)	5.5934e-02 (2)	-1.1773e-02 (2)	-6.7800e-03 (1)	23
		2.0590e-05 (3)	1.3952e-03 (10)	8.0801e-05 (5)	5.3434e-02 (3)	-2.4488e-02 (2)	-7.4822e-03 (4)	27
		3.2122e-05 (4)	6.1748e-04 (10)	1.4618e-05 (5)	3.7772e-02 (4)	-2.7515e-02 (2)	-7.8231e-03 (7)	32
	H	3.3835e-03 (7)	2.4995e-02 (8)	8.2174e-03 (5)	2.8646e-01 (2)	-8.7677e-02 (3)	-4.1769e-02 (1)	26
		9.1415e-05 (2)	1.9169e-02 (10)	3.0148e-04 (7)	2.7296e-01 (2)	-1.5575e-01 (3)	-5.2631e-02 (3)	27
		1.1109e-04 (4)	1.3482e-02 (10)	1.9128e-06 (4)	1.9554e-01 (3)	-1.7056e-01 (4)	-5.3982e-02 (5)	30
A10	R	5.2595e-04 (9)	4.1831e-04 (3)	4.5649e-04 (5)	5.8193e-02 (5)	4.5510e-06 (8)	4.3283e-03 (9)	39
		3.5464e-04 (10)	2.6111e-04 (5)	4.1130e-05 (3)	5.2442e-02 (1)	2.8521e-03 (9)	-1.6688e-04 (9)	37
		5.9881e-04 (10)	1.8259e-04 (5)	2.1800e-05 (6)	4.7466e-02 (5)	-8.7045e-03 (9)	-3.5148e-03 (9)	44
	H	3.8262e-03 (8)	5.4614e-03 (1)	2.0587e-03 (1)	3.0626e-01 (5)	-4.3766e-04 (9)	4.5102e-02 (9)	33
		4.3679e-03 (10)	2.1873e-03 (5)	8.0792e-05 (3)	2.7710e-01 (3)	1.0389e-02 (9)	1.6912e-02 (9)	39
		7.4776e-03 (10)	9.6089e-04 (5)	6.4794e-05 (5)	2.6029e-01 (5)	-5.1440e-02 (9)	-1.3468e-02 (9)	43

Algorithms: A1- 2LB-MOPSO, A2- MOCLPSO, A3- MO_PSO, A4- NSGA2_SBX, A5- NSGA2_PCX, A6- GDE3, A7- MO_DE, A8- MOSaDE, A9- DEMOWSA, A10- MTS.

Table 5. 6 The Rank of mean value for R & H indicators from 25 runs on test problems 14-19, $M=5$.

		14.S_DTLZ2	15.R_DTLZ2	16.S_DTLZ3	17.WFG1	18.WFG8	19.WFG9	Rank
A1	R	5.4529e-05 (2)	1.2536e-04 (5)	8.3365e-05 (3)	4.6188e-02 (2)	3.7450e-03 (4)	4.8164e-03 (3)	19
		1.6134e-05 (2)	6.6551e-05 (4)	2.6002e-05 (5)	4.5282e-02 (2)	-2.1381e-03 (3)	1.3512e-03 (3)	19

	H	2.0549e-05 (4)	4.8766e-05 (4)	3.0066e-05 (6)	4.4579e-02 (5)	-5.7425e-03 (4)	7.4545e-04 (3)	26
		1.9994e-04 (2)	1.8542e-03 (5)	1.5256e-03 (3)	5.2233e-01 (1)	-8.0651e-02 (4)	5.5085e-02 (3)	18
		2.8296e-05 (3)	4.2756e-04 (3)	3.7325e-04 (5)	5.1333e-01 (2)	-1.8624e-01 (4)	-9.3983e-02 (4)	21
		3.2575e-05 (5)	3.0897e-04 (4)	4.9858e-04 (6)	5.0677e-01 (4)	-2.7593e-01 (3)	-9.4519e-02 (4)	26
A2	R	1.5907e-04 (4)	1.3783e-04 (6)	2.0891e-04 (5)	4.6995e-02 (3)	6.7409e-03 (6)	7.9943e-03 (7)	31
		9.9432e-05 (6)	1.3242e-04 (8)	1.3830e-04 (7)	4.6813e-02 (5)	4.3560e-03 (8)	2.7489e-03 (6)	40
		7.7412e-05 (7)	1.1969e-04 (9)	9.9511e-05 (7)	4.6624e-02 (9)	1.8882e-03 (9)	8.4349e-04 (4)	45
	H	2.2024e-03 (4)	2.4695e-03 (6)	5.1083e-03 (6)	5.2940e-01 (2)	-4.0853e-02 (6)	1.0031e-01 (5)	29
		6.9101e-04 (7)	2.3465e-03 (8)	2.2355e-03 (7)	5.2770e-01 (5)	-1.0290e-01 (7)	-7.3761e-03 (6)	40
		4.6728e-04 (6)	2.3260e-03 (8)	1.3292e-03 (7)	5.2612e-01 (8)	-1.4090e-01 (8)	-8.7113e-02 (5)	42
A3	R	2.8395e-04 (9)	1.0172e-04 (4)	4.0586e-04 (10)	5.7287e-02 (8)	1.2396e-02 (9)	1.4515e-02 (10)	50
		1.4434e-04 (9)	8.0349e-05 (6)	2.7612e-04 (9)	5.4525e-02 (9)	4.5919e-03 (9)	4.7651e-03 (8)	50
		1.2707e-04 (10)	1.0349e-04 (8)	2.3567e-04 (9)	5.3081e-02 (10)	-3.3998e-03 (6)	2.5741e-03 (8)	51
	H	3.5268e-03 (7)	6.9792e-04 (4)	1.1336e-02 (10)	6.4317e-01 (8)	6.2030e-02 (9)	1.6483e-01 (8)	46
		6.1777e-04 (6)	7.0216e-04 (5)	7.9362e-03 (9)	6.1329e-01 (9)	-7.5609e-02 (8)	5.1445e-03 (8)	45
		6.6857e-04 (7)	1.6636e-03 (7)	9.4070e-03 (8)	5.9822e-01 (10)	-1.8899e-01 (6)	-4.6243e-02 (7)	45
A4	R	2.352e-06 (1)	2.134e-05 (1)	5.304e-06 (1)	6.398e-02 (10)	2.796e-03 (2)	1.964e-03 (2)	17
		-9.508e-06 (1)	-5.551e-06 (1)	-1.432e-08 (1)	4.955e-02 (8)	-7.838e-03 (1)	-2.089e-03 (1)	13
		2.811e-06	-1.183e-05	-1.473e-08	3.694e-02	-1.181e-02	-2.253e-03	9

	H	(1)	(1)	(1)	(4)	(1)	(1)	
		3.752e-5 (1)	1.878e-04 (1)	2.507e-05 (1)	7.025e-01 (10)	-2.120e-01 (1)	-1.573e-01 (2)	16
		-6.518e-06 (2)	-1.557e-04 (1)	5.859e-11 (1)	5.629e-01 (8)	-3.498e-01 (1)	-2.124e-01 (1)	14
		5.535e-06 (3)	-1.830e-04 (1)	2.042e-16 (1)	4.381e-01 (2)	-4.165e-01 (1)	-2.154e-01 (1)	9
A5	R	2.002e-04 (6)	7.245e-05 (3)	4.553e-05 (2)	4.777e-02 (5)	3.001e-03 (3)	-7.903e-04 (1)	20
		2.587e-04 (10)	6.924e-05 (5)	2.876e-04 (10)	4.731e-02 (7)	-6.877e-04 (5)	-9.847e-04 (2)	39
		1.250e-04 (9)	7.799e-05 (6)	3.908e-04 (10)	4.462e-02 (6)	-1.655e-03 (7)	-8.432e-04 (2)	40
	H	4.126e-03 (9)	8.342e-04 (2)	1.364e-03 (2)	5.371e-01 (5)	-2.010e-01 (2)	-1.918e-01 (1)	21
		4.610e-03 (10)	8.917e-04 (6)	2.518e-02 (10)	5.327e-01 (7)	-2.592e-01 (3)	-1.940e-01 (2)	38
		9.661e-04 (8)	8.260e-04 (6)	4.353e-02 (10)	5.068e-01 (5)	-2.677e-01 (4)	-1.919e-02 (8)	41
A6	R	2.0537e-04 (7)	1.9155e-04 (7)	2.2656e-04 (6)	5.6468e-02 (7)	5.8783e-03 (5)	7.0251e-03 (5)	37
		3.5491e-05 (4)	5.4718e-05 (3)	1.3649e-05 (3)	4.5536e-02 (3)	-7.5371e-03 (2)	2.4667e-03 (5)	20
		2.1511e-05 (5)	3.3115e-05 (3)	2.5426e-07 (2)	4.6089e-03 (1)	-1.1345e-02 (2)	1.9398e-03 (6)	19
	H	1.7376e-03 (3)	3.1397e-03 (7)	2.6169e-03 (5)	6.3496e-01 (7)	-4.2629e-02 (5)	5.6182e-02 (4)	31
		3.7278e-05 (4)	-1.7555e-06 (2)	7.6879e-06 (2)	5.2167e-01 (4)	-2.6760e-01 (2)	-1.0011e-01 (3)	17
		1.0020e-05 (4)	-1.3436e-04 (2)	4.9525e-10 (2)	5.2961e-02 (9)	-3.4706e-01 (2)	-1.1315e-01 (2)	21
A7	R	2.4338e-04 (8)	2.0670e-04 (8)	2.7012e-04 (8)	5.2164e-02 (6)	6.7938e-03 (7)	7.9079e-03 (6)	43
		1.1276e-04 (8)	1.3848e-04 (9)	1.3910e-04 (8)	4.6042e-02 (4)	-2.1333e-03 (4)	2.2364e-03 (4)	37
		1.0325e-04 (8)	9.7070e-05 (7)	1.5829e-04 (8)	4.5384e-02 (7)	-6.4978e-03 (3)	9.2791e-04 (5)	38

	H	3.0647e-03 (6)	7.0107e-03 (8)	9.0693e-03 (8)	5.9089e-01 (6)	-6.5829e-03 (8)	1.0556e-01 (6)	42
		7.2848e-04 (8)	4.2420e-03 (9)	3.2433e-03 (8)	5.2128e-01 (3)	-1.5128e-01 (6)	-3.8637e-02 (7)	41
		1.2504e-03 (9)	3.1872e-03 (9)	9.5658e-03 (9)	5.1433e-01 (6)	-1.6791e-01 (7)	-8.0591e-02 (6)	46
A8	R	1.9650e-04 (5)	2.3688e-04 (9)	2.7026e-04 (9)	5.9179e-02 (9)	2.1266e-02 (10)	1.1421e-02 (8)	50
		3.7089e-05 (5)	1.2100e-04 (7)	4.2957e-05 (6)	5.5157e-02 (10)	1.8656e-02 (10)	7.6963e-03 (10)	48
		1.5043e-05 (3)	7.7357e-05 (5)	8.9693e-06 (5)	1.9670e-02 (3)	9.4860e-03 (10)	7.2382e-03 (10)	36
	H	2.2131e-03 (5)	8.9669e-03 (9)	7.2009E-03 (7)	6.7337e-01 (9)	4.3299e-01 (10)	1.9572e-01 (9)	49
		-1.9121e-04 (1)	1.1850e-03 (7)	1.3146e-04 (6)	6.3612e-01 (10)	3.1404e-01 (10)	1.3173e-01 (9)	43
		-2.5991e-04 (1)	4.1870e-04 (5)	1.6230e-06 (4)	2.7218e-01 (1)	1.5836e-01 (9)	1.0625e-01 (10)	30
A9	R	4.4348e-04 (10)	7.0549e-04 (10)	2.5939e-04 (7)	4.7182e-02 (4)	9.3102e-03 (8)	1.2250e-02 (9)	48
		2.7842e-05 (3)	2.5883e-04 (10)	2.4993e-05 (4)	4.6881e-02 (6)	3.0397e-03 (7)	3.0019e-03 (7)	37
		7.8107e-06 (2)	1.3375e-04 (10)	6.9797e-06 (4)	4.5889e-02 (8)	-3.9639e-03 (5)	2.5642e-03 (7)	36
	H	1.5551e-02 (10)	2.1254e-02 (10)	9.0716e-03 (9)	5.3115e-01 (3)	-2.7957e-02 (7)	1.3861e-01 (7)	46
		4.9841e-05 (5)	1.3054e-02 (10)	5.1144e-05 (4)	5.2895e-01 (6)	-1.7012e-01 (5)	-9.3784e-02 (5)	35
		2.0109e-06 (2)	9.8835e-03 (10)	8.0469e-07 (3)	5.2053e-01 (7)	-2.5771e-01 (5)	-1.2020e-01 (3)	30
A10	R	1.3550e-04 (3)	6.7509e-05 (2)	9.2207e-05 (4)	2.3255e-02 (1)	2.3668e-03 (1)	6.9322e-03 (4)	15
		1.0010e-04 (7)	4.1210e-05 (2)	8.1611e-06 (2)	2.0744e-02 (1)	1.1614e-03 (6)	4.7799e-03 (9)	27
		7.5100e-05 (6)	2.5831e-05 (2)	4.5948e-07 (3)	1.8953e-02 (2)	-5.0574e-06 (8)	3.5798e-03 (9)	30
	H	3.9290e-03	8.9860e-04	1.9106e-03	5.3158e-01	9.9301e-02	2.2328e-01	32

		(8)	(3)	(4)	(4)	(3)	(10)	
		2.9544e-03 (9)	4.3716e-04 (4)	5.0563e-05 (3)	4.8225e-01 (1)	1.5081e-01 (9)	2.0877e-01 (10)	36
		2.5226e-03 (10)	2.5537e-04 (3)	2.1745e-06 (5)	4.4656e-01 (3)	1.6656e-01 (10)	5.8660e-02 (9)	40

Algorithms: A1- 2LB-MOPSO, A2- MOCLPSO, A3- MO_PSO, A4- NSGA2_SBX, A5- NSGA2_PCX, A6- GDE3, A7- MO_DE, A8- MOSaDE, A9- DEMOWSA, A10- MTS.

Table 5. 7 Summation of ranks based on R & H indicators on all test problems.

Algorithm	FEs=5000 ($R+H$)			FEs=50000 ($R+H$)			FEs=500000 ($R+H$)			Overall Summation		
	M=2	M=3	M=5	M=2	M=3	M=5	M=2	M=3	M=5	M=2	M=3	M=5
A1	22 (1)	18 (1)	37 (2)	40 (2)	30 (1)	40 (3)	52 (2)	42 (2)	52 (3)	114 (1)	90 (1)	129 (2)
A2	120 (10)	79 (7)	60 (5)	118 (10)	92 (10)	80 (8)	129 (10)	100 (10)	87 (9)	369 (10)	271 (10)	227 (6)
A3	84 (6)	80 (8)	96 (9)	106 (9)	88 (9)	95 (10)	111 (9)	88 (9)	96 (10)	301 (9)	256 (9)	287 (10)
A4	35 (2)	49 (3)	33 (1)	42 (3)	49 (3)	27 (1)	63 (4)	50 (3)	18 (1)	140 (2)	148 (3)	78 (1)
A5	94 (8)	68 (5)	41 (3)	93 (6)	73 (5)	77 (6)	71 (6)	71 (7)	81 (7)	258 (6)	212 (5)	199 (5)
A6	77 (5)	61 (4)	68 (6)	32 (1)	31 (2)	37 (2)	46 (1)	27 (1)	40 (2)	155 (3)	119 (2)	145 (3)
A7	93 (7)	81 (9)	85 (7)	98 (8)	79 (7)	78 (7)	78 (7)	70 (6)	84 (8)	269 (7)	230 (6)	247 (8)
A8	75 (4)	103 (10)	99 (10)	59 (4)	88 (8)	91 (9)	58 (3)	63 (5)	66 (4)	192 (4)	254 (8)	256 (9)
A9	108 (9)	49 (2)	94 (8)	97 (7)	54 (4)	72 (5)	65 (5)	62 (4)	66 (5)	270 (8)	165 (4)	232 (7)
A10	61 (3)	72 (6)	47 (4)	91 (5)	76 (6)	63 (4)	99 (8)	87 (8)	70 (6)	251 (5)	235 (7)	180 (4)

Algorithms: A1- 2LB-MOPSO, A2- MOCLPSO, A3- MO_PSO, A4- NSGA2_SBX, A5- NSGA2_PCX, A6- GDE3, A7- MO_DE, A8- MOSaDE, A9- DEMOWSA, A10- MTS.

5.5.3 Comparison among 2LB-MOPSO Variants

In order to demonstrate the benefits of selecting the *gbest* or *lbest* from a neighborhood in both parameter and objective spaces of a non-domination sorted external archive of large size, as well as the advantage to retaining the *gbest* or *lbest* of every particle in the same neighborhood for a few iterations, further experiments are conducted on three modified versions of the proposed 2LB-MOPSO.

1. 2LB-MOPSO_M1: Two *Lbests* based MOPSO with *count*=1 as all current MOPSO variants in the literature. Selecting the two *lbests* from two random bins, and the archive only includes non-dominated front 1 solutions. This version is similar to most MOPSO variants except having two *lbests* instead of one *gbest* and one *pbest* for each particle.
2. 2LB-MOPSO_M2: Two *Lbests* based MOPSO with *count*=1. Selecting the two *lbests* from two random bins of the non-domination sorted external archive of a reasonably large size. This version can show the benefit of non-domination sorted external archive with a reasonably large size when comparing to 2LB-MOPSO_M1 with only front 1 solutions in the external archive.
3. 2LB-MOPSO_M3: Two *Lbests* based MOPSO with *count*=1. The two *lbests* are selected from two neighboring bins of the non-domination sorted external archive. This version can show the benefit of selecting the *gbest* or *lbest* from the external archive within a neighborhood in both parameter space and objective space. When comparing our proposed 2LB-MOPSO to the 2LB-MOPSO_M3, we can observe the advantage of retaining the *gbest* or *lbest* of every particle in the same neighborhood for a few reasonable iterations as determined by the *count* parameter.

We compare mean values of 25 runs for *R* indicator achieved by the three modified versions of 2LB-MOPSO and our proposed 2LB-MOPSO at $5e+5$ function evaluations on six problems, which are No. 2 SYMPART (two-objectives), No. 7 S_ZDT6 (two-objectives), No. 9 R_DTLZ2 (three-objectives), No. 10 S_DTLZ3 (three-objectives), No. 18 WFG8 (five-objectives) and No. 19 WFG9 (five-objectives). Table 5.8 ranks these four different versions based on the mean values of *R* indicator and add the rankings together as presented in the last column. From the summation of rankings in the last column of Table 5.8, improvements due to the three contributions in our proposed 2LB-MOPSO can be easily

observed. Hence, it is obvious that each of our novel contributions plays a clear role in making our proposed 2LB-MOPSO to perform substantially better than the state-of-the-art MOPSO variants.

Table 5. 8 R _indicator obtained by the variants of 2LB-MOPSO at 500k function evaluations

Variants	2.SYMPART	7.S_ZDT6	9.R_DTLZ2	10.S_DTLZ3	18.WFG8	19.WFG9	Rank
2LB-MOPSO_M1	1.1222e-04 (4)	6.4552e-02 (4)	3.8002e-04 (4)	2.1906e-04 (4)	1.2664e-03 (4)	1.8550e-03 (4)	24
2LB-MOPSO_M2	6.8550e-05 (2)	2.8462e-03 (3)	1.6950e-04 (1)	9.0054e-05 (3)	-4.1225e-03 (3)	1.7650e-03 (3)	15
2LB-MOPSO_M3	7.2885e-05(3)	2.3142e-03 (2)	1.7703e-04 (2)	8.1804e-05 (2)	-8.6358e-03 (1)	1.3703e-03 (2)	12
2LB-MOPSO	1.3453e-06 (1)	3.5901e-04 (1)	1.8229e-04 (3)	1.1170e-05 (1)	-5.7425e-03 (2)	7.4545e-04 (1)	9

5.5.4 Comparison with SMM

We perform experimental comparison to highlight the performance differences between SMM and 2LB-MOPSO algorithms. Four test problems from [306] are considered in this comparison. The first function is a two-objective minimizing problem with a discontinuous Pareto optimal front. The second problem is the modeling for the design of welded Beam, which is two-objective minimization problem. The third problem designs a multiple disc brake, which is a two-objective minimizing problem. The last problem deals with the designing of a Four Bar Truss which is a two-objective minimizing problem. All problems' definitions are given in Appendix D. As some of these problems are constrained, we apply the epsilon-constraint handling method presented [319] to solve them. To compare the SMM and the proposed 2LB-MOPSO, the population size, the number of Pareto optimal solutions found and the function evaluations required are shown in Table 5.9. As 100 is used as the archive size for two-objective problems, 2LB-MOPSO easily achieves full size of the Pareto optimal solutions by using a population size of 50 and smaller number of function evaluations. However, the SMM fails to evolve the whole population to be the front 1

solutions. We also show the obtained front 1 curves in Figure 5.4 in which for ease of comparison, we reproduce the front 1 curves published in [306]. Our proposed 2LB-MOPSO can obtain better quality front 1 curves with good spread, while the front 1 curves presented in [306] lose diversity frequently. For the first problem, our 2LB-MOPSO can obtain four discontinuous front 1 curves with good spreads, while the SMM finds 40 front 1 solutions out of 100 population with almost 6 times function evaluations of ours. For the second problem, with less than half of function evaluations used in SMM, 2LB-MOPSO is able to find required number of front 1 solutions, while SMM find only 52 feasible solutions. For the third problem, we can easily find that 2LB-MOPSO can achieve more feasible solutions with better diversity than SMM by using much smaller number of function evaluations. For the last problem, 2LB-MOPSO obtains totally dominating solutions when comparing to the SMM with only 0.2 times the function evaluations used in SMM.

Table 5.9 Comparison between SMM and 2LB-MOPSO

Problem	Population size		# of Front 1 solution		Max function evaluations	
	SMM	2LB-MOPSO	SMM	2LB-MOPSO	SMM	2LB-MOPSO
1	100	50	40	100	29,572	5,000
2	100	50	42	100	18,389	8,000
3	100	50	52	100	6385	2,000
4	100	50	91	100	2525	1,000

Problem 1

Problem 2

Figure 5. 4 Pareto fronts obtained by the 2LB-MOPSO and SMM (Vertical and Horizontal axes repret two objectives of the four problems)

5.6 Multi-objective Robust PID Controller Tuning using 2LB-MOPSO

5.6.1 Introduction on Robust PID Controller Tuning

More than 90% of industrial controllers are still implemented based on PID control algorithms as no other controller matches the simplicity, clear functionality and ease of implementation. Several approaches have been reported in literature for tuning the parameters of PID controllers. Ziegler-Nichols and Cohen-Coon are the most commonly used conventional methods for tuning PID controllers [353], [354]. A recent PID control survey [355], suggested the need for an efficient tuning method for multiple design objectives. Designing a PID controller to satisfy multiple objectives is an active research area [356]-[370]. Time domain performance [357], [362], [363], [366], frequency domain performance [366] and robust performance criterion [356], [359], [360], [368], [367] based designs of PID controllers are reported in the literature.

Robustness is an important criterion in controller design because most of the real systems are vulnerable to external disturbance, measurement noise and model uncertainty [371],

[372]. There are two approaches to deal with robust optimal controller design problems. One is the structure specified controller [373]-[378] and the other is the output feedback controller [379]. Since the order of the robust output feedback controller is much higher than that of the system, it is not easy to implement the controller for higher order systems in practical engineering applications. To overcome this difficulty, the structure specified approach solves the robust optimal control problem from the suboptimal perspective [373].

Many researchers have employed mixed H_2/H_∞ optimal design strategy for optimal robust PID controller [373]-[378], [380]. In [373], authors propose the design of simple genetic algorithm (GA) based mixed H_2/H_∞ controller for SISO system by minimizing ISE (H_2 norm) subjected to robust stability and disturbance attenuation constraints. GA is adopted to design structure specified H_∞ optimal controllers by minimizing the balanced robust performance criteria which combine the robust stability and disturbance attenuation performance measures, for practical applications [374]. But, their procedure requires prior domain knowledge, i.e., the Routh-Hurwitz criterion for reducing the size of the domain of every design parameter. In [377], authors suggest design of two-stage GA based optimal disturbance rejection PID controller for a servo motor system. In the first stage, GA is applied to determine H_∞ disturbance rejection value and in the second stage, GA is used to minimize ITSE performance index subjected to the disturbance rejection constraint. The orthogonal simulated annealing algorithm (OSA) is applied to design the optimal robust PID controller for MIMO system by minimizing combined ISE and balanced robust performance criteria [375]. An intelligent genetic algorithm (IGA) is proposed to the design of mixed H_2/H_∞ optimal robust PID controller [376]. In all of the above mentioned papers [373]-[378], the mixed H_2/H_∞ optimal robust PID controller design problem was solved using single objective optimization algorithms by combining different objectives or by treating some objectives as constraints.

The design of multi-objective robust PID controller using various multi-objective optimization algorithms is proposed in [356], [359], [360], [368], [367]. In [359], considering load disturbance rejection, robustness to model uncertainty and set point tracking as objectives and actuator limit as a constraint, Multi-objective Robust Control Design (MRCD) genetic algorithm is used to solve the design problem. However, the weighting function for the design of robust controller and also robust stability and disturbance attenuation constraints are not considered. In [356], [368], [367], the multi-

objective robust PID controller design problem is formulated as a solution of linear matrix inequalities. Recently, an intelligent multi-objective simulated annealing algorithm was proposed for designing robust PID controllers to satisfy disturbance attenuation, robust stability and set point tracking [360]. The problem is treated as a three objective optimization problem. But, for performance analysis and ease of visualization of the Pareto front, disturbance attenuation and robust stability performance measures are combined into a balanced robust performance criterion. Pareto front is drawn between set point tracking and balanced robust performance criteria [381], [382].

In section 5.6, proposed 2LB-MOPSO algorithm is applied for the design of robust PID controller by minimizing ISE and balanced robust performance criterion for robust performance. For the design of optimal robust PID controller, two different systems, namely MIMO distillation column model [376], [383] and MIMO super maneuverable F18/HARV fighter aircraft system [374], [360], [375], [376] are considered. The two systems included in this work have more than one objective conflicting with each other. Our aim is to find Pareto optimal trade-off solutions representing the best possible compromises among all objectives. The 2LB-MOPSO is developed taking into consideration two common goals, namely fast convergence to the Pareto front and good distribution of solutions along the front. The performance of 2LB-MOPSO is compared with that of NSGA-II [295] by using the Hypervolume indicator [384].

5.6.2 Multi-objective Robust PID Controller

Consider a control system with n_i inputs and n_o outputs, as shown in Figure 5.5, where $P(s)$ is the nominal plant, $\Delta P(s)$ is the plant perturbation, $G_c(s)$ is the controller, $r(t)$ is the reference input, $u(t)$ is the control input, $e(t)$ is the tracking error, $d(t)$ is the external disturbance, and $y(t)$ is the output of the system [376].

Figure 5.5 Control system with plant perturbation and external disturbance

Without loss of generality, the plant perturbation $\Delta P(s)$ is assumed to be bounded by a known stable function matrix $W_1(s)$

$$\bar{\sigma}(\Delta P(j\omega)) \leq \bar{\sigma}(W_1(j\omega)), \forall \omega \in [0, \infty) \quad (5.3)$$

where $\bar{\sigma}(A)$ denotes the maximum singular value of a matrix A .

5.6.2.1 Robust PID Controller

If a controller $G_c(s)$ is designed in such a way that the nominal closed loop system ($\Delta P(s) = 0$ and $d(t) = 0$) is asymptotically stable, the robust stability performance satisfies the following inequality

$$J_a = \|W_1(s)T(s)\|_\infty < 1 \quad (5.4)$$

and the disturbance attenuation performance satisfies the following inequality

$$J_b = \|W_2(s)S(s)\|_\infty < 1 \quad (5.5)$$

Then the closed loop system is also asymptotically stable with $\Delta P(s)$ and $d(t)$, where $W_2(s)$ is a stable weighting function matrix specified by designers. $S(s)$ and $T(s) = I - S(s)$ are the sensitivity and complementary sensitivity functions of the system, respectively

$$S(s) = (I + P(s)G_c(s))^{-1} \quad (5.6)$$

$$T(s) = P(s)G_c(s)(I + P(s)G_c(s))^{-1} \quad (5.7)$$

H_∞ -norm in (5.4) and (5.5) is defined as

$$\|A(s)\|_\infty \equiv \max_{\omega} \bar{\sigma}(A(j\omega)) \quad (5.8)$$

where $\bar{\sigma}(A)$ denotes the maximum singular value of a matrix A . Robust controller design problem lies in how to find a controller to stabilize the closed loop system and attenuate the external disturbance. The principal criterion for this problem is simultaneously minimizing the robust stability performance J_a and disturbance attenuation performance J_b [374]. Minimization of balanced performance criterion $J_\infty = (J_a^2 + J_b^2)^{1/2}$ [376] is considered to minimize both J_a and J_b simultaneously. For improving the system performance, robust stability and disturbance attenuation together are often not enough in the control system design. Optimal tracking performance is also an appealing factor in many practical control engineering applications [373]. The most commonly used performance measure is the ISE because of its easy implementation and it being mathematically convenient both for analysis and computation. Minimization of ISE is a good compromise among reduction of rise time to limit the effect of large initial error, reduction of peak overshoot and reduction of settling time to limit the effect of small error lasting for a long time [385]. The minimization of tracking error ISE (J_2 i.e., H_2 norm) should be taken into account

$$J_2 = \int_0^\infty e^T(t)e(t)dt \quad (5.9)$$

where $e(t) = r(t) - y(t)$ is the error determined with $\Delta P(s) = 0$ and $d(t) = 0$.

In designing optimal robust controllers with fixed structures, both the tracking behavior and robust stability are considered [386]. In this work, the PID controller is designed for obtaining better tracking performance by minimizing ISE i.e. J_2 and better robust performance against model uncertainty and disturbance by minimizing balanced robust performance criterion J_∞ . Both the objectives J_2 and J_∞ are minimized simultaneously subjected to the robust stability and disturbance attenuation constraints by using multi-

objective optimization algorithm 2LB-MOPSO. Thus, the problem of multiobjective robust PID controller has been defined as follows:

$$\text{Minimize } J = \{J_2, J_\infty\} \quad (5.10)$$

subjected to robust stability and disturbance attenuation constraints as defined in (5.4) and (5.5).

5.6.2.2 Robust PID Controller Structure

In general, transfer function of a structure-specified controller can be expressed as follows:

$$G_c(s) = \frac{N_c(s)}{D_c(s)} = \frac{B_m s^m + B_{m-1} s^{m-1} + \dots + B_0}{s^n + a_{n-1} s^{n-1} + \dots + a_0} \quad (5.11)$$

where m and n are fixed orders of $N_c(s)$ and $D_c(s)$ respectively. Also B_k is defined in (5.12).

$$B_k = \begin{bmatrix} b_{k11} & \dots & b_{k1n_i} \\ \vdots & \ddots & \vdots \\ b_{kn_01} & \dots & b_{kn_0n_i} \end{bmatrix} \quad \text{for } k = 0, 1, \dots, m. \quad (5.12)$$

where n_i and n_o are the number of inputs and outputs respectively.

Most of the conventional controllers used in industrial control systems have fundamental structures such as PID and lead/lag configurations. Such controllers are special cases of the structure-specified controllers. For the PID controller, we have $n = 1, m = 2$ and $a_0 = 0$, i.e.

$$G_c(s) = \frac{B_2 s^2 + B_1 s + B_0}{s} \quad (5.13)$$

In case of the system with complicated plant and a single simple-structure controller, one seems to have less freedom of controller parameters to tune the system to achieve the H_∞ optimal design objectives. In this situation, we must increase the order m and n of the controller in (5.11).

Recently, several MOPSO techniques are applied to PID controller design. In [387], a novel variant of a multi-objective particle swarm optimization algorithm is proposed to the design of PID controllers taking into account the classical design objectives of set-point tracking and output disturbance rejection. In order to overcome the drawbacks of current PID design methods for main steam temperature control of boiler unit in a thermal power plant, a new method is proposed for multi-objective PID controller to ensure reliability and robustness in its performance. Adaptively weighted PSO (AWPSO) technique is applied to the parameter optimization design [388]. In this work, we employ 2LB-MOPSO variant.

5.6.3 Test Systems

In order to validate the performance of the proposed 2LB-MOPSO based multi-objective robust PID optimal controller, two different MIMO systems, namely distillation column model (Test System-I) and super maneuverable F18/HARV fighter aircraft system (Test System-II) are considered and the details are given below.

5.6.3.1 Test System-I

We consider the methanol-ethanol distillation column by Chien and Ogunnaike [389]. In [376], [383], robust PID controller was designed for this highly coupled distillation column model. In this system, tray temperatures are regulated, as inferential variables for composition, by manipulating liquid reflux and vapor boilup rates. The transfer function of the plant is given as follows:

$$P(s) = \begin{bmatrix} \frac{-33.98}{(98.02s+1)(0.42s+1)} & \frac{32.63}{(99.6s+1)(0.35s+1)} \\ \frac{-18.85}{(75.43s+1)(0.30s+1)} & \frac{34.84}{(110.5s+1)(0.03s+1)} \end{bmatrix} \quad (5.14)$$

where the outputs 1 and 2 are temperatures of tray 21 and tray 7 and inputs 1 and 2 are the liquid reflux and vapor boilup rates, respectively.

The plant uncertainties $\Delta P(s)$ are unknown but bounded by the following stable function:

$$W_1(s) = \frac{100s+1}{s+1000} I_{2 \times 2} \quad (5.15)$$

and for attenuate disturbance, a weighting function $W_2(s)$ consisting of low-pass filter

$$W_2(s) = \frac{s+1000}{1000s+1} I_{2 \times 2} \quad (5.16)$$

is considered.

