Novel particle swarm optimizers with hybrid, dynamic and adaptive neighborhood structures

Liang, Jing

2008

Liang, J. (2008). Novel particle swarm optimizers with hybrid, dynamic and adaptive neighborhood structures. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/41803>

<https://doi.org/10.32657/10356/41803>

Novel Particle Swarm Optimizers with Hybrid, Dynamic & Adaptive Neighborhood Structures

Liang Jing

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for the degree of
Doctorate of Philosophy

2008

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

.....
Date

.....
Liang Jing

ACKNOWLEDGMENTS

I would like to take this opportunity to express my gratitude to those who have provided me with help and encouragement during the development of the present works. I would thank them all, but there are some people who need special recognition.

First of all, I would like to thank my supervisor, Dr. Ponnuthurai Nagaratnam Suganthan. He has provided a fabulously good level of supervision throughout my Ph.D study. It has been truly a great pleasure and an honor to work with him. All these work would not have been possible without his help.

I am especially grateful to Qin Kai, Tang Ke and Huang Ling, who worked together with me. The valuable suggestions and help they provided assisted me in completing these works.

Thanks also to Dr. Kalyanmoy Deb. He has provided valuable suggestions on the work and I have benefited greatly from his vast technical expertise and insight. Dr. Chi Chiu Chan who has generously provided me with his time in the guidance of the simulation of the FBG sensor network is also not to be forgotten.

Meanwhile, I also want to express my appreciation to Dr. S. Baskar who once I had the opportunity to work with. The arguments and discussions in those days were valuable and helpful.

Special thanks to Dr. N. Hansen, Dr. Y-P Chen, Dr. A. Auger, S. Tiwari, Dr. Thomas Philip Runarsson, Dr. Efrén Mezura-Montes, Dr. Maurice Clerc, and Dr. Carlos A. Coello Coello, who once worked with me to select and define the standard benchmark functions set for the special session CEC'05 or CEC'06.

Not to be forgotten are my dear parents and all my good friends who had provided me with their constant support and love over the past years both directly and indirectly.

Last but not least, my greatest gratitude towards my husband Diao Xiaoning. His encouragement put me through these years towards the completion of my Ph.D study. He also supported me in various aspects all these years with patience. Without him the completion of this study would not be possible.

CONTENTS

ACKNOWLEDGMENTSi

CONTENTSii

SUMMARYiv

LIST OF TABLESv

LIST OF FIGURESvi

LIST OF ABBREVIATIONSviii

Chapter 1 Introduction..... 1

 1.1 Motivation..... 1

 1.2 Objectives 2

 1.3 Major Contributions of the Thesis 2

 1.4 Organization of the Thesis 3

Chapter 2 Background and Literature Survey5

 2.1 Optimization 5

 2.2 Evolutionary Algorithms 7

 2.3 Particle Swarm Optimizer..... 12

 2.4 The Original Algorithm..... 13

 2.5 Modifications 16

 2.6 Applications 20

Chapter 3 CLPSO for Single Objective Optimization..... 24

 3.1 Comprehensive Learning Particle Swarm Optimizer 24

 3.2 Introduction to the Test Functions 34

 3.3 Experimental Results and Discussions 44

 3.4 Conclusion 52

Chapter 4 DMS-PSO for Single Objective Optimization..... 52

 4.1 Dynamic Multi-Swarm Particle Swarm Optimizer 53

 4.2 Experiments 63

 4.3 Conclusion 76

Chapter 5 DMS-L-PSO for Constrained Optimization 77

 5.1 Constrained Optimization 77

- 5.2 Constraint-Handling Techniques..... 78
- 5.3 DMS-L-PSO with a New Constraint-Handling Mechanism 81
- 5.4 Experiments 87
- 5.5 Conclusion 94
- Chapter 6 DMS-PSO for Multi-Objective Optimization..... 95
 - 6.1 Multi-objective Optimization..... 95
 - 6.2 Multi-objective Particle Swarm Optimization 96
 - 6.3 DMS-PSO for Multi-Objective Optimization Problems..... 98
 - 6.4 Experiments 105
 - 6.5 Conclusion 117
- Chapter 7 Application..... 119
 - 7.1 FBG Sensor Network..... 119
 - 7.2 Improving the Performance of an FBG Sensor Network using Dynamic Multi-Swarm PSO..... 121
 - 7.3 Improving the Performance of an FBG Sensor Network using Tree Search Dynamic Multi-Swarm PSO (TS-DMS-PSO) 128
 - 7.4 Conclusion 132
- Chapter 8 Conclusions and Recommendations 133
 - 8.1 Conclusions..... 133
 - 8.2 Recommendations for Further Research..... 135
- AUTHOR’S PUBLICATIONS..... 138
- BIBLIOGRAPHY 141
- Appendix A..... 158
- Appendix B..... 181
- Appendix C..... 200

SUMMARY

Many real world problems can be formulated as optimization problems with various parameters to be optimized. Some problems only have one objective to be optimized, some may have multiple objectives to be optimized at the same time and some need to be optimized subjecting to one or more constraints. Thus numerous optimization algorithms have been proposed to solve these problems. Particle Swarm Optimizer (PSO) is a relatively new optimization algorithm which has shown its strength in the optimization world. This thesis presents two PSO variants, Comprehensive Learning PSO and Dynamic Multi-Swarm PSO, which have good global search ability and can solve complex multi-modal problems for single objective optimization. The latter one is extended to solve constrained optimization and multi-objective optimization problems successfully with a novel constraint-handling mechanism and a novel updating criterion respectively. Subsequently, the Dynamic Multi-Swarm PSO is applied to determine the Bragg wavelengths of the sensors in an FBG sensor network and a tree search structure is designed to improve the accuracy and reduce the computation cost.

LIST OF TABLES

Table 3-1 Mean Value of $\overline{R_2/R_1}$ for Sphere and Rastrigin's Functions.....29