5.6.3.2 Test System-II

We consider the longitudinal control system of the super maneuverable F18/HARV fighter aircraft [374]-[376], [360] in horizontal flight at an altitude of 15,000 (ft) with Mach number 0.24, airspeed $V_T = 238.7$ (ft/s), attack angle $\alpha = 25$ (deg), and pitch angle $\beta = 25$ (deg). The trim value of the path angle is $\beta - \alpha = 0$ (deg) and the trim pitch rate is $q = 0$ (deg/s). The longitudinal dynamics of the system can be described as

$$\begin{aligned} \frac{dx}{dt} &= Ax + Bu \\ y &= Cx \end{aligned} \quad (5.17)$$

where A, B and C are given as

$$A = \begin{bmatrix} -0.0750 & -24.0500 & 0 & -36.1600 \\ -0.0009 & -0.1959 & 0.9896 & 0 \\ -0.0002 & -0.1454 & -0.1677 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (5.18)$$

$$B = \begin{bmatrix} -0.0230 & 0 & -0.0729 & 0.0393 & -0.0411 & 0.1600 \\ -0.0002 & -0.0001 & -0.0004 & 0 & -0.0003 & -0.0003 \\ -0.0067 & -0.0007 & -0.0120 & 0.0007 & 0.0005 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (5.19)$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.20)$$

$$x = [V_T \quad \alpha \quad q \quad \beta]^T \quad (5.21)$$

$$u = [u_{TV} \quad u_{AS} \quad u_{SS} \quad u_{LE} \quad u_{TE} \quad u_T]^T \quad (5.22)$$

where u_{TV} , u_{AS} , u_{SS} , u_{LE} , u_{TE} , u_T are the perturbations in symmetric thrust vectoring vane deflection, symmetric aileron deflection, symmetric stabilator deflection, symmetric leading edge flap deflection, symmetric trailing edge flap deflection, and throttle position, respectively. Note that the rank of matrix B is only three. By employing the pseudo-control technique, the six control inputs (u_{TV} , u_{AS} , u_{SS} , u_{LE} , u_{TE} , u_T) are transformed into three linearly independent variables. Therefore, the system can be rewritten as,

$$\frac{dx}{dt} = Ax + B_v v \quad (5.23)$$

where B_v and v are given as,

$$B_v = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad (5.24)$$

$$v = \begin{bmatrix} -0.0230 & 0 & -0.0729 & 0.0393 & -0.0411 & 0.1600 \\ -0.0002 & -0.0001 & -0.0004 & 0 & -0.0003 & -0.0003 \\ -0.0067 & -0.0007 & -0.0120 & -0.0006 & 0.0007 & 0.0005 \end{bmatrix} u \quad (5.25)$$

The bound $W_1(s)$ for the plant perturbation $\Delta P(s)$

$$W_1(s) = \frac{0.0125s^2 + 1.2025s + 1.25}{s^2 + 20s + 100} I_{3 \times 3} \quad (5.26)$$

and for the attenuate disturbance, a weighting function $W_2(s)$ consisting of low-pass filter

$$W_2(s) = \frac{0.25s + 0.025}{s^2 + 0.4s + 10000000} I_{3 \times 3} \quad (5.27)$$

are considered.

5.6.4 Implementation of Multi-Objective PID Controller

5.6.4.1 Test System-I

For the highly coupled distillation column MIMO model, the PID controller structure is given in (5.13), where

$$\mathbf{B}_2 = \begin{bmatrix} b_{211} & b_{212} \\ b_{221} & b_{222} \end{bmatrix}, \mathbf{B}_1 = \begin{bmatrix} b_{111} & b_{112} \\ b_{121} & b_{122} \end{bmatrix}, \mathbf{B}_0 = \begin{bmatrix} b_{011} & b_{012} \\ b_{021} & b_{022} \end{bmatrix} \quad (5.28)$$

The design variables for this system are

$$X_D = [b_{211}, b_{212}, b_{221}, b_{222}, b_{111}, b_{112}, b_{121}, b_{122}, b_{011}, b_{012}, b_{021}, b_{022}] \quad (5.29)$$

with the lower and upper bounds of the design variables are taken as $-200 \leq X_{Di} \leq +200$ for $i = 1, 2, 3, 4, 5, \dots, 12$.

5.6.4.2 Test System-II

For the longitudinal control system of the super maneuverable F18/HARV fighter aircraft the PID controller structure is given in (5.13), where

$$\mathbf{B}_2 = \begin{bmatrix} b_{211} & b_{212} & b_{213} \\ b_{221} & b_{222} & b_{223} \\ b_{231} & b_{232} & b_{233} \end{bmatrix}, \mathbf{B}_1 = \begin{bmatrix} b_{111} & b_{112} & b_{113} \\ b_{121} & b_{122} & b_{123} \\ b_{131} & b_{132} & b_{133} \end{bmatrix}, \mathbf{B}_0 = \begin{bmatrix} b_{011} & b_{012} & b_{013} \\ b_{021} & b_{022} & b_{023} \\ b_{031} & b_{032} & b_{033} \end{bmatrix}. \quad (5.30)$$

The design variables for this system are

$$X_D = [b_{211}, b_{212}, b_{213}, b_{221}, b_{222}, b_{223}, b_{231}, b_{232}, b_{233}, \\ b_{111}, b_{112}, b_{113}, b_{121}, b_{122}, b_{123}, b_{131}, b_{132}, b_{133}, \\ b_{011}, b_{012}, b_{013}, b_{021}, b_{022}, b_{023}, b_{031}, b_{032}, b_{033}] \quad (5.31)$$

The lower and upper bounds of the design variables are taken as

$$-20,000 \leq X_{Di} \leq +20,000 \quad \text{for } i = 1, 2, 3, 4, 5, \dots, 27 \quad (5.32)$$

5.6.5 Simulation Results

All the simulations are carried on Pentium Core2duo PC operating @2.2 GHz with 3 GB RAM. In order to validate the performance of the 2LB-MOPSO based multi-objective robust PID controller, highly coupled distillation column and super maneuverable F18/HARV fighter aircraft MIMO test systems are considered. In order to show the superior performance of 2LB-MOPSO over other existing established MOEA, NSGA-II [295] is applied to test the same control systems. Owing to the randomness of the nature-inspired optimization algorithms, 25 independent trials are conducted. The best, mean, and worst value of best solutions with ISE combined with balanced robust performance criteria from 25 trials obtained by 2LB-MOPSO and NSGA-II are presented in Table 5.12. For the purpose of comparison, earlier reported results of OSA [375], IGA methods [376] are taken. The best solutions of the obtained Pareto fronts by those algorithms are used to compare the performance of the multi-objective algorithms in Tables 5.10 and 5.11. The Hypervolume indicator [384] is used to assess the approximated Pareto fronts obtained by 2LB-MOPSO and NSGA-II. The Hypervolume indicator measures the Hypervolume of that portion of the objective space that is weakly dominated by an approximation set A , and it is to be maximized. Here, we consider the Hypervolume difference to a reference set R , and we will refer to this indicator as H , which is defined as $H=H(R)-H(A)$, where smaller values correspond to higher quality - in contrast to the original Hypervolume. In order to compare the results of the two algorithms, we combined two sets of results obtained by two algorithms over all 25 runs, applying the non-dominance sorting on them to obtain the overall non-dominated solutions as the reference set R . The Hypervolume indicator H takes into account both the convergence performance to the Pareto front and distribution of solutions in approximating the front. It indicates better performance with smaller indicator value. Maximum number of functional evaluations (Max_Feval) is used as the stopping criterion. The best obtained Pareto fronts using NSGA-II and 2LB-MOPSO are shown in Figure 5.6. Max_Feval is set at 10,000 for Test System-I and Test System-II. As both problems have constraints, we apply the epsilon-constraint handling method [319] to solve them. For 2LB-MOPSO, the parameters are set as in the previous section 5.5: *count* and *n_bins* are defined as 5 and 10; population size $NP=50$, inertia weight $\omega=0.729$; $c_1=c_2=2.05$; $V_{\max}=0.25(X_{D_{\max}}-X_{D_{\min}})$. Both NSGA-II and 2LB-MOPSO have the same size 100 for the external archive for the two test systems.

5.6.5.1 Results for Test System-I

2LB-MOPSO is used to design a multi-objective robust PID controller for distillation column plant by minimizing ISE for unit step reference input and balanced robust performance criteria. The best obtained Pareto front out of 25 trials is shown in Figure 5.6. For the purpose of comparison, the obtained Pareto front using NSGA-II method is also given. The parameters J_2 , J_∞ and $J = J_2 + J_\infty$ were considered to enable a convenient comparison of the 2LB-MOPSO with existing methods such as Riccati based and IGA [376], [383] and also NSGA-II method [295]. Best PID controller in terms of J using 2LB-MOPSO is given in (5.33).

$$G_c(s) = \frac{\begin{bmatrix} -9.1462 & -3.0599 \\ -0.4071 & 0.4046 \end{bmatrix} s^2 + \begin{bmatrix} -91.2012 & 18.5016 \\ -20.0168 & 20.4916 \end{bmatrix} s + \begin{bmatrix} -72.2656 & 104.9353 \\ -63.7446 & 92.2075 \end{bmatrix}}{s} \quad (5.33)$$

The step responses of the system with plant perturbation $\Delta P(s)$ in (5.15) using IGA-based, NSGA-II based and 2LB-MOPSO based controllers are shown in Figures 5.7, 5.8 and 5.9 respectively. From the step response characteristics, it is clear that the step responses (y_1 and y_2) of the best optimal controller designed using 2LB-MOPSO are more robust than those designed using other methods against plant perturbation. The controller outputs (u_1 and u_2) using reported IGA based method and 2LB-MOPSO based controller are given in Figs. 5.10 and 5.11, respectively. It is clear that 2LB-MOPSO based controller output is better than that generated by the IGA-based method.

5.6.5.2 Results for Test System-II

2LB-MOPSO algorithm is used to design a multi-objective robust PID controller for F18/HARV fighter aircraft system by minimizing ISE for the reference input $r(t) = [0, 1 - e^{-3t}, 1 - e^{-6t}]$ and balanced robust performance criteria. The best obtained Pareto front out of 25 trials for the test system-II is shown in Figure 5.6. For the purpose of comparison, NSGA-II method and the already reported OSA and IGA based controllers are considered. The best PID controller obtained using 2LB-MOPSO is given in (5.34).

$$\begin{aligned}
G_c(s) = & \frac{\begin{bmatrix} 6.7047e+003 & -1.1386e+004 & -1.7767e+004 \\ -6.5330e+003 & -6.4182e+003 & 8.5945e+001 \\ 2.0000e+004 & -2.0000e+004 & 2.0000e+004 \end{bmatrix} s^2}{s} \\
& + \frac{\begin{bmatrix} 2.0000e+004 & -2.0000e+004 & -2.0000e+004 \\ -2.0000e+004 & -2.0000e+004 & -8.0985e+003 \\ 6.1823e+003 & -2.0000e+004 & 2.0000e+004 \end{bmatrix} s}{s} \\
& + \frac{\begin{bmatrix} 2.0000e+004 & -1.5947e+004 & -2.0000e+004 \\ -1.7599e+004 & -6.7764e+003 & -5.4814e+003 \\ -3.7193e+003 & -2.0000e+004 & 5.2109e+003 \end{bmatrix}}{s}
\end{aligned} \tag{5.34}$$

Tables 5.10 and 5.11 show the best performance of various PID controllers in terms of J . 2LB-MOPSO results are significantly better than earlier reported results obtained by the Riccati based, the OSA based, IGA based and NSGA-II based methods. Moreover, all the results obtained by 2LB-MOPSO satisfy the constraints in Equations (5.4) and (5.5). Table 5.12 shows the best, mean, and worst value of best solutions with ISE combined with balanced robust performance criteria out of each approximate Pareto front from 25 trials for 2LB-MOPSO and NSGA-II. It can be observed that the results of 2LB-MOPSO is relatively more stable with smaller standard deviations indicating superior robust performance of 2LB-MOPSO based optimal PID controller over that of NSGA-II based. To demonstrate the robust performance capability of the 2LB-MOPSO and IGA based optimal controllers, the system was simulated with the reference inputs $r(t) = [0, 1 - e^{-3t}, 1 - e^{-6t}]^T$ and external disturbance of $d(t) = 0.01e^{-0.2t} \cos(31623t)[1, 1, 1]^T$ and plant perturbation $\Delta P(s)$ as given in (5.26). The output responses (y_1 , y_2 and y_3) of the system with the IGA-based and the 2LB-MOPSO based methods are shown in Figures 5.12 and 5.13, respectively. Figures 5.12 and 5.13 clearly indicate that the robust performance of 2LB-MOPSO based optimal PID controller is better than the performance of IGA based controller. The best, mean, worst and standard deviation values of Hypervolume indicator H for the approximate Pareto fronts out of 25 trials obtained by 2LB-MOPSO and NSGA-II are shown in Table 5.13. As the Hypervolume indicator captures both the convergence performance to the Pareto front and distribution of solutions in the approximated front, the Hypervolume indicator values of obtained Pareto front can be directly compared. Smaller Hypervolume indicator H values correspond to higher quality. Hence, it can easily be observed that the Hypervolume indicators obtained by 2LB-MOPSO are much better than the results obtained by NSGA-II on test systems with respect to all comparison categories with small standard deviations. The Hypervolume indicators presented in Table 5.13 indicate better robust performance of 2LB-

MOPSO based optimal PID controller with faster convergence to the Pareto front and better distribution of solutions along the front than those generated by NSGA-II. The superior performance can also be observed from Figure 5.6, in which the best Pareto fronts obtained by 2LB-MOPSO dominate the ones obtained by NSGA-II.

Table 5. 10 Best performance of various PID controllers in terms of $J = J_2 + J_\infty$ for Test System-I

Method	J_2	J_∞	J
Riccati based [376], [383]	42.6753	1.1588	43.8341
IGA [376]	16.7777	0.8520	17.6297
NSGA-II [295]	10.5009	1.0244	11.5253
2LB-MOPSO	9.7505	1.0179	10.7685

Table 5. 11 Best performance of various PID controllers in terms of $J = J_2 + J_\infty$ for Test System-II

Method	J_2	J_∞	J
OSA [375]	0.0019	0.4374	0.4393
IGA [376]	0.0092	0.1396	0.1488
NSGA-II [295]	3.5037e-03	0.1154	0.1189
2LB-MOPSO	6.7506e-05	0.0828	0.0829

Table 5. 12 Statistical performance of 2LB-MOPSO and NSGA-II in terms of J for Test System-I and II

Test System	Statistical Performance on J	2LB-MOPSO	NSGA-II
I	best	10.7685	11.5252
	Mean	10.7710	12.4924
	Worst	10.7786	20.4050
II	best	0.0829	0.1189
	Mean	0.0829	0.1255
	Worst	0.0831	0.1456

Table 5. 13 Hypervolume indicators H obtained by 2LB-MOPSO and NSGA-II for Test System-I and II

Test System	Algorithms	Best	Mean	Worst	STD
I	2LB-MOPSO	1.8185e-005	3.4397e-002	7.7381e-002	2.9096e-002
	NSGA-II	4.2141e-001	5.3263e-001	6.0805e-001	5.8706e-002
II	2LB-MOPSO	8.6349e-007	3.1670e-006	6.1601e-006	2.0662e-006
	NSGA-II	7.6992e-003	1.5240e-002	3.2741e-002	8.5482e-003

Figure 5. 6 Best Obtained Pareto fronts for Test System-I and II using 2LB-MOPSO and NSGA-II

Figure 5. 7 Step response of Test System-I with plant perturbation using IGA method

Figure 5. 8 Step response of Test System-I with plant perturbation using NSGA-II method

Figure 5. 9 Step response of Test System-I with plant perturbation using 2LB-MOPSO method

Figure 5. 10 The controller outputs (u_1 and u_2) using reported IGA method

Figure 5.11 The controller outputs (u_1 and u_2) using 2LB-MOPSO method

Figure 5.12 Response of Test System-II with plant perturbation and external disturbance using IGA method

Figure 5.13 Response of Test System-II with plant perturbation and external disturbance using 2LB-MOPSO method

5.7 Multi-Objective Design of Monopulse Antenna using 2LB-MOPSO

5.7.1 Introduction on Design of Monopulse Antenna

The conventional way of enhancing angular accuracy amounts to taking several measurements while the antenna rotates through an area of interest, and then to compare the results. However, this method has its drawbacks even if the antenna is properly calibrated. As the measurements are taken one after the other, the target has moved to another place in-between and the aspect angle has changed too. This change of aspect angle can lead to significant variation in the strength of the echo signal (this is called fluctuation) and render a comparison of consecutive measurements utterly useless. The monopulse technique was invented to eliminate this source of measurement error. A monopulse antenna [390]-[393] also takes several measurements with beams pointing in different directions, but as the name implies, these measurements are taken simultaneously, with a single pulse. The word "monopulse" implies that with a single pulse, the antenna can gather angle information, as opposed to spewing out multiple narrow-beam pulses in different directions and looking for the maximum return. Therefore this technique can determine angle very precisely. Monopulse antennas are in widespread use in military applications like target-tracking radars and missile-seeker heads. Civilian applications include automotive radars, secondary radars for air traffic control and control systems, which need to know the precise whereabouts of a TV-, GPS- or other type of satellite [390]-[394].

A key issue in the design of monopulse antennas is that the sum pattern and the difference pattern have to be synthesized by the same array configuration. In this context Lopez *et al.* [395] proposed an interesting method that is based on a subarray configuration and uses a standard binary Genetic Algorithm (GA) to determine the weights of the subarrays. One of the excitation sets (for the sum or difference pattern) is assumed to be known (and optimum) and the other set is synthesized by using a subarray configuration to reduce the feeding complexity. The optimization in [395] has been performed by considering a cost function

constituted by a single term penalizing a maximum side lobe level (SLL) exceeding a prescribed value. Caorsi *et al.* [396] took a similar approach where the same synthesis problem has been faced by applying the Differential Evolution (DE) method [24], in which hybrid chromosomes (constituted by real and integer genes) are used to avoid the need for coding and decoding the real variables (weights of the subarrays). In [396] an objective function is formulated and minimized in order to define, for each array element, the corresponding subarray, the weights of all subarrays, and the excitation sets of the difference pattern. In [397] the approach of [396] is extended to the optimization of the directivity of the difference pattern by means of a hybrid real/integer DE algorithm.

As can be perceived from a literature survey, the design of monopulse antenna arrays can be formulated in several possible ways and with emphasis on various aspects of the final output expected. Under such circumstances there may not exist a single optimal solution but rather a whole set of possible solutions of equivalent quality [24]. A natural choice for handling this kind of design problems is to use Multi-objective Optimization (MO) algorithms [398], [328] that deal with such simultaneous optimization of multiple, possibly conflicting, objective functions. To the best of our knowledge, there has been no research work reporting the design of monopulse antenna arrays from an MO perspective. This work can be treated as a first humble attempt towards this direction.

Proposed 2LB-MOPSO is employed for two purposes: firstly to design monopulse arrays that could simultaneously minimize the Maximum Side-Lobe Level (MSLL) and principal lobe Beam Width (BW), and secondly to study the effects of number of elements and number of subarrays on the performance of the antenna array by observing the shape of the Pareto Fronts generated with 2LB-MOPSO for various combinations of these two numbers. For the multi-objective design of monopulse array, a fuzzy membership function based approach described in [399] is taken to select the best compromise solution from the Pareto Front. Comparison with the single objective design results with real parameter Particle Swarm Optimization (PSO) [400] reflects the superiority of the multi-objective approach in terms of final accuracy of design results. Since multi-objective approach is superior to single objective cases where more than one design objectives are combined through weighted sum, the Pareto Fronts generated by a reliable MO algorithm, like 2LB-MOPSO, can provide a means of identifying the optimal number of design variables (through number of elements and number of subarrays).

5.7.2 Formulation of the Design Problem

An antenna array is a configuration of individual radiating elements that are arranged in space and can be used to produce a directional radiation pattern. For a linear antenna array with $2N$ isotropic radiators the array factor can be expressed as below

$$AF(\theta) = \sum_{n=-N}^{-1} a_n e^{j\left(n+\frac{1}{2}\right)kd \cos \theta} + \sum_{n=1}^N a_n e^{j\left(n-\frac{1}{2}\right)kd \cos \theta}, \quad (5.35)$$

where a_n is the excitation of the n - th radiating elements, k is the wave number of the medium in which the antenna is located, d is the distance between the elements, and θ defines the angle at which $AF(\theta)$ is calculated with respect to a direction orthogonal to the array.

The required sum pattern is obtained by the excitations a_n^s , $n = -N, \dots, -1, 1, \dots, N$, which are assumed to be symmetric about the array centre and fixed. Thus, we will have $a_n^s = a_{-n}^s$. The excitations are obtained by the Dolph-Chebyshev method [401]. Using the symmetry property the array factor reduces to the expression (5.36).

$$AF_s(\theta) = \sum_{n=1}^N a_n^s \cdot \cos \left[\frac{1}{2} (2n-1) kd \cos \theta \right]. \quad (5.36)$$

Excitations for the difference pattern are obtained by:

$$a_n^d = a_n^s \sum_{p=1}^P \delta_{c_n p} g_p \quad n = 1, 2 \dots N. \quad (5.37)$$

$\delta_{c_n, p}$ represents the Kronecker delta function [402] i. e. $\delta_{c_n, p} = 1$ if $c_n = p$, otherwise $\delta_{c_n, p} = 0$. In order to obtain the difference pattern, the excitations must be anti-symmetric i. e. $a_n^s = -a_{-n}^s$. Thus, the array factor for the difference pattern reduces to expression (5.38).

$$AF_d(\theta) = \sum_{n=1}^N a_n^d \cdot \sin \left[\frac{1}{2} (2n-1) kd \cos \theta \right]. \quad (5.38)$$

$AF_d(\theta)$ is a function of θ which is symmetric about 0° . Let $\theta_{\max}$ be the angle at which $AF_d(\theta)$ attains global maxima. We calculate $AF_d(\theta)$ for discrete values of θ . Let those discrete values be represented by set $\psi = [0, \pi/2]$. Let the discrete steps in which $AF_d(\theta)$ is calculated be $\Delta\theta$.

For obtaining multi-objective formulation of the present problem we need to find the maximum sidelobe level and the width of the principal lobe. The maximum sidelobe level (MSLL) is taken as the decibal level of the maximum sidelobe. To calculate the maximum sidelobe we calculate where the array factor reaches local maxima, and the maximum value of all the local maxima gives us the SLL value.

Let, $\zeta = [\theta \in \psi | \{AF_d(\theta) > AF_d(\theta - \Delta\theta)\} \wedge \{AF_d(\theta) > AF_d(\theta + \Delta\theta)\} \wedge \{\theta \neq \theta_{\max}\}]$ be the set of angles where local maxima of $AF_d(\theta)$ occur. One null of the principal lobe is located at 0° because of the anti-symmetric property of monopulse antenna. Let, $\Phi = \{\theta \in \psi | AF_d(\theta) < AF_d(\theta - \Delta\theta) \wedge AF_d(\theta) < AF_d(\theta + \Delta\theta) \wedge \theta \neq 0^\circ\}$ be the set of angles where local minima of $AF_d(\theta)$ is reached. Let the local minimum closest to 0° be α . Therefore $\alpha = \min(\Phi)$. Now we are at a position to define the two objective functions:

$$f_1 = \max\left(\frac{AF_d(\theta_{\max})}{AF_d(\zeta)}\right). \quad (5.39a)$$

$$f_2 = \min(\Phi). \quad (5.39b)$$

Figure 5.14 Geometry of Subarrayed linear array

5.7.3 Study of Optimal Pareto Fronts for Designing Monopulse Antenna

For a monopulse antenna array of $2N$ elements and P subarrays we will have N binary coded variables and P real coded variables. The N binary coded variables signify the subarray grouping information C_n and the P real coded variables indicate the subarray weights g_p . Note the following example.

Example 1: Suppose we have a 6 element ($N = 3$) antenna arrays. We want to design a monopulse array with 2 subarrays ($P = 2$). Then a possible subarray grouping can be, $c_1 = 1, c_2 = 2, c_3 = 1$. If the subarray weights be $g_1 = 0.5, g_2 = 0.8$, then the excitations of the array elements to produce difference pattern are, $a_1^d = a_{-1}^d = .5a_1^s$; $a_2^d = a_{-2}^d = .8a_2^s$; and $a_3^d = a_{-3}^d = .5a_3^s$.

For 2LB-MOPSO, the parameters are set as in the previous section 5.5: *count* and *n_bins* are defined as 5 and 10; population size $NP=50$, inertia weight $\omega=0.729$; $c_1 = c_2 = 2.05$; $V_{\max} = 0.25 * \text{DimRange}$. The maximum function evaluations are set as 15000.

The followings are primarily meant to study how the parameters such as number of elements and number of subarrays affect design of monopulse antennas in terms of two important figures of merit: the BW and MSL. For a fixed number of elements we can use an MO algorithm to decide the number of subarrays that produces a good trade-off between the two design objectives yielding a Pareto Front (PF) that has its knee points closer to the utopia point. Then we fix the number of elements and number of subarrays to find the best solution from the PF considering the same two design objectives, but this is taken up in the next section. In this section we will investigate the effects of the number of elements and subarrays on the PF and thus on the final design results obtained through 2LB-MOPSO. For space constraints in what follows we shall provide only certain sample results from our parametric study of the design problem.

Case 1: 20 Element Array

Fixing the number of elements to 20, we run 2LB-MOPSO varying the number of subarrays P from 2 to 10 in steps of 2.

Table 5.14 gives the optimal compromise results presenting the two objective values of the best solutions obtained from 5 parameter settings in this case. The corresponding Pareto Fronts have been shown in Figure 5.15.

Table 5. 14 Optimal Compromise table for 20 Element Array

No. of Subarrays(P)	BW	MSLL
2	11.43	-17.38
4	11.43	-17.24
6	11.43	-17.24
8	12.04	-24.40
10	11.73	-20.19

Figure 5. 15 Trade-off Curves for 20 element array ($N=10$)

A close inspection of Figure 5.15 shows that the best trade-off can be achieved for 8 subarrays as points near the PF are closest to the utopia point corresponding to least values of MSL and BW in comparison to the Pareto Fronts obtained with other numbers of subarrays, which can also be observed from Table 5.14 by comparing the compromising objective values. It becomes also clear that having subarrays greater than 8 will increase the complexity of monopulse antenna without any improvement in the radiation pattern.

Case 2: 40 Element Array

Figure 5.16 shows the trade-off curve obtained for 40 element array with number of subarrays = 2, 4, 8, 16. In this case we can observe that all the trade-off curves have similar performance, the best trade-off can be obtained with $P = 8$ with mostly spreading range. The OPF is very close to that obtained for $P = 4$. However increasing the number of subarrays does not improve the quality of solutions.

Figure 5. 16 Trade-off Curve for 40 element array ($N=20$)

Case 3: NUMBER OF SUBARRAYS CONSTANT

Here we have fixed the number of subarrays to $P = 8$. In this case we can observe that keeping P constant increasing N does not improve the solution beyond a certain limit, as is evident from the trade-off curves provided in Figure 5.17.

Figure 5. 17 Trade-off Curve for fixed number of subarrays ($P=8$)

The aim of this section was to investigate how the optimal combination of two vital parameters related to the design problem viz. number of elements and number of subarrays can be estimated using an MO algorithm. The next section is devoted to the actual design of the monopulse array using 2LB-MOPSO with the subarray weights and the element grouping kept as decision variables.

5.7.4 Design of Monopulse Antenna using 2LB-MOPSO

In case 1 of Section 5.7.3, we discussed that for 20 element array design, 8 subarrays gives a considerably good PF showing better compromise between the two objectives. In this section we systematically find the best compromise solutions for 20 element array, 8 subarrays and compare them with the results from standard single objective optimization algorithms namely PSO. For PSO we followed the same hybrid real/integer coding as reported in [396], [397]. For PSO, we used swarm size = 50, acceleration coefficients $c_1 = c_2 = 2.05$, inertia weight ω linearly decreasing from 0.9 to 0.4 and for d -th component of maximum velocity $V_{\max} = 0.25 * \text{DimRange}$ where DimRange is the difference between the maximum and minimum values of the decision variable. The maximum function evaluations for PSO are set as 15000.

The best compromise solution was chosen from the PF using the method described in [399].

The i^{th} objective function f_i is represented by a membership function μ_i defined as

$$\mu_i = \begin{cases} 1 & f_i \leq f_i^{\min} \\ \frac{f_i^{\max} - f_i}{f_i^{\max} - f_i^{\min}} & f_i^{\min} < f_i < f_i^{\max} \\ 0 & f_i \geq f_i^{\max} \end{cases} \quad (5.40)$$

where $f_i^{\min}$ and $f_i^{\max}$ are the minimum and maximum value of the i^{th} objective solution among all nondominated solutions, respectively. For each nondominated solution q , the normalized membership function μ^q is calculated as:

$$\mu^q = \frac{\sum_{i=1}^{N_{obj}} \mu_i^q}{\sum_{k=1}^{N_s} \sum_{i=1}^{N_{obj}} \mu_i^k}, \quad (5.41)$$

where N_s is the number of non-dominated solution. The best compromise is the one having the maximum value of μ^q . Below we provide the design results for five different cases corresponding to five different numbers of subarrays. The sum pattern corresponds to a Dolph-Chebyshev array with distance between elements $d = \lambda/2$ and SLL=-25dB.

The fitness function for single objective algorithms was taken as $f_1 + f_2$ where f_1 and f_2 are given by (5.39a) and (5.39b). As results we provide the best solutions found in 25 independent trials of each algorithm.

20 Elements , 8 Subarrays

Table 5. 15 Subarray Configuration

Algorithms	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
2LB-MOPSO	4	3	2	6	2	5	8	1	7	1
PSO	1	4	5	6	8	3	7	8	6	2

Table 5. 16 Subarray Weights

Algorithms	g_1	g_2	g_3	g_4
2LB-MOPSO	0.2794	1.0000	0.8568	0.5844
PSO	0.1870	0.1379	0.9578	0.5136
	g_5	g_6	g_7	g_8
2LB-MOPSO	0.1762	1.0000	1.0000	0.9549
PSO	0.7257	0.8885	0.9781	0.9575

Table 5. 17 Design Objectives Achieved

Algorithms	2LB-MOPSO	PSO
BW (degrees)	12.04	13.74
SLL	-24.40	-20.66

Figure 5.18 Normalized patterns for 20 element array and 8 subarrays

A keen observation of Table 5.17 and Figure 5.18 shows that in the test cases, 2LB-MOPSO achieves much better design objectives as well as array factors with lower SLL in comparison with both the single-objective algorithms PSO.

5.8 Conclusions

This chapter pointed out that almost all multi-objective particle swarm optimization (MOPSO) implementations currently available in the literature used exclusively the non-dominated solutions in the external archive of the current iteration as the *gbest* or *lbest* of every particle. As this approach emphasizes the elitism at the expense of diversity, this approach has the potential to converge to a small section of the true Pareto front if the size of the non-dominated set is small for several iterations during the evolution. In order to resolve this flaw, the *gbest* or *lbest* of every particle are chosen from a non-domination sorted archive of reasonably large size. As no current MOPSO variant takes into consideration the actual positions of the two guides, it is possible that the two guides can be located far apart in the parameter and/or objective space. In this situation, the search behavior of the MOPSO can potentially become chaotic. To resolve this, we propose a novel two local best (*lbests*) based MOPSO in which the two *lbests* are located close to each other in the vicinity of the best front sections. Since in all MOPSO variants, the *gbest* or *lbest* of each particle was chosen independently in each iteration. This process also results in chaotic search behavior.

The threshold *count* was included in the 2LB-MOPSO to keep the *gbests* or *lbests* in a particular region for a few iterations. Therefore, the 2LB-MOPSO was able to focus the search around small regions in the vicinity of the best front sections while maintaining the diversity due to the usage of two *lbests* and binning. Our simulation results demonstrate the improved performances due to our original contributions.

2LB-MOPSO is applied to design multi-objective robust PID controllers. Multi-objective robust controller design problem is formulated by minimizing ISE and balanced robust performance criteria. Two MIMO test systems, namely distillation column plant and longitudinal control system of the super maneuverable F18/HARV fighter aircraft are taken to validate the performance of the 2LB-MOPSO. For the purpose of comparison, 2LB-MOPSO and NSGA-II as well as earlier reported Riccati, IGA and OSA methods are considered. Statistical performances of PID controllers designed using 2LB-MOPSO in 25 independent trials are reported. In addition, Hypervolume testing is also conducted on the results obtained by 2LB-MOPSO and NSGA-II. The performance of various PID controllers is compared with respect to sum of the ISE and balanced robust performance criteria. Comparative studies reveal that 2LB-MOPSO results are better than earlier reported results and the results obtained by NSGA-II. The proposed 2LB-MOPSO gives consistent performance in terms of lesser mean value of sum of ISE and balanced robust performance criteria.

The approach to the synthesis problem of the difference patterns of monopulse antenna arrays in a multi-objective optimization framework is presented. 2LB-MOPSO is applied over different instances of the design problem, keeping minimum Maximum Sidelobe Level (MSLL) and principal lobe Beam Width (BW) as two design objectives to be simultaneously achieved. The subarray grouping information and weights are obtained from the best compromise solution of the OPF corresponding to N and P as determined before. The best compromise solution for $N = 10$ and $P = 2, 4, 6, 8, 10$ are obtained from their PF and the figure of merit of solution (i. e. MSLL and BW) are shown to beat that obtained with well-known single objective algorithms PSO. It is demonstrated that the optimal 20 element array design should be with 8 subarrays. Increasing the number of subarrays increases the complexity of design without improving the quality of solution appreciably. In conclusion we can say that MO algorithms have a dual role in designing a monopulse antenna. They can be used both for fixing N and P and also for determining the subarray configuration and

subarray weights. The method of design presented in this work can be directly put to use by the antenna designers. Finally we would like to point out that in this work we presented only one of the possible synthesis problems. There may be some other design objectives based on different formulations and it will be interesting to extend the multi-objective approach to those in future. Moreover, other MO algorithms [335][336] can also be tested on such problems.

Chapter 6

Multi-objective Particle Swarm Optimizer with Ensemble of External Archives

6.1 Introduction

Numerous optimization problems have more than one objective in conflict with each other. As there is no single solution for these problems, our aim is to find Pareto optimal trade-off solutions that represent the best possible compromises among the objectives. As evolutionary algorithms (EAs) and swarm intelligence algorithms (SIs) are able to find multiple Pareto-optimal solutions in one single run, development of nature-inspired optimization algorithms to solve multi-objective optimization problems has attracted much interest and several multi-objective nature-inspired optimization algorithms have been suggested [278], [295], [296], [320], [295], [292]. While most of these algorithms were developed taking into consideration two common goals, namely fast convergence to the Pareto-optimal front and good distribution of solutions along the front, each algorithm employs a unique combination of specific techniques to achieve these goals.

In order to avoid solutions becoming clustered, some techniques must be used so that solutions can adequately represent the whole of the Pareto front. The ε -dominance is one such technique [321] that does not allow more than one solution with difference in all (i^{th}) objective values less than ε_i to be non-dominated by each other. Hence, removal of such similar solutions improves diversity in the fixed sized external archive. However, the ε non-domination sorting method requires the user to choose individual ε_i values depending on the relative ranges of the objectives soon after initialization and in the final true Pareto front. In addition, it is also possible that different ε_i values may be needed for each objective during the different stages of the search process. In order to resolve difficulties in choosing suitable ε values, an ensemble of ε non-domination sorted external archives is proposed to use and demonstrate advantages of this approach by using a multi-objective particle swarm optimization algorithm (MOPSO) [332].

6.2 MOPSOs with Diversity Preservation

In MO algorithms, an important issue is what approach to use for guaranteeing the diversity of the solutions. Fitness sharing, mating constraints and crowding are some tools used to maintain the diversity in the population [322], [296], [323]. Zitzler [323] uses an external population, in the framework of SPEA, in which some of the non-dominated solutions found until the current generation are stored. In order to avoid the size of this population growing beyond a pre-defined size, S , a clustering technique is used in which the entire population is divided into S clusters and the centroid of each cluster is used as representatives. PAES [296] also uses an external population of non-dominated solutions which accepts new members if there is any available slot or the new candidate dominates one or more solutions in the archive, and these dominated solutions are removed from the archive. When the new candidate and existing solutions in the archive do not dominate each other and there is no available slot in the archive, the solutions in the less crowded regions are chosen. Here, the objective space is divided into regions (or grids) and the candidate is accepted if it is in a less crowded region (cell of the grid) than the other members of the archive. Otherwise it is rejected.

Elitism is implemented in NSGA-II [295] by evaluating the combined parents and offspring population and by choosing the non-dominated individuals in this combined population as the next generation population. When there is a need to choose between two or more candidates the one in the less crowded region is selected. The use of a grid as a way to reduce the size of the non-dominated front is also mentioned by Coello *et al.* [278].

Laumanns *et al.* [321] define the ε -dominance concept used to identify an ε -approximate Pareto set which, while representative of the Pareto optimal front, has also a diverse set of solutions. The ε -dominance strategy guarantees both the convergence towards the Pareto Optimal Front and a set of well distributed solutions. The ε -dominance does not allow more than one solution with difference in all (i^{th}) objective values less than ε_i to be non-dominated by each other. Hence, removal of such similar solutions improves diversity in the fixed sized external archive. However, the ε non-domination sorting method requires the user to choose individual ε_i values depending on the relative ranges of the objectives soon after initialization and in the final true Pareto front. Ideally different ε_i values are required even for a particular objective in an MOP during the different stages of evolution.

6.3 MOPSO with Ensemble of Epsilon-non-domination Sorted External Archives

6.3.1 ε -Dominance

The general idea of this mechanism is to divide the i^{th} objective function space into bins of size ε_i . By dividing every objective, we obtain cells. The approach accepts a new solution into the ε -dominance front if 1) it is the only solution in the cell to which it belongs, 2) it dominates other solution(s) or 3) it competes against other non-dominated solutions inside the cell, but it is closer to the origin vertex of the cell. A graphical illustration of epsilon-dominance concept is shown in Figure 6.1 in which the solution P ε -dominates the entire region ABCDA (with respect to minimization objectives), whereas according to the original dominance definition (*i.e.* $e_1=e_2=0$), P dominates the region PECFP. Solutions Q and P are

non-dominated with each other. However, since the distance between the solution Q and the vertex of the corresponding grid A is larger than the solution P 's distance to the vertex A , solution Q is ε -dominated by solution P . As there can be only one solution in one ε -cell (such as AGHIA) after ε -dominance sorting, the solution Q is deleted from non-dominated front and P is kept.

This algorithm was very attractive both from a theoretical and from a practical point of view. However, in order to achieve the best performance, it is necessary to provide the size of the cell by specifying the ε values for each objective which is problem-dependent and sensitive. Some ε values used by researchers are shown in Table 6.1 to demonstrate that different ε values are required for different objectives of the same problem if different algorithms are used. These ε values were obtained by performing numerous trial-and-error runs [322] over a large range with different values for different objectives. This approach is not feasible if the objective functions are computation intensive [324] or if there are several objectives in an MOP. In addition, it is also possible that different epsilon values may be needed for a single objective during different stages of evolution.

Two objectives: f_1 and f_2

Solutions: P and Q

Epsilon Values of two objectives: e_1 and e_2

Two Distances: AP and AQ , $AP < AQ$

Figure 6. 1 The ε -dominance concept (minimizing f1 and f2 objectives).

Table 6. 1 ε values used in different algorithms

MOEA Problem	Fast Steady-State ε - MOEA [322]	Enhanced ε -MOPSO [279]	Online Adaptive Search [326]
ZDT1 (2-objectives)	$\varepsilon = [0.75 \ 0.80]$	--	$\varepsilon = [0.75 \ 0.75]$
ZDT2 (2-objectives)	$\varepsilon = [0.0076 \ 0.0076]$	--	$\varepsilon = [0.0076 \ 0.0076]$
ZDT3 (2-objectives)	$\varepsilon = [0.00261 \ 0.00261]$	--	$\varepsilon = [0.00261 \ 0.00261]$
ZDT4 (2-objectives)	$\varepsilon = [0.0058 \ 0.0058]$	--	$\varepsilon = [0.0074 \ 0.0074]$
ZDT6 (2-objectives)	$\varepsilon = [0.0067 \ 0.0067]$	--	$\varepsilon = [0.0067 \ 0.0067]$
DTLZ1 (3-objectives)	$\varepsilon = [5 / 24 \ 5/24 \ 5/10]$	$\varepsilon = [0.07 \ 0.07 \ 0.07]$	--
DTLZ2 (3-objectives)	$\varepsilon = [0.06 \ 0.06 \ 0.066]$	$\varepsilon = [0.12 \ 0.12 \ 0.12]$	--
DTLZ3 (3-objectives)	$\varepsilon = [0.042 \ 0.0425 \ 0.04]$	$\varepsilon = [0.22 \ 0.22 \ 0.22]$	--
DTLZ4 (3-objectives)	$\varepsilon = [0.07 \ 0.07 \ 0.03]$	--	--
DTLZ8 (3-objectives)	$\varepsilon = [0.02 \ 0.02 \ 0.04]$	--	--
DTLZ2 (4-objectives)	$\varepsilon = [1 / 8 \ 1/8 \ 1/8 \ 1/6]$	--	--

6.3.2 Non-domination Sorted External Archive

Generally, an external archive is used to store the non-dominated solutions obtained so far in the search process. Since the size of the non-dominated set can be very large and the complexity of updating the archive increases with the archive size, the size of the archive is usually restricted to a pre-specified value which is usually the same as the finally required approximation solution set size. In our MOPSO implementation, the external archive is always the same size as the final approximate solution set size even if all the solutions are not non-dominated solutions as shown in Figure 6.2. In other words, we truncate the archive as $|P(t)| + |Q(t)| \rightarrow |P(t+1)|$ where cardinality of $P(t)$ and $P(t+1)$ are the same $|P(t)| = |P(t+1)|$ as the required final solution set size and $|Q(t)|$ is the number of new solutions generated in

generation t .

Initially, the archive includes all solutions at initialization. In every generation, all new offspring $Q(t)$ is combined with members in the archive $P(t)$ to obtain the mixed-temporary population. The epsilon non-domination sorting [321], [289] is applied to this mixed-temporary archive to obtain the sorted archive $R(t)$. During this process, all the sorted solutions retain two indicators namely, the front rank F_R and crowding distance value D_N . The sorted solutions with the lowest front rank will be included in the new archive first. When the size of the archive hits the permitted maximum size of $P(t)$, we apply the crowding distance [295] to select members from front F_4 (in Figure 6.2) in archive $R(t)$ to be included in $P(t+1)$.

- $P(t)$: External archive of generation t
- $Q(t)$: New solutions of generation t
- $R(t)$: Mixed-temporary archive
- $P(t+1)$: New external archive of generation $t+1$

Figure 6. 2 An illustration of external archive updating process

6.3.3 Two local bests (*Lbest*) Based MOPSO (2LB-MOPSO)

We employ a two *lbests* based MOPSO variant [328], which is introduced in the chapter 5. In order to select an *lbest* for a particle, an objective is first randomly selected followed by a random selection of a bin of the chosen objective. The second *lbest* is selected from a neighboring bin to the previously selected bin. These two *lbests* are the solutions with the smallest front numbers within the respective bins. As each particle's velocity is adjusted by the two *lbests* from two neighboring bins, the flight of each particle will be in the direction between the positions of two *lbests* and orientated to improve upon the current non-dominated solutions. The experimental results clearly show that these two *lbests* based MOPSO performed better than the previous MOPSO implementations.

Given that the two *lbests* of every particle are selected somewhat randomly, it will not be effective if every particle is assigned with a different pair of *lbests* in every generation because the flight of each particle will be almost random in such situations. Therefore, after assigning a pair of *lbests* to a particle, the number of generations the particle fails to contribute a solution to the archive $P(t)$ is recorded as *count*. If the *count* exceeds a pre-specified threshold, the particle is re-assigned to another pair of *lbests*.

6.3.4 2LB-MOPSO with Ensemble of External Archives

The 2LB-MOPSO [328] with an ensemble of external archives is shown in Figure 6.3. The first step is to initialize the population, their velocities, the maximum function evolutions (Max_FEs) and other parameters of algorithm as shown in Table 6.2. We evaluate the objective values of the initialized particle positions and assign them to three external archives $P_{\epsilon_1}(t)$, $P_{\epsilon_2}(t)$ and $P_{\epsilon_3}(t)$ with different epsilon values ϵ_1 , ϵ_2 and ϵ_3 . The epsilon values are selected based on our experiments. We divide the population of particles equally and assign $NP/3$ particles to each archive.

To update a particle's position in every generation, we select 2 *lbests* from the external archive to which the particle is assigned. After updating the dimension values of each

particle using the equations in Table 6.2, we need to check whether any dimension values of particle exceeds of the specified search bounds. We assign the corresponding dimensions with bound values and evaluate the objectives of the particles. We combine the three groups of new particles together to be the new solutions, combine them with the solutions in each archive, update the three archives by using the non-dominance sorting with 3 different epsilon values and truncate the archive size by applying the crowding-distance-sorting. If only a particular ε value is suitable for an objective, the new solutions generated using that particular ε value will dominate the solutions generated by the other two ε values and populate all three archives. Therefore, this ensemble approach has the inherent ability to benefit from the most suitable ε value. When the iterations reach maximum permitted function evaluations (Max_FEs), we combine all the solutions in the three archives and select the best solutions to be the approximated Pareto solutions.

Table 6. 2 The 2LB-MOPSO Algorithm

Initialize the swarm: Initialize NP (=60) number of particles randomly in the D -dimensional search space. Evaluate the fitness values of all particles. Set all current positions to be the external archive.

$V_{\max} = 0.25(X_{\max} - X_{\min}); \quad \omega = 0.729; \quad cc = [2.05 \ 2.05];$
 $V = (2 * rand(NP, 1) - 1) * (0.25 * (X_{\max} - X_{\min})); \quad // \text{ Initialize velocity}$
 $n_bin = 10 \quad // \text{ initialize the number of bins}$

Optimization Loop:

Select $lbests$ from the $P(t)$ external archive.

For $i=1$ to $Feval_max$ *// iteration loop*

For $j=1$ to NP *// iteration loop for updating velocity, position of each particle*

If $count(j) > 6$ *// updating lbests*

Reassign bins and select two $lbests$ for particle j ;

Else

Update the $lbests$ within the same bins for particle j ;

Endif

$V(j) = \omega * (V(j) + cc(1) * rand() * (lbest(j) - particle(j)) + cc(2) * rand() * (lbest(j + NP) - particle(j)))$
// updating velocity

$V_i(d) = \min(V_{\max}(d), \max(-V_{\max}(d), V_i(d)))$ *// Limit the velocity for each dimension*

$X_i(d) = X_i(d) + V_i(d)$ *// updating position of each particle*

If any dimension exceeds the search space

```
    Assign the corresponding dimensions with bound values;  
    Endif  
    Evaluate the fitness values of  $particle(j)$ ;  
EndFor  $j$   
Do non-domination sorting with epsilon gridding among combined  $P(t)$  archive and  $Q(t)$   
new particles.  
Update  $count$  of each particle according to the last paragraph in Section 6.3.3.  
Stop if a stop criterion is satisfied  
EndFor  $i$ 
```

Figure 6.3 The flowchart of ε -Dominance 2LB-MOPSO Algorithm with an Ensemble of External Archives

6.4 Experimental Results and Discussion

6.4.1 2LB-MOPSO Variants Considered in the Experimental Study

We compare the following four variants:

Method A: ε -Dominance 2LB-MOPSO with External Archive $\varepsilon = 0.05$;

Method B: ε -Dominance 2LB-MOPSO with External Archive $\varepsilon = 0.005$;

Method C: ε -Dominance 2LB-MOPSO with External Archive $\varepsilon = 0.0005$;

Method D: ε -Dominance 2LB-MOPSO with ensemble of external archives $\varepsilon = 0.05$, $\varepsilon = 0.005$ and $\varepsilon = 0.0005$;

These four variants are referred to as Method A or Method B and so on in the subsequent sections.

6.4.2 Performance Measures

In order to compare the performances of the above four 2LB-MOPSOs, we require quantitative performance metrics. There are two goals in a multi-objective optimization: 1) convergence to the Pareto-optimal front and 2) diversity of approximated solutions to the true Pareto-optimal front. We use two metrics which also applied in chapter 5, 1) R indicator (I_{R2}), 2) Hypervolume difference to a reference set ($I_{\bar{H}}$) [318].

6.4.3 Parameter Settings

For the each objective of these ten multi-objectives problems considered in our simulations,

we set the single epsilon values as 0.05; 0.005; 0.0005 and ensemble of those three epsilon values, as stated in subsection 6.4.1. As we explained in Section 6.3, the “*count*” value is set as a pre-specified threshold to decide when to re-assign a particle to another pair of *lbests* if the particle fails to generate a solution to enter the archive $P(t)$ consecutively for “*count*” number of iterations. For all problems, we use $count = 6$ in all four variants.

In our simulations, all 2LB-MOPSOs are run for a maximum of 500,000 function evaluations (FES) with 15 runs. Population size is $NP = 60$. The archive sizes are $|P(t)| = 100$ for two-objective problems, $|P(t)| = 150$ for three-objective problems and $|P(t)| = 300$ for five-objective problems. The maximum number of non-dominated solutions used to calculate the performance metrics is the same as the archive size for each problem in all four variants.

6.4.4 Benchmark Problems and Portfolio Optimization

We make use of ten Multi-objective problems which are four two-objective problems (SYMPART, S_ZDT4, R_ZDT4 and S_ZDT6), four three-objective problems (R_DTLZ2, S_DTLZ3, WFG8 and WFG9), and two five-objective problems (R_DTLZ2 and WFG1). They are selected from the CEC07 Special Session and Competition on “Performance Assessment on Multi-objective Optimization Algorithms”¹ as the representative test functions [309], which are list in Appendix C.

In order to show the advantage of ensemble of external archives in a real scenario, we apply the developed algorithm to solve the Portfolio Optimization problem which involves selection of different assets to invest to maximize the overall return and minimize the overall risk. The mathematical representation of portfolio optimization was introduced by Markowitz in the 1950s and he was rewarded with a Nobel Prize in Economics in 1990 [328]. The Markowitz model assumes that the investors would like to maximize the return under a certain risk level or minimize the risk with a certain return level [330], this model makes use of the mean and variance of normalized historical asset prices to measure the expected portfolio return and risk [331]. The model can be expressed as a two-objective optimization problem as follows:

¹ <http://www3.ntu.edu.sg/home/EPNSugan/>

$$\text{Min:} \quad \sigma^2 = \sum_{i=1}^D \sum_{j=1}^D w_i w_j \sigma_{ij} \quad (6.1)$$

$$\text{Max:} \quad r_p = \sum_{i=1}^D w_i r_i \quad (6.2)$$

$$\text{Subject to:} \quad \sum_{i=1}^D w_i = 1 \quad w_i \geq 0 \quad (6.3)$$

where D is the number of assets available to be selected into the portfolio which can be defined as the dimension of the optimization problem; w_i is the weight of i^{th} asset; σ^2 stands for the portfolio risk and σ_{ij} is the covariance between asset i and asset j . If $i=j$, σ_{ij} is the variance of that particular asset; r_p is the average portfolio return over a certain historical period, while r_i is the average individual return of asset i over the historical period. The experiment is conducted using the standard Markowitz (mean-variance) model presented above. The test data used in the simulation is downloaded from Yahoo-finance, up to more than 4000 stocks, 100 days' prices are considered with ending date at 31st Dec. 2007. For the MOPSOs, we use the same parameter settings described in the subsection 6.4.3, except the three epsilon values are modified to 5e-10, 5e-11 and 5e-12, due to the high precision of the particular system. For the generality, we applied the four variants presented in subsection 6.4.1 into randomly selected 50 assets out of the 4000 ones. In order to compare the results of the four variants, we combined all 4 sets of results over all 15 runs, do non-dominance sorting to obtain the overall non-dominated solutions as the reference set. Then we use the two performance metrics introduced in subsection 6.4.2 to compare the performances of the four variants. The results are presented in Table 6.5.

6.4.5 Discussions on Experimental Results

In Tables 6.3, 6.4 and 6.5, we can observe that the R indicator and H indicator values at the end of evolution for all four methods. The Median results of Methods A, B, C and D are ranked and small *rank* value indicates comparatively better performance. The h values presented in Tables 6.3, 6.4 and 6.5 are the results of t -tests as well as Wilcoxon tests between the results of Method D and the results achieved by Methods A, B and C. An h value of nonzero indicates that the performances of the two algorithms are statistically

different with 95% certainty. An h value of one means that the result of Method D is statistically better, negative one means that the result of Method D is worse and zero implies that the performances are not statistically different.

From the summations of $Rank$ value (computed on median result) and the summations of h value presented in the last two rows of Tables 6.3 and 6.4, we can easily find that Method D which is the ensemble of ε -dominance sorted external archives is comprehensively better than Methods A, B and C. The summations of $Rank$ value of the Methods A, B and C are approximately equal to each other implying that there is no one particular ε value best suited for all ten problems. We also can find the consistent conclusion in Table 6.5 implying that Method D outperforms the other three methods.

By observing the h values of t -test as well as Wilcoxon tests in Tables 6.3 and 6.4, we can also conclude that no one particular ε value best suited for all ten problems. For instance, according to the R indicator, Methods A and D are comparable to each other while outperforming Methods B and C in solving problems S_ZDT4 (2-objectives) and R_DTLZ2 (3-objectives). But, when we solve the problem R_DTLZ2 (5-objectives), Methods B, C, and D are obviously superior in performance than the Method A. Methods B and D achieve statistically similar best results when we solve problem S_ZDT6 (2-objectives). If we compare the results of problem R_ZDT4 (2-objectives) in the H indicator Table, it shows that the Method A and D yield the comparatively better performance than the other 2 Methods. Over all ten test problems, there is only one problem WFG1 (5-objectives), the results of all four methods are statistically similar to each other.

Figures 6.4 and 6.5 show the approximated Pareto fronts obtained by using Methods A-D (median run) on S_ZDT4 and R_ZDT4. From these figures, we can easily observe that the Method D can achieve the best performance, while some single ε -Dominance Methods failed totally. As a whole, among all the three single archived methods, only some of them can successfully approximate the non-dominated solutions of these difficult test problems. The ε -Dominance 2LB-MOPSO with an ensemble of external archives yields overall superior performance.

Table 6.3 The results for R indicator after $5e+5$ functions evaluations

TEST		METHOD A	METHOD B	METHOD C	METHOD D
------	--	----------	----------	----------	----------

MOPSO with Ensemble of External Archives

PROBLEM		($\varepsilon = 0.05$)	($\varepsilon = 0.005$)	($\varepsilon = 0.0005$)	(ensemble)
SYMPART (2objectives)	Best	1.5692e-04	4.4070e-05	3.8364e-05	3.0792e-05
	Median (Rank)	2.1917e-04 (4)	6.1786e-05 (3)	5.6932e-05 (2)	3.5321e-05 (1)
	Worst	2.8925e-04	1.1196e-04	8.3905e-05	4.8569e-05
	Mean	2.1057e-04	6.8756e-05	5.8125e-05	3.7716e-05
	Std	4.1326e-05	1.9673e-05	1.6194e-05	4.8616e-06
<i>t/Wilcoxon</i>	<i>h</i>	1/1	1/1	1/1	-
S_ZDT4 (2objectives)	Best	5.8214e-03	7.9352e-03	5.6235e-03	5.0025e-03
	Median (Rank)	8.0451e-03 (2)	1.1215e-02 (3)	1.3245e-02 (4)	7.6601e-03 (1)
	Worst	1.4141e-02	4.8129e-02	5.5226e-02	1.4062e-02
	Mean	8.7241e-03	1.4337e-02	2.6424e-02	8.4598e-03
	Std	2.7254e-03	1.2002e-02	2.3045e-02	2.3651e-03
<i>t/Wilcoxon</i>	<i>h</i>	0/0	1/1	1/1	-
R_ZDT4 (2objectives)	Best	1.7216e-03	3.1241e-03	4.0023e-03	1.3241e-03
	Median (Rank)	2.4194e-03 (2)	3.5395e-03 (3)	4.5793e-03 (4)	2.4136e-03 (1)
	Worst	2.8947e-03	4.6002e-03	4.9078e-03	2.7644e-03
	Mean	2.3219e-03	3.7546e-03	4.4965e-03	2.1674e-03
	Std	6.7658e-04	7.5120e-04	4.4843e-04	7.4106e-04
<i>t/Wilcoxon</i>	<i>h</i>	1/1	1/1	1/1	-
S_ZDT6 (2objectives)	Best	3.5004e-03	1.1854e-03	1.8422e-03	1.1984e-03
	Median (Rank)	3.9055e-03 (4)	1.3595e-03 (1)	2.1152e-03 (3)	1.3645e-03 (2)
	Worst	4.4448e-03	1.6454e-03	2.5514e-03	1.6200e-03
	Mean	3.9502e-03	1.3968e-03	2.1696e-03	1.3943e-03
	Std	4.7379e-04	2.3225e-04	3.5772e-04	2.1237e-04
<i>t/Wilcoxon</i>	<i>h</i>	1/1	0/0	1/1	-
R_DTLZ2 (3objectives)	Best	2.3773e-04	2.5004e-04	2.7001e-04	2.5201e-04
	Median (Rank)	2.8432e-04 (2)	2.9064e-04 (3)	3.0184e-04 (4)	2.7862e-04 (1)
	Worst	3.2461e-04	3.3303e-04	3.2544e-04	3.0441e-04
	Mean	2.7544e-04	2.9351e-04	2.9761e-04	2.7134e-04
	Std	2.3901e-05	2.4041e-05	1.9500e-05	1.4449e-05
<i>T/Wilcoxon</i>	<i>h</i>	0/0	1/1	1/1	-
S_DTLZ3 (3objectives)	Best	2.3292e-04	2.5439e-04	1.7623e-04	1.8202e-04
	Median (Rank)	2.6725e-04 (1)	2.8594e-04 (4)	2.8371e-04 (3)	2.7006e-04 (2)
	Worst	3.0064e-04	3.0113e-04	3.0492e-04	2.9252e-04
	Mean	2.7620e-04	2.8030e-04	2.5624e-04	2.3925e-04
	Std	2.6479e-05	2.4752e-05	7.8695e-05	8.2295e-05
<i>t/Wilcoxon</i>	<i>h</i>	1/1	1/1	1/1	-
WFG8 (3objectives)	Best	-2.1536e-02	-2.3341e-02	-2.2812e-02	-3.4049e-02
	Median (Rank)	-1.8741e-02 (4)	-2.0532e-02 (3)	-2.1525e-02 (2)	-3.1945e-02 (1)
	Worst	-1.5339e-02	-1.7714e-02	-2.0237e-02	-2.9748e-02
	Mean	-1.8845e-02	-2.0327e-02	-2.1444e-02	-3.1645e-02
	Std	1.5267e-03	1.6144e-03	6.4846e-04	1.2214e-03
<i>t/Wilcoxon</i>	<i>h</i>	1/1	1/1	1/1	-
WFG9	Best	-8.2347e-03	-9.1341e-03	-1.2836e-02	-1.4048e-02
	Median	-7.8258e-03	-9.1394e-03	-9.3097e-03	-1.3546e-02

(3objectives)	(Rank)	(4)	(3)	(2)	(1)
	Worst	-7.8357e-03	-8.2155e-03	-9.2261e-03	-1.3260e-02
	Mean	-8.0194e-03	-8.8585e-03	-9.6407e-03	-1.3627e-02
	Std	1.5005e-03	8.3008e-04	9.1546e-04	8.5475e-04
<i>t/Wilcoxon</i>	<i>h</i>	1/1	1/1	1/1	-
R_DTLZ2 (5objectives)	Best	1.2073e-04	1.2054e-04	1.1748e-04	1.1912e-04
	Median (Rank)	1.3617e-04 (3)	1.3296e-04 (2)	1.3888e-04 (4)	1.3136e-04 (1)
	Worst	1.6180e-04	1.6120e-04	1.5688e-04	1.5368e-04
	Mean	1.4150e-04	1.3954e-04	1.3849e-04	1.3254e-04
	Std	6.4234e-04	5.5277e-04	2.7240e-04	3.0573e-04
<i>t/Wilcoxon</i>	<i>h</i>	1/1	0/0	0/0	-
WFG1 (5objectives)	Best	4.3745e-02	4.3944e-02	4.5439e-02	4.3312e-02
	Median (Rank)	4.5535e-02 (2)	4.5835e-02 (3)	4.5931e-02 (4)	4.5518e-02 (1)
	Worst	4.6325e-02	4.6241e-02	4.6249e-02	4.6006e-02
	Mean	4.5237e-02	4.5528e-02	4.5934e-02	4.5219e-02
	Std	7.4324e-04	6.7477e-04	2.7121e-04	7.5736e-04
<i>t/Wilcoxon</i>	<i>h</i>	0/0	0/0	0/0	-
Summation Of Rank (Median)		26	28	32	12
Summation of <i>h</i> value (<i>t/Wilcoxon</i>)		7/7	7/7	8/8	NA

Table 6. 4 The results for H indicator after $5e+5$ functions evaluations

TEST PROBLEM		METHOD A ($\varepsilon = 0.05$)	METHOD B ($\varepsilon = 0.005$)	METHOD C ($\varepsilon = 0.0005$)	METHOD D (ensemble)
SYMPART (2objectives)	Best	4.5379e-04	1.1874e-04	1.1576e-04	9.1141e-05
	Median (Rank)	1.4478e-03 (3)	1.6856e-04 (2)	3.6451e-03 (4)	1.0601e-04 (1)
	Worst	1.9345e-03	3.8254e-03	4.1345e-03	1.4343e-04
	Mean	1.2454e-03	8.6663e-04	2.0387e-03	1.1236e-04
	Std	5.1622e-04	1.5358e-03	2.0324e-03	1.4119e-05
<i>t/Wilcoxon</i>	<i>h</i>	1/1	1/1	1/1	-
S_ZDT4 (2objectives)	Best	1.6441e-02	2.2635e-02	1.5924e-02	1.4102e-02
	Median (Rank)	2.3136e-02 (2)	3.2224e-02 (3)	3.8438e-02 (4)	2.1223e-02 (1)
	Worst	4.0512e-02	1.4304e-01	1.6943e-01	4.0231e-02
	Mean	2.4854e-02	4.1808e-02	8.0125e-02	2.2302e-02
	Std	7.8247e-03	3.5925e-02	7.1239e-02	6.0528e-03
<i>t/Wilcoxon</i>	<i>h</i>	1/1	1/1	1/1	-
R_ZDT4 (2objectives)	Best	6.3145e-03	9.4148e-03	9.6947e-03	5.7417e-03
	Median (Rank)	7.2463e-03 (2)	1.0047e-02 (3)	1.2751e-02 (4)	7.2297e-03 (1)
	Worst	7.9003e-03	1.5040e-02	1.7799e-02	7.9051e-03
	Mean	7.0537e-03	1.1201e-02	1.3115e-02	6.9688e-03
	Std	7.8695e-04	3.3121e-03	4.4132e-03	1.0106e-03
<i>t/Wilcoxon</i>	<i>h</i>	0/0	1/1	1/1	-

S_ZDT6 (2objectives)	Best	8.0244e-03	2.3951e-03	4.4741e-03	2.4009e-03
	Median (Rank)	8.7614e-03 (4)	2.8562e-03 (1)	4.9881e-03 (3)	2.9144e-03 (2)
	Worst	9.6655e-03	3.1647e-03	5.3240e-03	3.1245e-03
	Mean	8.8171e-03	2.8153e-03	4.9287e-03	2.8133e-03
	Std	8.2187e-04	3.8731e-04	4.2805e-04	3.7225e-04
<i>t/Wilcoxon</i>	<i>h</i>	1/1	0/0	1/1	-
R_DTLZ2 (3objectives)	Best	3.9215 e-03	4.5243 e-03	4.2457 e-03	4.4123 e-03
	Median (Rank)	5.0412 e-03 (2)	5.2421 e-03 (4)	5.2378 e-03 (3)	4.8020 e-03 (1)
	Worst	6.3365 e-03	6.4377 e-03	6.1421 e-03	5.6007 e-03
	Mean	5.0413 e-03	5.2321 e-03	5.2489 e-03	4.9024 e-03
	Std	6.9855e-04	5.9423e-04	4.6799e-04	3.8226e-04
<i>t/Wilcoxon</i>	<i>h</i>	1/1	1/1	1/1	-
S_DTLZ3 (3objectives)	Best	5.0635e-04	6.0833e-04	2.6127e-04	2.7952e-04
	Median (Rank)	6.3500e-04 (1)	7.6723e-04 (4)	7.3321e-04 (3)	7.2009e-04 (2)
	Worst	8.8413e-04	8.3004e-04	8.0446e-04	7.9324e-04
	Mean	7.2234e-04	7.5760e-04	6.3256e-04	5.4217e-04
	Std	2.8417e-05	1.7408e-05	5.7468e-05	5.6359e-05
<i>t/Wilcoxon</i>	<i>h</i>	1/1	1/1	1/1	-
WFG8 (3objectives)	Best	-1.1544e-01	-1.3493e-01	-1.3732e-01	-1.7874e-01
	Median (Rank)	-1.0213e-01 (4)	-1.2394e-01 (3)	-1.3291e-01 (2)	-1.7494e-01 (1)
	Worst	-9.2335e-02	-1.1503e-01	-1.2473e-01	-1.7283e-01
	Mean	-1.0402e-01	-1.2382e-01	-1.3172e-01	-1.7524e-01
	Std	6.6357e-03	6.0237e-03	3.7481e-03	3.1189e-03
<i>t/Wilcoxon</i>	<i>h</i>	1/1	1/1	1/1	-
WFG9 (3objectives)	Best	-6.4932e-02	-6.5405e-02	-6.6528e-02	-8.3245e-02
	Median (Rank)	-5.1744e-02 (4)	-5.3748e-02 (3)	-5.7336e-02 (2)	-7.9520e-02 (1)
	Worst	-4.6231e-02	-5.1645e-02	-5.5118e-02	-7.5111e-02
	Mean	-5.2542e-02	-5.6034e-02	-5.7814e-02	-7.9347e-02
	Std	5.2294e-03	5.6451e-03	2.3654e-03	1.9414e-03
<i>t/Wilcoxon</i>	<i>h</i>	1/1	1/1	1/1	-
R_DTLZ2 (5objectives)	Best	2.4457e-03	4.1314e-03	5.4428e-03	3.4216e-03
	Median (Rank)	7.6325e-03 (3)	7.2445e-03 (2)	7.8390e-03 (4)	4.6050e-03 (1)
	Worst	8.4120e-03	9.2347e-03	9.4054e-03	8.1255e-03
	Mean	6.7297e-03	6.8345e-03	7.4458e-03	4.9128e-03
	Std	1.8134e-03	1.7487e-03	1.1489e-03	1.5065e-03
<i>t/Wilcoxon</i>	<i>h</i>	1/1	1/1	1/1	-
WFG1 (5objectives)	Best	5.0153e-01	5.0263e-01	5.1432e-01	4.9523e-01
	Median (Rank)	5.1574e-01 (2)	5.1812e-01 (3)	5.2104e-01 (4)	5.1470e-01 (1)
	Worst	5.2112e-01	5.2191e-01	5.2250e-01	5.1991e-01
	Mean	5.1433e-01	5.1601e-01	5.1943e-01	5.1300e-01
	Std	6.0484e-03	5.9351e-03	3.0251e-03	7.0028e-03
<i>t/Wilcoxon</i>	<i>h</i>	0/0	0/0	0/0	-
Summation of Rank (Median)		27	28	31	12
Summation					

of <i>h</i> value (<i>t</i> / <i>Wilcoxon</i>)	8/8	8/8	9/9	NA
---	-----	-----	-----	-----------

Table 6. 5 The results for *H* and *R* indicators after 5e+5 functions evaluations in Portfolio Optimization Application

Portfolio Optimization Problem (2objectives)		METHOD A ($\varepsilon = 5e-10$)	METHOD B ($\varepsilon = 5e-11$)	METHOD C ($\varepsilon = 5e-12$)	METHOD D (ensemble)
<i>H</i> INDICATOR	Best	9.5124e-04	8.2005e-04	8.9721e-04	3.6015e-05
	Median (Rank)	1.3753e-03 (4)	1.3551e-03 (2)	1.3608e-03 (3)	9.5528e-04 (1)
	Worst	1.4721e-03	1.4761e-03	1.5062e-03	1.2629e-03
	Mean	1.2827e-03	1.2616e-03	1.2765e-03	6.5755e-04
	Std	1.8311e-04	2.0249e-04	2.2050e-04	4.9739e-04
<i>t</i> / <i>Wilcoxon</i>	<i>h</i>	1/1	1/1	1/1	-
<i>R</i> INDICATOR	Best	7.0070e-04	6.0428e-04	6.6136e-04	1.0245e-05
	Median (Rank)	1.0141e-03 (4)	9.9900e-04 (2)	1.0031e-03 (3)	6.9238e-04 (1)
	Worst	1.0852e-03	1.0883e-03	1.1101e-03	9.1723e-04
	Mean	9.4175e-04	9.2797e-04	9.4001e-04	4.7135e-04
	Std	1.4010e-04	1.5177e-04	1.6330e-04	3.6813e-04
<i>t</i> / <i>Wilcoxon</i>	<i>h</i>	1/1	1/1	1/1	-

Figure 6. 4 Pareto fronts obtained by using Methods A-D (median run) on Test Problem 2 (S_ZDT4)

Figure 6. 5 Pareto fronts obtained by using Methods A-D (median run) on Test Problem 3 (R_ZDT4)

6.5 Conclusions

In this chapter, we demonstrated that multi-objective particle swarm optimization implementation with an ensemble of ε -dominance external archives is suitable when solving problems with different characteristics. Different problems require the user to choose individual ε values depending on the relative ranges of the objectives soon after initialization and in the final true Pareto front. Some problems especially the real world applications require different ε values for different objectives during different stages of evolution. If we use the same ε value for every objective, such problems may not be solved satisfactorily. On the other hand, when we use an ensemble of external archives sorted using significantly different ε values, it is highly likely that different objectives may find at least one ε value appropriate thereby yielding satisfactory results. Our experimental results on recent ten difficult multi-objective problems as well as the portfolio optimization problem show that the ensemble approach always performs either better than or comparable to any single archive implementations in the statistical sense. In addition, the almost equal aggregate rankings of single archive implementations imply that they perform differently on each problem. Hence, we can conclude that instead of implementing an nature-inspired

optimization algorithm with just a single ε non-domination sorted external archive, using an ensemble of external archives is more efficient in particular if the computation cost of evaluating the objectives is high or there are several objectives to be optimized. Even though we realized the ensemble of external archives using different numerical values for the ε parameter, it is also possible to employ different diversity preservation methods with different parameter values to form ensemble of external archives.