Table 3-2 Global Optimum, Search Ranges and Initialization Ranges44

Table 3-3 Results for 10-D Problems46

Table 3-4 Results for 30-D Problems51

Table 4-1 Best Functions Error Values Achieved for 10-D Functions 1-664

Table 4-2 Best Functions Error Values Achieved for 10-D Functions 7-1265

Table 4-3 Best Functions Error Values Achieved for 10-D Functions 13-1865

Table 4-4 Best Functions Error Values Achieved for 10-D Functions 19-2566

Table 4-5 Successful FES & Success Performance for 10-D66

Table 4-6 Best Functions Error Values Achieved for 30-D Functions 1-667

Table 4-7 Best Functions Error Values Achieved for 30-D Functions 7-1267

Table 4-8 Best Functions Error Values Achieved for 30-D Functions 13-1868

Table 4-9 Best Functions Error Values Achieved for 30-D Functions 19-2568

Table 4-10 Successful FES & Success Performance for 30-D69

Table 4-11 Algorithm Cost71

Table 4-12 Success Rates of the 11 Algorithms for 10-D73

Table 4-13 Normalized Success Performance (SP/SPbest) for 10-D74

Table 4-14 Success Rates of the 11 Algorithms for 30-D74

Table 4-15 Normalized Success Performance (SP/SPbest) for 30-D75

Table 5-1 Error Values Achieved for Problems 1-688

Table 5-2 Error Values Achieved for Problems 7-1289

Table 5-3 Error Values Achieved for Problems 13-1889

Table 5-4 Error Values Achieved for Problems 19-2490

Table 5-5 Number of FES to Achieve the Fixed Accuracy Level90

Table 5-6 Success Rate92

Table 5-7 Normalized Success Performance93

Table 6-1 Convergence Metric (γ) Comparison of the Four Algorithms107

Table 6-2 Diversity Metric (Δ) Comparison of the Four Algorithms.....107

Table 6-3 Unary ϵ Value Comparison of the Four Algorithms108

Table 6-4 Binary ϵ Values for Problem SCH109

Table 6-5 Binary ϵ Values for Problem FON110

Table 6-6 Binary ϵ Values for Problem KUR.....111

Table 6-7 Binary ϵ Values for Problem ZDT1112

Table 6-8 Binary ϵ Values for Problem ZDT2113

Table 6-9 Binary ϵ Values for Problem ZDT3114

Table 6-10 Binary ϵ Values for Problem ZDT4115

Table 6-11 Binary ϵ Values for Problem ZDT6117

Table B-1 Data Set for Test Problem g19.....193

Table B-2 Data Set for Test Problem g20.....195

Table B-3 Details of the 24 Test Problems198

Table B-4 $f(x^*)$ and the Bounds for the 24 Problems199

LIST OF FIGURES

Fig. 2-1 Illustration of Global Optimum and Local Optima.....	6
Fig. 2-2 The General Flowchart of EA	9
Fig. 2-3 EA vs Repeated Local Search	10
Fig. 2-4 The Pictures of Flying Birds in the Nature	12
Fig. 2-5. Flowchart of the Original PSO.....	15
Fig. 2-6 Some Topology Structures for Local Version of PSO.....	18
Fig. 3-1 Selection of Exemplar Dimensions for Particle i	26
Fig. 3-2 The CLPSO's and The Original PSO's Possible Search Regions per Variable in a Swarm with Five Members	28
Fig. 3-3 Comparison of PSO and CLPSO's Potential Search Space.....	30
Fig. 3-4 Each Particle's P_c with a Population Size of 30	31
Fig. 3-5 CLPSO's Results on Six Test Functions with Different Refreshing Gap m	32
Fig. 3-6 Flowchart of the CLPSO Algorithm	33
Fig. 3-7 Pseudo Code of Composition Function.....	38
Fig. 3-8 Construct a Two-Dimensional Composition Function Using Five Sphere Functions.....	39
Fig. 3-9 The Landscape Maps of Group D problems	43
Fig. 3-10 The Median Convergence Characteristics of 10- D Test Functions	49
Fig. 4-1 DMS-PSO's Search.....	54
Fig. 4-2 Sub-Flowchart 1 for DMS-L-PSO	57
Fig. 4-3 Illustration of Local Search Phase for a Population with 10 Particles	58
Fig. 4-4 Sub-Flowchart 2 for DMS-L-PSO (Local Search Phase)	59
Fig. 4-5 Sub-Flowchart 3 for DMS-L-PSO (Convergence Phase)	60
Fig. 4-6 The Flowchart of DMS-L-PSO	61
Fig. 4-7 Convergence Maps of PSO, DMS-PSO, DMS-PSO with Adaptive P_c , DMS-L-PSO on 10- D Rastrigin's problem.....	62
Fig. 4-8 Empirical Distribution Over All Functions for 10- D	73
Fig. 4-9 Empirical Distribution Over All Functions for 30- D	75
Fig. 5-1 Illustration of the Search Behavior of a Particle i	84
Fig. 5-2 Empirical Distribution over All Functions.....	93
Fig. 6-1 External Archive Updating.....	99
Fig. 6-2 The Crowding Distance Calculation	100
Fig. 6-3 Illustration of Choosing Local Best for Each Sub-Swarm	100
Fig. 6-4 Illustration of an Extreme Example for Pbest Updating	102
Fig. 6-5 An Illustration About PbestUpdating.....	103
Fig. 6-6 The Flowchart of the DMS-MO-PSO	104
Fig. 6-7 Pareto Fronts Generated by the Four Algorithms on SCH.....	109
Fig. 6-8 Pareto Fronts Generated by the Four Algorithms on FON	110
Fig. 6-9 Pareto Fronts Generated by the Four Algorithms KUR.....	111
Fig. 6-10 Pareto Fronts Generated by the Four Algorithms on ZDT1.....	112
Fig. 6-11 Pareto Fronts Generated by the Four Algorithms on ZDT2.....	113
Fig. 6-12 Pareto Fronts Generated by the Four Algorithms on ZDT3.....	114
Fig. 6-13 Pareto Fronts Generated by the Four Algorithms on ZDT4.....	115
Fig. 6-14 Pareto Fronts Generated by the Four Algorithms on ZDT6.....	116

Fig. 7-1 Schematic Diagram for N-FBG Network..... 120

Fig. 7-2 The Spectrums of the 10-FBGs Sensor Network from the OSA 123

Fig. 7-3 The Mean RMS Values of the Wavelength Detection Error ($\xi = 0.1$ pm) 124

Fig. 7-4 Comparison of the Computational Cost ($\xi = 0.1$ pm)..... 124

Fig. 7-5 The Mean RMS Values of the Wavelength Detection Error Due to the Simple GA, CMA-ES and DMS-PSO for Overlapping Situation 125

Fig. 7-6 Experimental Setup for Two FBGs 126

Fig. 7-7 Spectrums Measured from OSA for Different Applied Strain Values.... 127

Fig. 7-8 Measured Bragg Wavelength vs. Applied Strain 128

Fig. 7-9 The Mean RMS Detection Errors for Partially Overlapped Case 130

Fig. 7-10 The Computation Costs for Partially Overlapped Case 130

Fig. 7-11 The Mean RMS Detection Errors for Overlapped Case 131

Fig. 7-12 The Computation Costs for Overlapped Case..... 131

Fig. A-1 3-D Map for 2-D Function F_1 158

Fig. A-2 3-D Map for 2-D Function F_2 159

Fig. A-3 3-D Map for 2-D Function F_3 160

Fig. A-4 3-D Map for 2-D Function F_4 160

Fig. A-5 3-D Map for 2-D Function F_5 161

Fig. A-6 3-D Map for 2-D Function F_6 162

Fig. A-7 3-D Map for 2-D Function F_7 162

Fig. A-8 3-D Map for 2-D Function F_8 163

Fig. A-9 3-D Map for 2-D Function F_9 164

Fig. A-10 3-D Map for 2-D Function F_{10} 164

Fig. A-11 3-D Map for 2-D Function F_{11} 165

Fig. A-12 3-D Map for 2-D Function F_{12} 166

Fig. A-13 3-D Map for 2-D Function F_{13} 167

Fig. A-14 3-D Map for 2-D Function F_{14} 167

Fig. A-15 3-D Map for 2-D Function F_{15} 169

Fig. A-16 3-D Map for 2-D Function F_{16} 170

Fig. A-17 3-D Map for 2-D Function F_{17} 170

Fig. A-18 3-D Map for 2-D Function F_{18} 172

Fig. A-19 3-D Map for 2-D Function F_{19} 172

Fig. A-20 3-D Map for 2-D Function F_{20} 173

Fig. A-21 3-D Map for 2-D Function F_{21} 174

Fig. A-22 3-D Map for 2-D Function F_{22} 175

Fig. A-23 3-D Map for 2-D Function F_{23} 176

Fig. A-24 3-D Map for 2-D Function F_{24} 178

Fig. C-1 The Search Space Near the Pareto-Optimal Region for SCH 200

Fig. C-2 The search space near the Pareto-optimal region for FON 201

Fig. C-3 The Search Space Near the Pareto-Optimal Region for KUR..... 201

Fig. C-4 The Search Space Near the Pareto-Optimal Region for ZDT1 202

Fig. C-5 The Search Space Near the Pareto-Optimal Region for ZDT2 203

Fig. C-6 The Search Space Near the Pareto-Optimal Region for ZDT3 203

Fig. C-7 The Search Space Near the Pareto-Optimal Region for ZDT4 204

Fig. C-8 The Search Space Near the Pareto-Optimal Region for ZDT6 205

LIST OF ABBREVIATIONS

- CEC:** IEEE Congress on Evolutionary Computation
- CPD:** Conventional Peak Detection
- CLPSO:** Comprehensive Learning Particle Swarm Optimizer
- DMS-C-PSO:** Dynamic Multi-Swarm Particle Swarm Optimizer with a Novel Constraint-Handling Mechanism
- DMS-L-PSO:** Dynamic Multi-Swarm Particle Swarm Optimizer with Local Search
- DMS-MO-PSO:** Dynamic Multi-Swarm Particle Swarm Optimizer for Multi-Objective Optimization.
- DMS-PSO:** Dynamic Multi-Swarm Particle Swarm Optimizer
- EA:** Evolutionary Algorithm
- EP:** Evolutionary Programming
- ES:** Evolution Strategies
- GA:** Genetic Algorithm
- FBG:** Fiber Bragg Grating
- MO:** Multi-Objective Optimization
- nm:** nano meter
- pm:** pico meter
- PSO:** Particle Swarm Optimizer
- TOF:** Tunable Optical Filter
- WDM:** Wavelength Division Multiplexed
- SQP:** Sequential Quadratic Programming

Chapter 1

Introduction

1.1 Motivation

Optimization forms an important part of our day-to-day life. Many scientific, engineering and economic problems involve optimization of a set of parameters. These problems include examples like minimizing the losses in a power grid by finding the optimal configuration of the components, or training a neural network to recognize face images. It is clear that there will always be a need for better optimization algorithms, since the complexity of the problems that we attempt to solve is ever increasing.

Numerous optimization algorithms have been proposed to solve these problems, with varying degrees of success. Swarm intelligence is a phenomenon in which many unsophisticated agents interact locally with their environment to produce global patterns of collective and emergent behavior. The Particle Swarm Optimizer (PSO) [1][2] is a relatively new technique based on swarm intelligence that has been empirically shown to solve many of these optimization problems well. Similar to other evolutionary computation techniques, the search is initialized with a group of randomly generated solutions in PSO. Individual solutions of the current population are evaluated using some fitness functions and better solutions for next generation are created. The particular characteristic of PSO is that no mutation or crossover operators are involved in the updating process. The old solutions are moved to the new position according to the historical information.

The major problem of PSO and some other evolutionary algorithms is that they tend to get trapped in local optima of objective functions. Therefore the primary objective of this research is to construct new and efficient algorithms, based on the swarm intelligence to improve the performance of current variants of the PSO, especially their global search abilities.

1.2 Objectives

The main objectives of this thesis are

- ✧ To develop better PSO algorithms with better global search ability to solve single objective optimization problems.
- ✧ To develop novel and efficient constraint-handling techniques to solve single objective constrained problems.
- ✧ To develop efficient PSO algorithms to solve multi-objective optimization problems.
- ✧ To use the novel PSO algorithms to improve the performance of an FBG sensor network.

1.3 Major Contributions of the Thesis

The major contributions of this thesis are:

- ✧ A comprehensive learning strategy is proposed, with which the particles obtain more freedom and a larger potential search range. The comprehensive learning strategy is combined in the PSO with inertia weight and a novel Comprehensive Learning Particle Swarm Optimizer (CLPSO) which possessing of better global search ability is constructed.
- ✧ The weaknesses of the existing benchmark functions for single objective optimization are analyzed and novel composition test functions are proposed. A composition function can own many different properties and the users can control the properties of the composition function through changing the basic functions and the parameters. Subsequently a standard benchmark function set is selected and defined.
- ✧ A new neighborhood topology for local PSO version and an adaptive self-learning strategy are proposed to result in Dynamic Multi-Swarm Particle Swarm Optimizer (DMS-PSO). By combining with a local search method, DMS-L-PSO, a PSO algorithm with good global and local search capabilities, is presented.
- ✧ A novel and efficient constraint-handling mechanism which utilizes multi swarms is proposed. With the help of the new constraint-handling method, DMS-L-PSO is extended to solve the constrained optimization problems.

CHAPTER 1 INTRODUCTION

- ✧ Novel **lbest** selection method and **pbest** and **lbest** updating schemes are proposed and combined with DMS-PSO to solve multi-objective optimization problems.
- ✧ DMS-PSO is applied to optimally design an FBG sensor network. A tree search structure is proposed according to the special property of this real world problem to reduce the computation time and increase the accuracy for the concerned FBG sensor network.