Chapter 7

Decomposition Based Multiobjective Evolutionary Algorithm with an Ensemble of Neighborhood Sizes

7.1 Introduction

The multiobjective optimization is introduced in previous section 5.2. An unconstrained multiobjective optimization problem (MOP) can be stated as follows [293]:

$$\begin{array}{ll} \text{Minimize} & F(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_m(\mathbf{x}))^T \\ \text{Subject to:} & x \in \Omega \end{array} \quad (7.1)$$

where Ω is the *decision (variable) space*, $F : \Omega \rightarrow R^m$ consists of m real-valued objective functions and R^m is called the *objective space*. In many real-world applications, since the objectives in equation (7.1) conflict with one another, no point in Ω can minimize all the objectives at the same time. A point $x^* \in \Omega$ is *Pareto optimal* if there is no point $x \in \Omega$ such that $F(x)$ dominates $F(x^*)$. $F(x^*)$ is then called a *Pareto optimal (objective) vector*. In other words, any improvement in a Pareto optimal point in one objective must lead to deterioration to at least one other objective. The set of all the Pareto optimal objective vectors is the *Pareto front* (PF) [293].

Recent years have witnessed significant progress in the development of evolutionary algorithms (EAs) and swarm intelligence algorithms (SIs) for dealing with multiobjective optimization problems (MOPs). Those algorithms aim at finding a set of representative Pareto optimal solutions in a single run. Most Multiobjective evolutionary algorithms (MOEAs) are Pareto dominance based, they push their population as a whole towards the PF, guided mainly by dominance-based fitness measures of individual solutions. NSGA-II [295], [322], SPEA-II [323] and PAES [334] are among the most popular Pareto-dominance based MOEAs.

MOEA/D (multiobjective evolutionary algorithm based on decomposition) [335], [336] is a recent multiobjective evolutionary algorithm. Using conventional aggregation approaches, MOEA/D decomposes the approximation of the PF into a number of single objective optimization subproblems. The objective of each subproblem is a (linear or nonlinear) weighted aggregation of all the individual objectives in the MOP. Neighborhood relations among these subproblems are defined based on the distances among their aggregation weight vectors. Each subproblem is optimized in MOEA/D by using information mainly from its neighboring subproblems. Neighborhood sizes are a crucial control parameter in MOEA/D [335], [336], [337]. A large neighborhood size promotes collaboration among dissimilar subproblems and thus is good for population diversity, while a small neighborhood size encourages combination of similar solutions and is good for local search. Arguably, different multiobjective problems need different neighborhood sizes, and even for a particular problem, using different neighborhood sizes at different search stages could improve the algorithm performance.

Ensemble learning has proven to be very efficient and effective for adjusting algorithmic control parameters and operators in an online manner [332], [338], [340], [341]. In this chapter, we propose to use an ensemble of different neighborhood sizes in MOEA/D and dynamically adjust their selection probabilities based on their previous performances. We compare the resultant algorithm, called ENS-MOEA/D with MOEA/D proposed in [336] on CEC 2009 MOEA competition benchmark problems [339]. Our results indicate that ensemble learning improve the performance of the best algorithm significantly.

7.2 Review on the MOEA/D

There are several variants of MOEA/D. In this chapter, we use MOEA/D with dynamical resource allocation, which won the CEC2009 multiobjective algorithm contest [336]. To decompose equation (7.1), it needs N evenly spread weight vectors $\lambda^1, \dots, \lambda^N$. Each $\lambda^j = (\lambda_1^j, \dots, \lambda_m^j)^T$ satisfies $\sum_{k=1}^m \lambda_k^j = 1$ and $\lambda_k^j \geq 0$ for all k and m . Let $z^* = (z_1^*, \dots, z_m^*)^T$ be the reference point, i. e. $z_i^* = \max \{f_i(x) \mid x \in \Omega\}$. Then, the problem of approximation of the PF of equation (7.1) can be decomposed into N scalar optimization subproblems and the objective function of the j -th subproblem is:

$$g^{te}(x \mid \lambda, z^*) = \max_{1 \leq i \leq m} \{\lambda_i \mid f_i(x) - z_i^*\} \quad (7.2)$$

z^* is often unknown before the search, the algorithm uses the lowest f_i -value found during the search to substitute z_i^* [336]. During the search, MOEA/D maintains:

- a population of N points $x^1, \dots, x^N \in \Omega$, where x^i is the current solution to i -th subproblem;
- $FV^1, \dots, FV^N$, where FV^i is the F -value of x^i , i.e., $FV^i = F(x^i)$ for each $i = 1, \dots, N$;
- $z = (z_1, \dots, z_m)^T$, where z_i is the best value found so far for objective f_i ;
- π^i : Utility of subproblem i , which measures the likelihood that x^i be improved in the later search.

For each weight vector, its T -neighborhood is the set of T closest weight vectors to it. Correspondingly, each solution and each subproblem have their T -neighborhoods, respectively.

At each generation, a set of the current solutions is selected based on their utility. For each selected solution x^i , MOEA/D does the following:

Step 1. Set the mating and update range P to be the T -neighborhood of x^i with a large probability, and the whole population otherwise.

Step 2. Randomly select three current solutions from P .

Step 3. Apply genetic operators on the above selected solutions to generate a new solution y , repley y if necessary. Compute $F(y)$.

Step 4. Replace a small number of solutions in P by y if y is better than them for their subproblems.

No solution will be replaced In Step 4 if y is not better than any solution in P for its subproblem. When such a case happens, we say that the update fails, otherwise, it is successful.

More details about the MOEA/D with dynamical resource allocation can be referred to the original works [336], [335].

7.3 Ensemble of Neighborhood Sizes for MOEA/D (ENS-MOEA/D)

In MOEA/D [336], a neighborhood of weight vector λ^i is defined as a set of its several closest weight vectors in $\lambda^i = (\lambda_1, \dots, \lambda_N)$. The neighborhood of the i th subproblem consists of all the subproblems with the weight vectors from the neighborhood of λ^i . To evolve the solution of a subproblem, only the current solutions to its neighboring subproblems are exploited. This requires specifying a neighborhood size which may have to be varied for each subproblem over different stages of evolution. Normally, a larger neighborhood size permits the corresponding subproblem to search more globally, while a smaller

neighborhood size performs a local search. Hence, by appropriately adjusting the neighborhood size for each subproblem performance of the MOEA/D can be enhanced. However, for diverse problems, the trial-and-error approach can be too demanding to tune neighborhood sizes. In addition, during different stages of evolution, different neighborhood sizes can be more effective to solve a single problem. Motivated by these observations, we employ an ensemble of neighborhood sizes which are adaptively selected according to their previous records of generating promising solutions.

In ENS-MOEA/D, several neighborhood sizes (*NS*) are used as a pool of candidates. During evolution, with respect to each subproblem, one neighborhood size will be chosen from the candidate pool proportionally to the candidates' previous experiences of generating improved solutions. Therefore, in ENS-MOEA/D, a gradual adaptation of the selection probability is performed for each problem according to previous *NS* values that have generated improved solutions. The certain fixed number of previous generations used to store the success probability is named as the Learning Period (*LP*). After the initial *LP* generations, the probabilities of choosing different *NS*s will be updated at each subsequent generation based on the *Success rate* of each *NS*. For example, at the generation *G*, the probability of choosing the k^{th} ($k = 1, 2, \dots, K$) *NS* is updated by:

$$p_{k,G} = \frac{R_{k,G}}{\sum_{k=1}^K R_{k,G}}, \text{ where } R_{k,G} = \frac{\sum_{g=G-LP}^{G-1} FEs_success_{k,g}}{\sum_{g=G-LP}^{G-1} FEs_{k,g}} + \varepsilon, (k=1,2,\dots,K; G > LP) \quad (7.3)$$

where $R_{k,G}$ represents the proportion of improved solutions generated by the k^{th} *NS* and successfully entering the next generation within the previous *LP* generations with respect to generation *G*. $FEs_{k,g}$ represents the total number of function evaluations used by the k^{th} *NS* within the previous *LP* generations respect to generation *G*, while $FEs_success_{k,g}$ means the number of function evaluations used by the the k^{th} *NS* generated improved solutions within the previous *LP* generations with respect to generation *G*. The small constant value $\varepsilon = 0.05$ 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 $R_{k,G}$ by $\sum_{k=1}^K R_k$ to calculate $p_{k,G}$.

As in the recent MOEA/D [336], the ENS-MOEA/D maintains a population of N points $x^1, \dots, x^N \in \Omega$, where x^i is the current solution to the i th subproblem. Each current solution is the center point of one subproblem with an associated weight vector. Hence, the number of weight vectors is also the same as the population size and the number of subproblems. For a subproblem with a particular weight vector, NS number of closest weight vectors are chosen based on the Euclidean distance to form the neighboring subproblems. The procedure of ENS-MOEA/D presented below:

- **Initialization:** Set generation counter as $G=0$; probability $P_{k,G}=1/K$. Generate a uniform spread of N weight vectors, $\lambda^i = (\lambda_1, \dots, \lambda_N)$. Compute the Euclidean distances between any two weight vectors and then find the NS closest weight vectors to each weight vector, where the NS is randomly chosen from k ($k = 1, 2, \dots, K$) NS by a uniform distribution to make sure that each NS value has equal probability of being selected. Generate an initial population $x^1, \dots, x^N \in \Omega$ by uniformly randomly sampling from the search space. Initialize the best value found so far for each objective, $z = (z_1, \dots, z_m)^T$ as well as the utility of each subproblem π^i . Initialize $FES_{k,G}$ and $FES_success_{k,G}$ for the k^{th} NS .
- **Optimization Loop:**

If The stopping criteria is not satisfied, repeat:

For each subproblem, based on the current probability $P_{k,G}$, select one NS from the k ($k = 1, 2, \dots, K$) NS to form the neighbourhood. Apply the original MOEA/D [336] to generate new solutions, $G = G + 1$; update $FES_{k,G}$ and $FES_success_{k,G}$ for the k^{th} NS .

If $G = Integer \times LP$,

Update the probability $P_{k,G}$ based on the current $FES_{k,G}$ and $FES_success_{k,G}$ for the k^{th} NS as shown in the equation 7.4. Reinitialize the $FES_{k,g}$ and $FES_success_{k,g}$ as 0 for the next LP loop.

Endif

Else Stop and output the final N solutions.

Endif

7.4 Experimental Setup and Results

In this section, ENS-MOEA/D has been tested on the 10 unconstrained test instances in CEC 2009 MOEA Competition [339] including the two-objective and three-objective problems. The IGD value is used to measure algorithm performances as in the CEC 2009 MOEA Competition, which indicates both the convergence speed as well as the diversity among solutions. The basic parameter settings are as follows, which are the same as in the original MOEA/D [336]. N : 600 for two objectives and 1000 for three objectives; In DE and mutation operators: $CR=1.0$ and $F=0.5$; Stopping criteria: simulation stops after 300,000 function evaluations for each test instance. In the parameter investigation of LP , it is set as 10, 25, 50 and 75. The four different NS s for the two-objective problems are set as 30, 60, 90 and 120, in which $NS=60$ is the original parameter setting in the MOEA/D [336]; while the NS s for the three-objective problems are set as 60, 80, 100, 120 and 140, where 100 is the original parameter setting for NS in the MOEA/D [336].

Table 7.2 in the shows the results of parameter investigation of LP for ENS-MOEA/D among 4 different values 10, 25, 50 and 75 on the 10 benchmark functions, the mean of IGD values over 25 runs are ranked. Base on the entire results, one conclusion can be found that the LP is not so sensitive to most of the benchmark functions. Form the overall rankings, 50 generations should be chosen as the most promising setting for LP . Hence in the final version of ENS-MOEA/D, we use 50 as the Learning Period (LP) for updating the assigning probabilities of different NS . Table 7.3 and 7.4 show the comparison among the ENS-MOEA/D with variants of MOEA/D with different values of over implementations of ensemble with only one NS . Within the variants of MOEA/D, the original fixed parameter setting for NS in the MOEA/D [336] is also included. The mean of IGD values over 25 runs among all the variants of MOEA/D and ENS-MOEA/D are ranked. Form these ranking, it is can easily observed that the ENS-MOEA/D yields superior performance over all other implementations with only one fixed setting of the neighborhood size by 300,000 function evaluations.

In addition, in order to investigate the both convergent abilities and diversity qualities of the original MOEA/D and the proposed ENS-MOEA/D through the entire search process, the IGD value plots along the whole search stage are shown in Figure 7.1. For each test function, the mean of IGD values out of 25 runs obtained by the original MOEA/D and the proposed ENS-MOEA/D at different search stages are recorded, the convergence graph is demonstrated in term of mean of the IGD values. For clear view of difference on evolution performance between two methods, the convergence graphs exhibit the IGD values during the main search stage, except the very early phase before 30,000 function evaluations (earlier 10% of the whole search stage). Furthermore, in order to compare the performance of the original MOEA/D and the proposed ENS-MOEA/D numerically at two different stages, Table 7.1 shows the mean of IGD values obtained by them out of 25 runs at 150,000 function evaluations and 300,000 function evaluations for all the 10 test functions. As the observation from the Figure 7.1 and Table 7.1, overall, the faster convergence speed with good distribution of solutions is obtained by the proposed ENS-MOEA/D over the original MOEA/D. For some problems, such as F7, F8 and F9, even though the final performances of two methods are nearly similar, but the better performance is achieved earlier by the proposed ENS-MOEA/D. In all the comparison categories of Table 7.1 to 7.4, best solutions are shown in bold. Moreover, in Table 7.1, the h values present the results of t -tests as well as Wilcoxon tests between the two methods. An h value of nonzero indicates that the performances of the two methods are statistically different with 95% certainty. An h value of one means that the result of ENS-MOEA/D is statistically better, negative one means that the result of ENS-MOEA/D is worse and zero implies that the performances are not statistically different.

Table 7. 1 Mean IGD obtained by two methods at two phases.

F	150,000 function evaluations			300,000 function evaluations		
	MOEA/D	ENS-MOEA/D	h	MOEA/D	ENS-MOEA/D	h
1	2.9481e-03	2.8657e-03	0/0	2.0211e-03	1.6423e-03	1/1
2	1.1294e-02	8.4730e-03	1/1	4.7824e-03	4.0487e-03	1/1
3	2.2059e-02	6.7624e-03	1/1	1.0552e-02	2.5916e-03	1/1
4	8.4205e-02	4.5530e-02	1/1	6.2411e-02	4.2070e-02	1/1
5	3.7242e-01	2.8351e-01	1/1	3.1666e-01	2.4811e-01	1/1
6	1.8868e-01	6.1571e-02	1/1	1.8860e-01	6.0847e-02	1/1
7	2.0903e-03	1.9375e-03	0/0	1.7668e-03	1.7286e-03	0/0

8	6.5673e-02	4.7500e-02	1/1	4.2793e-02	3.1006e-02	1/1
9	4.2086e-02	3.6835e-02	1/1	2.9236e-02	2.7874e-02	1/1
10	5.8056e-01	2.4346e-01	1/1	4.8577e-01	2.1173e-01	1/1

Table 7. 2 Results of parameter investigation of *LP*

F	<i>LP=10</i>	<i>LP=25</i>	<i>LP=50</i>	<i>LP=75</i>
1	1.9860e-03(2)	2.1233e-03(4)	1.6423e-03(1)	2.0495e-03(3)
2	4.8229e-03(2)	5.2834e-03(4)	4.0487e-03(1)	4.8603e-03(3)
3	8.4398e-03(4)	4.8241e-03(3)	2.5916e-03(1)	4.6226e-03(2)
4	6.1736e-02(4)	5.9504e-02(3)	4.2070e-02(1)	5.9261e-02(2)
5	3.0492e-01(4)	2.8743e-01(3)	2.4811e-01(2)	2.4723e-01(1)
6	1.9266e-01(2)	2.5994e-01(3)	6.0847e-02(1)	3.1457e-01(4)
7	1.6889e-03(1)	1.8336e-03(3)	1.7286e-03(2)	2.0421e-03(4)
8	3.0489e-02(1)	5.5337e-02(4)	3.1006e-02(3)	3.0571e-02(2)
9	1.2703e-01(4)	9.2639e-02(3)	2.7874e-02(1)	8.2577e-02(2)
10	4.5652e-01(4)	4.4261e-01(3)	2.1173e-01(1)	4.3687e-01(2)
Ranking	28	33	14	25

Table 7. 3 Results of the variants of MOEA/D with different *NS* and ENS-MOEA/D on two-objective problems F1 to F7 (Original MOEA/D sets *NS*=60)

F	<i>NS=30</i>	<i>NS=60</i>	<i>NS=90</i>	<i>NS=120</i>	ENS-MOEA/D
1	1.7454e-03(2)	2.0211e-03(3)	2.6472e-03(4)	3.1065e-03(5)	1.6423e-03(1)
2	5.7314e-03(3)	4.7824e-03(2)	6.0706e-03(4)	7.4653e-03(5)	4.0487e-03(1)
3	1.0964e-02(5)	1.0552e-02(4)	7.5243e-03(3)	4.4294e-03(2)	2.5916e-03(1)
4	5.8660e-02(2)	6.2411e-02(4)	6.2793e-02(5)	6.1326e-02(3)	4.2070e-02(1)
5	3.5256e-01(5)	3.1666e-01(4)	2.9491e-01(3)	2.6224e-01(2)	2.4811e-01(1)
6	2.4952e-01(5)	1.8860e-01(3)	2.0682e-01(4)	1.3369e-01(2)	6.0847e-02(1)
7	1.9660e-03(3)	1.7668e-03(2)	2.4637e-03(4)	2.7014e-03(5)	1.7286e-03(1)
Ranking	25	22	27	24	7

Table 7. 4 Results of the variants of MOEA/D with different *NS* and ENS-MOEA/D on three-objective problems F8 to F10 (Original MOEA/D sets *NS*=100)

	F8	F9	F10	Ranking
$NS=60$	4.3024e-02(5)	1.0481e-01(6)	4.2104e-01(3)	14
$NS=80$	3.7810e-02(2)	9.3060e-02(3)	4.1454e-01(2)	7
$NS=100$	4.2793e-02(4)	2.9236e-02(2)	4.8577e-01(6)	12
$NS=120$	4.7825e-02(6)	9.4638e-02(5)	4.7328e-01(5)	16
$NS=140$	4.1981e-02(3)	9.3111e-02(4)	4.2249e-01(4)	10
ENS-MOEA/D	3.1006e-02(1)	2.7874e-02(1)	2.1173e-01(1)	3

Figure 7. 1 Convergence graphs in term of mean of IGD obtained by two methods

7.5 Conclusions

In this chapter, an ensemble of different neighborhood sizes of the subproblems with self-adaptation is integrated with MOEA/D (ENS-MOEA/D). The trial-and-error procedure of tuning suitable numerical values for neighborhood sizes when solving different problems can be avoided. Moreover, during one whole of evolution, reasonable values for this critical parameter should be adjusted to enhance the computational effort. This difficulty can also be overcome by applying an ensemble of different reasonable neighborhood sizes with self adaptation. Our experimental results on the unconstrained problems in the “Special Session and Competition on Multiobjective Optimization” held at the *CEC 2009* show that ENS-MOEA/D yields superior performance when compared with MOEA/D with different fixed neighborhood sizes.

Chapter 8

Self-adaptive Differential Evolution with Modified Multi-trajectory Search for Large Scale Optimization

8.1 Introduction

Differential Evolution (DE) is a simple yet powerful population-based stochastic search technique for solving global optimization problems in continuous search domain. DE generates offspring by perturbing the solutions with a scaled difference of selected population vectors, and employs parent-offspring comparison which allows replacement of a parent only if the offspring is better or equivalent to its parent. In DE, there exist many trial vector generation strategies and crucial control parameters that significantly influence the optimization performance of the DE. Therefore, in order to successfully solve a specific optimization problem, it is generally required to perform a time-consuming and computationally demanding trial-and-error search for the most appropriate strategy and to tune the associated parameter values. Moreover, during the search process, the DE

population may evolve through different regions in the search space thereby requiring different strategies and parameter settings. Hence, it is desirable to adaptively determine appropriate strategies and associated parameter values at different stages of evolution. In this chapter, we apply the Self-adaptive DE (SaDE) [139], [342], [343] to make both crossover strategies and their associated parameters self-adaptive by learning from their previous experiences in generating improved solutions. Consequently, suitable generation strategies along with parameter settings can be determined adaptively to match different search phases. At each generation, a strategy together with associated parameter values will be assigned to each parent according to the selection probabilities learnt from the previous generations.

The JADE [169] introduces a new mutation strategy, “DE/current-to-*pbest*” with optional external archive. As a generalization of DE/current-to-best, DE/current-to-*pbest* utilizes not only the best solution information but also the information of other good solutions of the current population. To be specific, any of the top $p\%$, $p \in (0, 100]$ solutions will be randomly chosen in DE/current-to-*pbest* to play the role designed exclusively for the single best solution in DE/current-to-best. In addition, the difference between the archived solutions and a member in the current population can be incorporated into the mutation operation. This strategy is able to diversify the population and provides approximate direction of evolution.

In this chapter, the SaDE is enhanced by utilizing MMTS, JADE mutation strategy (SaDE-MMTS) and the binomial (uniform) crossover, the exponential crossover and without crossover to test the 19 problems from the Soft Computing Special Session and Competition on Large Scale Global Optimization [346]. The step sizes are also self-adapted during every MMTS step. The success rates of both SaDE and the MMTS are computed and consequently, future function evaluations for both search algorithms are assigned according to their recent past performance.

8.2 SaDE with MMTS

In this section, Self-adaptive DE (SaDE) is first introduced, the JADE mutation and the original multi-trajectory search (MTS). Finally, we explain how we combine them to form the Self-adaptive DE with a Modified multi-trajectory search (MMTS) (SaDE-MMTS) [344], [345].

8.2.1 Self-adaptive DE (SaDE)

To achieve good performance by applying the conventional DE to a given problem, it is necessary to perform a trial-and-error search for the most appropriate trial vector generation strategy and fine-tune its associated control parameter values, *i.e.* the values of CR , F and NP . 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 employed Self-adaptive DE (SaDE) [139], in which both trial vector generation strategies and their associated control parameter values can be gradually self-adapted according to their previous records of generating promising solutions.

SaDE maintains a strategy candidate pool. 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 improved solutions and applied to perform the mutation operation. The probability of being chosen in the current generation to generate offspring is proportional to generating improved offspring in the past iterations.

In SaDE, the population size NP is a user-specified parameter according to the pre-estimated complexity of the given problem. CR is usually more sensitive to problems with different characteristics, while F is closely related to the convergence speed [24]. Hence, the parameter F is approximated by a normal distribution with mean value of 0.5 and standard deviation of 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 of the current population. Therefore, both exploitation (with small F values) and exploration (with large F values) are maintained throughout the entire evolution process. The proper choice of CR can lead to successful optimization performance while a wrong choice may deteriorate the performance. Therefore, in SaDE, gradual adaptation of the range of CR values is considered for a given problem according to previous CR values that have generated trial vectors successfully entering the next generation. Details on the updating of probabilities with respect to each strategy, updating the *Success and Failure Memories* and Learning Period (LP) can be found in the original SaDE work [139].

8.2.2 JADE Mutation

Compared to DE/rand, the greedy strategies such as DE/current-to-best and DE/best benefit from their fast convergence property by incorporating best solution information in the evolutionary search. However, the best solution information may also cause premature convergence. As a compromise, a new mutation strategy, named DE/current-to-*pbest* with small sized archive called *arch*, is proposed in [169]. Any of the top $p\%$, $p \in (0, 100]$ solutions can be randomly chosen to play the role of the *pbest* solution in DE/current-to-*pbest*. The archive operation is made very simple to avoid computation overhead. The archive is initiated to be empty. Then, after each generation, the parent solutions that fail in the selection process (parent-offspring comparison) are added to the archive. If the archive size exceeds a certain threshold, defined as the population size NP , then some solutions are randomly removed from the archive to keep the archive size as NP . The archive provides information about the evolution path and improves the diversity of the population. In DE/current-to-*pbest*/1 with archive, a mutation vector is generated as follows:

$$\text{"DE/rand-to-*pbest*/1"}: \mathbf{V}_{i,G} = \mathbf{X}_{r_1^i,G} + F \times (\mathbf{X}_{pbest,G} - \mathbf{X}_{r_2^i,G}) + F \times (\mathbf{X}_{r_3^i,G} - \mathbf{X}_{r_4^i,G}^{\square}) \quad (8.1)$$

where $\mathbf{X}_{r_1^i,G}$, $\mathbf{X}_{r_2^i,G}$, $\mathbf{X}_{r_3^i,G}$ and $\mathbf{X}_{pbest,G}$ are randomly selected from the current population named *pops*, while $\mathbf{X}_{r_4^i,G}^{\square}$ is randomly chosen from the union, $pops \cup arch$, of the current population and the archive. The scale factor $F \in (0, 1+)$ is a positive real number that controls the rate of population evolution or the diversity of the population.

8.2.3 MTS

The Modified Multiple Trajectory Search (MMTS) was applied to solve the unconstrained real-parameter large scale optimization problem [347], [348]. The original MTS employs search agents, which search for better solutions by moving with different step sizes in the parameter space from the original positions in each dimension. The step size was defined within a search domain to fit the requirement of global and local search. In this chapter, we integrate a MMTS into the SaDE to implement the line search along each dimension one by one. The details will be presented in the next section.

8.2.4 SaDE-MMTS

Optimization algorithms perform differently when solving different optimization problems due to their distinct characteristics. But, some of them often lose their efficacy when applied to solve complex problem instances with high dimensions. There are two main difficulties for those optimization methods whose performance deteriorates quickly as the dimensionality of the search space increases. First one is the high demand on exploration capabilities of the optimization methods. When the solution space of a problem increases exponentially with the number of problem dimensions, more efficient search strategies are required to explore all promising regions within a given time budget. Second, the complexity of a problem characteristics may increase with increasing dimensionality, e.g. uni-modality in lower dimensions may become multi-modality in higher dimensions for some problems. Due to these reasons, a successful search strategy in lower dimensions may no longer be capable of finding the optimal solution in higher dimension [349], [350].

In order to solve the complex problem instances especially with high dimensions, the JADE mutation strategy with small sized archive is coupled with two basic crossover operators (binomial crossover and exponential crossover) as well as no crossover option. The mutation strategies with exponential crossover and without any crossover are appropriate for solving linked problems. The binomial crossover is effective for solving separable problems when the CR is low while it is also effective for solving non-separable problems when CR is large. SaDE-MMTS also benefits from the self-adaptation of trial vector generation strategies and control parameter adaptation schemes in the SaDE by learning from their previous experiences to fit different characteristic of the problems and different search requirements of evolution phases. In the SaDE-MMTS, 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. 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 in the pool. At the beginning, the probabilities with respect to each strategy are initialized as $1/K$, *i.e.* all strategies have the equal probability to be chosen. 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 population is recorded as

$ns_{k,G}$ while the number of trial vectors generated by the k^{th} strategy that are discarded is recorded as $nf_{k,G}$. *Success and Failure Memories* are introduced to store these numbers within a fixed number of previous generations hereby named Learning Period (LP). Once two 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 those numbers calculated in the current generation can be stored in the memories.

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 the generation G , the probability of choosing the k^{th} ($k = 1, 2, \dots, K$) strategy is updated by:

$$p_{k,G} = \frac{S_{k,G}}{\sum_{k=1}^K S_{k,G}}, \text{ where } S_{k,G} = \frac{\sum_{g=G-LP}^{G-1} ns_{k,g}}{\sum_{g=G-LP}^{G-1} ns_{k,g} + \sum_{g=G-LP}^{G-1} nf_{k,g}} + \varepsilon, \quad (k=1,2,\dots,K; G > LP) \quad (8.2)$$

where $S_{k,G}$ represents the success rate of the trial vectors generated by the k^{th} strategy and successfully entering the next generation within the previous LP generations with respect to generation G . The small constant value $\varepsilon = 0.01$ is used to avoid the possible null success rates. To ensure that the probabilities of choosing strategies are always summed to 1, $S_{k,G}$ are further divided by $\sum_{k=1}^K S_{k,G}$ to obtain $p_{k,G}$. Obviously, the larger the success rate for the k^{th} strategy within the previous LP generations is, the larger the probability of applying it to generate trial vectors in the current generation will be. The JADE mutation strategy contributes $pbest$ as well as the evolution path to the SaDE-MMTS.

The MMTS is used periodically for a certain number of function evaluations, which is also determined by an adaptation procedure. At the beginning of optimization procedure, the SaDE and the MMTS are firstly conducted sequentially within one search cycle by using same number of function evaluations. Then the success rates of both SaDE and MMTS are calculated similarly as in equation (8.2). Subsequently, function evaluations are assigned to SaDE and MMTS in each search cycle proportional to the success rates of both search methods.

An adaptation approach is proposed to adaptively determine the initial step size parameter used in the MMTS every time the MMTS is applied. In each MMTS phase, we calculate the

average of all mutual dimension-wise distances between current population members (AveDis), select one of the five linearly reducing factors (LRF) from 1 to 0.02, 5 to 0.02, 10 to 0.02, 20 to 0.02 and 40 to 0.02, and apply this LRF to scale AveDis over the evolution. After that, the step size will be further reduced when a better solution is found along a particular dimension.

The search agents in each MMTS step are selected from the current DE population by using a Clearing procedure [351]. The number of the search agents in each MMTS step is linearly reduced from 4 to 1, which associates the variation of search requirement from “global” to “local”. Consequently, the SaDE-MMTS is proposed to solve 19 problems proposed for the Special Issue of Soft Computing on Scalability of Evolutionary Algorithms for Large Scale Continuous Optimization Problems [346]. The proposed SaDE-MMTS is presented in Table 8.1.

Table 8.1 The algorithm description of SaDE-MMTS

INITIALIZATION: Set the generation counter $G = 0$, and randomly initialize a population (*pops*) 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 mean value of CR (CRm_k), strategy probability ($p_{k,G}$, $k = 1, \dots, K$, K is the no. of available strategies), learning period (LP). Evaluate the initial population.

INITIAL OPTIMIZATION STEP

Conduct both SaDE and MMTS in one search cycle for a certain number of function evaluations, determine the success rates for both approaches. Assign updated number of function evaluations proportionally according to the success rates of SaDE and MMTS in the next search cycle for both search methods.

OPTIMIZATION LOOP

WHILE stopping criterion is not satisfied.

Conduct both SaDE and MMTS in one search cycle, update the success rates for both approaches. Reassign updated number of function evaluations proportionally according to the success rates of SaDE and MMTS for the next search cycle for both search methods.

Step1. 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 (8.2)

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

END

END

Step2. Assign trial vector generation strategy by using stochastic universal sampling to

target vector $\mathbf{X}_{i,G}$. Assign control parameters to each target vector $\mathbf{X}_{i,G}$. Assign $\mathbf{X}_{pbest,G}$, $\mathbf{X}_{r1,G}$ and $\mathbf{X}_{r2,G}$ for 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 $G \geq LP$

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 $CR_{m,i} > 1$

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

END

END FOR

END FOR

/* Assign $pbest$ vector and the other two distinct parameter vectors for each target vector */

FOR $i = 1$ to NP

Randomly select the $\mathbf{X}_{pbest,G}$ for the target vector $\mathbf{X}_{i,G}$ from the top $p\%$ out of the $pops$.

Randomly select the $\mathbf{X}_{r1,G}$ for the target vector $\mathbf{X}_{i,G}$ from the $pops$.

Randomly select the $\mathbf{X}_{r2,G}$ for the target vector $\mathbf{X}_{i,G}$ from the $pops \cup arch$.

END FOR

Step3. Generate a new population where each trial vector $\mathbf{U}_{i,G}^k$ is generated according to associated trial vector generation strategy k out of the following listed three combinations and generated parameters F_i and $CR_{k,i}$.

Generation Strategy 1 JADE mutation with exponential crossover;

Generation Strategy 2 JADE mutation with binomial crossover;

Generation Strategy 3 JADE mutation with no crossover;

Randomly reinitialize a value to the variable of trial vector $\mathbf{U}_{i,G}^k$ within the search space it is outside its boundaries.

Step4. Selection and updating

FOR $i = 1$ to NP

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

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

$\mathbf{X}_{i,G+1} = \mathbf{U}_{i,G}^k$, $f(\mathbf{X}_{i,G+1}) = f(\mathbf{U}_{i,G}^k)$

```

 $ns_{k,G} = ns_{k,G} + 1$ 
Store  $CR_{k,i}$  into  $CRMemory_k$ , store  $X_{i,G}$  into JADE archive, if the archive size is full,
truncate the archive by removing a randomly selected archived member.
    IF  $f(U_{i,G}) < f(X_{best,G})$ 
         $X_{best,G} = U_{i,G}$ ,  $f(X_{best,G}) = f(U_{i,G})$ 
    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.
Increment the generation count  $G = G + 1$ 
Step5. IF search cycle of SaDE is completed
    FOR each search agent selected by Clearing
        Apply MMTS along each dimension with initial step size as  $LRF \times AveDis$ .
    ENDFOR
ENDIF
END WHILE

```

8.3 Experimental Results and Discussion

19 test functions (F1-F19) that should be used for the experimental study for the Special Issue of Soft Computing on Scalability of Evolutionary Algorithms and other Metaheuristics for Large Scale Continuous Optimization Problems [346] are considered in the simulation. Here is a summary of the 19 test problems:

The *F1-F11* functions are the following:

- *Shifted Unimodal Functions:*
 - *F1: Shifted Sphere Function*
 - *F2: Shifted Schwefel's Problem 2.21*
- *Shifted Multimodal Functions:*
 - *F3: Shifted Rosenbrock's Function*
 - *F4: Shifted Rastrigin's Function*
 - *F5: Shifted Griewank's Function*
 - *F6: Shifted Ackley's Function*

- *Shifted Unimodal Functions*
 - *F7: Shifted Schwefel's Problem 2.22*
 - *F8: Shifted Schwefel's Problem 1.2*
 - *F9: Shifted Extended f10*
 - *F10: Shifted Bohachevsky*
 - *F11: Shifted Schaffer*

The hybrid composition functions *F12-F19* are obtained combining two functions from the set *F1-F11*.

8.3.1 Parameter Setting

Experiments were conducted on all 19 minimization problems with 50, 100, 200, 500, and 1,000 dimensions. To solve these problems, the population size of SaDE-MMTS is set as 60. The maximum number of fitness evaluations is 5,000. Each run stops when the maximal number of evaluations is exhausted. The size of JADE archive is the same as the population size and the *pbest* selection ratio from the population is set as 0.5 [169], i.e. $p=50\%$ or top 30 of DE population is used as the *pbests*. The initial control parameters for binomial crossover and exponential crossover are set as ($F=0.5$, $CR=0.2$) and ($F=0.5$, $CR=0.9$) respectively. Furthermore, the parameter F is approximated by a normal distribution with mean value 0.5 and standard deviation 0.3, denoted by $N(0.5, 0.3)$ and CR obeys a normal distribution with mean value CR_m obtained by self-adaptation and standard deviation 0.1, denoted by $N(CR_m, 0.1)$ [139]. The learning period (LP) is set as 50 generations [139].

According to the requirements [346], solution quality for each function is measured as defined as: $f(x)-f(op)$, when the FEs reaches the maximum evaluations. Here op is the optimal parameters of the function. Maximum, median, minimum and average error values of the best individual found in each of the 25 runs are presented in Tables 8.2 and 8.3. The SaDE-MMTS is compared with the three evolutionary algorithms (DE, Real-coded CHC, and G-CMA-ES) used as the baseline algorithms in the Special Issue of Soft Computing on Scalability of Evolutionary Algorithms and other Metaheuristics for Large Scale Continuous Optimization Problems. Statistical comparison on the average error is carried out by applying non-parametric tests and presented in Table 8.4. The computational running time of

the SaDE-MMTS is also provided in Table 8.5. The computer system runs Windows XP (SP2) with Pentium(R) 4 3.00GHz, **RAM:** 2 GB. Matlab 7.1 is used as the programming language.

8.3.2 Analysis of the Results

In this section, the experimental results obtained with the proposed algorithm are analyzed. In particular, a statistical comparison on the average error is conducted against the original SaDE with JADE mutation as well as the reference algorithms specified for this special issue. Finally, the computational running time of the proposed SaDE-MMTS is reported.

From the experimental results in Tables 8.2 and 8.3, we can observe that out of the 19 test functions, problems 1, 3, 5, 6, 7, 10, 12, 15, 16 and 19 can be comparatively easily solved by the proposed SaDE-MMTS irrespective of the scaling of dimensions. Among those 10 functions, functions 1, 6 and 7 are separable, functions 12 and 16 are two composition functions which combine separable functions 1 and 9. By performing line search with MMTS, they should be optimized variable by variable. Also benefiting from the binomial crossover, the SaDE-MMTS performs perfectly on them. Moreover, with the greedy strategy in mutation phase, high precision of solutions can be achieved on those functions. In contrast, the basic functions 3, 5 and 10 are non-separable functions which can be solved by using the exponential crossover with control parameter adaptation schemes and without any crossover. The SaDE-MMTS performs well on them also. The perfect performance is difficult to be achieved on both separable and highly linked functions by SaDE-MMTS which is a compromise between the dimension-wise search as well as general directional search. Consequently, the performance of the SaDE-MMTS when solving functions 8, 9 and 11 deteriorates when the number of dimensions is increased. Comparatively, the four composition functions, namely 13, 14, 17 and 18, combine the relatively hardest basic functions 3, 4 and 9. The proposed SaDE-MMTS is facing difficulty in solving them, especially when the number of dimensions is greater than 200.

The indicative results obtained by three reference algorithms specified for this special issue [346] show that DE with exponential crossover achieves the best performance than other two indicative algorithms. Therefore, in Table 8.4, only a comparison between SaDE-MMTS and DE on the whole set of functions and dimensions is presented. In Table 8.4, for each function, the better value out of the two competing average errors obtained by SaDE-MMTS

and DE are show in bold, as well it is assigned the rank value as 1, while the other one is given the rank as 2. In the last column of this table, all the rank values are accumulated to show the performance difference of two algorithms. Overall the SaDE-MMTS performs superior than the reference DE algorithm for all five different dimensions. Accordingly, the proposed SaDE-MMTS is able to achieve compromised performance on both separable and non-separable problems. With increasing number of dimensions, the SaDE-MMTS still performs in a stable manner on most of the test functions demonstrating a good scalability characteristic.

In order to demonstrate the improvement of the proposed SaDE-MMTS when compared to the original SaDE with JADE mutation, four out of the 19 test functions with 1000 dimensions (F3, F9, F13 and F19) are selected to be solved by both methods. The average errors obtained by both methods on this function group are presented in Table 8.6. The table shows clearly the superior performance achieved by SaDE-MMTS compared to the original SaDE with JADE mutation.

Table 8. 2 Experimental Results of 25 runs obtained by SaDE-MMTS on Problem 1-10

		F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
50D	Best	0	0	0	0	0	0	0	4.4545e-10	8.1081e-02	0
	Median	0	0	0	0	0	0	0	3.0454e-09	1.1018e-01	0
	Worst	0	0	0	0	0	0	0	1.7532e-08	1.9025e-01	0
	Average	0	0	0	0	0	0	0	4.1349e-09	1.3584e-01	0
	Std	0	0	0	0	0	0	0	6.1641e-09	1.1641e-01	0
100D	Best	0	0	0	0	0	0	0	6.1262e-06	1.2856e-01	0
	Median	0	0	0	0	0	0	0	2.6778e-04	2.2137e-01	0
	Worst	0	0	0	0	0	0	0	3.9786e-04	5.9347e-01	0
	Average	0	0	0	0	0	0	0	3.0522e-04	3.1831e-01	0
	Std	0	0	0	0	0	0	0	3.1615e-04	2.3154e-02	0
200D	Best	0	0	0	0	0	0	0	9.2076e+00	5.7459e-01	0
	Median	0	1.0229e+00	0	1.9849e-02	0	0	0	2.3691e+01	1.0815e+00	0
	Worst	0	3.2684e+00	0	1.9798e-01	0	0	0	5.3190e+01	2.0170e+00	0
	Average	0	1.3410e+00	0	8.0899e-02	0	0	0	2.6753e+01	1.2403e+00	0
	Std	0	1.4616e+00	0	9.4616e-02	0	0	0	8.5501e+00	2.1961e+00	0
500D	Best	0	8.6415e+00	0	1.4695e+00	0	0	0	1.5694e+02	4.6115e+00	0
	Median	0	1.2131e+01	0	3.1960e+00	0	0	0	2.8418e+02	2.6313e+01	0
	Worst	0	1.6156e+01	0	5.8169e+00	0	0	0	5.8474e+02	4.5152e+01	0
	Average	0	1.2546e+01	0	3.8563e+00	0	0	0	3.0164e+02	2.8122e+01	0
	Std	0	5.8941e+00	0	4.4961e-01	0	0	0	4.5110e+01	4.5113e+00	0
1000D	Best	0	4.0546e+01	0	2.5841e+01	0	0	0	1.0561e+03	5.4512e+01	0
	Median	0	4.6414e+01	0	3.0655e+01	0	0	0	1.2564e+03	8.5612e+01	0
	Worst	0	5.5642e+01	0	4.0229e+01	0	0	0	2.8645e+03	1.2348e+02	0
	Average	0	4.8875e+01	0	3.2145e+01	0	0	0	1.4628e+03	9.2524e+01	0
	Std	0	6.8005e+00	0	4.5145e+00	0	0	0	3.6545e+02	1.5116e+01	0

All the results below 1e-14 have been approximated to 0.

Table 8. 3 Experimental Results of 25 runs obtained by SaDE-MMTS on Problem 11-19

		F11	F12	F13	F14	F15	F16	F17	F18	F19
50D	Best	3.4100e-05	0	4.0288e-01	5.5227e-09	0	0	3.3926e-01	2.0392e-03	0
	Median	4.7037e-05	0	1.4239e+00	3.3544e-08	0	0	3.4407e-01	8.1103e-03	0
	Worst	8.8619e-05	0	1.2352e+01	5.0622e-08	0	0	6.9469e-01	1.1422e-02	0
	Average	5.1919e-05	0	4.2352e+00	3.9366e-08	0	0	4.7842e-01	9.3882e-03	0
	Std	7.1961e-06	0	4.6191e+00	6.2614e-09	0	0	5.4941e-02	1.8441e-03	0
100D	Best	6.3085e-05	0	5.0705e+00	3.2652e-04	0	0	3.1848e+00	8.1542e-03	0
	Median	1.3069e-04	0	2.3406e+01	8.2050e-03	0	0	1.0016e+01	3.1257e-02	0
	Worst	4.0392e-04	0	4.5811e+01	1.6313e-02	0	0	3.0398e+01	7.5505e-02	0
	Average	2.0017e-04	0	3.3080e+01	1.0282e-02	0	0	1.1719e+01	4.7029e-02	0
	Std	2.1949e-04	0	5.1641e+00	6.1658e-03	0	0	5.1648e+00	6.4165e-03	0
200D	Best	9.9286e-05	0	3.6922e+01	7.0715e-03	0	0	2.4531e+01	7.0611e-02	0
	Median	1.9458e-04	0	8.2768e+01	1.6604e-02	0	0	3.0080e+01	2.9554e-01	0
	Worst	5.5211e-04	0	9.3254e+01	2.2528e-02	0	0	5.6547e+01	3.6500e-01	0
	Average	2.3926e-04	0	8.8902e+01	1.5751e-02	0	0	3.5030e+01	3.3546e-01	0
	Std	7.0554e-05	0	1.4615e+01	1.7506e-02	0	0	5.0456e+00	3.5964e-01	0
500D	Best	9.1615e+00	0	2.5314e+02	5.1634e-02	0	0	6.8454e+01	1.0024e+00	0
	Median	2.0164e+01	0	3.0522e+02	2.6440e-01	0	0	8.8405e+01	1.1516e+00	0
	Worst	3.6515e+01	0	4.6881e+02	6.4615e-01	0	0	1.6554e+02	1.3578e+00	0
	Average	2.5328e+01	0	3.2784e+02	4.0112e-01	0	0	9.8075e+01	1.1898e+00	0
	Std	8.5114e+00	0	5.0064e+01	4.5131e-01	0	0	2.5641e+01	2.3564e-01	0
1000D	Best	6.9116e+01	0	4.5582e+02	9.8451e+01	0	0	1.5451e+02	5.6018e+01	0
	Median	9.3347e+01	0	5.4631e+02	1.3401e+02	0	0	1.8556e+02	7.6545e+01	0
	Worst	5.1612e+02	0	9.9085e+02	1.5131e+02	0	0	2.0164e+02	9.6299e+01	0
	Average	1.8516e+02	0	6.5502e+02	1.4007e+02	0	0	1.9045e+02	8.2054e+01	0
	Std	2.1674e+02	0	8.7559e+01	3.0847e+01	0	0	3.4516e+01	1.2610e+01	0

All the results below 1e-14 have been approximated to 0.

Table 8. 4 Comparison on the average error of 25 runs obtained by four algorithms

		F1	F2	F3	F4	F5
50D	DE	0.00E+00	3.29E-01	2.90E+01	1.51E-13	0.00E+00
	SADE-MMTS	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
100D	DE	0.00E+00	4.34E+00	7.81E+01	4.23E-13	0.00E+00
	SADE-MMTS	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
200D	DE	0.00E+00	1.93E+01	1.77E+02	3.58E-12	0.00E+00
	SADE-MMTS	0.00E+00	1.34E+00	0.00E+00	8.08E-02	0.00E+00
500D	DE	0.00E+00	5.33E+01	4.74E+02	9.22E-03	0.00E+00
	SADE-MMTS	0.00E+00	1.25E+01	0.00E+00	3.85E+00	0.00E+00
1000D	DE	0.00E+00	8.44E+01	9.69E+02	1.32E+00	0.00E+00
	SADE-MMTS	0.00E+00	4.88E+01	0.00E+00	3.21E+01	0.00E+00
		F6	F7	F8	F9	F10
50D	DE	1.42E-13	0.00E+00	3.54E+00	2.73E+02	0.00E+00
	SADE-MMTS	0.00E+00	0.00E+00	4.13E-09	1.35E-01	0.00E+00
100D	DE	3.13E-13	0.00E+00	3.47E+02	5.06E+02	0.00E+00
	SADE-MMTS	0.00E+00	0.00E+00	3.05E-04	3.18E-01	0.00E+00
200D	DE	6.54E-13	0.00E+00	5.33E+03	1.01E+03	0.00E+00
	SADE-MMTS	0.00E+00	0.00E+00	2.67E+01	1.24E+00	0.00E+00
500D	DE	1.65E-12	0.00E+00	6.11E+04	2.52E+03	0.00E+00
	SADE-MMTS	0.00E+00	0.00E+00	3.01E+02	2.81E+01	0.00E+00
1000D	DE	3.30E-12	0.00E+00	2.46E+05	5.13E+03	0.00E+00
	SADE-MMTS	0.00E+00	0.00E+00	1.46E+03	9.25E+01	0.00E+00
		F11	F12	F13	F14	F15
50D	DE	5.60E-05	5.27E-13	2.44E+01	2.58E-08	0.00E+00
	SADE-MMTS	5.19E-05	0.00E+00	4.23E+00	3.93E-08	0.00E+00
100D	DE	1.29E-04	6.18E-11	6.17E+01	1.30E-07	0.00E+00
	SADE-MMTS	2.00E-04	0.00E+00	3.30E+01	1.02E-02	0.00E+00
200D	DE	2.59E-04	9.36E-10	1.36E+02	2.71E-07	0.00E+00

	SaDE-MMTS	2.39E-04	0.00E+00	8.89E+01	1.57E-02	0.00E+00
500D	DE	6.71E-04	6.98E-09	3.58E+02	9.01E-07	0.00E+00
	SaDE-MMTS	2.53E+01	0.00E+00	3.27E+02	4.01E-01	0.00E+00
1000D	DE	1.35E-03	1.70E-08	7.29E+02	9.95E-01	0.00E+00
	SaDE-MMTS	1.85E+02	0.00E+00	6.55E+02	1.40E+02	0.00E+00
		F16	F17	F18	F19	SumRank
50D	DE	1.51E-09	6.83E-01	1.20E-04	0.00E+00	30
	SaDE-MMTS	0.00E+00	4.78E-01	9.38E-03	0.00E+00	21
100D	DE	3.53E-09	1.28E+01	2.86E-04	0.00E+00	29
	SaDE-MMTS	0.00E+00	1.17E+01	4.70E-02	0.00E+00	22
200D	DE	7.26E-09	3.70E+01	4.70E-04	0.00E+00	30
	SaDE-MMTS	0.00E+00	3.50E+01	3.35E-01	0.00E+00	21
500D	DE	2.05E-08	1.11E+02	1.22E-03	0.00E+00	29
	SaDE-MMTS	0.00E+00	9.80E+01	1.18E+00	0.00E+00	22
1000D	A1	4.19E-08	2.35E+02	2.37E-03	0.00E+00	29
	SaDE-MMTS	0.00E+00	1.90E+02	8.20E+01	0.00E+00	22

All the results below 1e-14 have been approximated to 0.

Table 8. 5 Matlab average running time on the 25 runs for the SaDE-MMTS

50D	100D	200D	500D	1000D
5mins	12mins	25mins	120mins	320mins

Table 8. 6 Comparison on the average error of 25 runs obtained by SaDE-MMTS and SaDE with JADE mutation on 4 functions.

1000D functions	F3	F9	F13	F19
<i>SaDE</i>	1.24e+10	3.51e+03	3.15e+09	6.27e+01
<i>SaDE-MMTS</i>	0	9.25e+01	6.55e+02	0

8.4 Conclusions

In order to solve complex problem instances especially with high dimensions, this chapter proposed SaDE-MMTS incorporating the JADE mutation strategy with small sized archive and two crossover operators (binomial crossover and exponential crossover) as well as no crossover option. Moreover, SaDE-MMTS employs adaptation of trial vector generation strategies and control parameters in the Self-adaptive Differential Evolution (SaDE) by learning from their previous experiences to suit different characteristics of the problems and different requirements of evolution phases. Furthermore, a Modified Multiple Trajectory Search (MMTS, i.e. line search) is used to improve the solutions generated by the SaDE.

The SaDE-MMTS hybridization benefits from SaDE, JADE mutation strategy, different crossover operators and line search to solve large scale continuous optimization problems with diverse characteristics. Exploration as well as exploitation capabilities are also enhanced simultaneously by combining these schemes. The SaDE-MMTS is simple and easy to implement possessing fast convergence while diversity is preserved. In our future work, more investigation will be conducted on stable parameter settings, additional DE-mutation strategies and modifications will be integrated within this current framework to improve further the scalability of the SaDE-MMTS. Furthermore, as we mentioned, the scalability of an optimization method depended on two main features, exploration capability of the optimization method and the robustness of search strategy over any dimensionality. More investigations will be conducted on improving these characteristics.

Chapter 9

Conclusions and Future work

In this chapter, the contributions of the thesis are summarized and some future research directions are suggested.

9.1 Conclusions

In order to enhance the diversity of the swarm in the basic PSO and Comprehensive Learning PSO (CLPSO), a diversity-enhancement process which uses an external memory of selected past solutions had been proposed to discourage premature convergence. An external memory was formed to hold selected past solutions with good diversity. Selected solutions from the external memory were periodically injected into the swarm to enhance the diversity. Experiments had been conducted on multimodal and composition test problems with and without coordinate rotations. The test results had indicated improved performance of the Diversity Enhanced PSO and CLPSO, namely DivEnh-PSO and DivEnh-CLPSO, in solving multimodal problems when compared to the basic PSO and CLPSO implementations without the diversity enhancement.

A hybrid evolutionary algorithm which improved the dynamic multi-swarm particle swarm

optimizer (DMS-PSO) by hybridizing with the Harmony Search (DMS-PSO-HS) had been presented. This new approach merged a harmony search into each sub-swarm of DMS-PSO after generating new solutions of PSO during every iteration. Different from the existing multi-swarm PSOs or local versions of PSO, the sub-swarms applied here were dynamic with small sizes as required by the harmony search. The whole DMS-PSO population was divided into a certain number of sub-swarms, each of which was also the individual HS population. These sub-swarms were regrouped frequently by using various regrouping schedules and information was exchanged among sub-swarms. Improved performance on the strongly multimodal and composition test functions had been demonstrated when comparing the DMS-PSO-HS with both the original DMS-PSO and the original HS.

To the best of our knowledge, all multi-objective particle swarm optimization (MOPSO) implementations exclusively used the non-dominated solutions in the external archive of the current generation as the *gbest* or *lbest* for every particle. As this approach emphasized the elitism at the expense of diversity, this approach had the potential to converge to a small section of the true Pareto Front if the size of the non-dominated set was small for several generations during the evolution. In order to resolve this flaw, we had proposed that the *gbest* or *lbest* of every particle should be chosen from a non-domination sorted archive of reasonably large size. As no current MOPSO variant took into consideration the actual positions of the two guides, it was possible that the two guides could be located far apart in the parameter space. In this situation, the search behavior of the MOPSO could potentially become chaotic. To overcome this, a novel two local best (*lbests*) based MOPSO had been proposed, in which the two *lbests* were located close to each other in the vicinity of the best fronts, i.e. two guides for each particle were selected from the neighborhood in both parameter space and objective space. The 2LB-MOPSO was able to focus the search around small regions in the vicinity of the best fronts while maintaining the diversity due to the usage of two *lbests* chosen from the special binning procedure. Our simulation results had demonstrated the improved performances of these contributions over the state-of-the-art MOEAs. The 2LB-MOPSO was applied to design multi-objective robust PID controllers for two MIMO systems, as well as a Multi-objective Monopulse Antennas system.

The ε -dominance concept integrated with an ensemble of external archives into the multi-objective particle swarm optimization algorithm (MOPSO) had been proposed. With this approach, the difficulty of tuning the numerical value of the ε parameter could be overcome

for every objective when solving different problem as well as when solving a single problem over different stages of evolution. The experimental results showed that an ensemble of external archives with different ε values yielded good spread and convergence to the true Pareto-optimal front for difficult problems with different characteristics. Furthermore, the proposed approach was applied to solve the optimal asset allocation problem in Portfolio Optimization. The experimental results had also shown the superior performance of the ensemble of external archives with different ε values over implementations with only one archive. As a conclusion, many other multiobjective optimization nature-inspired algorithms can be also improved with Ensemble of External Archives.

Continuing to take advantage of the Ensemble approach, an ensemble of different neighborhood sizes of the subproblems was applied into MOEA/D (ENS-MOEA/D). The trial-and-error procedure of tuning the suitable numerical values for neighborhood sizes when solving different problems could be omitted. Moreover, the ensemble approach was improved with self-adaptation. In this manner, during one whole evolution stage, the requirement of reasonable values for this critical parameter is also needed to be adjusted to rationalize the computational effort. Our experimental results had shown that ENS-MOEA/D yielded superior performance compared with the same MOEA/D with a single fixed neighborhood size.

Finally, in order to solve complex problem instances especially with high dimensions, another hybrid evolutionary algorithm had been proposed, known as the SaDE-MMTS. This approach incorporated the JADE mutation strategy with small sized archive as well as two crossover operators (binomial crossover and exponential crossover) and a no crossover option. SaDE-MMTS employed self-adaptation of trial vector generation strategies and control parameters in the SaDE by learning from their previous experiences to suit different characteristics of the problems and different requirements of evolution phases. Furthermore, MMTS was applied frequently to refine several diversely distributed SaDE solutions at different search stages. In order to satisfy both the requirements of global search and local search, ensemble of initial MMTS step sizes with a self-adaption scheme had been defined based on the average of all mutual dimension-wise distances between current SaDE population members. The success rates of both SaDE and MMTS were determined and compared, so that consequent function evaluations for both search algorithms were assigned

proportionally to their recent past performance Exploration as well as exploitation capabilities had also been enhanced simultaneously by combining these schemes.

9.2 Future work

- From the research experiences obtained from experimenting with the several proposed algorithms, such as MOPSO with Ensemble of External Archives, ENS-MOEA/D and SaDE-MMTS, it was understood that the ensemble approach performed better when the individual techniques used have distinct characteristics. It is natural to extend the research into the following directions: the performance of PSOs on computationally expensive problems can be improved by introducing ensemble of values for the PSO parameter settings, as well as ensemble of topologies application into PSO populations. However, there remain challenging problems to be investigated, including:
 - How to choose different values for the critical PSO parameters, such as inertia weight ω , the maximum velocity V_{max} , and so on.
 - Among different topologies of Static, Ring, Star, Von Neumann as well as Pyramid, which ones should be selected to form the ensemble of topologies for the PSO population.
 - What are the optimal number of parameters, the optimal number of values for each parameter, as well as the optimal number of topologies to be used.
- In multiobjective optimization, the non-domination sorting procedure is normally used to build the approached Pareto Front, while the Decomposition is also used as a basic strategy in traditional multi-objective optimization. However, no work has published yet on combination of both techniques. The new idea is arisen here. Each sub-problem in MOEA/D is optimized by using information of the non-dominated solutions from its neighboring sub-problems, instead of using information of randomly selected neighboring sub-problems as in the original MOEA/D. In this configuration, taking the advantage of non-domination sorting, the new solutions are generated based on the most promising neighboring sub-problems. The good spread of solutions can be also achieved with fast convergence when solving the MO test problems involving arbitrary complexities.

- Most of the current Decomposition based MOEAs are DE based approaches. It is worth to investigate other EAs or SIs as the essential search method in the MOEA/D. For instance, 2LB-MOPSO based MOEA/D, the two lbests could be selected from its neighboring subproblems in different ways. If two lbests are chosen from dissimilar subproblems, it will advance the information exchange among the diverse subproblems, and thus it speeds up the convergence of the whole population; while if two lbests are from nearest subproblems, it is good for local search in a local neighborhood, which maintains the diversity for the whole population. However, in some cases, two lbests from dissimilar subproblems benefit on diversity recovery if some subproblems are trapped in a locally optimal regions; while two lbests are from nearest subproblems or even the same subproblem within the global optima region is also able to facilitate the convergence if the global optima area is already found.
- In DE, binomial crossover and exponential crossover are attracted the attentions differently on solving the problems with various characteristics. When generating the new target vectors from their parents and mutant vectors, the former one is applied by emphasizing the convergent ability of the population, while the latter one is used to tend to maintain the diversity. Experimental investigations will be carried out on both crossover operators on different test problems. Modifications and combinations will be made on them, to satisfy the various requirements when solving optimization problems, involving efficiency, robustness, complexity and so on.
- As one of the most difficult topics in optimization, the scalability of an optimization method is an important challenge to be tackled further. As we mentioned in previous work, to solve those problems, exploration capability of the optimization method and the robustness of search strategy over any dimensionality are two most significant features of optimization methods. More investigations will be conducted on improving these characteristics. An effective archive may be used to improve the exploration capability of search methods. Since the solution space of a problem increases exponentially with the number of problem dimensions, more efficient search route of the population should be guided by the archive to explore all promising regions within a given time budget. Moreover, the key parameters of each algorithm should be reinvestigated. Some dynamically changing parameter settings or ensemble of parameter settings could be incorporated with the original algorithms. If these important parameters are sensitive to

the dimensionality of problems, they can be self-adapted effectively according to the different problem properties and different search requirements during different evolution stages.

- We will conduct more research works applying the developed algorithms. Our developed EAs and SIs can handle many difficult test problems successfully. Hence, it is worth applying our research to solve process control, energy systems, economy problems and tracking problems. We are planning to investigate the use of SI or EA as a learning algorithm for the target dynamics in a target tracking problem. Our objective is to study the most appropriate models of optimization scheme in modeling various dynamics.

Author's Publications

Journals Papers

1. S. Z. Zhao and P. N. Suganthan, "Multi-objective Evolutionary Algorithm with Ensemble of External Archives", *Int. J. of Innovative Computing, Information and Control*, Vol. 6, No. 1, pp 1713-1726, April 2010.
2. S. Z. Zhao and P. N. Suganthan, "Two *lbests* Based Multi-objective Particle Swarm Optimizer", *Engineering Optimization*, vol. 38, issue 4, April 2011, Pages 3735-3742.
3. S. Z. Zhao, P. N. Suganthan and S. Das, "Self-adaptive Differential Evolution with Multi-trajectory Search for Large Scale Optimization", *Soft Computing*, vol. 43, issue 1, January 2011, pages 1 - 17.
4. S. Z. Zhao and P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer with harmony search", *Expert Systems with Applications*, vol. 38, issue 4, April 2011, Pages 3735-3742..
5. S. Z. Zhao, M. Willjuice Iruthayarajan, S. Baskar and P. N. Suganthan, "Multi-objective Robust PID Controller Tuning using Two *lbests* Multi-objective Particle Swarm Optimization" *Information Sciences*, vol. 181, issue 16, 15 August 2011, Pages 3323-3335..
6. S. Z. Zhao, P. N. Suganthan and Q. F. Zhang, "Decomposition Based Multiobjective Evolutionary Algorithm with an Ensemble of Neighborhood Sizes", *IEEE Trans. on Evolutionary Computation*, Aug 2011, DOI 10.1109/TEVC.2011.2166159.

7. A. Zhou, B. Y. Qu, H. Li, S. Z. Zhao, P. N. Suganthan and Q. Zhang, "Multiobjective Evolutionary Algorithms: A Survey of the State-of-the art," *Swarm and Evolutionary Computation*, Mar 2011, DOI: 10.1016/j.swevo.2011.03.001.
8. S. Z. Zhao and P. N. Suganthan, "Diversity Enhanced Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Problems", minor revised and resubmitted to *Information Sciences*, August. 2010.
9. S. Z. Zhao and P. N. Suganthan, "Investigations into the Exponential Crossover of Differential Evolution", submitted to *IEEE Trans. on Evolutionary Computation*, August 2011.

Conference Publications

1. S. Z. Zhao, J. J. Liang, P. N. Suganthan, M. F. Tasgetiren, "Dynamic Multi-swarm Particle Swarm Optimizer with Local Search for Large Scale Global Optimization", *IEEE Congress on Evolutionary Computation*, pp. 3845-3852, Hong Kong, June 2008.
2. S. Z. Zhao, P. N. Suganthan, "Diversity Enhanced Particle Swarm Optimizer for Global Optimization of Multimodal Problems", *IEEE Congress on Evolutionary Computation*, Norway, pp. 590-597, May 2009.
3. V. L. Huang, S. Z. Zhao, R. Mallipeddi, P. N. Suganthan, "Multi-objective Optimization Using Self-adaptive Differential Evolution Algorithm", *IEEE Congress on Evolutionary Computation*, pp. 190-194, Norway, May 2009.
4. S. Z. Zhao, P. N. Suganthan and S. Das, "Dynamic Multi-Swarm Particle Swarm Optimizer with Sub-regional Harmony Search", *IEEE Congress on Evolutionary Computation*, Barcelona, Spain, July 2010.
5. G. G. Roy, P. Chakroborty, S. Z. Zhao, S. Das and P. N. Suganthan, "Artificial Foraging Weeds for Global Numerical Optimization over Continuous Spaces", *IEEE Congress on Evolutionary Computation*, Barcelona, Spain, pp. 1189-1196, July 2010.
6. S. Z. Zhao, B. Y. Qu, M. Willjuice Iruthayarajan, S. Baskar and P. N. Suganthan, "Multi-objective Robust PID Controller Tuning using Multi-objective Differential Evolution" *the Eleventh International Conference on Control, Automation, Robotics and Vision (ICARCV 2010)*, Singapore, December, 2010.

7. S. Z. Zhao, P. N. Suganthan and S. Das, "Self-adaptive Differential Evolution with Modified Multi-trajectory Search for CEC'2010 Large Scale Optimization" *International Conference on Swarm, Evolutionary and Memetic Computing (SEMCCO 2010)*, vol. 6466/2010, 1-10, Chennai, India, Dec 16-18, 2010.
8. S. Z. Zhao, P. N. Suganthan and B. Y. Qu, "Multiobjective Particle Swarm Optimizer with Dynamic epsilon-dominance Sorting", *Second World Congress on Nature and Biologically Inspired Computing (NaBIC2010)*, Kitakyushu, Japan, Dec 15 – 17, 2010.
9. S. Z. Zhao, P. N. Suganthan, S. Pal, S. Das and A. Basak, "Multi-Objective Design of Monopulse Antenna with Two-lbests based Multi-objective Particle Swarm Optimizer", *The First International Conference On Computational Problem-Solving (ICCP2010)*, Li Jiang, China, Dec 3-5, 2010. (Best Student Paper Award).
10. Subhrajit Roy, Sk Minhazul Isam, Saurav Ghosh, S. Z. Zhao, P.N. Suganthan and Swagatam Das, "Design of Two Channel Quadrature Mirror Filter Bank: A Multi-objective approach", accepted by SEMCCO 2011.
11. Sk Minhazul Isam, Saurav Ghosh, Subhrajit Roy, S. Z. Zhao and P.N. Suganthan and Swagatam Das, "Synthesis and Design of Thinned Planar Concentric Circular Antenna Array- A Multi-Objective Approach", accepted by SEMCCO 2011.
12. Saurav Ghosh, Subhrajit Roy, Sk. Minhazul Islam, S. Z. Zhao, P.N. Suganthan and Swagatam Das, "Non-uniform Circular-shaped Antenna Array Design and Synthesis- A Multi-Objective Approach", accepted by SEMCCO 2011.

Technical Report

1. Qingfu Zhang, Aimin Zhou, S. Z. Zhao, P. N. Suganthan and Wudong Liu, Santosh Tiwari, "Multiobjective Optimization Test Instances for the CEC 2009 Special Session and Competition", Technical Report CES-887, University of Essex and Nanyang Technological University, 2008.