1.4 Organization of the Thesis

Chapter 2 presents a brief introduction to the theory of optimization and evolutionary algorithms. Then the Particle Swarm Optimizer (PSO) is described and a review of the existing PSO algorithm variants is presented.

Chapter 3 introduces the Comprehensive Learning Particle Swarm Optimizer and the composition test functions. The comparison results of the proposed CLPSO and other eight PSO versions are presented to show the superiority of the CLPSO.

Chapter 4 presents the Dynamic Multi-Swarm Particle Swarm Optimizer (DMS-L-PSO) which employs a periodically changing neighborhood topology structure, an adaptive self-learning strategy and a local search phase. DMS-L-PSO is tested using the standard benchmark functions set of CEC'05 and compared with other ten evolutionary algorithms.

Chapter 5 gives a brief review of the existing constraint-handling methods and then introduces the novel constraint-handling mechanism. DMS-L-PSO with the new constraint-handling technique (DMS-C-PSO) is described and tested on the standard set of benchmark functions proposed in CEC'06.

Chapter 6 presents a brief review of the existing multi-objective PSO algorithms. This is followed by the description of a new **pbest** and **lbest** updating strategy and a new **lbest** choosing schedule. DMS-PSO for multi-objective optimization (DMS-MO-PSO) is introduced and the results of DMS-MO-PSO on a group of test functions are presented to show the good search ability of the DMS-MO-PSO.

In Chapter 7, the FBG sensor network is introduced and DMS-PSO is applied to improve the performance of this FBG sensor network. In order to reduce the computation cost and increase the accuracy, a tree search structure is combined within the DMS-PSO. The results show that the proposed algorithm can accurately

CHAPTER 1 INTRODUCTION

determine the Bragg wavelengths of the sensors, when the spectrums of the FBGs are partially or completely overlapped.

Chapter 8 gives a conclusion of this thesis and recommendations for further research.

The appendices describe the standard benchmark functions set proposed in CEC'05, CEC'06 and the test functions used in Chapter 6 respectively.

Chapter 2

Background and Literature Survey

This chapter introduces the theory of optimization and evolutionary algorithms. Then a description of the Particle Swarm Optimizer (PSO) is presented, followed by a review of the existing PSO versions.

2.1 Optimization

The task of optimization is finding one or more solutions to minimize or maximize the given objective functions. Formally an optimization problem can be defined as follows:

$$\begin{aligned}
 &\text{Maximize/Minimize} && f_i(\mathbf{x}), i=1, \dots, M, \quad \mathbf{x}=[x_1, x_2, \dots, x_D] && (2-1) \\
 &\text{subject to:} && g_j(\mathbf{x}) \leq 0, j=1, \dots, J \\
 &&& h_k(\mathbf{x}) = 0, k=1, \dots, K \\
 &&& \mathbf{x} \in [\mathbf{X}_{\min}, \mathbf{X}_{\max}]^D
 \end{aligned}$$

$f_i(\mathbf{x})$ is called "objective function". $g_j(\mathbf{x})$ and $h_k(\mathbf{x})$ are "inequality constrained function" and "equality constrained function" respectively. A maximization problem $\max(f)$ can be transformed into a minimization problem as $\min(-f)$, thus in this thesis, optimization is used to mean minimization. When an optimization problem involves only one objective function, it is called single-objective optimization problem; and when an optimization problem addresses more than one objective function, it is called a multi-objective optimization problem. When an optimization problem requires some of the parameters satisfying one or more constraints, it is known as a constrained optimization problem; otherwise it is known as an unconstrained optimization problem. Usually the bounds constraints, $\mathbf{x} \in [\mathbf{X}_{\min}, \mathbf{X}_{\max}]^D$, are easy to handle. Optimization problem with only bounds constraints, $\mathbf{x} \in [\mathbf{X}_{\min}, \mathbf{X}_{\max}]^D$, are regarded as

CHAPTER 2 BACKGROUND AND LITERATURE SURVEY

unconstrained problems. If we denote the feasible region as F and the whole search space as S , $x \in F$ if $x \in S$ and all constraints are satisfied. In this case, x is called a feasible solution. For unconstrained optimization problem, $F = S$.

A problem which can be put into the form as (2-1) is also called Nonlinear Program (NLP). Because NLP is a difficult field, researchers have identified special cases for study. A particularly well studied case is the one where all the constraints g and h are linear. The name for such a problem, unsurprisingly, is "linearly constrained optimization". If, as well, the objective function is quadratic at most, this problem is called Quadratic Programming (QP). An even more special case of great importance is where the objective function and the constraints are entirely linear; this is called Linear Programming (LP).

A solution which satisfies $f(x^*) \leq f(x), \forall x \in S$ is called global optimum and it is possible to have more than one global optimum for an optimization problem. Except for the global optima, for a complex problem that there always exist solutions which only satisfy $f_L(x^*) \leq f_L(x), \forall x \in L$, where $L \subset S$ and S denotes the search range. Fig. 2-1 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.

Fig. 2-1 Illustration of Global Optimum and Local Optima

For every pair of points within the search range of a function $f(x)$, if every point on the straight line segment that joins them is also within the range, it is considered to be convex over the search range S . Convex functions are continuous and have

CHAPTER 2 BACKGROUND AND LITERATURE SURVEY

nice differentiability properties and they have no local optima that are not global. If the objective function of a problem is convex and the search range S is convex, it is called convex programming problem.

Many local optimization algorithms have been proposed, such as the Steepest Descent, interior-reflective Newton method and the Quasi-Newton method. Most of these algorithms start with an initial point $\mathbf{x}_0$ and search for a local optimum near the starting point. These methods are called the traditional methods in this thesis. Newton's method assumes that the function can be locally approximated as quadratic in the region around the optimum, and uses the first and second derivatives (gradient and Hessian) to find the stationary point. Quasi-Newton methods are based on Newton's method to find the stationary point of a function, where the gradient is 0. In Quasi-Newton methods the Hessian matrix of second derivatives of the function to be minimized does not need to be computed. The Hessian is updated by analyzing successive gradient vectors instead. The first Quasi-Newton method was proposed by W.C. Davidon in 1959 [3]: the DFP updating formula was proposed in 1963 by Fletcher and Powell [4]. One of the most common Quasi-Newton methods is the Broyden-Fletcher- Goldfarb-Shanno (BFGS) method, which was proposed independently by Broyden, Fletcher, Goldfarb, and Shanno, in 1970 [5][6][7][8]. The BFGS method derived from the Newton's method in optimization is a class of hill-climbing optimization techniques that seeks the stationary point of a function, where the gradient is 0.

If the task is finding the global optimum, it is possible to repeat this process starting at different start points to obtain the best solution among all the obtained local optima. When the optimization problem is more complex and with a huge number of local optima (such as Rastrigin's function and Weierstrass function, Figs A-9 and A-11, it is obvious that these methods cannot satisfy the requirement. Thus, the global optimization algorithms are needed, by which a global optimum can be found regardless of the start point $\mathbf{x}_0$. Evolutionary Algorithms (EAs) can solve this problem.

2.2 Evolutionary Algorithms

Evolutionary Algorithms (EAs) are developed based on the natural selection and survival of the fittest in the biological world. Most organisms evolve with two

CHAPTER 2 BACKGROUND AND LITERATURE SURVEY

primary processes: selection and reproduction. Selection determines the individuals which survive and reproduce, and reproduction mixes and recombined the genes of their offspring. When sperm and ova fuse, matching chromosomes line up with one another and then cross-over partway along their length, thus swapping genetic material. Evolutionary algorithms are guided stochastic search methods mimicking the metaphor of natural biological evolution. Computer simulations of evolution started as early as in 1954 with the work of Nils Aall Barricelli [9]. And then in 1957, Alex Fraser published a series of papers on simulation of artificial selection of organisms [10]. In the 1958, Hans Bremermann adopted a population of solution with recombination, mutation, and selection operators which are elements of modern genetic algorithms to optimization [11]. Then it is popularized by Holland who applied Evolutionary Algorithm to formally study adaptation in nature for the purpose of applying the mechanisms into computer science [12].

Different from the traditional optimization techniques, they 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 makes the population move towards regions of the space from which good solutions have already been seen. The flowchart of a typical evolutionary algorithm is presented in Fig. 2-2.

CHAPTER 2 BACKGROUND AND LITERATURE SURVEY

Fig. 2-2 *The General Flowchart of EA*

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 larger potential search spaces, thus the adverse influence of initialization is less when compared with local search algorithms. For complex multi-modal problems, EAs have higher chance to avoid getting stuck into a local optimum and can achieve better and more stable results. The illustration of search behavior of EAs and repeated local search is given in Fig. 2-3. Another important advantage of EAs is that they do not have to know any rules of the problem - they work by their own internal rules. The evolution is executed through comparing the fitness values of the individuals. The output of a system or the results of experiments can be comparison criteria of EAs. Thus they can solve problems which do not have exact mathematic modals and cannot be solved using the traditional methods introduced before which use the first and second derivatives in the updating process. This is very useful for complex or loosely defined problems. At the same time, it eliminates the need to compute derivative information. Sometimes it is very time-consuming to numerically obtain derivative values for complex problems.