Bibliography

- [1] A. M. Anile, V. Cutello, G. Nicosia, R. Rascuna, and S. Spinella, "Comparison among Evolutionary Algorithms and Classical Optimization Methods for Circuit Design Problems," *Proc of IEEE Conference on Evolutionary Computation*. vol. 1 Vancouver, Canada, 2005, pp. 765-772.
- [2] J. Kennedy and R. C. Eberhart, "A new optimizer using particle swarm theory," *Proc. of the Sixth International Symposium on Micromachine and Human Science*, Nagoya, Japan, pp. 39-43, 1995.
- [3] Y. Jin, "A Comprehensive Survey of Fitness Approximation in Evolutionary Computation," *Soft Computing*, vol. 9, pp. 3-12, 2005.
- [4] L. Rokach, "Taxonomy for Characterizing Ensemble Methods in Classification Tasks: A Review and Annotated Bibliography," *Computational Statistics and Data Analysis*, vol. 53, pp. 4046-4072, July 2009.
- [5] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin: Springer-Verlag, 1992.
- [6] J. H. Holland, "Outline for a logical theory of adaptive systems," *Journal of the Association for Computing Machinery*, 3:297-314, 1962.
- [7] J. H. Holland, *Adaptation in natural and artificial systems*. The University of Michigan Press, Ann Arbor, MI, 1975.
- [8] I. Rechenberg, "Cybernetic solution path of an experimental problem," Royal Aircraft Establishment, Library translation No. 1122, Farnborough, Hants., UK, August 1965.
- [9] I. Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, Stuttgart, 1973.
- [10] I. Rechenberg, *Evolutionsstrategie*, volume 1 of *Werkstatt Bionik und Evolutionstechnik*. Frommann- Holzboog, Stuttgart, 1994.
- [11] H.-P. Schwefel, *Kybernetische Evolution als Strategie der experimentellen Forschung in der Stromungstechnik*. Diplomarbeit, Technische Universität Berlin, 1965.
- [12] H.-P. Schwefel, "Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie," in *Interdisciplinary Systems Research*. Birkhauser, Basel, vol. 26, 1977.

- [13] L. J. Fogel, "Toward inductive inference automata," *Proc. of the International Federation for Information Processing Congress*, pages 395-399, Munich, 1962.
- [14] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence through Simulated Evolution*. Wiley, New York, 1966.
- [15] X. Yao and Y. Liu, "Scaling up evolutionary programming algorithms," *Proc. of the Seventh Annual Conference on Evolutionary Programming (EP98)*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1998, pp. 103--112.
- [16] D. B. Fogel, *System identification through simulated evolution: A machine learning approach to modeling*. Needham Heights, MA 02194: Ginn Press, 1991.
- [17] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. on Evolutionary Computation*, vol. 3, pp. 82-102, 1999.
- [18] R. Mallipeddi and P. N. Suganthan, "Evaluation of Novel Adaptive Evolutionary Programming on Four Constraint Handling Techniques," *Proc. of IEEE Congress on Evolutionary Computation*, Hong Kong, June 2008.
- [19] R. Storn and K. V. Price, "Differential evolution-A simple and efficient heuristic for global optimization over continuous Spaces," *Journal of Global Optimization*, vol.11, pp.341-359. 1997.
- [20] 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.
- [21] K. V. Price, "An Introduction to Differential Evolution". in: *Corne, D., Dorigo, M., and Glover, F. (eds.): New Ideas in Optimization*. McGraw-Hill, London (UK) 79-108, 1999.
- [22] R. Gänperle, 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, pp.293-298, 2002.
- [23] J. Rönkkönen, S. Kukkonen, and K. V. Price. "Real-parameter optimization with differential evolution," *Proc. of 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, Edinburgh, Scotland, IEEE Press, Sept. 2005, pp. 506–513.
- [24] R. Storn, K. V. Price, and J. Lampinen, "*Differential Evolution - A Practical Approach to Global Optimization*", in *Natural Computing Series* Berlin, 2005.
- [25] R. C. Eberhart, P. K. Simpson and R. W. Dobbins *Computational Intelligence PC tools*, Boston: MA: Academic Press Professional.
- [26] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," *Proc. of the IEEE Congress on Evolutionary Computation (CEC 1998)*, Piscataway, NJ, pp. 69-73, 1998.

- [27] H. Y. Fan and Y. Shi, "Study on Vmax of particle swarm optimization," *Proc. of the Workshop on Particle Swarm Optimization 2001*, Indianapolis, IN, 2001.
- [28] Y. Shi and R. C. Eberhart, "Particle swarm optimization with fuzzy adaptive inertia weight," *Proc. of the Workshop on Particle Swarm Optimization*, Indianapolis, IN, 2001.
- [29] P. J. Angeline, "Evolutionary optimization versus particle swarm optimization: philosophy and performance differences," *Evolutionary Programming VII: Proceedings of the Seventh Annual Conference on Evolutionary Programming*, 1998.
- [30] M. Clerc, "The swarm and the queen: towards a deterministic and adaptive particle swarm optimization," *Proc of the IEEE Congress on Evolutionary Computation (CEC 1999)*, pp. 1951-1957, 1999.
- [31] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 58-73, 2002.
- [32] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2000)*, San Diego, CA, pp. 84-88, 2000.
- [33] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," *Evolutionary Programming VII: Proceedings of the Seventh Annual Conference on Evolutionary Programming*, New York, pp. 591-600, 1998.
- [34] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," *Proc. of the 1999 Congress on Evolutionary Computation*, Piscataway, NJ, pp. 1945-1950, 1999.
- [35] K. E. Parsopoulos and M.N. Vrahatis, "UPSO: 'A unified particle swarm optimization scheme'", in *Proceedings of the International Conference of Computational Methods in Sciences and Engineering, ICCMSE 2004*, Zeist, The Netherlands, 2004, pp. 868-873.
- [36] K. E. Parsopoulos and M. N. Vrahatis, "Parameter selection and adaptation in Unified Particle Swarm Optimization," *Mathematical and Computer Modelling* 46, pp.198-213, 2007.
- [37] A. Ratnaweera, S. Halgamuge, and H. Watson, "Self-organizing hierarchical particle swarm optimizer with time varying accelerating coefficients," *IEEE Transactions on Evolutionary Computation*, Vol. 8 (3), pp. 240 - 255, June 2004.
- [38] Z. He, C. Wei, L. Yang, X. Gao, S. Yao, R. C. Eberhart, and Y. Shi, "Extracting rules from fuzzy neural network by particle swarm optimization," *Proc. of IEEE Congress on Evolutionary Computation (CEC 1998)*, Anchorage, Alaska, USA, 1998.
- [39] X. Chen and Y. Li, "An improved stochastic PSO with high exploration ability," *Proc. of IEEE Swarm Intell. Symp.*, May 2006, pp. 228-235.

- [40] J. Kennedy, R. C. Eberhart and Y. Shi, *Swarm Intelligence*, Morgan Kaydmann, 2001.
- [41] J. Kennedy, "Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance," *Proc. of IEEE Congress on Evolutionary Computation (CEC 1999)*, Piscataway, NJ. pp. 1931-1938, 1999.
- [42] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," *Proc. of the IEEE congress on evolutionary computation (CEC2002)* pp. 1671–1676, 2002.
- [43] R. Mendes, P. Cortes, M. Rocha, and J. Neves, "Particle swarms for feedforward neural net training," *Proc. of the international joint conference on neural networks*, pp. 1895–1899, 2002.
- [44] R. Mendes, J. Kennedy and J. Neves, "Watch thy neighbor or how the swarm can learn from its environment," *Proc. of the IEEE swarm intelligence symposium (SIS)*, pp. 88–94, 2003.
- [45] P. N. Suganthan, "Particle swarm optimizer with neighborhood operator," *Proc. of the IEEE Congress on Evolutionary Computation (CEC 1999)*, Piscataway, NJ, pp. 1958-1962, 1999.
- [46] X. Hu and R. C. Eberhart, "Multi-objective optimization using dynamic neighborhood particle swarm optimization", *Proc. of the IEEE Congress on Evolutionary Computation*, Honolulu, Hawaii USA. pp. 1677-1681.
- [47] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler, maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8, pp. 204-210, 2004.
- [48] J. J. Liang, A. K. Qin, P. N. Suganthan and S. Baskar, "Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions", *IEEE T. on Evolutionary Computation*, Vol. 10, No. 3, pp. 281-295, June 2006.
- [49] S. Chamaani, S.A. Mirtaheri and M.A. Shooredeli, "Design of very thin wide band absorbers using modified local best particle swarm optimization," *International Journal of Electronics and Communications*, August 2008, pp. 549-556.
- [50] V. Miranda and N. Fonseca, "New evolutionary particle swarm algorithm (EPSO) applied to voltage/VAR control," *The 14th Power Systems Computation Conference (PSCC'02)*, Servile, Spain, 2002.
- [51] T. M. Blackwel and P. J. Bentley, "Don't push me! Collision-avoiding swarms," *Proc. of the IEEE Congress on Evolutionary Computation (CEC 2002)*, Honolulu, Hawaii USA, 2002.
- [52] T. Krink, J. S. Vesterstroem, and J. Riget, "Particle swarm optimisation with spatial particle extension," *Proc. of IEEE Congress on Evolutionary Computation (CEC 2002)*, Honolulu, Hawaii USA, 2002.

- [53] M. Lovbjerg and T. Krink, "Extending particle swarms with self-organized criticality," *Proc. of IEEE Congr. Evol. Comput.*, May 2002, vol. 2, pp. 1588–1593.
- [54] J. Robinson, S. Sinton, and Y. Rahmat-Samii, "Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna," *IEEE Antennas and Propagation Society International Symposium and URSI National Radio Science Meeting*, San Antonio, TX, 2002.
- [55] Q. He and L. Wang L, "A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization," *Applied Mathematics and Computation*, Volume: 186 Issue: 2 pp: 1407-1422, MAR 15 2007.
- [56] A. El-Dib, H. Youssef, M. El-Metwally and Z. Osman, "Load flow solution using hybrid particle swarm optimization," *Proc. of Int. Conf. Elect., Electron., Comput. Eng.*, Sep. 2004, pp. 742–746.
- [57] S. Naka, T. Genji, T.Yura, and Y. Fukuyama, "A hybrid particle swarm optimization for distribution state estimation," *IEEE Trans. Power Syst.*, pp. 60–68, Feb. 2003.
- [58] P. Angeline, "Using selection to improve particle swarm optimization," *Proc. of IEEE Int. Conf. Evol. Comput.*, May 1998, pp. 84–89.
- [59] V. Miranda and N. Fonseca, "EPSO best of two worlds meta-heuristic applied to power system problems," *Proc. of IEEE Congr. Evol. Comput.*, May 2002, vol. 2, pp. 1080–1085.
- [60] V. Miranda and N. Fonseca, "EPSO evolutionary particle swarm optimization, a new algorithm with applications in power systems," in *IEEE/PES Transmission and Distribution Conf. Exhibition Asia Pacific*, Oct. 2002, vol. 2, pp. 745–750.
- [61] W. J. Zhang and X.-F. Xie, "DEPSO: Hybrid particle swarm with differential evolution operator," *Proc. of IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2003, vol. 4, pp. 3816–3821.
- [62] H. Talbi and M. Batouche, "Hybrid particle swarm with differential evolution for multimodal image registration," *Proc. of IEEE Int. Conf. Ind. Technol.*, Dec. 2004, vol. 3, pp. 1567–1572.
- [63] P. Moore and G.Venayagamoorthy, "Evolving combinational logic circuits using particle swarm, differential evolution and hybrid DEPSO," *Int. J. Neural Syst.*, vol. 16, no. 2, pp. 163–177, 2006.
- [64] S. Kannan, S. Slochanal, and N. Padhy, "Application of particle swarm optimization technique and its variants to generation expansion problem," *ELSERNIER Electric Power Syst. Res.*, vol. 70, no. 3, pp. 203–210, Aug. 2004.
- [65] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, pp. 225–239, June 2004.

- [66] S. K. S. Fan and E. Zahara, "A hybrid simplex search and particle swarm optimization for unconstrained optimization," *European Journal of Operational Research*, Volume: 181 Issue: 2 Pages: 527-548, SEP 1 2007.
- [67] Z. G. Lian, X.S. Gu and B. Jiao, "A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan," *Chaos, Solitons & Fractals*, Volume 35, Issue 5, pp. 851-861, March 2008.
- [68] L. D. S. Coelho, "A quantum particle swarm optimizer with chaotic mutation operator," *Chaos, Solitons & Fractals*, Volume 37, Issue 5, pp. 1409-1418, September 2008.
- [69] J. Sun, B. Feng and W. Xu, "Particle swarm optimization with particles having quantum behavior", In *Proceedings of congress on evolutionary computation*, Portland (OR, USA), 2004. p. 325–31.
- [70] J. Sun, B. Feng and W. Xu, "Adaptive parameter control for quantum-behaved particle swarm optimization on individual level", In *Proceedings of IEEE international conference on systems, man and cybernetics*, Big Island (HI, USA), 2005. pp. 3049–54.
- [71] M. S. Arumugam and M.V.C. Rao, "On the improved performances of the particle swarm optimization algorithms with adaptive parameters, cross-over operators and root mean square (RMS) variants for computing optimal control of a class of hybrid systems," *Applied Soft Computing*, Volume 8, Issue 1, pp. 324-336, January 2008.
- [72] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," *Proc. of the World Multiconference on Systemics, Cybernetics and Informatics*, Piscataway, NJ, pp. 4104-4109, 1997.
- [73] A. Cervantes, I. Galvan and P. Isasi, "A comparison between the Pittsburgh and Michigan approaches for the binary PSO algorithm," *Proc. of the IEEE Congress on Evolutionary Computation (CEC 2000)*, vol. 1, pp. 290 - 297, 2005.
- [74] D. K. Agrafiotis and W. Cedeno, "Training feedforward neural networks using multi-phase particle swarm optimization," *Journal of Medicinal Chemistry*, pp. 1098-1107, 2002.
- [75] L. Y. Chuang, H.W. Chang and C.J. Tu, "Improved binary PSO for feature selection using gene expression data", *Computational Biology and Chemistry*, Volume: 32 Issue: 1 pp. 29-38, FEB 2008.
- [76] B. Ye, J. Sun and W.B. Xu, "Solving the hard knapsack problems with a binary particle swarm approach", *Computational Intelligence in Bioinformatics*, PT 3, PROCEEDINGS, pp. 155-163, 2006.
- [77] F. Afshinmanesh, A. Marandi and A. Rahimi-Kian, "A Novel Binary Particle Swarm Optimization Method Using Artificial Immune System," *The International Conference on Computer as a Tool (EUROCON 2005)*, vol. 1, pp.217 - 220, 2005.

- [78] K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimization method for constrained optimization problems," *Proc. of the Euro-International Symposium on Computational Intelligence*, 2002.
- [79] X. Hu and R. C. Eberhart, "Solving constrained nonlinear optimization problems with particle swarm optimization," *Proc. of the Sixth World Multiconference on Systemics, Cybernetics and Informatics 2002 (SCI 2002)*, Orlando, USA, 2002.
- [80] N. Iwasaki and K. Yasuda, "Adaptive particle swarm optimization using velocity feedback," *International Journal of Innovative Computing Information and Control*, Volume: 1, Issue: 3, pp. 369-380, SEP 2005.
- [81] R. A. Krohling and L.D. Coelho, "Coevolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems," *IEEE Transactions On Systems, Man, And Cybernetics-Part B: Cybernetics*, Volume: 36, Issue: 6, pp. 1407-1416, DEC 2006.
- [82] A. E. M. Zavala, A.H. Aguirre and E.R.V. Diharce, "Particle Evolutionary Swarm Optimization with linearly decreasing epsilon-tolerance," *Advances in Artificial Intelligence*, Volume: 3789, pp. 641-651, MICAI 2005.
- [83] R. Brits, A. P. Engelbrecht, and F. van den Bergh, "A niching particle swarm optimizer," *Proc. of the 4th Asia-Pacific Conference on Simulated Evolution and Learning 2002 (SEAL 2002)*, Singapore. pp. 692-696, 2002.
- [84] R. Brits, A. P. Engelbrecht and F. van den Bergh, "Scalability of niche PSO," *Proc. of the IEEE Swarm Intelligence Symposium 2003 (SIS 2003)*, Indianapolis, Indiana, USA. pp. 228-234, 2003.
- [85] M. Iwamatsu, "Multi-species particle swarm optimizer for multimodal function optimization," *IEICE Transactions on Information and Systems*, Volume: E89D, Issue: 3, pp. 1181-1187, MAR 2006.
- [86] J. H. Seo, C.H. Im and C.G. Heo, "Multimodal function optimization based on particle swarm optimization," *IEEE Transactions on Magnetics*, Volume: 42, Issue: 4, pp. 1095-1098, APR 2006.
- [87] P. S. Shelokar, P. Siarry and V.K. Jayaraman, "Particle swarm and ant colony algorithms hybridized for improved continuous optimization" *Applied Mathematics and Computation*, Volume: 188, Issue: 1, pp. 129-142, MAY 1 2007.
- [88] Y. T. Kao and E. Zahara, "A hybrid genetic algorithm and particle swarm optimization for multimodal functions," *Applied Soft Computing*, Volume: 8, Issue: 2, pp. 849-857, MAR 2008.
- [89] J. Kennedy and W. M. Spears, "Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator," *Proc. of International conference on evolutionary computation*, pp. 78-83, 1998.

- [90] G. Pamparä N. Franken and A. P. Engelbrecht, "Combining particle swarm optimization with angle modulation to solve binary problems," *Proc. of the IEEE congress on evolutionary computation (CEC2005)*, pp. 225–239.
- [91] M. Clerc, "Discrete particle swarm optimization, illustrated by the traveling salesman problem," *New optimization techniques in engineering*, pp. 219–239, Berlin, 2004.
- [92] M. Clerc, "Particle swarm optimization," London: ISTE. 2006.
- [93] A. Moraglio, C. Di Chio and R. Poli, "Geometric particle swarm optimization," *Proc. of the European conference on genetic programming (EuroGP)*, pp. 125–136. Berlin, 2007.
- [94] C. Mohan and B. Al-Kazemi, "Discrete particle swarm optimization," *Proc. Workshop on Particle Swarm Optimization*, Indianapolis, IN, Apr. 2001.
- [95] X. Hu and R. C. Eberhart, "Tracking dynamic systems with PSO: where's the cheese?" *Proc. of the workshop on particle swarm optimization*, Indianapolis, IN. 2001.
- [96] K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimizer in noisy and continuously changing environments," *Artificial intelligence and soft computing* pp. 289–294, Anaheim: IASTED/ACTA. 2001
- [97] R. C. Eberhart and Y. Shi, "Tracking and optimizing dynamic systems with particle swarms," *Proc. of the IEEE congress on evolutionary computation (CEC2001)*, pp. 94–100, Seoul, Korea. 2001.
- [98] A. Carlisle and G. Dozier, "Adapting particle swarm optimization to dynamic environments," *Proc of international conference on artificial intelligence*, pp. 429–434, Las Vegas, NE. 2000.
- [99] A. Carlisle and G. Dozier, "Tracking changing extrema with particle swarm optimizer," Auburn University, Technical Report CSSE01-08. 2001.
- [100] X. Hu and R. C. Eberhart, "Adaptive particle swarm optimization: detection and response to dynamic systems," *Proc. of the IEEE congress on evolutionary computation (CEC2002)*, pp. 1666–1670, Honolulu.
- [101] K. E. Parsopoulos and M. N. Vrahatis, "On the computation of all global minimizers through particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, pp. 211–224, 2004.
- [102] X. Li and K. H. Dam, "Comparing particle swarms for tracking extrema in dynamic environments," *Proc. of the IEEE congress on evolutionary computation (CEC2003)*, pp. 1772–1779.
- [103] S. Janson and M. Middendorf, "A hierarchical particle swarm optimizer for dynamic optimization problems," *Proc. of evoworkshops 2004: 1st European workshop on evolutionary algorithms in stochastic and dynamic environments*, pp. 513–524.

-
- [104] T. M. Blackwell and J. Branke, "Multi-swarms, exclusion and anti-convergence on dynamic environments," *IEEE Transactions on Evolutionary Computation*, vol. 10, pp. 459–472, 2006.
- [105] D. Parrot and X. Li, "Locating and tracking multiple dynamic optima by a particle swarm model using speciation," *IEEE Transactions on Evolutionary Computation*, vol. 10, pp. 440–458, 2006.
- [106] T. M. Blackwell, "Particle swarm optimization in dynamic environments," *Evolutionary computation in dynamic environments*, pp. 29–49. Springer, Berlin. DOI 10.1007/978-3-540-49774-5-2.
- [107] J. Pugh, A. Martinoli and Y. Zhang, "Particle swarm optimization for unsupervised robotic learning," *Proc. of IEEE swarm intelligence symposium (SIS)*, pp. 92–99, 2005.
- [108] A. I. El-Gallad, M. El-Hawary, A. A. Sallam, and A. Kalas, "Swarm intelligence for hybrid cost dispatch problem," *Proc. of Canadian Conf. Elect. Comput. Eng.*, 2001, vol. 2, pp. 753–757.
- [109] Z. L. Gaing, "Constrained dynamic economic dispatch solution using particle swarm optimization," *Proc. of IEEE Power Eng. Soc. General Meeting*, 2004, pp. 153–158.
- [110] A. I. S. Kumar, K. Dhanushkodi, J. J. Kumar, and C. K. C. Paul, "Particle swarm optimization solution to emission and economic dispatch problem," *Proc. of Conf. Convergent Technol. Asia-Pacific Region*, 2003, vol. 1, pp. 435–439.
- [111] H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, and Y. Nakanishi, "A particle swarm optimization for reactive power and voltage control considering voltage security assessment," *IEEE Trans. Power Syst.*, vol.15, no. 4, pp. 1232–1239, Nov. 2000.
- [112] B. Zhao, C. X. Guo, and Y. J. Cao, "A multiagent-based particle swarm optimization approach for optimal reactive power dispatch," *IEEE Trans. Power Syst.*, vol. 20, no. 2, pp. 1070–1078, May 2005.
- [113] A. A. A. Esmin, G. Lambert-Torres, and A. C. Zambroni de Souza, "A hybrid particle swarm optimization applied to loss power minimization," *IEEE Trans. Power Syst.*, vol. 20, no. 2, pp. 859–866, May 2005.
- [114] M. A. Abido, "Optimal power flow using particle swarm optimization," *Int. J. Elect. Power Energy Syst.*, vol. 24, no. 7, pp. 563–571, Oct. 2002.
- [115] B. Zhao, C. X. Guo, and Y. J. Cao, "Improved particle swarm optimization algorithm for OPF problems," *Proc. of IEEE/PES Power Syst. Conf. Expo.*, 2004, pp. 233–238.
- [116] S. He, J. Y. Wen, E. Prempan, Q. H. Wu, J. Fitch, and S. Mann, "An improved particle swarm optimization for optimal power flow," *Proc. of Int. Conf. Power Syst. Technol.*, 2004, vol. 2, pp. 1633–1637.

- [117] M. A. Abido, "Optimal design of power-system stabilizers using particle swarm optimization," *IEEE Trans. Energy Conversion*, vol. 17, no. 3, pp. 406–413, Sep. 2002.
- [118] S. P. Ghoshal, "Optimizations of PID gains by particle swarm optimizations in fuzzy based automatic generation control," *Electric Power Syst. Res.*, vol. 72, no. 3, pp. 203–212, Dec. 2004.
- [119] L. Chun-Feng and J. Chia-Feng, "Evolutionary fuzzy control of flexible AC transmission system," *Inst. Elect. Eng. Proc. Generation, Transmission Distrib.*, vol. 152, no. 4, pp. 441–448, 2005.
- [120] I. N. Kassabalidis, M. A. El-Sharkawi, R. J. Marks, II, L. S. Moulin, and A. P. ves da Silva, "Dynamic security border identification using enhanced particle swarm optimization," *IEEE Trans. Power Syst.*, vol. 17, no. 3, pp. 723–729, Aug. 2002.
- [121] H. Emara, M. Ammar, A. Bahgat, and H. Dorrah, "Stator fault estimation in induction motors using particle swarm optimization," in *Proc. IEEE Conf. Electric Machines and Drives*, Jun. 2003, vol. 3, pp. 1469–1475.
- [122] G. Bao, D. Zhang, J. Shi, and J. Jiang, "Optimal design for cogging torque reduction of transverse flux permanent motor using particle swarm optimization algorithm," in *Proc. IEEE Power Electron. Motion Control Conf.*, Aug. 2004, vol. 1, pp. 260–263.
- [123] R. C. Eberhart and Y. Shi, "Evolving artificial neural networks," in *Proc. of International Conference on Neural Networks and Brain*, Beijing, P. R. China, pp. PL5-PL13, 1998.
- [124] R. C. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources," in *Proc. of the IEEE Congress on Evolutionary Computation (CEC 2001)*, Seoul, Korea, 2001.
- [125] J. Robinson and Y. Rahmat-Samii, "Particle swarm optimization in electromagnetics," *IEEE Transactions on Antennas and Propagation*, vol. 52 (2), pp. 397-407, FEB 2004.
- [126] Z. L. Gaing, "Particle swarm optimization to solving the economic dispatch considering the generator constraints," *IEEE Transactions on Power Systems*, vol. 18 (3), pp. 1187-1195, AUG 2003.
- [127] Z. L. Gaing, "A particle swarm optimization approach for optimum design of PID controller in AVR system," *IEEE Transactions on Energy Conversion*, vol. 19 (2), pp.384-391, JUN 2004.
- [128] A. Salman, I. Ahmad and S. Al-Madani "Particle swarm optimization for task assignment problem" *Microprocessors and Microsystems*, vol. 26 (8) pp. 363-371, NOV 2002.

- [129] W. J. Xia and Z.M. Wu, "An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems" *Computers & Industrial Engineering*, vol. 48(2), pp. 409-425, MAR 2005.
- [130] M. F. Tasgetiren, Y. C. Liang, M. Sevkli and G. Gencyilmaz, "A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem", *European Journal of Operational Research*, vol. 177 (3), pp. 1930-1947, MAR 16 2007.
- [131] M. P. Wachowiak, R. Smolikova, Y. F. Zheng, J. M. Zurada and A. S. Elmaghraby, "An approach to multimodal biomedical image registration utilizing particle swarm optimization", *IEEE Transactions On Evolutionary Computation*, vol. 8 (3), pp. 289-301, JUN 2004.
- [132] K. W. Chau, "Particle swarm optimization training algorithm for ANNs in stage prediction of Shing Mun River", *Journal of Hydrology*, vol. 329 (3-4), pp. 363-367, OCT 2006.
- [133] A. Chatterjee, K. Pulasinghe, K. Watanabe and K. Izumi, "A particle-swarm-optimized fuzzy-neural network for voice-controlled robot systems", *IEEE Transactions on Industrial Electronics*, vol. 52 (6), pp. 1478-1489, DEC 2005.
- [134] K. Veeramachaneni, L. A. Osadciw and P. K. Varshney, "An adaptive multimodal biometric management algorithm", *IEEE Transactions on Systems Man and Cybernetics Part C-Applications and Reviews*, vol. 35 (3), pp. 344-356, AUG 2005.
- [135] T. H. Kim, I. Maruta and T. Sugie, "Robust PID controller tuning based on the constrained particle swarm optimization", *Automatica*, vol. 44 (4), pp. 1104-1110, APR 2008.
- [136] A. R. Yildiz, "A novel particle swarm optimization approach for product design and manufacturing", *International Journal of Advanced Manufacturing Technology*, vol. 40 (5-6), pp. 617-628, JAN 2009.
- [137] W.T. Li, Y.Q. Hei, X.W. Shi, S.F. Liu and Z.Q. Lv, "An Extended Particle Swarm Optimization Algorithm for Pattern Synthesis of Conformal Phased Arrays", *International Journal of RF and Microwave Computer-Aided Engineering*, vol. 20 (2), pp. 190-199, MAR 2010.
- [138] K. Price, "Genetic annealing," *Dr. Dobbs's Journal*, pp. 127 - 132, October 1994.
- [139] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential Evolution Algorithm with Strategy Adaptation for Global Numerical Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, pp. 398-417, April 2009.
- [140] A. Iorio and X. Li, "Solving Rotated Multi-objective Optimization Problems Using Differential Evolution," in *Australian Conference on Artificial Intelligence* Cairns, Australia, 2004, pp. 861-872.

- [141] S. Das, A. Abraham, and U. K. Chakraborty, "Differential Evolution Using a Neighborhood-Based Mutation Operator," *IEEE Transactions on Evolutionary Computation*, vol. 13, pp. 526-553, June 2009.
- [142] J. Lampinen and I. Zelinka, "Mixed Integer-Discrete-Continuous Optimization with Differential Evolution," in *5th International Mendel Conference on Soft Computing* Brno, Czech Republic, 1999.
- [143] G. Pampara, A. P. Engelbrecht, and N. Franken, "Binary Differential Evolution," in *IEEE Congress on Evolutionary Computation* Vancouver, Canada, 2006, pp. 1873-1879.
- [144] R. Gänperle, S. D. Müller, and P. Koumoutsakos, "A Parameter Study for Differential Evolution," in *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation* Interlaken, Switzerland: WSEAS Press, 2002, pp. 293-298.
- [145] S. Das, A. Konar, and U. K. Chakraborty, "Two Improved Differential Evolution Schemes for Faster Global Search," *Proceedings of the 2005 conference on Genetic and Evolutionary Computation*, pp. 991-998, 2005.
- [146] J. Lampinen and I. Zelinka, "On Stagnation of the Differential Evolution Algorithm," *Proceedings of MENDEL 2000, 6th International Mendel Conference on Soft Computing*, pp. 76-83, 2000.
- [147] D. Zaharie, "Control of Population Diversity and Adaptation in Differential Evolution Algorithms," In: *Proceedings of the 9th International Conference on Soft Computing, Brno 41-46*, 2003.
- [148] R. Storn and K. Price, "Differential Evolution : A Simple Evolution Strategy for Fast Optimization," *Dr. Dobbs's Journal*, vol. 22, pp. 18-24, April 1997.
- [149] J. Brest, S. Greiner, B. Boscosvic, 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, pp. 646-657, Dec 2006.
- [150] R. Storn, "On the Usage of Differential Evolution for Function Optimization", in *Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS)* Berkeley: IEEE, 1996, pp. 519-523.
- [151] K. V. Price, "An Introduction to Differential Evolution," in *New Ideas in Optimization*, D. Corne, Dorigo, M., and Glover, F., Ed. London (UK), 79-108: McGraw-Hill, 1999.
- [152] D. Zaharie, "Influence of crossover on the behavior of Differential Evolution Algorithms," *Applied Soft Computing*, vol. 9, pp. 1126-1138 JUN 2009
- [153] E. Mezura-Montes, J. Velazquez-Reyes, and C. A. Coello Coello, "Modified Differential Evolution for constrained optimization," in *IEEE Congress on Evolutionary Computation*, 2006, pp. 25-32.
- [154] D. Zaharie, "Critical Values for the Control Parameters of Differential Evolution," in *Proceedings of Eighth International Conference on Soft Computing*, Brno, Czech Republic, 2002, pp. 62-67.

-
- [155] D. Zaharie, "Critical Values for the Control Parameters of Differential Evolution," in *Proceedings of MENDEL 2002, Eighth International Conference on Soft Computing*, Brno, Czech Republic, 2002, pp. 62-67.
- [156] J. Rönkkönen and J. Lampinen, "On Using Normally Distributed Mutation Step Length for the Differential Evolution Algorithm," in *Proceedings of MENDEL 2003, Ninth International MENDEL Conference on Soft Computing*, Brno, Czech Republic, 2003, pp. 11-18.
- [157] U. K. Chakraborty, S. Das, and A. Konar, "Differential Evolution with Local neighborhood," in *Proceedings of Congress on Evolutionary Computation*, Vancouver, BC, Canada, 2006, pp. 2042-2049.
- [158] H. A. Abbass, "The Self-adaptive Pareto Differential Evolution Algorithm," *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 1, pp. 831-836, 2002.
- [159] J. Liu and J. Lampinen, "A Fuzzy Adaptive Differential Evolution Algorithm," *Soft Computing*, vol. 9, pp. 448-462, Jun 2005.
- [160] M. G. H. Omran, A. Salman, 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.
- [161] J. Teo, "Exploring Dynamic Self-adaptive Populations in Differential Evolution," *Soft Computing*, vol. 10, pp. 673-686, 2006.
- [162] D. Zaharie and D. Petcu, "Adaptive Pareto Differential Evolution and its Parallelization," in *Proceedings of 5th International Conference on Parallel Processing and Applied Mathematics*, Czestochowa, Poland, 2003, pp. 261-268.
- [163] H. Y. Fan and J. Lampinen, "A Trigonometric Mutation Operation to Differential Evolution," *International Journal of Global Optimization*, vol. 12, pp. 105-129, 2003.
- [164] U. Maulik. and I. Saha., "Modified Differential Evolution based Fuzzy Clustering for Pixel Classification in Remote Sensing Imagery," *Pattern Recognition*, vol. 42, pp. 2135--2149, Jan 2009.
- [165] J. Liu and J. Lampinen, "On Setting the Control Parameter of the Differential Evolution Method," in *Proc. 8th Int. Conf. Soft Computing*, 2002, pp. 11-18.
- [166] D. Zaharie and D. Petcu, "Adaptive Pareto differential evolution and its parallelization," in *Proc. of 5th International Conference on Parallel Processing and Applied Mathematics*, Czestochowa, Poland, 2003, pp. 261-268.
- [167] M. M. Ali and A. Torn, "Population Set Based Global Optimization Algorithms: Some Modifications and Numerical Studies," *Computers and Operations Research*, vol. 31, pp. 1703-1725, 2004.
- [168] 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 pp. 646-657 Dec 2006.
- [169] Z. Jingqiao and A. C. Sanderson, "JADE: Adaptive Differential Evolution with Optimal External Archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, pp. 945-958, October 2009.