CHAPTER 2 BACKGROUND AND LITERATURE SURVEY

(a) Repeated Local Search

(b-1) EA Step 1

(b-2) EA Step 2

(b-3) EA Step 3

(b-4) EA Step 4

Fig. 2-3 EA vs Repeated Local Search

CHAPTER 2 BACKGROUND AND LITERATURE SURVEY

Generally, there are three mainstream methods of evolutionary algorithms:

- Genetic Algorithms (GAs), is originally proposed by Holland in 1962 [12][13]. Genetic algorithms use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover (also called recombination). The operator which plays an important role in GA is the recombination. Mutation acts as an assistant operator.
- Evolutionary Programming (EP), is developed by Fogel [14][15] in the context of evolving finite state-machines to be used in the prediction of time series and relies on mutation operator. Members of the population are viewed as part of a specific species rather than members of the same species therefore each parent generates an offspring, using a $(\mu + \mu)$ survivor selection.
- Evolution Strategies (ES), is devised by Rechenberg and Schwefel [16][17][18][19][20]. Different from GAs, the main operator in ES is mutation, while recombination is an assisting operator. The simplest ES operates on a population with only two points: the current point (parent) and offspring which is generated by mutation. Only if the offspring is better than the parent, it becomes parent in next generation. Otherwise the parent is kept. This is called $(1+1)$ -ES. If λ offsprings are generated and compared with the parent, it is called $(1+\lambda)$ -ES. New derivatives of ES often use a population of μ parents and also recombination as an additional operator (called $(\mu/\rho+, \lambda)$ -ES).
- Genetic Programming (GP), was first reported and then popularized by John Koza [21][22]. The main difference between genetic programming and other evolutionary algorithms is the representation of the solution. Genetic programming creates computer programs in the lisp or scheme computer languages as the solution. It represents computer programs as tree structures. Trees can be easily evaluated in a recursive manner. Every tree node has an operator function and every terminal node has an operand, making mathematical expressions easy to evolve and evaluate.

Particle swarm optimization (PSO) is a relatively new evolutionary algorithm which is different from the above discussed evolutionary algorithms. Recently PSO has attracted more and more researcher and become an important embranchment of EA. It employs a population which evolves in each generation as other EA algorithms, but no crossover and mutation operators are used. It

CHAPTER 2 BACKGROUND AND LITERATURE SURVEY

simulates the behaviors of swarms such as fish schools, bird flocks. Its structure and properties will be described in detail in the following section.

2.3 Particle Swarm Optimizer

Fig. 2-4 The Pictures of Flying Birds in the Nature

Particle swarm optimization (PSO) is one of the evolutionary algorithms based on swarm intelligence. The PSO was first designed to simulate birds seeking food. Birds would find food through social cooperation with other birds within a neighborhood. Two beautiful pictures of the flying birds in the nature are provided

CHAPTER 2 BACKGROUND AND LITERATURE SURVEY

in Fig. 2-4. Suppose there is a group of birds searching food in an area. There are small pieces of food near the food centre and usually the nearer from the food centre, the bigger the food becomes. No bird knows where the food center is. So what's the best strategy to find the food? The effective one is to follow the bird which has found the biggest pieces. PSO just simulates this scenario and uses it to solve optimization problems.

In PSO, each single solution is a "bird" in the search space. We call it a "particle". All particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current best particles. Therefore, the particles have a tendency to fly towards better and better search area over the course of the search process. Since its introduction in 1995 by Kennedy and Eberhart [1][2], PSO has attracted a lot of attention as evidenced by the research results have been reported. The first book on PSO, *Swarm Intelligence*, coauthored by James Kennedy, Russell Eberhart and Yuhui Shi was published in 2001 by Morgan Kaufmann Publishers [24].

2.4 The Original Algorithm

The original PSO algorithm is described as below:

$$V_i^d \leftarrow V_i^d + c_1 * rand1_i^d * (pbest_i^d - X_i^d) + c_2 * rand2_i^d * (gbest^d - X_i^d) \quad (2-2)$$

$$X_i^d \leftarrow X_i^d + V_i^d \quad (2-3)$$

where c_1 and c_2 in the equation are the acceleration constants, which represent the weighting of stochastic acceleration terms that pull each particle toward **pbest** and **gbest** positions. $rand1_i^d$ and $rand2_i^d$ are two random numbers in the range [0,1]; $\mathbf{X}_i = (X_i^1, X_i^2, \dots, X_i^D)$ represents the position of the i th particle; $\mathbf{pbest}_i = (pbest_i^1, pbest_i^2, \dots, pbest_i^D)$ represents the best previous position (the position giving the best fitness value) of the i th particle; $\mathbf{gbest} = (gbest^1, gbest^2, \dots, gbest^D)$ represents the best previous position of the population; $\mathbf{V}_i = (V_i^1, V_i^2, \dots, V_i^D)$ represents the rate of the position change (velocity) for particle i .

Eqns. (2-2) and (2-3) describe the flying trajectory of the particles. Velocities of

CHAPTER 2 BACKGROUND AND LITERATURE SURVEY

the particles are dynamically updated by using Eqn. (2-2) and Eqn. (2-3) is the position updating equation of the particles. Eqn. (2-2) consists of three parts. The first part is the momentum part. The velocity is changed from current velocity rather than being changed abruptly. This part is to improve the global search ability of the particles by preventing the particles converging too fast. The second part is the “cognitive” part, which represents how the particles learn from their own flying experience. The third part is the “social” part, which represents how the particles learn from the group's flying experience [25].

When updating the velocity of a particle using Eqn. (2-2), different dimensions have different $rand1_i^d$ and $rand2_i^d$. Some researchers [27][28][29] use the following updating equation:

$$V_i^d \leftarrow V_i^d + c_1 * rand1_i^d * (pbest_i^d - X_i^d) + c_2 * rand2_i^d * (gbest^d - X_i^d) \quad (2-4)$$

Comparing the two variants in Eqns. (2-2) and (2-4), the former one has a larger search space due to independent updating of each dimension, while the second is dimension-dependant and has a smaller search space due to the same random numbers being used for all dimensions. Eqn. (2-2) always yields better performance on unrotated problems than the rotated version of the problems since it treats the different dimensions as separate parts and the relativity among the dimensions are not considered in the updating equation. Eqn. (2-2) performs consistently on unrotated and rotated problems [30]. As the first updating strategy achieves a larger search space and always performs better, we use Eqn. (2-2) in our study. The particles' velocity on each dimension is clamped to a maximum velocity **Vmax**. Large **Vmax** makes particles have the potential to fly far past good solution areas while a small **Vmax** makes particles have the potential to be trapped into local optima. Usually a fixed constant value is predefined by the user. The value can be decided by experiments. Generally either the full or half length of the search range is used.

The PSO algorithm is simple in concept, easy to implement and computationally efficient. Like the other evolutionary algorithms, PSO algorithms are a population based search algorithm with random initialization, and the behaviors of the individuals are influenced by the other members of the population. The difference is that in PSO each particle which flies through the solution space has the ability to remember its previous best position and uses the historical information to guide the

CHAPTER 2 BACKGROUND AND LITERATURE SURVEY

flying direction of the particle [31]. Compared to the other evolutionary algorithms, the original version of PSO is faster in initial convergence while slower in refining [32][33]. The flowchart of the original PSO is as given in Fig. 2-5.

ps : population size

k : generation counter from 1 to max_gen

d : dimension

$c_1 = c_2 = 2$

max_gen : maximum generations

i : particle's id counter from 1 to ps

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

Fig. 2-5 Flowchart of the Original PSO[1][2]

2.5 Modifications

Many researchers have worked on improving the performance of PSO in various ways and developed many interesting variations of the PSO algorithm.

2.5.1 Modifying the Parameters

Shi and Eberhart introduced one new parameter, the inertia weight, into the original PSO algorithm [25]. The velocity updating equation of PSO with inertia weight is:

$$V_i^d \leftarrow \omega * V_i^d + c_1 * rand1_i^d * (pbest_i^d - X_i^d) + c_2 * rand2_i^d * (gbest^d - X_i^d) \quad (2-5)$$

Eqn. (2-5) is the same as the Eqn. (2-2) except the inertia weight ω . The inertia weight is an important parameter and it can balance between the global and local search capabilities. A large inertia weight facilitates global, while a small inertia weight facilitates local search.

The inertia weight is further investigated in [48] and a linearly decreasing inertia weight is introduced by Shi and Eberhart [47] to the PSO. The inertia weight is kept large in the initial search period to make the particles have better global search ability and avoid falling into local minima and then a small inertia weight is employed at the end of PSO running to refine the best solution found so far. This version is used as the basic PSO in the experimental comparisons of this thesis. They further designed fuzzy systems 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 output of the fuzzy systems. The drawback of this method is that there are more parameters need to be set to construct this fuzzy system. In [49], the inertia weight was set as a random varying number between 0.5 and 1.0 rather than time decreasing from 0.9 to 0.4.

A so called constriction coefficient is proposed in [42][43] by Clerc to ensure the PSO's convergence. The velocity updating equation becomes:

$$V_i^d \leftarrow k * [V_i^d + c_1 * rand1_i^d * (pbest_i^d - X_i^d) + c_2 * rand2_i^d * (gbest^d - X_i^d)] \quad (2-6)$$

$$\text{with } k = \frac{2}{2 - \varphi - \sqrt{\varphi^2 - 4\varphi}} \quad \text{where } \varphi = c_1 + c_2, \varphi > 4 \quad (2-7)$$

When Clerc's constriction method is used, φ is commonly set to 4.1 and the constant multiplier k is approximately 0.729. If we replace k with the inertia weight ω and make c_1 and c_2 meet the condition $\varphi = c_1 + c_2$, $\varphi > 4$, the PSO algorithm

CHAPTER 2 BACKGROUND AND LITERATURE SURVEY

with the constriction factor can be considered as a special case of the PSO with inertia weight [44]. The constriction method with the commonly used setting becomes equivalent to the PSO with inertia weight when $\omega \approx 0.729$ and $c_1 = c_2 = 1.49445$. Clerc's constriction method guarantees the convergence of the PSO while it employs the same parameters in the whole search process. This causes the particles not to have a large diversity at the beginning as PSO with the linearly decreasing inertia weight. Comparing these two versions, the PSO with linearly decreasing inertia weight can achieve better performance on multimodal problems because the large diversity at the beginning prevents the premature convergence. But the designed constriction method has better convergence ability and always performs better on unimodal problems.