-
- [170] Z. Yang, J. He, and X. Yao, "Making a Difference to Differential Evolution," in *Advances in Metaheuristics for Hard Optimization*: Springer Berlin Heidelberg, 2007, pp. 415-432.
- [171] N. Noman and H. Iba, "Accelerating Differential Evolution using Adaptive Local Search," *IEEE Transactions on Evolutionary Computation*, vol. 12, pp. 107-125, Feb 2008.
- [172] Z. Yang, K. Tang, and Y. Xin, "Large Scale Evolutionary Optimization using Cooperative Evolution," *Information Sciences*, vol. 178, pp. 2985-2999, August 2008.
- [173] S. Tsutsui, M. Yamamura, and T. Higuchi, "Multi-parent Recombination with Simplex Crossover in Real Coded Genetic algorithms," in *Genetic Evolutionary Computation Conference*, 1999, pp. 657-664.
- [174] S. Das, A. Konar, and U. K. Chakraborty, "Improving Particle Swarm Optimization with Differentially Perturbed Velocity," in *ACM-SIGEVO Proceedings of Genetic Evolutionary Computation Conference*, Washington DC, 2005, pp. 177-184.
- [175] P. W. Moore and K. Venayagamoorthy, "Evolving Digital Circuit using Hybrid Particle Swarm Optimization and Differential Evolution," *International Journal of Neural Systems*, vol. 6, pp. 163-177, 2006.
- [176] M. G. H. Omran, A. P. Engelbrecht, and A. Salman, "Bare Bones Differential Evolution," *European Journal of Operational Research*, vol. 196, pp. 128-139, July 2009.
- [177] Z. F. Hao, G. H. Guo, and H. Huang, "A Particle Swarm Optimization Algorithm with Differential Evolution," in *Proceedings of 6th International Conference on Machine Learning Cybernetics*, 2007, pp. 1031-1035.
- [178] X. Xu, Y. Li, S. Fang, Y. Wu, and F. Wang, "A Novel Differential Evolution Scheme Combined with Particle Swarm Intelligence," in *IEEE Congress on Evolutionary Computation*, 2008, pp. 1057-1062.
- [179] S. Das, A. Konar, and U. K. Chakraborty, "Annealed Differential Evolution," in *IEEE Conference on Evolutionary Computation* Singapore, 2007, pp. 1926-1933.
- [180] M. Dorigo and L. M. Gambardella, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 53-66, 1997.
- [181] J. P. Chiou, C.-F. Chang, and C.-T. Su, "Ant direction hybrid differential evolution for solving large capacitor placement problems," *IEEE Transactions on Power Systems*, vol. 19, pp. 1794-1800, November 2004.
- [182] X. Zhang, H. Duan, and J. Jin, "DEACO: Hybrid Ant Colony Optimization with Differential Evolution," in *IEEE Congress on Evolutionary Computation*, 2008, pp. 921-927.
- [183] D. Dasgupta, "Artificial Immune Systems and Their Applications," Berlin: Springer-Verlag, 1999.

-
- [184] X. He and L. Han, "A Novel Binary Differential Evolution algorithm based on Artificial Immune System," in *IEEE Congress on Evolutionary Computation* Singapore, 2007, pp. 2267–2272.
- [185] A. Biswas, S. Dasgupta, S. Das, and A. Abraham, "A Synergy of Differential Evolution and Bacterial Foraging Algorithm for Global Optimization," *Neural Network World*, vol. 17, pp. 607–626, 2007.
- [186] V. Tirronen, F. Neri, T. Kärkkäinen, K. Majava, and T. Rossi, "An Enhanced Memetic Differential Evolution in Filter Design for Defect Detection in Paper Production," *Evolutionary Computation*, vol. 16, pp. 529–555, December 2008.
- [187] A. Caponio, F. Neri, and V. Tirronen, "Super-fit Control Adaptation in Memetic Differential Evolution Frameworks," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 13, pp. 811–831, June 2009.
- [188] L. S. Coelho and V. C. Mariani, "Combining of Chaotic Differential Evolution and Quadratic Programming for Economic Dispatch Optimization with valve-point effect " *IEEE Transactions on Power Systems*, vol. 21, pp. 989–996, 2006.
- [189] N. Noman and H. Iba, "Differential Evolution for Economic Load Dispatch Problems," *Electric Power Systems Research*, vol. 78, pp. 1322–1331, August 2008.
- [190] X. Yuan, L. Wang, Y. Zhang, and Y. Yuan, "A Hybrid Differential Evolution Method for Dynamic Economic Dispatch with Valve-point Effects," *Expert Systems with Applications*, vol. 36, pp. 4042–4048, March 2009.
- [191] J. P. Chiou, "A Variable Scaling Hybrid Differential Evolution for Solving Large-scale Power Dispatch Problems," *IET Generation, Transmission & Distribution*, vol. 3, pp. 154–163, February 2009.
- [192] H. R. Cai, C. Y. Chung, and K. P. Wong, "Application of Differential Evolution Algorithm for Transient Stability Constrained Optimal Power Flow," *IEEE Transactions on Power Systems*, vol. 23, May 2008.
- [193] C. H. Liang, C. Y. Chung, K. P. Wong, and X. Z. Duan, "Parallel Optimal Reactive Power Flow based on Cooperative Co-evolutionary Differential Evolution and Power System Decomposition," *IEEE Transactions on Power Systems*, vol. 22, February 2007.
- [194] M. Varadarajan and K. S. Swarup, "Solving Multi-objective Optimal Power Flow using Differential Evolution," *IET Generation, Transmission & Distribution*, vol. 2, pp. 720–730, 2008.
- [195] M. Basu, "Optimal Power Flow with FACTS Devices using Differential Evolution," *International Journal of Electrical Power and Energy Systems*, vol. 30, pp. 150–156, February 2008.
- [196] S. Sayah and K. Zehar, "Modified Differential Evolution Algorithm for Optimal Power Flow with Non-smooth Cost Functions," *Energy Conversion and Management*, vol. 49, pp. 3036–3042, November 2008.
- [197] G. Y. Yang, Z. Y. Dong, and K. P. Wong, "A Modified Differential Evolution Algorithm with Fitness Sharing for Power System Planning," *IEEE Transactions on Power Systems*, vol. 23, May 2008.

-
- [198] S. Kannan and P. Murugan, "Solutions to Transmission Constrained Generation Expansion Planning using Differential Evolution," *European Transactions on Electrical Power*, July 2008.
- [199] T. Sum-Im, G. A. Taylor, M. R. Irving, and Y. H. Song, "Differential Evolution Algorithm for Static and Multistage Transmission Expansion Planning," *IET Generation, Transmission & Distribution*, vol. 3, pp. 365-384, April 2009.
- [200] Chung-Fu Chang, Ji-Jen Wong, Ji-Pyng Chiou, and C.-T. Su, "Robust Searching Hybrid Differential Evolution Method for Optimal Reactive Power Planning in Large-scale Distribution Systems," *Electric Power Systems Research*, vol. 77, pp. 430-437, April 2007.
- [201] A. Qing, "Dynamic Differential Evolution Strategy and Application in Electromagnetic Inverse Scattering Problems," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 44, pp. 116-125, January 2006.
- [202] S. Yang and A. Qing, "Design of High-power Millimeter-wave TM₀₁-TE₁₁ Mode Converters by the Differential Evolution Algorithm," *IEEE Transactions on Plasma Science*, vol. 33, pp. 1372-1376, 2005.
- [203] D. G. Kurup, M. Himdi, and A. Rydberg, "Synthesis of Uniform Amplitude Unequally Spaced Antenna Arrays using the Differential Evolution Algorithm," *IEEE Transactions on Antennas and Propagation*, vol. 51, pp. 2210-2217, September 2003.
- [204] S. Yang and Z. Nie, "Mutual Coupling Compensation in Time Modulated Linear Antenna Arrays," *IEEE Transactions on Antennas and Propagation*, vol. 53, pp. 372-376, January 2005.
- [205] H. Yousefi, H. Handroos, and A. Soleymani, "Application of Differential Evolution in System Identification of a Servo-hydraulic System with a Flexible Load," *Mechatronics*, vol. 18, pp. 513-528, November 2008.
- [206] H. Tang, S. Xue, and C. Fan, "Differential Evolution Strategy for Structural System Identification," *Computers and Structures*, vol. 86, pp. 2004-2012, November 2008.
- [207] I. L. Lopez Cruz, L. G. Van Willigenburg, and V. S. G., "Efficient Differential Evolution Algorithms for Multimodal Optimal Control Problems," *Applied Soft Computing*, vol. 3, pp. 97-122, September 2003.
- [208] M. W. Iruthayarajan and S. Baskar, "Evolutionary Algorithms based Design of Multivariable PID Controller," *Expert Systems with Applications*, vol. 36, pp. 9159-9167, 2009.
- [209] A. Biswas, S. Das, A. Abraham, and S. Dasgupta, "Design of Fractional-order PI λ D μ Controllers with an Improved Differential Evolution," *Engineering Applications of Artificial Intelligence*, vol. 22, pp. 343-350, March 2009.
- [210] S. Aydin and H. Temeltas, "Fuzzy-Differential Evolution Algorithm for Planning Time-Optimal Trajectories of a Unicycle Mobile Robot on a Predefined Path," *Advanced Robotics*, vol. 18, pp. 725-748, 2004.
- [211] J. Chakraborty, A. Konar, L. C. Jain, and U. K. Chakraborty, "Cooperative Multi-Robot Path Planning using Differential Evolution," *Journal of Intelligent and Fuzzy Systems*, vol. 20, pp. 13-27, April 2009.

-
- [212] S. Das and A. Konar, "Automatic Image Pixel Clustering with an Improved Differential Evolution " *Applied Soft Computing*, vol. 9, pp. 226-236 Jan 2009
- [213] P. Besson, V. Popovici, J.-M. Vesin, J.-P. Thiran, and M. Kunt, "Extraction of Audio Features Specific to Speech Production for Multimodal Speaker Detection," *IEEE Transactions on Multimedia*, vol. 10, pp. 63-73, January 2008.
- [214] I. De Falco, A. D. Cioppa, D. Maisto, and F. Tarantino, "Differential Evolution as a Viable Tool for Satellite Image Registration," *Applied Soft Computing*, vol. 8, pp. 1453-1462, September 2008.
- [215] J. Ilonen, J. K. Kamarainen, and J. Lampinen, "Differential Evolution Training Algorithm for Feed-forward Neural networks," *Neural Processing Letters*, vol. 17, pp. 93-105, 2003.
- [216] N. Chauhan, V. Ravi, and D. K. Chandra, "Differential Evolution Trained Wavelet Neural Networks: Application to Bankruptcy Prediction in Banks," *Expert Systems with Applications*, vol. 36, pp. 7659-7665, May 2009.
- [217] L. S. Coelho and F. A. Guerra, "B-spline Neural Network Design using Improved Differential Evolution for Identification of an Experimental Nonlinear Process," *Applied Soft Computing*, vol. 8, pp. 1513-1522, September 2008.
- [218] R. Storn, "Differential Evolution Design of an IIR-filter " in *IEEE International Conference on Evolutionary Computation* IEEE, 1996, pp. 268-273.
- [219] R. Storn, "Designing Nonstandard Filters with Differential Evolution," *IEEE Signal Processing Magazine*, vol. 22, pp. 103-106, 1995.
- [220] S. Das and A. Konar, "Two-dimensional IIR Filter Design with Modern Search Heuristics: A comparative study," *International Journal of Computational Intelligence and Applications*, vol. 6, pp. 329-355, 2006.
- [221] S. Chattopadhyay, S. K. Sanyal and A. Chandra, " Design of FIR Filter Using Differential Evolution Optimization & to Study its Effect as a Pulse-Shaping Filter in a QPSK Modulated System", *International Journal of Computer Science and Network Security*, VOL.10 No.1, January 2010.
- [222] R. Xu, G. K. Venayagamoorthy, and D. C. Wunsch, "Modeling of Gene Regulatory Networks with Hybrid Differential Evolution and Particle Swarm Optimization," *Neural Networks*, vol. 20, pp. 917-927, October 2007.
- [223] N. Noman and H. Iba, "Inferring Gene Regulatory Networks using Differential Evolution with Local Search Heuristics," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 4, pp. 634-647, October 2007.
- [224] K. Suresh, D. Kundu, S. Ghosh, S. Das, A. Abraham, and S. Y. Han, "Multi-objective Differential Evolution for Dynamic Clustering with Application to Micro-array Data Analysis," *Sensors, Molecular Diversity Preservation International* vol. 9, pp. 3981- 4004, 2009.
- [225] H. Silverio and L. R. Bitello, "A Differential Evolution Approach for Protein Folding using a Lattice Model," *Journal of Computer Science and technology*, vol. 22, 2007.
- [226] B. V. Babu and R. Angira, "Modified Differential Evolution for Optimization of Non-linear Chemical Processes," *Computers & Chemical Engineering*, vol. 30, pp. 989-1002, 2006.

- [227] M. Srinivas and G. P. Rangaiah, "A Study of Differential Evolution and Tabu Search for Benchmark, Phase Equilibrium and Phase Stability Problems," *Computers & Chemical Engineering*, vol. 31, pp. 760-772, July 2007.
- [228] 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*, vol. 23, pp. 327-339, Feb 1999.
- [229] Z. Fan, J. Liu, T. Sørensen, and P. Wang, "Improved Differential Evolution based on Stochastic Ranking for Robust Layout Synthesis of MEMS Components," *IEEE Transactions on Industrial Electronics*, vol. 56, pp. 937-948, April 2009.
- [230] R. Storn, "System Design by Constraint Adaptation and Differential Evolution," *IEEE Transactions on Evolutionary Computation*, vol. 3, pp. 22-34, April 1999.
- [231] G. C. Onwubolu, "Design of Hybrid Differential Evolution and Group Method of Data Handling Networks for Modeling and Prediction," *Information Sciences*, vol. 178, pp. 3616-3634, September 2008.
- [232] M. K. Venu, R. Mallipeddi, and P. N. Suganthan, "Fiber Bragg Grating Sensor Array Interrogation using Differential Evolution," *Optoelectronics and Advanced Materials - Rapid Communications*, vol. 2, pp. 682-685, 2008.
- [233] S. Z. Zhao, B. Y. Qu, M. Willjuice Iruthayarajan, S. Baskar and P. N. Suganthan, "Multi-objective Robust PID Controller Tuning using Multi-objective Differential Evolution" *the Eleventh International Conference on Control, Automation, Robotics and Vision (ICARCV 2010)*, Singapore, December, 2010.
- [234] S. Z. Zhao, P. N. Suganthan, S. Pal, S. Das and A. Basak, "Multi-Objective Design of Monopulse Antenna with Two-lbests based Multi-objective Particle Swarm Optimizer", *The First International Conference On Computational Problem-Solving (ICCP2010)*, Li Jiang, China, Dec 3-5, 2010. (Best Student Paper Award)
- [235] Y-P Chang and C-N Kob, "A PSO method with nonlinear time-varying evolution based on neural network for design of optimal harmonic filters", *Expert Systems with Applications*, vol. 36(3), pp. 6809-6816, 2009.
- [236] J. J. Liang, C. C. Chan, V. L. Huang and P. N. Suganthan, "Improving the performance of FBG sensor networks using dynamic multi-swarm particle swarm optimizer", *J. of Optoelectronics and Advanced Materials, Rapid-communications*, Issue 8, pp. 373-378, 2007.
- [237] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space", *IEEE Transactions on Evolutionary Computation*, Volume 6, Issue 1, Feb 2002 Page(s):58 – 73.
- [238] D. E. Goldberg and J. Richardson, "Genetic Algorithms with Sharing for Multi-modal Function Optimization", In Grefenstette, editor, "Genetic Algorithms and their Applications" (ICGA'87), pages 41-49.
- [239] T. Krink, R. Thomsen, and P. Rickers. "Applying Self-Organised Criticality to Evolutionary Algorithms". In *Parallel Problem Solving form Nature - PPSN VI*, volume 1, 375–384.
- [240] M. Lozano, F. Herrera and J R. Cano, "Replacement strategies to preserve useful diversity in steady-state genetic algorithms", *Information Sciences*, 178(23), pp. 4421-4433, Dec 1 2008.

-
- [241] R. Thomsen, P. Rickers, and T. Krink. "A Religion-Based Spatial Model for Evolutionary Algorithms". In *Parallel Problem Solving from Nature - PPSN VI*, volume 1, 817–826.
 - [242] M. Ventresca and H.R. Tizhoosh "A diversity maintaining population-based incremental learning algorithm", *Information Sciences*, 178(21), pp. 4038-4056, Nov 1 2008.
 - [243] T. Krink, J. S. Vestertroem, and J. Riget. "Particle swarm optimization with spatial particle extension", In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002)*, Honolulu, Hawaii, 2002.
 - [244] C. K. Monson and K. D. Seppi "Adaptive diversity in PSO", *Proc. of the 8th annual conference on Genetic and evolutionary computation*, Seattle, Washington, USA, pp. 59-66, 2006.
 - [245] J. Riget and J. S. Vesterstrom. "A diversity-guided particle swarm optimizer — the ARPSO", *Technical Report 2002-02*, Department of Computer Science, University of Aarhus, 2002.
 - [246] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont, "Evolutionary Algorithms for Solving Multi-Objective Problems", Kluwer Academic Publishers, New York, May 2002.
 - [247] D. W. Corne, N. R. Jerram, J. D. Knowles, and M. J. Oates, "PESA-II: Region based Selection in Evolutionary Multiobjective Optimization". *Proc. of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pp. 283–290, San Francisco, California, 2001.
 - [248] P. K. Tripathi , S. Bandyopadhyay and S. K. Pal "Multi-Objective Particle Swarm Optimization with time variant inertia and acceleration coefficients" *Information Sciences*, 177(22), pp.5033-5049, Nov 15 2007.
 - [249] S. Z. Zhao, P. N. Suganthan, "Diversity enhanced particle swarm optimizer for global optimization of multimodal problems," pp. 590-597, 2009 *IEEE Congress on Evolutionary Computation*, 2009.
 - [250] A. L. Barker and W. N. Martin, "Population Diversity and Fitness Measures Based on Genomic Distances", Uni. of Virginia, Computer Science Department, *Technical Report TR-99-24*, July 31, 1999.
 - [251] C. Mattiussi, M. Waibel and D. Floreano, "Measures of Diversity for Populations and Distances between Individuals with Highly Reorganizable Genomes", *Evolutionary Computation*, vol. 12, num. 4, 2004, p. 495—515.
 - [252] R. Myers and E.R. Hancock, "Genetic Algorithms for Ambiguous Labeling Problems" *Pattern Recognition*, 33(4): 685-704 (2000).
 - [253] M. A. Bedau, M. Zwick and A. Baham, "Variance and Uncertainty Measures of Population Diversity Dynamics", In *Advances in System Science and Applications* Special Issue I. pp. 7-12.
 - [254] J. Riget and J. S. Vesterstrom. "A diversity-guided particle swarm optimizer — the ARPSO", *Technical Report 2002-02*, Department of Computer Science, University of Aarhus, 2002.
 - [255] R. K. Ursem, "Diversity-Guided Evolutionary Algorithms", *Proc. of the 7th International Conference on Parallel Problem Solving from Nature*, 2002, pp: 462-474.
 - [256] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger and S. Tiwari, "Problem Definitions and Evaluation Criteria for the CEC2005 Special Session on Real-Parameter Optimization", *Technical Report*, Nanyang Technological

- University, Singapore, May 2005 AND KanGAL Report #2005005, IIT Kanpur, India.
- [257] J. J. Liang, A. K. Qin, P. N. Suganthan and S. Baskar, "Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions", *IEEE T. on Evolutionary Computation*, Vol. 10, No. 3, pp. 281-295, June 2006.
 - [258] J. J. Liang, P. N. Suganthan and K. Deb, "Novel composition test functions for numerical global optimization," in *Proc. of Swarm Intelligence Symp.*, June 2005.
 - [259] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans on Evolutionary Computation*, vol. 1, pp. 67-82, 1997.
 - [260] Broyden, C.G., "The Convergence of a Class of Double-rank Minimization Algorithms," *Journal of the Institute of Mathematics and Applications.*, Vol. 6, pp 76-90, 1970.
 - [261] Fletcher, R., "A New Approach to Variable Metric Algorithms," *Computer Journal*, Vol. 13, pp 317-322, 1970.
 - [262] Goldfarb, D., "A Family of Variable Metric Updates Derived by Variational Means," *Mathematics of Computing*, Vol. 24, pp 23-26, 1970.
 - [263] Shanno, D.F., "Conditioning of Quasi-Newton Methods for Function Minimization," *Mathematics of Computing*, Vol. 24, pp 647-656, 1970.
 - [264] F. van den Bergh and A. P. Engelbrecht. A cooperative approach to particle swarm optimisation. *IEEE Transactions on Evolutionary Computation*, (3): 225-239, June 2004.
 - [265] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in amultidimensional complex space", *IEEE Transactions on Evolutionary Computation*, Volume 6, Issue 1, Feb 2002 Page(s):58 – 73.
 - [266] K. Zielinski, R. Laur: "Stopping Criteria for a Constrained Single-Objective Particle Swarm Optimization Algorithm", *Informatica*, 31(1):51-59, 2007, ISSN 0350-5596.
 - [267] Davidon, W.C., "Variable Metric Method for Minimization," A.E.C. Research and Development Report, ANL-5990, 1959, M. Young, The Technical Writers Handbook. Mill Valley, CA: University Science, 1989.
 - [268] Fletcher, R. and M.J.D. Powell, "A Rapidly Convergent Descent Method for Minimization," *Computer Journal*, Vol. 6, pp 163-168, 1963.
 - [269] J. J. Liang, and P. N. Suganthan, "Dynamic Multi-Swarm Particle Swarm Optimizer," *Proc. of IEEE Int. Swarm Intelligence Symposium*, pp. 124-129, 2005.
 - [270] K. Lee and Z. W. Geem, "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice", *Computer Methods in Applied Mechanics and Engineering* 194, 3902–3933, 2005.
 - [271] M. G. H. Omran and M. Mahdavi, "Global-best harmony search". *Applied Mathematics and Computation*, 198 (2), 643–656, 2008.
 - [272] Z. W. Geem, "Particle-swarm harmony search for water network design", *Engineering Optimization*, Vol. 41, 297 - 311, 4 April 2009.
 - [273] J. Kennedy, and R. C. Eberhart, "Particle swarm optimization ". *Proceedings of IEEE International Conference on Neural Networks*, Piscataway, NJ. pp. 1942-1948, 1995.
 - [274] S. Z. Zhao and P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer with harmony search", *Expert Systems with Applications*, vol. 38, issue 4, April 2011, Pages 3735-3742.

-
- [275] M. A. Abido, "Two-Level of Non-dominated Solutions Approach to Multi-objective Particle Swarm Optimization", *Proc. of the Genetic and Evolutionary Computation Conf. (GECCO-2007)*, July 7–11, 2007, London, UK.
- [276] J. Branke and S. Mostaghim, "About Selecting the Personal Best in Multi-Objective Particle Swarm Optimization" *Proc. of the 9th Int. Conf. on Parallel Problem Solving from Nature – PPSN IX*, Reykjavik, Iceland, Sept. 9-13, 2006, pp. 523-532.
- [277] J. E. Alvarez-Benitez, R. M. Everson and J. E. Fieldsend, "A MOPSO algorithm based exclusively on pareto dominance concepts", *3rd Int. Conf. on Evolutionary Multi-Criterion Optimization (EMO 2005)*, pp. 459–473, Guanajuato, Mexico, 2005. LNCS 3410, Springer Verlag.
- [278] C. A. Coello Coello, G. T. Pulido and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization", *IEEE Trans. on Evolutionary Computation*, 8(3):256–279, June 2004.
- [279] J. Hao, J. H. Zheng and L. J. Chen, "Multi-Objective Particle Swarm Optimization Algorithm Based on Enhanced ϵ –Dominance", *IEEE Int. Conf. on Engineering of Intelligent Systems*, April 2006, pp. 1 – 5.
- [280] S. L. Ho, S. Y. Yang, G. Z Ni, W. C. Lo Edward and H. C. Wong, "A particle swarm optimization based method for multiobjective design optimizations", *IEEE Transactions on Magnetics*, 41(5):1756–1759, May 2005.
- [281] X. Hu and R. C. Eberhart, "Multi-objective optimization using dynamic neighborhood particle swarm optimization", *Proc. of the IEEE Congress on Evolutionary Computation*, Honolulu, Hawaii USA. pp. 1677-1681, 2002.
- [282] V. L. Huang, P. N. Suganthan and J. J. Liang, "Comprehensive Learning Particle Swarm Optimizer for Solving Multiobjective Optimization Problems", *Int. Journal of Intelligent Systems*, vol. 21, pp. 1-18, 2006.
- [283] D. Ireland, A. Lewis, S. Mostaghim and J. W. Lu, "Hybrid Particle Guide Selection Methods in Multi-Objective Particle Swarm Optimization", *2nd IEEE Int. Conf. on e-Science and Grid Computing*, Dec. 2006 pp. 116 – 116.
- [284] X. D. Li, "A non-dominated sorting particle swarm optimizer for multiobjective optimization", *Proc. of Genetic and Evolutionary Computation Conf. (GECCO'2003)*, pp. 37–48. Springer. Lecture Notes in Computer Science, Vol. 2723, July 2003.
- [285] S. Mostaghim and J. R. Teich, "Strategies for finding local guides in multi-objective particle swarm optimization (MOPSO)", *Proc. of the IEEE Swarm Intelligence Symposium (SIS 2003)*, Indianapolis, Indiana, USA. pp. 26-33, 2003.
- [286] S. Mostaghim and J. R. Teich, "The role of e-dominance in multi-objective particle swarm optimization methods", *Proc. of IEEE Congress on Evolutionary Computation*, Canberra, Australia. pp. 1764-1771, 2003.
- [287] K. E. Parsopoulos and M. N. Vrahatis. "Recent approaches to global optimization problems through particle swarm optimization". *Natural Computing*, Vol.1, No.2–3, pp. 235–306, 2002.
- [288] M. Salazar-Lechuga and J. Rowe, "Particle swarm optimization and fitness sharing to solve multi-objective optimization problems", *In Congress on Evolutionary Computation*, pp. 1204–1211, Edinburgh, Scotland, UK, Sept. 2005.
- [289] M. R. Sierra and C. A. Coello Coello, "Improving PSO-based multi-objective optimization using crowding, mutation and ϵ -dominance", *3rd Int. Conf. on Evolutionary Multi-Criterion Optimization (EMO 2005)*, pp. 505 -519, Mexico, 2005. LNCS 3410, Springer-Verlag.

-
- [290] M. R. Sierra and C. Coello Coello, "Multi-objective particle swarm optimizers: A survey of the state--of--the--art", *Int. J. of Computational Intelligence Research*, 2(3):287--308, 2006.
 - [291] G. Toscano-Pulido, C. A. Coello Coello and L.V. Santana-Quintero, "EMOPSO: A multi-objective particle swarm optimizer with emphasis on efficiency", *4th Int. Conf. on Evolutionary Multi-Criterion Optimization (EMO 2007)*, MAR 05-08, 2007, 272-285.
 - [292] K. Zielinski, R. Laur, "Adaptive Parameter Setting for a Multi-Objective Particle Swarm Optimization Algorithm", *Proc. of IEEE Congress of Evolutionary Computation*, 2007, Singapore. pp. 3585 – 3592.
 - [293] K. Deb, "Multiobjective Optimization Using Evolutionary Algorithms", Chichester, U.K. Wiley, 2001.
 - [294] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, New York, 2002.
 - [295] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II", *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.
 - [296] J. D. Knowles and D. W. Corne, "Approximating the nondominated front using the Pareto archive evolutionary strategy", *Evolutionary Computation*, 8(2): 149-172, 2000.
 - [297] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms", *Evol. Comput.*, vol. 2, no. 3, pp. 221–248, 1994.
 - [298] K. C. Tan, *Multiobjective Evolutionary Algorithms and Applications*, Springer-Verlag, 2005.
 - [299] E. Zitzler, K. Deb, and L. Thiele, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results", *Evolutionary Computation*, 8(2):173–195, Summer 2000.
 - [300] J. E. Fieldsend and S. Singh, "A multiobjective algorithm based upon particle swarm optimisation, an efficient data structure and turbulence", *Proc. of 2002 U.K. Workshop on Computational Intelligence*, pp. 37–44, Birmingham, UK, September 2002.
 - [301] Q. Jiang, M. Huang and C. Wang, "A Novel Method for Finding Good Local Guides in Multi-objective Particle Swarm Optimization", *3rd Int. Conf. on Natural Computation*, Vol. 3, Aug. 2007 pp. 737 – 741.
 - [302] P. K. Tripathi, S. Bandyopadhyay and S. K. Pal, "Multi-Objective Particle Swarm Optimization with time variant inertia and acceleration coefficients", *Information Sciences* 177 (22): 5033-5049 NOV 15 2007.
 - [303] J. J. Liang, A. K. Qin, P. N. Suganthan and S. Baskar, "Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions", *IEEE T. on Evolutionary Computation*, Vol. 10, No. 3, pp. 281-295, June 2006.
 - [304] K. E. Parsopoulos, D. E. Tasoulis and M. N. Vrahatis. "Multi-objective optimization using parallel vector evaluated particle swarm optimization", *Proc. of international conference on artificial intelligence and applications (IASTED)*, Innsbruck, Austria, 2004.
 - [305] Schaffer, J. D. "Multiple objective optimization with vector evaluated genetic algorithms", *Proc. of First Int. Conf. on Genetic Algorithms*, pp. 93-100. Lawrence Erlbaum, (1985).

-
- [306] T. Ray and K. M. Liew, "A swarm metaphor for multi-objective design optimization", *Engineering Optimization*, 34(2):141–153, March 2002.
 - [307] K. Deb, L. Thiele, M. Laumanns and E. Zitzler, "Scalable Multi-Objective Optimization Test Problems", *CEC 2002*, pp. 825 - 830, IEEE Press, 2002.
 - [308] S. Huband, P. Hingston, L. Barone and L. While, "A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit", *IEEE Trans. on Evolutionary Computation*, Oct. 2006. Vol. 10, No. 5, pp. 477-506.
 - [309] V. L. Huang, A. K. Qin, K. Deb, E. Zitzler, P. N. Suganthan, J. J. Liang, M. Preuss, S. Huband, "Problem Definitions for Performance Assessment on Multi-objective Optimization Algorithms" *Technical Report*, <http://www.ntu.edu.sg/home/epnsugan> Nanyang Technological University, Singapore, 2007.
 - [310] G. Rudolph, B. Naujoks and M. Preuss, "Capabilities of EMOA to Detect and Preserve Equivalent Pareto Subsets". *Evolutionary Multi-Criterion Optimization (EMO 2007)*, pp. 36-50. Vol. 4403, Springer, Berlin. 2007.
 - [311] D. Sharma, A. Kumar, K. Deb and K. Sindhya, "Hybridization of SBX based NSGA-II and sequential quadratic programming for solving multi-objective optimization problems", *Proc. of the IEEE Congress on Evolutionary Computation*, Singapore. pp. 3003 – 3010, 2007.
 - [312] A. Kumar, D. Sharma and K. Deb, "A hybrid multi-objective optimization procedure using PCX based NSGA-II and sequential quadratic programming", *Proc. of the IEEE Congress on Evolutionary Computation*, Singapore, pp. 3011 – 3018, 2007.
 - [313] S. Kukkonen and J. Lampinen, "Performance assessment of Generalized Differential Evolution 3 (GDE3) with a given set of problems", *Proc. of the IEEE Congress on Evolutionary Computation*, Singapore, pp. 3593 – 3600, 2007.
 - [314] K. Zielinski and R. Laur, "Differential evolution with adaptive parameter setting for multi-objective optimization", *Proc. of IEEE Congress of Evolutionary Computation*, 2007, Singapore. pp. 3585 – 3592.
 - [315] V. L. Huang, A. K. Qin, P. N. Suganthan and M. F. Tasgetiren, "Multi-objective optimization based on self-adaptive differential evolution algorithm", *Proc. of the IEEE Congress on Evolutionary Computation*, Singapore. pp. 3601 – 3608, 2007.
 - [316] A. Zamuda, J. Brest, B. Boskovic and V. Zumer, "Differential evolution for multiobjective optimization with self adaptation", *Proc. of the IEEE Congress on Evolutionary Computation (CEC 2007)*, Singapore. pp. 3617 - 3624.
 - [317] L-Y. Tseng and C. Chen, "Multiple trajectory search for multiobjective optimization", *Proc. of the IEEE Congress on Evolutionary Computation (CEC 2007)*, Singapore. pp. 3609 - 3616.
 - [318] J. Knowles, 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.
 - [319] M. Laumanns, L. Thiele and E. Zitzler, "An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method", *European Journal of Operational Research*, Vol. 169, No. 3, pp. 932-942, March 2006.
 - [320] C. A. Liu and Y.P. Wang, "A new dynamic multi-objective optimization evolutionary algorithm", *International Journal of Innovative Computing Information and Control*, 4(8): 2087-2096, Aug 2008.

-
- [321] M. Laumanns, L. Thiele, K. Deb and E. Zitzler, "Combining Convergence and Diversity in Evolutionary Multi-objective Optimization", *Evolut. Computation*, Vol. 10, pp. 263-282, 2002.
 - [322] K. Deb, M. Mohan and S. Mishra, "A Fast Multi-objective Evolutionary Algorithm for Finding Well-Spread Pareto-Optimal Solutions." *KanGAL Report No. 2003002*, (February, 2003).
 - [323] E. Zitzler and L. Thiele, "An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach", *TIK Report 43; Computer Engineering and Communication Networks Lab*; Swiss Federal Institute of Technology; Zurich; Switzerland, May, 1998.
 - [324] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Computing* 9(1), 2005, pp. 3–12.
 - [325] J. Hao, J. H. Zheng and L. J. Chen, "Multi-Objective Particle Swarm Optimization Algorithm Based on Enhanced ε -Dominance", *IEEE Int. Conf. on Engineering of Intelligent Systems*, April 2006, pp. 1 – 5.
 - [326] J. B. Kollat and P. M. Reed, "The Value of Online Adaptive Search: A Performance Comparison of NSGAI, e-NSGAI and eMOEA", in C. A. Coello Coello and A. H. Aguirre and E. Zitzler (editors), *Evolutionary Multi-Criterion Optimization. Third Int. Conf., EMO 2005*, pp. 386-398, Springer. Lecture Notes in Computer Science Vol. 3410, Guanajuato, México, March 2005.
 - [327] M. R. Sierra and C. A. Coello Coello, "Improving PSO-based multi-objective optimization using crowding, mutation and ε - dominance", *3rd Int. Conf. on Evolutionary Multi-Criterion Optimization (EMO 2005)*, pp. 505 -519, Mexico, 2005, LNCS 3410, Springer-Verlag.
 - [328] S. Z. Zhao and P. N. Suganthan, "Two lbests Based Multi-objective Particle Swarm Optimizer", *Engineering Optimization*, vol. 43, issue 1, January 2011, pages 1 - 17.
 - [329] H. Markowitz, W. F. Sharpe and M. Miller, "Founders of Modern Finance: Their Prize Winning Concepts And 1990 Nobel Lectures", 1991.
 - [330] H. Markowitz, "Portfolio Selection", *Journal of Finance*, vol. 7, pp. 77-91, 1952.
 - [331] C. C. Lin and Y. T. Liu, "Genetic algorithms for portfolio selection problems with minimum transactions lots," *European Journal of operational Research*, vol. 185, pp. 393-404, 2008.
 - [332] S. Z. Zhao and P. N. Suganthan, "Multi-objective Evolutionary Algorithm with Ensemble of External Archives", *Int. J. of Innovative Computing, Information and Control*, Vol. 6, No. 1, pp 1713-1726, April 2010.
 - [333] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization," in *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, K. C. Giannakoglou, D. T. Tsahalis, J. P ériaux, K. D. Papailiou, and T. Fogarty, Eds., Athens, Greece, 2001, pp. 95–100.
 - [334] J. D. Knowles and D. W. Corne, "The pareto archived evolution strategy: A new baseline algorithm for multiobjective optimisation," in *Proc. of Congress on Evolutionary Computation (CEC'99)*, Washington D.C., 1999, pp. 98–105.
 - [335] Q. Zhang and H. Li, "MOEA/D: A Multi-objective Evolutionary Algorithm Based on Decomposition", *IEEE Trans. Evolutionary Computation*, vol.11, no. 6, pp712-731 2007.