In the classical PSO, there are two parameters named acceleration constants c_1 and c_2 which are used to balance the stochastic acceleration terms to pull each particle towards **pbest** and **gbest**. They also determine the balance of the global and local search ability of PSO. With a larger c_1 , the particles will have the potential to fly to the best position found by all the particles, and with a larger c_2 , the particles will have the potential to search around the best positions found by themselves. Thus, in fact c_1 improves the local search around **gbest** while a larger c_2 maintains the particles with a larger diversity and improves the global search capability of the particles. Ratnaweera *et al.* proposed self-adaptive acceleration coefficients in [45] and then in [46], they proposed a Self-Organizing Hierarchical Particle Swarm Optimizer with time varying acceleration coefficients. In the Self-Organizing Hierarchical Particle Swarm Optimizer, the inertia weight is set to be 0. Only the "social" part and the "cognitive" part of the particle swarm strategy are considered to estimate the new velocity of each particle and the particles are reinitialized whenever they are stagnated in the search space.

Fan and Shi suggested that the dynamically changing **Vmax** may improve the PSO's performance and introduced a linearly decreasing **Vmax** in [26].

2.5.2 Using Topologies

There are two versions of PSOs namely the global and the local PSO. In the global version of PSO, each particle dynamically adjusts its velocity and position

CHAPTER 2 BACKGROUND AND LITERATURE SURVEY

according to the best solution found so far by itself and the best solution found so far by the whole swarm. While in the local version of PSO, each particle adjusts its velocity and position according to its personal best and the best solution achieved so far within its neighborhood. The local version of PSO can be transformed into the global version if we set the neighborhood of each particle as the whole swarm. Since the information of the best solution is passed to each particle immediately, the global version of PSO converges fast, but with potential to converge to a local minimum. While in the local version of PSO, a particle is only influenced by the particles in its neighborhood, so it might have less chances to get trapped in the local optima and has more chances to find better solutions [24][50].

Fig. 2-6 Some Topology Structures for Local Version of PSO [51]

A lot of researchers have designed different types of neighborhood structures in PSO to improve its performance. It is found that PSO with small neighborhoods might perform better on complex problems while PSO with large neighborhoods would perform better on simple problems [50]. Kennedy and Mendes [51] tested PSOs with different shaped neighborhoods, such as pyramid structure, star structure, “small” structure, von Neumann and randomly generated neighborhoods. These topology structures are showed in Fig. 2-6.

CHAPTER 2 BACKGROUND AND LITERATURE SURVEY

Suganthan [52] proposed a dynamically adjusted neighborhood which is adjusted according to the distances among particles and a predefined criterion. The local version PSO is transformed into global version at the end of PSO running. And then in [53], Hu and Eberhart also proposed a dynamic neighborhood for multi-objective optimization using the PSO. The neighborhood of each particle is also dynamically adjusted according to the distances among the particles. But in [53], the distance is calculated in the objective space than the decision space as in [52]. In each generation, for each particle, m closest particles are selected to be its new neighborhood.

Some researchers modified the velocity updating equation and make a particle learn from more than two exemplars. Veeramachaneni and his group developed a new version of PSO, Fitness-Distance-Ratio based PSO (FDR-PSO), with near neighbor interactions [41]. When updating each velocity dimension, the FDR-PSO algorithm selects one other particle, **nbest**, which has highest fitness-to-distance ratio being updated, in the velocity updating equation. In [54], Mendes and Kennedy introduced a fully informed particle swarm optimization algorithm, in which all the neighbors of a particle are involved in calculating the velocity instead of using the previous best positions in the original particle swarm optimization algorithm. The influence of each particle to its neighbors is weighted based on its fitness value and the neighborhood size. Parsopoulos and Vrahatis combined the global version and local version together to construct a Unified Particle Swarm Optimizer (UPSO) [56][57]. In UPSO, the velocity of each particle is updated according to its personal best, global best and local best at the same time. And then it was extended to solve constrained problems and dynamic tracking problems in [58][59].

2.5.3 Hybrid PSO Algorithms

Some researchers tried different ways to combine the PSO with the other techniques, especially the other evolutionary computation techniques, to improve PSO's performance. Many researchers applied the mutation operator, which is always used in other evolutionary algorithms to PSO in order to increase the diversity of the population and to have the ability to escape local minima [46][60][61][63][29]. Some researchers even mutate parameters such as ϕ , c_1 and c_2 [30] and the inertia weight w [61]. Angeline applied selection operation in PSO to preserve the

CHAPTER 2 BACKGROUND AND LITERATURE SURVEY

particles with the best performance into the next generation [32]. Lovbjerg *et al.* applied crossover operation to swap information between two particles to have a large diversity and better global search ability [55].

There are some hybrid PSOs which are constructed through combining other evolutionary algorithms and PSO. Robinson *et al.* [65] applied PSO first followed by a GA in their profiled corrugated horn antenna optimization problem and it was claimed that better results were achieved. Krink and Lovbjerg [62] combined particle swarm optimization, genetic algorithm, and hill-climbing search algorithm together to construct the so called the lifecycle model. Hendtlass and M. A. Randall [66] combined ant colony optimization with PSO. Hendtlass [67] occasionally applied differential evolution on a bad particle to replace the poorly performing particle with a better one.

Some researches used different methods to maintain the diversity to prevent particles converging too fast and getting trapped into the local optima. Lovbjerg and Krink relocated the particles when they are too close to each other in [61]. In [63][29], collision-avoiding mechanisms are combined into the PSOs to prevent particles from colliding with each other and increase the diversity of the swarm. Xie *et al.* proposed a dissipative particle swarm in which the negative entropy is added to add the diversity and discourage premature convergence.

Bergh and Engelbrecht [68] proposed a Cooperative Particle Swarm Optimizer (CPSO) combining cooperative search in PSO. The solution vector is split into smaller vector partitions and multiple swarms are employed to optimize these components cooperatively. In [28], the population of particles is divided into subpopulations, which would breed within their own subpopulation or with a member of another with some probability. Hence the diversity of the population can be increased.

2.6 Applications

PSO is simple in concept. It has few parameters to adjust and is easy to implement. It has found applications in many areas. In general, all application areas which other evolutionary application techniques are good at are good applications areas for PSO too.

CHAPTER 2 BACKGROUND AND LITERATURE SURVEY

2.6.1 Binary 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 [34] calculate the velocity using the same equation as Eqn. (2-3) and a logistic function $s(v) = 1/(1 + \exp(-v))$ to determine the state of x_i^d . If a randomly generated number within $[0,1]$ is less than $s(v_i^d)$, then x_i^d is set to be 1, otherwise it is set to be 0. Cervantes *et al.* [35] introduce a repulsive force to favor particle competition in the above binary PSO and employed it in two different classification systems - Pittsburgh and the Michigan approaches. Ting *et al.* presented a hybrid particle swarm optimization scheme using binary particle swarm optimization method and real coded particle swarm optimization method to solve the unit commitment problem in [36].

Agrafiotis and Cedeno [37] treated Eqn. (2-3) as probabilities and use the roulette wheel to determine whether the new corresponding bit is 1 or 0 in the next generation. Mohan and Al-kazemi [38] proposed five binary variations of PSO namely direct approach, quantum approach, bias vector approach, and mixed search approach. Rastegar *et al.* [39] defined the concepts of trajectories and velocities the same as [34], but instead of using the sigmoid transformation, learning automata are used to determine the position of the particles. Afshinmanesh *et al.* [40] proposed a novel binary PSO method based on the theory of immunity in biology.

2.6.2 Constrained Optimization

An important issue in the constrained optimization is how to handle the constraints. Many constraint-handling techniques have been proposed in the evolutionary algorithms literatures. By combining those constraint-handling methods with PSO, different PSO variants for constrained optimization have been proposed. Parsopoulos and Vrahatis [72] employed the penalty function and convert the constrained optimization problems into a non-constrained optimization problem. Hu and Eberhart preserved feasible solutions and repair the infeasible solutions in [73]. The preservation of feasible solutions method and penalty function method are presented in [75]; Chunlin [76] used subpopulation and used MOPSO in each

CHAPTER 2 BACKGROUND AND LITERATURE SURVEY

subpopulation; [78] preferred the feasible solution to the infeasible solution and sorted the feasible solutions according to the objective function while it sorted the infeasible solutions according to the sum of constraints violation.

2.6.3 Multi-objective Optimization

Different from the single-objective optimization, there exists more than one objective in the multi-objective optimization. The goal of the multi-objective optimization is to find a set of non-dominated solutions, the Pareto front. Some researchers convert multi-objective problem to a single objective optimization problem. In [64], the so-called weighted aggregation approach was employed. All the objectives were summed to form a weighted combination. The weights can be dynamically changing during the search process. Some researchers handled the objectives one by one and optimized only one objective at a time. Some researchers kept the non-dominated solutions and moved the particles toward the found non-dominated solutions to make particles converge to the true Pareto front [81][84][85]. A more detailed review will be presented in Chapter 6.

2.6.4 Min-max Problems

PSO has been also applied to solve min-max problems or the problems which can be converted to min-max problems [79][80][81]. Laskari *et al.* [81] embedded the maximum part in calculation of the fitness values to convert the min-max problem to a minimization problem. In [79], multi-PSO strategy was used. Two PSOs were employed to solve these two optimization problems independently and the two PSOs cooperated through the fitness calculation

2.6.5 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 [97][98]. In [70], deflection, stretching and repulsion techniques were incorporated into the original particle

CHAPTER 2 BACKGROUND AND LITERATURE SURVEY

swarm optimization to avoid particles moving toward already found global minima so that the PSO can have more chances to find as many global minima as possible.

2.6.6 Dynamic Tracking

Usually the optima of the optimization problems are fixed, but sometimes the landscape of an optimization problem may change. In this situation, previously found good solutions may become bad in the future. Different PSO variants were proposed to track dynamic systems [99][100][101][102].

2.6.7 Other Applications

In addition to the above application areas, PSO has been successfully applied to solve many other problems. It has been applied to evolve weights and structure of neural networks [103][104], register 3D-to-3D biomedical image [105], play games [106], analyze human tremor [107], control reactive power and voltage [108], and many other problems.

Chapter 3

Comprehensive Learning Particle Swarm Optimizer for Single Objective Optimization

This chapter presents a variant of the particle swarm optimizers called the Comprehensive Learning Particle Swarm Optimizer (CLPSO). CLPSO is designed to solve single objective optimization problems with only bound constraints. It uses a novel 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 addition to introducing the CLPSO, an analysis of the existing benchmark functions is presented and novel composition functions are described. Experiments are conducted on four groups of benchmark functions and the results demonstrate good performance of the CLPSO in solving multimodal problems when compared with eight other recent variants of the PSO.

3.1 Comprehensive Learning Particle Swarm Optimizer

3.1.1 Comprehensive Learning Strategy

Although there are numerous variants for the PSO, premature convergence when solving multimodal problems is still the main deficiency of the PSO. In the original PSO, the following velocity updating equation is generally used:

$$V_i^d \leftarrow w * V_i^d + c_1 * rand1_i^d * (pbest_i^d - X_i^d) + c_2 * rand2_i^d * (gbest^d - X_i^d) \quad (3-1)$$

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 environment is complex with numerous local solutions. As $f(\mathbf{x}) = f([x^1, x^2, \dots, x^D])$, the fitness

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

value of a particle is possibly determined by all dimensions. A particle which has discovered the region 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 is designed.

In this new learning strategy, we use the following velocity updating equation:

$$V_i^d \leftarrow w * V_i^d + c * rand_i^d * (pbest_{f_i(d)}^d - X_i^d) \quad (3-2)$$

where $f_i = [f_i(1), f_i(2), \dots, f_i(D)]$ defines which particle's *pbest* particle i should follow. $pbest_{f_i(d)}^d$ can be the corresponding dimension of any particle's *pbest* including its own *pbest*, and the decision depends on probability P_c , referred to as the learning probability. The learning probability can take different values for different particles. For each dimension of particle i , we generate a random number with uniform distribution in the range $[0, 1]$. If this random number is larger than P_{c_i} , this dimension will learn from its own *pbest*, otherwise it will learn from another particle's *pbest*. We employ the tournament selection procedure when the particle's dimension learns from another particle's *pbest* as follows:

1. We first randomly choose two particles out of the population which excludes the particle whose velocity is updated;
2. We compare the fitness values of these 2 particles' **pbests** and select the better one. In CLPSO, we define the fitness value the larger the better, which means when solving minimization problems, we will use the negative function value as the fitness values;
3. We use the winner's *pbest* as the exemplar to learn from for that dimension. If all exemplars of a particle are its own *pbest*, we will randomly choose one dimension to learn from another particle's **pbest's** corresponding dimension. The details of choosing f_i are given in Fig. 3-1.

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

Fig. 3-1 Selection of Exemplar Dimensions for Particle i .

All these $pbest_{f_i}$ can generate 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, we allow the particle to learn from the same exemplars until the particle ceases improving for a certain number of generations called the refreshing gap m , then we re-assign f_i for the particle. We observe three main differences between the CLPSO and the original PSO:

- i. Instead of using particle's own $pbest$ and $gbest$ as the exemplars, all particles' **pbests** can potentially be used as the exemplars to guide a particle's flying direction.

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

- ii. Instead of learning from the same exemplar particle for all dimensions, each dimension of a particle in general can learn from different **pbests** for different dimensions for a few generations. In other words, each dimension of a particle may learn from the corresponding dimension of different particle's **pbest**.
- iii. Instead of learning from two exemplars (**gbest** and **pbest**) at the same time in every generation as in the original PSO Eqns. (2-1) and (2-3), each dimension of a particle learns from just one exemplar for a few generations.

3.1.2 CLPSO's Search Behavior

The above operations increase the swarm's diversity to yield improved performance when solving complex multimodal problems. In the original PSO, for a certain dimension, if the **pbest** and **gbest** are on opposite sides of the particle's current position **X**, the **pbest** and **gbest** may make the particle oscillate. However, the **gbest** is more likely to provide a larger momentum, as $|\mathbf{gbest} - \mathbf{X}|$ is likely to be larger than the $|\mathbf{pbest} - \mathbf{X}|$. Hence, the **gbest** may influence the particle to move in its direction even if it is in a local optimum region. If **pbest** and **gbest** are on the same side of the particle's current position and if it points to a local optimum, the particle will move in that direction and it may be impossible to jump out of the local optimum area once its **pbest** falls into the same local optimum region where the **gbest** is. However, in our new learning strategy, the particle can fly in other directions by learning from other particles' **pbest** when the particle's **pbest** and **gbest** fall into the same local optimum region. Hence, our new learning strategy has the ability to jump out of local optimum via the cooperative behavior of the whole swarm.

In order to compare the original PSO's and CLPSO's potential search spaces, first we omit the old velocity $w * V_i^d$ component. If we let c_1 , c_2 in the original PSO and c in CLPSO all be equal to one, the update equations of the original PSO and CLPSO reduce to the following equations:

$$\text{PSO: } V_i^d \leftarrow \text{rand}1_i^d * (\mathbf{pbest}_i^d - X_i^d) + \text{rand}2_i^d * (\mathbf{gbest}^d - X_i^d) \quad (3-3)$$

$$\text{CLPSO: } V_i^d \leftarrow \text{rand}_i^d * (\mathbf{pbest}_{f(i,d)}^d - X_i^d) \quad (3-4)$$

Let us consider the fourth particle in a swarm with five members as an example. The potential search spaces of the original PSO and the CLPSO on one dimension

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

are plotted as a line in Fig. 3-2. For the fourth particle whose position is X_4 , three different cases are illustrated in Fig. 3-2 : a) $X_4 \geq \min(pbest_j) \& X_4 \leq \max(pbest_j)$ b) $X_4 < \min(pbest_j)$; c) $X_4 > \max(pbest_j)$, $j = [1, 2, 3, 4, 5]$. In this example, $pbest_2$ is the $gbest$, $\min(pbest_j)$ is the $pbest_1$, and $\max(pbest_j)$ is the $pbest_5$.

Fig. 3-2 The CLPSO's and the Original PSO's Possible Search Regions per Variable in a Swarm with Five Members

Let the length of the potential space of the PSO and CLPSO for the d^{th} dimension of the i^{th} particle be r_{1i}^d and r_{2i}^d respectively. Extending the three cases to the d^{th} dimension of the i^{th} particle in a swarm of size ps , the potential search ranges for the i^{th} particle of PSO and CLPSO are:

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

PSO:

$$r_{1i}^d = L_{1i}^d + L_{2i}^d = |gbest^d - X_i^d| + |pbest_i^d - X_i^d| \quad (3-5)$$

CLPSO:

$$\left\{ \begin{array}{l} \text{if } X_i^d \geq \min(pbest_j^d) \& X_i^d \leq \max(pbest_j^d) \\ \quad \text{then } r_{2i}^d = L_{3i}^d + L_{4i}^d = \max(pbest_j^d) - \min(pbest_j^d) \\ \text{if } X_i^d < \min(pbest_j^d) \\ \quad \text{then } r_{2i}^d = \max(L_{3i}^d, L_{4i}^d) = \max(pbest_j^d) - X_i^d \\ \text{if } X_i^d > \max(pbest_j^d) \\ \quad \text{then } r_{2i}^d = \max(L_{3i}^d, L_{4i}^d) = X_i^d - \min(pbest_j^d) \end{array} \right.$$

$$\Rightarrow r_{2i}^d = \max(pbest_j^d, X_i^d) - \min(pbest_j^d, X_i^d)$$

$$i = 1, 2, \dots, ps, j = 1, 2, \dots, ps, d = 1, 2, \dots, D \quad (3-6)$$

Hence, the volumes of PSO's and CLPSO's potential search spaces for the i^{th} particle are $R_{1i} = \prod_i^D r_{1i}$ and $R_{2i} = \prod_i^D r_{2i}$ respectively. $\overline{R_1}$ and $\overline{R_2}$ are the mean values of the volumes of PSO's and CLPSO's potential search spaces for the whole swarm. In order to compare the potential search spaces of PSO and CLPSO, both algorithms are run 20 times on a (unimodal) sphere function and a (multimodal) Rastrigin's function defined in Section 3.2.2. $\overline{R_1}$, $\overline{R_2}$ and $\overline{R_2/R_1}$ in each iteration are recorded. Table 3-1 presents $\overline{\overline{R_2/R_1}}$'s mean value of the 20 runs. $\overline{R_1}$ and $\overline{R_2}$ and $\overline{R_2/R_1}$ versus the iterations are plot in Fig. 3-3.

Table 3-1 Mean Value of $\overline{\overline{R_2/R_1}}$ for Sphere and Rastrigin's Functions in 20 runs

	Sphere	Rastrigin's
mean($\overline{\overline{R_2/R_1}}$)	4.2059e+003	1.5971e+004

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

Fig. 3-3 Comparison of PSO and CLPSO's Potential Search Space

From Table 3-1 and Fig. 3-3, it is observed that CLPSO's updating strategy yields a larger potential search space than that of the original PSO. The multimodal Rastrigin's function's $\text{mean}(\overline{R_2}/\overline{R_1})$ is ten times larger than that of the unimodal sphere function. By increasing each particle's potential search space, the diversity is also increased. As each particle's **pbest** is possibly at a good area, the search of CLPSO is neither blind nor random. Compared to the original PSO, CLPSO searches more promising regions to find the global optimum. Experimental results also support this description.