-
- [336] Q. Zhang, W. Liu and H. Li, "The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances" in *Proc. of Congress on Evolutionary Computation (CEC 2009)*, Trondheim, Norway, 2009, pp. 203-208.
 - [337] H. Ishibuchi, Y. Sakane, N. Tsukamoto and Y. Nojima, "Effects of using two neighborhood structures on the performance of cellular evolutionary algorithms for many-objective optimization", in *Proc. of Congress on Evolutionary Computation (CEC 2009)*, Trondheim, Norway, 2009, pp. 2508-2515.
 - [338] E. L. Yu and P. N. Suganthan, "Ensemble of niching algorithms", *Information Sciences*, Vol. 180, No. 15, pp. 2815-2833, Aug. 2000.
 - [339] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, and S. Tiwari, "Multiobjective optimization test instances for the CEC 2009 special session and competition," *Technical Report CES-487*, The School of Computer Science and Electronic Engineering, University of Essex, Tech. Rep., 2008.
 - [340] R. Mallipeddi, S. Mallipeddi and P. N. Suganthan, "Ensemble strategies with adaptive evolutionary programming", *Information Sciences*, vol. 180, no. 9, May 2010, pp. 1571-1581.
 - [341] R. Mallipeddi, P. N. Suganthan, "Ensemble of Constraint Handling Techniques", *IEEE Trans. on Evolutionary Computation*, Vol. 14, No. 4, pp. 561 - 579 , Aug. 2010.
 - [342] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," *Proc. IEEE Congr. Evol. Comput. (CEC 2005)* Edinburgh, Scotland, IEEE Press, Sept. 2005, pp. 1785-1791.
 - [343] V. L. Huang, A. K. Qin, and P. N. Suganthan, "Self-adaptive differential evolution algorithm for constrained real-parameter optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC 2006)*, Jul. 2006, pp. 17-24.
 - [344] S. Z. Zhao, P. N. Suganthan and S. Das, "Self-adaptive Differential Evolution with Multi-trajectory Search for Large Scale Optimization", *Soft Computing*, DOI: 10.1007/s00500-010-0645-4.
 - [345] S. Z. Zhao, P. N. Suganthan and S. Das, "Self-adaptive Differential Evolution with Modified Multi-trajectory Search for CEC'2010 Large Scale Optimization" *International Conference on Swarm, Evolutionary and Memetic Computing (SEMCCO 2010)*, vol. 6466/2010, 1-10, Chennai, India, Dec 16-18, 2010.
 - [346] F. Herrera M. Lozano D. Molina, "Test Suite for the Special Issue of Soft Computing on Scalability of Evolutionary Algorithms and other Metaheuristics for Large Scale Continuous Optimization Problems". Available: <http://sci2s.ugr.es/eamhco/CFP.php>.
 - [347] L. Y. Tseng and C. Chen, "Multiple trajectory search for multiobjective optimization," *Proceeding 2007 IEEE Congress on Evolutionary Computation*, pp. 3609-3616, 2007.
 - [348] L. Y. Tseng and C. Chen, "Multiple trajectory search for large scale global optimization," *Proceeding 2008 IEEE Congress on Evolutionary Computation*, pp. 3052-3059, 2008.
 - [349] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, and Z. Yang, "Benchmark Functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization," *Technical Report*, Nature Inspired Computation and Applications Laboratory, USTC, China, & Nanyang Technological University, Singapore, Nov. 2007.
 - [350] K. Tang, Xiaodong Li, P. N. Suganthan, Z. Yang and T. Weise, "Benchmark Functions for the CEC'2010 Special Session and Competition on Large Scale Global Optimization," *Technical Report*, Nature Inspired Computation and Applications

- Laboratory, USTC, China, <http://nical.ustc.edu.cn/cec10ss.php>, & Nanyang Technological University, 2009.
- [351] A. Párowski, A clearing procedure as a niching method for genetic algorithms, *Proc. of the IEEE Int. Conf. on Evolutionary Computation*, New York, USA, 1996, pp. 798–803.
- [352] S. García, D. Molina, M. Lozano, and F. Herrera, "A Study on the use of Non-parametric Tests for Analyzing the Evolutionary Algorithms' Behaviour: A Case Study on the CEC'2005 Special Session on Real Parameter Optimization," *Journal of Heuristics*, 2007.
- [353] J. B. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," *ASME Transactions*, v64 (1942), pp. 759-768.
- [354] K. J. Astrom, T. Hagglund. *PID Controllers: Theory, design, and tuning*. Instrument society of America. 2nd Edition, 1995.
- [355] K. H. Ang, G. Chang, Y. Li. PID control system Analysis, Design and Technology. *IEEE trans. Contr. Syst. Technol.* 2005; 13(4):559-577. DOI:10.1109/TCST.2005.847331.
- [356] H. Bervani, T. Hiyama. Multiobjective PI/PID control design using an iterative linear matrix inequalities algorithm. *Int. Journal of Control, Automation and Systems* 2007; 5(2),117-127.
- [357] A. Chipperfield, P. Fleming. Multiobjective gas turbine engine controller design using genetic algorithm. *IEEE Trans. On Industrial Electronics* 1996; 43(5):583-587. DOI: 10.1109/41.538616.
- [358] W-D Chang and S-P Shih, "PID controller design of nonlinear systems using an improved particle swarm optimization approach", *Communications in Nonlinear Science and Numerical Simulation*, vol. 15, Issue 11, November 2010, Pages 3632-3639.
- [359] A. Herreros, E. Baeyens, J. Peran. Design of PID-type controllers using multiobjective Genetic Algorithms. *ISA Transactions* 2002; 41:457-472. DOI: 10.1016/S0019-0578(07)60102-5.
- [360] M.-H. Hung, A.-S. Shu, S.-J. Ho, S.-F. Hwang, S.-Y. Ho. A novel intelligent multiobjective simulated annealing algorithm for designing robust PID controllers. *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans* 2008; 38(2):319-330. DOI: 10.1109/TSMCA.2007.914793.
- [361] C-C Kao, C-W Chuang and R-F Fung, "The self-tuning PID control in a slider-crank mechanism system by applying particle swarm optimization approach", *Mechatronics*, vol. 16, Issue 8, October 2006, Pages 513-522.
- [362] C.-L. Lin, H.-Y. Jan, N.-C. Shieh. GA-based multiobjective PID and control for a linear brushless DC motor. *IEEE/ASME Trans. On Mechatronics* 2003; 8(1):56-65. DOI:10.1109/TMECH.2003.809136.
- [363] C.-L. Lin, H.-Y. Jan. Multiobjective PID control for a linear brushless DC motor: An evolutionary approach. *IEE Proc. Electr. Power Appl.* 2002; 149(6):397-406. DOI:10.1049/ip-epa:20020487.
- [364] L. dos Santos Coelho and D. L. de Andrade Bernert, "PID control design for chaotic synchronization using a tribes optimization approach", *Chaos, Solitons & Fractals*, vol. 42, Issue 1, 15 October 2009, Pages 634-640.
- [365] R.R. Sumar, A.A.R. Coelho and L.D. Coelho, "Computational intelligence approach to PID controller design using the universal model", *Information Sciences*, Vol. 180 (20), pp. 3980-3991, OCT 2010.

- [366] C.M. Seaman, A.A. Desrochers, G. F. List. Multiobjective optimization of a plastic injection molding process. *IEEE Trans. Control Syst. Tech.* 1994; 2(3):157-168. DOI: 10.1109/87.317974.
- [367] C.-S. Tseng, B.-S. Chen. Multiobjective PID control design in uncertain robotic systems using neural network elimination scheme. *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans* 2001; 31(6):632-644. DOI: 10.1109/3468.983419.
- [368] R.-H. C. Takahashi, P.-L.D. Peres, P.-A.V. Ferreira. Multiobjective H_2/H_∞ Guaranteed cost PID design. *IEEE Control Syst. Mag.* 1997; 17(5):37-47. DOI: 10.1109/37.621468.
- [369] J.H. Zhang, J. Zhuang and H.F. Du, "Self-organizing genetic algorithm based tuning of PID controllers", *Information Sciences*, Vol. 179 (7), pp. 1007-1018, MAR 2009.
- [370] M. Zamani, M. Karimi-Ghartemani, N. Sadati and M. Parniani, "Design of a fractional order PID controller for an AVR using particle swarm optimization", *Control Engineering Practice*, vol. 17, Issue 12, December 2009, Pages 1380-1387.
- [371] J. Doyle, B. Francis, A. Tannenbaum. *Feedback Control Theory*. Macmillan: New York, 1992.
- [372] K. Zhou, J.C. Doyle, K. Glover. *Robust and Optimal Control*. Prentice Hall: New Jersey, 1996.
- [373] B.-S. Chen, Y.-M. Cheng, C.-H. Lee. A genetic approach to mixed H_2/H_∞ optimal PID control. *IEEE Control Syst. Mag.* 1995; 15(5):51–60.
- [374] B.-S. Chen, Y.-M. Cheng, A structure-specified H_∞ optimal control design for practical applications: a genetic approach. *IEEE Trans. Contr. Syst. Technol.* 1998; 6(6):707–718. DOI: 10.1109/87.726532.
- [375] S.-J. Ho, S.-Y. Ho, L.-S. Shu. OSA: orthogonal simulated annealing algorithm and its application to designing mixed H_2/H_∞ optimal controllers. *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans* 2004; 34(5):588–600. DOI: 10.1109/TSMCA.2004.832834.
- [376] S.-J. Ho, S.-Y. Ho, M.-H. Hung, L.-S. Shu, H.-L. Huang. Designing structure-specified mixed H_2/H_∞ optimal controllers using an intelligent genetic algorithm IGA. *IEEE Trans. Contr. Syst. Technol.* 2005; 13(6):1119-1124. DOI:10.1109/TCST.2005.857403.
- [377] R. A. Krohling, and J.P. Rey. Design of optimal disturbance rejection PID controllers using genetic algorithms. *IEEE Trans. Evol. Comput.* 2001; 5(2):78–82. DOI: 10.1109/4235.910467.
- [378] I. Kitsios, T. Pimenides, P.A. Groumpos. A genetic algorithm for designing H_∞ structured specified controllers. *Proc. IEEE Int. Conf. Control Applications*, Mexico, 2001; 1196–1201. DOI: 10.1109/CCA.2001.974035.
- [379] S. Kanev, C. Scherer, M. Verhagegen, B.D. Schutter. Robust output-feedback controller design via local BMI optimization. *Automatica* 2004; 40:1115-1127. DOI:10.1016/j.automatica.2004.01.028.
- [380] M. Zamani, N. Sadati and M. K. Ghartemani, "Design of an H-infinity PID Controller Using Particle Swarm Optimization", *International Journal of Control Automation and Systems*, Volume: 7, Issue: 2, P ages: 273-280, APR 2009.
- [381] M. W. Iruthayarajan and S. Baskar, "Evolutionary algorithms based design of multivariable PID controller", *Expert Systems with Applications*, Vol. 36, Issue 5, Pages 9159-9167, July 2009.
- [382] A. Biswas, S. Das, A. Abraham, and S. Dasgupta, "Design of fractional-order PID controllers with an improved differential evolution", *Engineering Applications of Artificial Intelligence*, Vol. 22, Issue 2, Pages 343-350, March 2009.

-
- [383] W. Tan, T. Chen, H.J. Marquez. Robust controller design and PID tuning for multivariable processes. *Asian J. Control* 2002; 4(4):439–451. DOI: 10.1111/j.1934-6093.2002.tb00085.x.
- [384] J. Knowles, L. Thiele and E. Zitzler, "A tutorial on the performance assessment of stochastic multi-objective optimizers", *TIK-Report* No. 214, Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland. 2006.
- [385] I. J. Nagrath, M. Gopal, "Control System Engineering", New Age International (P) Ltd, New Delhi, 3rd Edn, 1999.
- [386] M. Jamshidi, R. A. Krohling, L. D. S. Coelho, P J. Fleming, "Robust Control Systems with Genetic Algorithms", CRC Press, 2002.
- [387] P. B. de Moura Oliveira, E. J. Solteiro Pires, J. Boaventura Cunha and Damir Vrančić, "Multi-Objective Particle Swarm Optimization Design of PID Controllers", *Lecture Notes in Computer Science*, Springer, Vol. 5518, 2009, pp. 1222-1230.
- [388] C. A. Kumar and N.K. Nair, "Multi-objective PID controller based on adaptive weighted PSO with application to steam temperature control in boilers", *International Journal of Engineering Science and Technology*, Vol. 2(7), 2010, pp. 3179-3184.
- [389] I.-L. Chien and B.A. Ogunnaike, "Modeling and control of a temperature-based high-purity distillation column", *Chem. Eng. Comm.*, Vol. 158, pp. 71-105, 1997.
- [390] I. M. Skolnik, Radar Handbook, McGraw-Hill, 1990.
- [391] S. M. Sherman, Monopulse Principles and Techniques, Artech House, 1984.
- [392] E. T. Bayliss, "Design of monopulse antenna difference patterns with low sidelobes", *Bell Syst. Tech. J.*, 47, pp. 623–650, 1968.
- [393] D. A. McNamara, "Synthesis of sum and difference patterns for two section monopulse arrays," *Proc. Inst. Elect. Eng.*, pt. H, vol. 135, no. 6, pp. 371–374, Dec. 1988.
- [394] R. S. Elliott, "Antenna theory and design", Prentice Hall, Englewood Cliffs, NJ, 1981.
- [395] P. López, J. A. Rodríguez, F. Ares, and E. Moreno, "Subarray weighting for the difference patterns of monopulse antennas: Joint optimization of subarray configurations and weights," *IEEE Trans. Antennas Propag.*, Vol. 49, No. 11, pp. 1606–1608, Nov. 2001.
- [396] S. Caorsi, A. Massa, M. Pastorino, and A. Randazzo, "Optimization of the difference patterns for monopulse antennas by a hybrid real/integer coded differential evolution method," *IEEE Trans. Antennas Propag.*, Vol. 53, No. 1, pp. 372–376, Jan. 2005.
- [397] A. Massa, M. Pastorino, and A. Randazzo, "Optimization of the directivity of a monopulse antenna with a subarray weighting by a hybrid differential evolution method," *IEEE Antennas Wireless Propag. Lett.*, Vol. 5, pp. 155–158, 2006.
- [398] K. Deb, Multi-Objective Optimization using Evolutionary Algorithms, John Wiley & Sons, 2001.
- [399] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Springer, 2007.
- [400] M. A. Abido, "A novel multiobjective evolutionary algorithm for environmental/economic power dispatch", *Electric Power Systems Research*, Elsevier, 65, 71- 81, 2003.
- [401] C. L. Dolph, "A current distribution for broadside arrays", *Proc. IRE*, vol. 34, pp. 335-348, June 1946.

- [402] M. Abramovitz and I. A. Stegun, *Handbook of Mathematical Functions*, Dover Publications, New York, 1965.
- [403] A. Abraham, L. C. Jain and R. Goldberg (Eds.), *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*, Springer Verlag, London, 2005.
- [404] J. D. Knowles and D. W. Corne, "Approximating the nondominated front using the pareto archived evolution strategy", *Evolutionary Computation*, vol. 8, no. 2, pp. 149-172, 2000.
- [405] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II", *IEEE Transactions on Evolutionary Computation*, Vol. 13, No. 2, Page(s) 284 – 302, April 2009.
- [406] G. Beni and J. Wang, "Swarm Intelligence in Cellular Robotic Systems", *Proc. of NATO Advanced Workshop on Robots and Biological Systems*, Tuscany, Italy, June 26–30, 1989.
- [407] N. A. Barricelli, "Esempi numerici di processi di evoluzione", *Methodos*, pp. 45–68, 1954.
- [408] J. L. Crosby, "Computer Simulation in Genetics", John Wiley & Sons, London, 1973.
- [409] A. Coloni, M. Dorigo et V. Maniezzo, "Distributed Optimization by Ant Colonies, actes de la première conférence européenne sur la vie artificielle", Paris, France, Elsevier Publishing, 134-142, 1991.
- [410] M. Dorigo, "Optimization, Learning and Natural Algorithms", PhD thesis, Politecnico di Milano, Italie, 1992.

Appendix A

Statistical Tests used for Comparing Algorithms

Statistical tests are used to compare the performance of different evolutionary algorithms on a given problem or on a given set of different problems.

In [352], the t -test is a parametric test and can be used to compare the performance of two different algorithms on each single problem. In our research works, t -test with 0.05 significance level is employed.

Appendix B

Problems of CEC 2005 [256]

Many benchmark numerical functions commonly used to evaluate and compare optimization algorithms may suffer from two problems. Firstly, global optimum lies at the center of the search domain. Secondly, local optima lie along the coordinate axes or no linkage among the variables/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 domain. 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.

Two composition functions are chosen from [258], 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

[258], which are not repeated here.

In the following, 20 test functions classified into four groups are listed, among which functions $f_1 - f_2$ are unimodal, functions $f_3 - f_{10}$ are unrotated multimodal, while the functions $f_{11} - f_{18}$ are rotated multimodal, and last two functions f_{19} and f_{20} are Composition problems. These 20 test functions ($f_1 - f_{20}$) are dimension-wise scalable.

Group A: Unimodal and Simple Multi-modal Problems:

1) *Sphere function*

$$f_1(x) = \sum_{i=1}^D x_i^2$$

2) *Rosenbrock's function*

$$f_2(x) = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$$

Group B: Unrotated Multimodal Problems:

3) *Ackley's function*

$$f_1(x) = -20 \exp(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2}) - \exp(\frac{1}{D} \sum_{i=1}^D \cos(2\pi x_i)) + 20 + e$$

4) *Griewank's function*

$$f_2(x) = \sum_{i=1}^D \frac{x_i^2}{4000} - \prod_{i=1}^D \cos(\frac{x_i}{\sqrt{i}}) + 1$$

5) *Weierstrass function*

$$f_3(x) = \sum_{i=1}^D (\sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k (x_i + 0.5))]) - D \sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k \cdot 0.5)],$$

$a=0.5, b=3, k_{\max}=20$

6) *Rastrigin's function*

$$f_4(x) = \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i) + 10)$$

7) *Non-continuous Rastrigin's function*

$$f_5(x) = \sum_{i=1}^D (y_i^2 - 10 \cos(2\pi y_i) + 10),$$

$$y_i = \begin{cases} x_i & |x_i| < 1/2 \\ \text{round}(2x_i)/2 & |x_i| \geq 1/2 \end{cases} \text{ for } i = 1, 2, \dots, D$$

8) *Schwefel's function*

$$f_6(x) = 418.9829 \times D - \sum_{i=1}^D x_i \sin(|x_i|^{1/2})$$

9) *Michalewicz's function*

$$f_7(x) = -\sum_{i=1}^D \sin x_i \left(\sin\left(\frac{i\pi x_i^2}{\pi}\right) \right)^{10}$$

10) *Sum of powers function*

$$f_8(x) = \sum_{i=1}^D |x_i|^{i+1}$$

Group C: Rotated Multimodal Problems:

In Group 1, some functions are separable and they can be solved by using D one-dimensional searches where D is the dimensionality of the problem. Hence, in Group 2, we have the corresponding six rotated multimodal problems.

11) *Rotated Ackley's function*

$$f_9(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D y_i^2}\right) - \exp\left(\frac{1}{D} \sum_{i=1}^D \cos(2\pi y_i)\right) + 20 + e, \mathbf{y} = \mathbf{M} \times \mathbf{x}$$

12) *Rotated Griewank's function*

$$f_{10}(x) = \sum_{i=1}^D \frac{y_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{y_i}{\sqrt{i}}\right) + 1, \mathbf{y} = \mathbf{M} \times \mathbf{x}$$

13) *Rotated Weierstrass function*

$$f_{11}(x) = \sum_{i=1}^D \left(\sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k (y_i + 0.5))] \right) - D \sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k \cdot 0.5)],$$

$$a=0.5, b=3, kmax=20, \mathbf{y}=\mathbf{M}\times\mathbf{x}$$

14) Rotated Rastrigin's function

$$f_{12}(x) = \sum_{i=1}^D (y_i^2 - 10\cos(2\pi y_i) + 10), \mathbf{y}=\mathbf{M}\times\mathbf{x}$$

15) Rotated Non-continuous Rastrigin's function

$$f_{13}(x) = \sum_{i=1}^D (z_i^2 - 10\cos(2\pi z_i) + 10)$$

$$z_i = \begin{cases} y_i & |y_i| < 1/2 \\ \text{round}(2y_i)/2 & |y_i| \geq 1/2 \end{cases} \text{ for } i = 1, 2, \dots, D, \mathbf{y}=\mathbf{M}\times\mathbf{x}$$

16) Rotated Schwefel's function

$$f_{14}(x) = 418.9829 \times D - \sum_{i=1}^D z_i$$

$$z_i = \begin{cases} y_i \sin(|y_i|^{1/2}) & \text{if } |y_i| \leq 500 \\ 0.001(|y_i| - 500)^2 & \text{if } |y_i| > 500 \end{cases}, \text{ for } i = 1, 2, \dots, D,$$

$$\mathbf{y} = \mathbf{y}' + 420.96, \mathbf{y}' = \mathbf{M} * (\mathbf{x} - 420.96)$$

17) Rotated Michalewicz's function

$$f_{15}(x) = -\sum_{i=1}^D \sin x_i (\sin(\frac{i\pi x_i^2}{\pi}))^{10}, \quad \mathbf{y}=\mathbf{M}\times\mathbf{x}$$

18) Rotated Sum of powers function

$$f_{16}(x) = \sum_{i=1}^D |x_i|^{i+1}, \quad \mathbf{y}=\mathbf{M}\times\mathbf{x}$$

In rotated Schwefel's function, in order to keep the global optimum in the search domain after rotation, noting that the original global optimum of Schwefel's function is at [420.96, 420.96, ..., 420.96], $\mathbf{y}' = \mathbf{M} * (\mathbf{x} - 420.96)$ and $\mathbf{y} = \mathbf{y}' + 420.96$ are used instead of $\mathbf{y} = \mathbf{M} \times \mathbf{x}$. Since Schwefel's function has better solutions out of the search domain $[-500, 500]^D$, when $|y_i| > 500$, $z_i = 0.001(|y_i| - 500)^2$, z_i is set in portion to the square distance between y_i and the bound.

Group D: Composition Problems

Composition functions are constructed using some basic benchmark problems to obtain more challenging problems with a randomly located global optimum and several randomly located deep local optima. The Gaussian function is used to combine the simple benchmark functions and blur the function's structures. The composition functions are asymmetrical multimodal problems, with different properties in different search regions. The details on how to construct this class of functions and six composition functions are presented in [258]. Two of the six composition functions defined in [258] are used here. Parameters settings for the following two composition functions:

19) Composition function 1 (CF1) in [258]:

The f_9 (CF1) are composed using ten sphere functions. The global optimum is easy to find once the global basin is found.

20) Composition function 2 (CF5) in [258]:

The f_{10} (CF5) is composed using ten different benchmark functions: two rotated Rastrigin's functions, two rotated Weierstrass functions, two rotated Griewank's functions, two rotated Ackley's functions and two sphere functions. The CF5 is more complex than CF1 since even after the global basin is found, the global optimum is not easy to locate.

Appendix C

Definitions of CEC07 Multi-objective Optimization Problems [309]

OCA2

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

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

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

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

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

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

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

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

D : dimension

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

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 $x_i = o_i, \quad i = 2, \dots, D$

Extended Shifted ZDT2 (S_ZDT2)

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

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

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

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

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

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 $x_i = o_i, \quad i = 2, \dots, D$

Extended Shifted ZDT4 (S_ZDT4)

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

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

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

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

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

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

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

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

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

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

Extended rotated ZDT4 (R_ZDT4)

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

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

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

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

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

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

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

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

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

Extended shifted ZDT6 (S_ZDT6)

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

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

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

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

$$\mathbf{z} = \mathbf{x} - \mathbf{0}, \quad \mathbf{x} = [x_1, x_2, \dots, x_D], \quad \mathbf{z} = [z_1, z_2, \dots, z_D]$$

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

$x_i \in [x \min_i, x \max_i]$, $\mathbf{xmin}=[x \min_1, x \min_2, \dots, x \min_D]$ 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 $x_i = o_i, i = 2, \dots, D$

Extended shifted DTLZ2 (S_DTLZ2)

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

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

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

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

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

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

The Pareto-optimal solutions $x_i^* = 0.5 + o_i (x_i^* \in \mathbf{x}_M)$ and the objective function values must satisfy: $\sum_{m=1}^M (f_m^*)^2 = 0.5$

Extended Rotated DTLZ2 (R_DTLZ2)

$$\begin{aligned}
 f_1(\mathbf{x}) &= \begin{cases} (1 + g(\mathbf{x}_M)) \cos(z_1' \pi/2) \cos(z_2' \pi/2) \dots \cos(z_{M-2}' \pi/2) \cos(z_{M-1}' \pi/2) + 1, & z_i \geq 0 \\ S(psum_1) \left((1 + g(\mathbf{x}_M)) \cos(z_1' \pi/2) \cos(z_2' \pi/2) \dots \cos(z_{M-2}' \pi/2) \cos(z_{M-1}' \pi/2) + 1 \right), & \text{otherwise} \end{cases} \\
 f_2(\mathbf{x}) &= \begin{cases} (1 + g(\mathbf{x}_M)) \cos(z_1' \pi/2) \cos(z_2' \pi/2) \dots \cos(z_{M-2}' \pi/2) \sin(z_{M-1}' \pi/2) + 1, & z_i \geq 0 \\ S(psum_2) \left((1 + g(\mathbf{x}_M)) \cos(z_1' \pi/2) \cos(z_2' \pi/2) \dots \cos(z_{M-2}' \pi/2) \sin(z_{M-1}' \pi/2) + 1 \right), & \text{otherwise} \end{cases} \\
 f_3(\mathbf{x}) &= \begin{cases} (1 + g(\mathbf{x}_M)) \cos(z_1' \pi/2) \cos(z_2' \pi/2) \dots \sin(z_{M-2}' \pi/2) + 1, & z_i \geq 0 \\ S(psum_3) \left((1 + g(\mathbf{x}_M)) \cos(z_1' \pi/2) \cos(z_2' \pi/2) \dots \sin(z_{M-2}' \pi/2) + 1 \right), & \text{otherwise} \end{cases} \\
 \vdots & \\
 f_{M-1}(\mathbf{x}) &= \begin{cases} (1 + g(\mathbf{x}_M)) \cos(z_1' \pi/2) \sin(z_2' \pi/2) + 1, & z_i \geq 0 \\ S(psum_{M-1}) \left((1 + g(\mathbf{x}_M)) \cos(z_1' \pi/2) \sin(z_2' \pi/2) + 1 \right), & \text{otherwise} \end{cases} \\
 f_M(\mathbf{x}) &= \begin{cases} (1 + g(\mathbf{x}_M)) \sin(z_1' \pi/2) + 1, & z_i \geq 0 \\ S(psum_M) \left((1 + g(\mathbf{x}_M)) \sin(z_1' \pi/2) + 1 \right), & \text{otherwise} \end{cases}
 \end{aligned}$$

$$g(\mathbf{x}_M) = \sum_{x_i \in \mathbf{x}_M} (z_i' - 0.5)^2$$

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

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

D : dimension

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

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

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

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

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

Extended shifted DTLZ3 (S_DTLZ3)

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

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

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

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

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

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

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

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

WFG1

Shape $h_{m=1:M-1} = \text{convex}_{x_m}$

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

$$\begin{aligned}
 t^1 \quad t^1_{i=1:k} &= y_i \\
 t^1_{i=k+1:n} &= \text{s_linear}(y_i, 0.35) \\
 t^2 \quad t^2_{i=1:k} &= y_i \\
 t^2_{i=k+1:n} &= \text{b_flat}(y_i, 0.8, 0.75, 0.85) \\
 t^3 \quad t^3_{i=1:n} &= \text{b_poly}(y_i, 0.02)
 \end{aligned}$$

$$\begin{aligned}
 t^4 \quad t_{i=1:M-I}^4 &= r_sum(\{y_{(i-1)k/(M-1)+1}, \dots, y_{ik/(M-1)}\}, \\
 &\quad \{2((i-1)k/(M-1)+1), \dots, 2ik/(M-1)\}) \\
 t_M^4 &= r_sum(\{y_{k+1}, \dots, y_n\}, \{2(k+1), \dots, 2n\})
 \end{aligned}$$

Optimal solutions:

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

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

WFG8

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

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

Optimal solutions:

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

$$z_{i=k+1:n} = 2i \times 0.35^{(0.02+49.98(\frac{0.98}{49.98} - (1-2u) \lfloor 0.5-u \rfloor + \frac{0.98}{49.98}))^{-1}}$$

$$u = r_sum(\{z_1, \dots, z_{i-1}\}, \{1, \dots, 1\}).$$

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.

WFG9

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

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

$$t_M^3 = \text{r_nonsep}(\{y_{k+1}, \dots, y_n\}, l)$$

Optimal solutions:

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

$$u = r_sum(\{z_{i+1}, \dots, z_n\}, \{1, \dots, 1\}).$$

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.

The six functions NO.14-19 are five-objective versions of six functions NO.8-13.

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	✓	✓	✓	concave	✓	X
2.SYMPART	$f_{1:2}$	✓	NS	M	✓	✓	✓	concave	✓	X
3.S_ZDT1	f_1	1	S	U	✓	✓	✓	convex	X	X
	f_2	✓	S	U	✓	✓	✓	convex	X	X
4.S_ZDT2	f_1	1	S	U	✓	✓	✓	concave	X	X
	f_2	✓	S	U	✓	✓	✓	concave	X	X
5.S_ZDT4	f_1	1	S	U	✓	✓	✓	convex	X	X
	f_2	✓	S	M	✓	✓	✓	convex	X	X
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	✓	✓	✓	concave	✓	X
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.

Appendix D

Definitions of Four test problems [306]

Function 1

$$\text{Minimize} \quad f_1(x) = x_1$$

$$\text{Minimize} \quad f_2(x) = (1+10x_2)(1 - (\frac{x_1}{1+10x_2})^2 - \frac{x_1 \sin(8\pi x_1)}{1+10x_2})$$

$$\text{subjective to} \quad 0 \leq x \leq 1$$

Function 2

$$\text{Minimize} \quad f_1(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2)$$

$$\text{Minimize} \quad f_2(x) = \delta(x) = \frac{2.1952}{x_3^3x_4}$$

$$\text{subjective to} \quad \tau(x) - 13,600 \leq 0$$

$$\sigma(x) - 30,000 \leq 0$$

$$x_1 - x_4 \leq 0$$

$$P - P_C(x) \leq 0$$

$$\tau(x) = \sqrt{(\tau')^2 + (\tau'')^2 + (x_2\tau'\tau'') / \sqrt{0.25(x_2^2 + (x_1 + x_3)^2)}}$$

$$\tau' = \frac{6,000}{\sqrt{2}x_1x_2}$$

$$\tau'' = \frac{6,000(14 + 0.5x_2)\sqrt{0.25(x_2^2 + (x_1 + x_3)^2)}}{2\{0.707x_1x_2(x_2^2 / 12 + 0.25(x_1 + x_3)^2)\}}$$

$$\sigma(x) = \frac{504,000}{x_3^2x_4}$$

$$P_C(x) = 64,746.022(1 - 0.0282346x_3)x_3x_4^3$$

where $0.125 \leq x_1 \leq 5.0$, $0.1 \leq x_1 \leq 10.0$, $0.1 \leq x_1 \leq 10.0$ and $0.125 \leq x_1 \leq 5.0$.

Function 3

$$\text{Minimize} \quad f_1(x) = 4.9 \times 10^{-5} (x_2^2 - x_1^2) (x_4 - 1)$$

$$\text{Minimize} \quad f_2(x) = \frac{9.82 \times 10^6 (x_2^2 - x_1^2)}{x_3 x_4 (x_2^3 - x_1^3)}$$

$$\text{subjective to} \quad (x_2 - x_1) - 20 \geq 0$$

$$30 - 2.5(x_4 + 1) \geq 0$$

$$0.4 - \frac{x_3}{3.14(x_2^2 - x_1^2)} \geq 0$$

$$1 - \frac{2.22 \times 10^{-3} x_3 (x_2^3 - x_1^3)}{(x_2^2 - x_1^2)^2} \geq 0$$

$$\frac{2.66 \times 10^{-2} x_3 x_4 (x_2^3 - x_1^3)}{(x_2^2 - x_1^2)} - 900 \geq 0$$

where $55 \leq x_1 \leq 80$, $75 \leq x_1 \leq 110$, $1000 \leq x_1 \leq 3000$ and $2 \leq x_1 \leq 20$.

Function 4

$$\text{Minimize} \quad f_1(x) = L(2x_1 + \sqrt{2}x_2 + \sqrt{x_3} + x_4)$$

$$\text{Minimize} \quad f_2(x) = \frac{FL}{E} \left(\frac{2}{x_1} + \frac{2\sqrt{2}}{x_2} - \frac{2\sqrt{2}}{x_3} + \frac{2}{x_4} \right)$$

$$\text{subjective to} \quad F/\sigma \leq x_1 \leq 3F/\sigma$$

$$\sqrt{2}F/\sigma \leq x_2 \leq 3F/\sigma$$

$$\sqrt{2}F/\sigma \leq x_3 \leq 3F/\sigma$$

$$F/\sigma \leq x_4 \leq 3F/\sigma$$

where $F = 10\text{kN}$, $E = 2.0\text{e}05\text{kn} / \text{cm}^2$, $L = 200\text{cm}$ and $\sigma = 10\text{kN} / \text{cm}^2$