As we have discussed in Chapter 2, there are two versions of PSOs namely the global and the local PSO. We proved here CLPSO has a larger promising regions than the global PSO, then what about local PSO? In local PSO, a particle i learns from the local best particle $lbest_i$, which performs best in the neighborhood. Since $\min(lbest_i^d) \geq \min(pbest_i^d)$ and $\max(lbest_i^d) \leq \max(pbest_i^d)$, CLPSO has a larger potential search space than local PSO also.

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

3.1.3 Learning Probability P_c

From the experiments, it is found that different P_c values yielded different results on the same problem if the same P_c value was used for all the particles in the population. On unrotated problems, smaller P_c values perform better while on the rotated problems different P_c values yield the best performance for different problems. Different P_c values yield similar results on simple unimodal problems while seriously affecting CLPSO's performance on multimodal problems. In order to address this problem in a generic manner, P_c is set as (3-7) to make sure 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. We empirically developed the following expression to set a P_{c_i} value for each particle:

$$P_{c_i} = 0.5 * (\exp(\frac{10(i-1)}{ps-1}) - 1) / (\exp(10) - 1) \tag{3-7}$$

Fig. 3-4 presents an example of P_c assigned for a population of 30. Each particle from 1 to 30 has a P_c value ranging from 0.0 to 0.5 .

Fig. 3-4 Each Particle's P_c with a Population Size of 30

3.1.4 Implementation of Search Bounds

In many practical problems, there are bounds on the variables' ranges. Suppose the search range for a problem is $[X_{min}, X_{max}]$. In order to prevent particles moving out of the search bounds, some researchers use the equation $X_i^d = \min(X_{max}^d, \max(X_{min}^d, X_i^d))$ to restrain a particle on the border. However,

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

this may convey the particle's wrong information and may lead to bad results when the global optimum is near the bounds of the ranges, but not on the bounds. In our algorithm, we use a different method to constrain the particles within the range as follows: *We calculate the fitness value of a particle and update its **pbest** and **gbest** only if the particle is in the range. Since all exemplars are within the range, the particle will eventually return to the search range.*

3.1.5 Adjusting the Refreshing Gap m

In CLPSO, except the acceleration constant c , there is another constant need to be optimized. That is the refreshing gap parameter m . In this section, six different kinds of 10-D test functions are used to investigate the impact of this parameter. They are the sphere, Rosenbrock's, Ackley's, Griewank's, Rastrigin's and rotated Rastrigin's functions as defined in Section 3.2. The CLPSO is run 20 times on each of these functions and the mean values of the final results are plotted in Fig. 3-5. As all test functions are minimization problems, the smaller the final result, the better it is. From Fig. 3-5, we can observe that m can influence the results. When m is zero, we obtained a faster convergence velocity and better results on the sphere function. For the other five test functions, better results were obtained when m is around 7. Hence, in the following experiments, the refreshing gap m is set at 7 for all test functions. The entire flowchart of the CLPSO is given in Fig. 3-6.

Fig. 3-5 CLPSO's Results on Six Test Functions with Different Refreshing Gap m

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

ps : population size Max_FES : maximum fitness evaluations
 FES : fitness evaluations counter from 1 to Max_FES
 i : particle's id counter from 1 to ps d : dimension
 X_i^d : i^{th} particle's d^{th} dimension's value w : inertia weight ($w_0 = 0.9, w_1=0.4$)
 $c = 1.49445$ m : refreshing gap
 $flag_i$: the number of generations the i^{th} particle has not improved its own **pbest**.

Fig. 3-6 Flowchart of the CLPSO Algorithm

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

3.2 Introduction to The Test Functions

3.2.1 Analysis of the Existing Test Functions:

With the purpose of testing the proposed algorithm, a set of benchmark functions need to be chosen. Various benchmark functions with various properties have been proposed by different researchers to evaluate a novel algorithm. But among their popular benchmark functions, there exist some weaknesses and these weaknesses have been made use of by some algorithms on purpose or not on purpose to make their results excellent. Some of these problems are listed below:

- i. Global optima have same parameter values for different dimensions:

Among those popular benchmark functions, most of their Global optima have same parameter values for different dimensions because of their symmetry, say the global optimum $\mathbf{o} = [a, a, a, \dots, a]$. For example, Rastrigin's function, Griewank's function's global optima are $[0, 0, 0, \dots, 0]$ and Rosenbrock's function's global optimum are $[1, 1, 1, \dots, 1]$. In this situation, if there exist some operators that copy one dimension's value to other dimensions, the global optimum will be found easily in a short time.

Take one neighborhood competition operator in [109] as an example:

$$m = (m_1, \dots, m_{i_1-1}, m_{i_1}, m_{i_1+1}, \dots, m_{i_2-1}, m_{i_2}, m_{i_2+1}, \dots, m_D) \rightarrow (m_1, \dots, m_{i_1-1}, m_{i_2}, m_{i_2+1}, \dots, m_{i_1+1}, m_{i_1}, m_{i_1+1}, \dots, m_D) = l$$

(3-8)

where m is the best solution in the population and l is newly generated solution, i_1 and i_2 are two random integers and $1 < i_1 < i_2 < D$.

In this way, if the algorithm has found global optimum for some dimension, it will be easy to copy to other dimensions, but it has no meaning when we solve the real problem. In general we do not have any apriori information about global optima for real world problem.

- ii. Global optimum on the origin:

In this case, the global optimum $\mathbf{o} = [0, 0, 0, \dots, 0]$, if we set a local range using following function:

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

$[l \cdot (1 - sRadius), l \cdot (1 + sRadius)]$ [109], where l is the search center and $sRadius$ is the local search radius, we could find that the local search range is much smaller when l is near the origin point, the global optimum's position, than when l is far from the origin point. This operator will affect algorithm's performance if the global optimum is not on the origin because of large local search range near global optimum.

iii. Global optimum lies in the center of the search range:

According to our observation, some algorithms have the potential to converge to the center of the given search range. The mean-centric crossovers are just good examples of this kind. When we randomly generate the initial population uniformly, the mean-centric method will have a trend to lead the population to the center of the search range.

iv. Global optimum on the bounds:

This problem always happen in some multi-objective optimization algorithms, some algorithms set the population running out of the search range to the near bounds [110]. In this way, if the global optimum is on the bounds, like in some popular multi-objective benchmark functions, the global optimum will be easily found. On the contrary, if there are some local optima on the bounds, it will be easy to fall into the local optima and fail to find the global optimum.

v. Local optima lying along the coordinate axes or there is no linkage among the different dimensions:

Most of the benchmark functions always have their symmetry grid structure and local optima lying along the coordinate axes. In this case, the information of the local optima could be used to search for global optima. Except this, there are some functions that are too simple and to find out conjugate directions you need nothing more than just n line searches to reach the optimum. Some co-evolutionary algorithms [68] and the one dimension mutation operator [109][114] just use these points to find out the global optimum efficiently.

By analyzing these problems, some suggestions are listed below to avoid these problems and obtain effective and valuable benchmark functions:

- i. Shift the global optima to a random position as shown in Eqn. (3-9) to make sure that the global optimum have different values for different dimensions for benchmark functions with problem 1 to 3:

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

$$F(\mathbf{x}) = f(\mathbf{x} - \mathbf{o}_{new} + \mathbf{o}_{old}) \quad (3-9)$$

where $F(\mathbf{x})$ is the new function, $f(\mathbf{x})$ is the old function, $\mathbf{o}_{old}$ is the old global optimum and $\mathbf{o}_{new}$ is the new setting global optimum which has different values for different dimensions and is not in the center of the search range.

- ii. Don't set the newly generated solutions to the near bounds when they are out of the search range for benchmark functions with problem 4. And in general, handling the bounds using this method is not recommended. According to our experiments, it's not good to set them on the bounds when there are local optima near the bounds; this operator gives a wrong hint for the whole population's search direction.
- iii. Rotate the functions which have local optima lying along the coordinate axes or have no linkage among the different dimensions. First an orthogonal matrix $\mathbf{M}$ should be generated. The original variable $\mathbf{x}$ is left multiplied by the orthogonal matrix $\mathbf{M}$ to get the new rotated variable $\mathbf{y} = \mathbf{M} * \mathbf{x}$. This variable $\mathbf{y}$ is used to calculate the fitness value f .

$$\text{If } \mathbf{M} = \begin{bmatrix} m_{11} & m_{12} & \dots & m_{1D} \\ m_{21} & m_{22} & \dots & m_{2D} \\ \dots & \dots & \dots & \dots \\ m_{D1} & m_{D2} & \dots & m_{DD} \end{bmatrix}, \quad \begin{matrix} \mathbf{x} = [x_1, x_2, \dots, x_D]^T \\ \mathbf{y} = [y_1, y_2, \dots, y_D]^T \end{matrix} \quad (3-10)$$

$$\text{then } y_i = m_{i1}x_1 + m_{i2}x_2 + \dots + m_{iD}x_D, \quad i = 1, 2, \dots, D \quad (3-11)$$

When one dimension in $\mathbf{x}$ vector is changed, all dimensions in vector $\mathbf{y}$ will be affected. Hence, the rotated function cannot be solved by just D one-dimensional searches. The orthogonal rotation matrix does not affect the shape of the functions. In this thesis, Salomon's method [115] is employed to generate the orthogonal matrix.

- iv. Except for the functions that need to be shifted, when we test a novel algorithm, functions having different properties should be included. For example, continuous functions, non-continuous functions, functions with high condition numbers, functions with global optimum on the bounds, functions with global optimum not on the bounds, functions with local optima with large basins of attractions and the global optima with narrow basins of attraction and functions with no clear structure in the fitness landscape are all useful types which can be

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

used to test the different properties of the novel algorithm. Some functions with some of these properties could be found in [116].

3.2.2 Novel Composition Test Functions

Based on these considerations, a group of novel composition test functions, which could satisfy most of the above discussed desired properties, has been constructed. The main idea is to compose some simple benchmark functions to construct a more difficult function, which has a randomly located global optimum and several randomly located deep local optima. Gaussian function is used to combine these benchmark functions and blur the function's structures.

$F(\mathbf{x})$: new composition function

$f_i(\mathbf{x})$: i^{th} basic function used to construct the composition function

n : number of basic functions

D : dimensions

$\mathbf{M}_i$: linear transformation matrix for each $f_i(\mathbf{x})$

$\mathbf{o}_i$: new shifted optimum position for each $f_i(\mathbf{x})$

The corresponding composition function will be:

$$F(\mathbf{x}) = \sum_{i=1}^n \{w_i * [f_i'((\mathbf{x} - \mathbf{o}_i) / \lambda_i * \mathbf{M}_i) + bias_i]\} + f_{bias} \quad (3-12)$$

w_i : weight value for each $f_i(\mathbf{x})$, calculated as below:

$$w_i = \exp\left(-\frac{\sum_{k=1}^D (x_k - o_{ik})^2}{2D\sigma_i^2}\right) \quad (3-13)$$

$$w_i = \begin{cases} w_i & w_i = \max(w_i) \\ w_i * (1 - \max(w_i)^{10}) & w_i \neq \max(w_i) \end{cases} \quad (3-14)$$

then normalize the weight $w_i = w_i / \sum_{i=1}^n w_i$

σ_i is used to control each $f_i(\mathbf{x})$'s coverage range, a small σ_i gives a narrow range for that $f_i(\mathbf{x})$. λ_i is used to stretch or compress the function, $\lambda_i > 1$ means stretch, $\lambda_i < 1$ means compress.

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

$\mathbf{o}_i$ defines the global and local optima's position, $bias_i$ defines which optimum is global optimum. Using $\mathbf{o}_i$, $bias_i$, a global optimum can be placed anywhere.

If $f_i(\mathbf{x})$ are different functions, and since they have different properties and heights, the biggest function value $f_{\max i}$ for 10 functions $f_i(\mathbf{x})$ is estimated and each basic function is normalized to similar heights as below to get a better mixture:

$f_i'(\mathbf{x}) = C * f_i(\mathbf{x}) / |f_{\max i}|$, C is a predefined constant.

$|f_{\max i}|$ is estimated using $|f_{\max i}| = f_i((\mathbf{x}' / \lambda_i) * \mathbf{M}_i)$, $\mathbf{x}' = [5, 5, \dots, 5]$.

The pseudo code of the composition function is presented in Fig. 3-7. An example of the composition function with five sphere functions as basic functions is plotted in Fig. 3-8.

Define $f_1, f_2, \dots, f_n$, $n = 10$, σ, λ , bias, C , $\mathbf{y} = [5, 5, \dots, 5]$.

Load data file $\mathbf{o}$ and rotated linear transformation matrix $\mathbf{M}_1, \mathbf{M}_2, \dots, \mathbf{M}_n$

For $i = 1 : n$

$$w_i = \exp\left(-\frac{\sum_{k=1}^D (x_k - o_{ik})^2}{2D\sigma_i^2}\right), \quad \tilde{fit}_i = f_i(((\mathbf{x} - \mathbf{o}_i) / \lambda_i) * \mathbf{M}_i)$$

$$f_{\max i} = f_i((\mathbf{y} / \lambda_i) * \mathbf{M}_i), \quad fit_i = C * \tilde{fit}_i / f_{\max i}$$

EndFor

$$SW = \sum_{i=1}^n w_i, \quad MaxW = \max(w_i)$$

For $i = 1 : n$

$$w_i = \begin{cases} w_i & \text{if } w_i = MaxW \\ w_i * (1 - MaxW.^{10}) & \text{if } w_i \neq MaxW \end{cases}$$

$$w_i = w_i / SW$$

EndFor

$$F(\mathbf{x}) = \sum_{i=1}^n \{w_i * [fit_i + bias_i]\}$$

$$F(\mathbf{x}) = F(\mathbf{x}) + f_{bias}$$

Fig. 3-7 Pseudo Code of Composition Function

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

Fig. 3-8 Construct a Two-Dimensional Composition Function Using Five Sphere Functions

3.2.3 Benchmark Function Set

As we wish to test the CLPSO on diverse test functions and our main objective is to improve PSO’s performance on multimodal problems, we choose two unimodal functions and fourteen multimodal benchmark functions [111][112][113]. All functions are tested on 10 and 30 dimensions. According to their properties, these functions are divided into four groups: *unimodal problems*, *unrotated multimodal problems*, *rotated multimodal problems* and *composition problems*. The properties and the formulas of these functions are presented below:

Group A: Unimodal and Simple Multi-modal Problems:

1) *Sphere function*

$$f_1(x) = \sum_{i=1}^D x_i^2 \tag{3-15}$$

2) *Rosenbrock’s function*

$$f_2(x) = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2) \tag{3-16}$$

The first problem is the sphere function and is easy to be solved. The second problem is the Rosenbrock’s 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’s function.

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

Group B: Unrotated Multimodal Problems:3) *Ackley's function*

$$f_3(x) = -20 \exp(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2}) - \exp(\frac{1}{D} \sum_{i=1}^D \cos(2\pi x_i)) + 20 + e \quad (3-17)$$

4) *Griewank's function*

$$f_4(x) = \sum_{i=1}^D \frac{x_i^2}{4000} - \prod_{i=1}^D \cos(\frac{x_i}{\sqrt{i}}) + 1 \quad (3-18)$$

5) *Weierstrass function*

$$f_5(x) = \sum_{i=1}^D (\sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k (x_i + 0.5))]) - D \sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k \cdot 0.5)]$$

$$a = 0.5, b = 3, k_{\max} = 20 \quad (3-19)$$

6) *Rastrigin's function*

$$f_6(x) = \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i) + 10) \quad (3-20)$$

7) *Non-continuous Rastrigin's function*

$$f_7(x) = \sum_{i=1}^D (y_i^2 - 10 \cos(2\pi y_i) + 10),$$

$$y_i = \begin{cases} x_i & |x_i| < 1/2 \\ \text{round}(2x_i)/2 & |x_i| \geq 1/2 \end{cases} \text{ for } i = 1, 2, \dots, D \quad (3-21)$$

8) *Schwefel's function*

$$f_8(x) = 418.9829 \times D - \sum_{i=1}^D x_i \sin(|x_i|^{1/2}) \quad (3-22)$$

In this group, there are six multimodal test functions. 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 thereby making it difficult to reach the global optimum. An interesting phenomenon of Griewank's function is that it is more difficult for lower dimensions than higher dimensions [118]. 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

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

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 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 six rotated multimodal problems.

9) Rotated Ackley's function

$$f_9(x) = -20 \exp(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D y_i^2}) - \exp(\frac{1}{D} \sum_{i=1}^D \cos(2\pi y_i)) + 20 + e, \mathbf{y} = \mathbf{M} * \mathbf{x} \quad (3-23)$$

10) Rotated Griewanks's function

$$f_{10}(x) = \sum_{i=1}^D \frac{y_i^2}{4000} - \prod_{i=1}^D \cos(\frac{y_i}{\sqrt{i}}) + 1, \mathbf{y} = \mathbf{M} * \mathbf{x} \quad (3-24)$$

11) Rotated Weierstrass function

$$f_{11}(x) = \sum_{i=1}^D \left(\sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k (y_i + 0.5))] \right) - D \sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k \cdot 0.5)],$$

$$a=0.5, b=3, k_{\max}=20, \mathbf{y} = \mathbf{M} * \mathbf{x} \quad (3-25)$$

12) Rotated Rastrigin's function

$$f_{12}(x) = \sum_{i=1}^D (y_i^2 - 10 \cos(2\pi y_i) + 10), \mathbf{y} = \mathbf{M} * \mathbf{x} \quad (3-26)$$

13) Rotated Non-continuous Rastrigin's function

$$f_{13}(x) = \sum_{i=1}^D (z_i^2 - 10 \cos(2\pi z_i) + 10) \quad (3-27)$$

$$z_i = \begin{cases} y_i & |y_i| < 1/2 \\ \text{round}(2y_i)/2 & |y_i| \geq 1/2 \end{cases} \text{ for } i=1, 2, \dots, D, \mathbf{y} = \mathbf{M} * \mathbf{x}$$

14) Rotated Schwefel's function

$$f_{14}(x) = 418.9829 \times D - \sum_{i=1}^D z_i \quad (3-28)$$

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

$$z_i = \begin{cases} y_i \sin(|y_i|^{1/2}) & \text{if } |y_i| \leq 500 \\ 0.001(|y_i| - 500)^2 & \text{if } |y_i| > 500 \end{cases}, \text{ for } i = 1, 2, \dots, D,$$

$$y = y' + 420.96, y' = \mathbf{M} * (\mathbf{x} - 420.96)$$

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

Group D: Composition Problems

Parameter settings for the following two composition functions:

Number of basic functions $n=10$.

$D = 10, 30$

bias = $[0, 100, 200, 300, 400, 500, 600, 700, 800, 900]$.

Hence, the first function $f_1(\mathbf{x})$ is always the function with the global optimum.

$C=2000$

$\mathbf{M}_1, \mathbf{M}_2, \dots, \mathbf{M}_n$ are $D \times D$ orthogonal rotation matrixes obtained by using Salmon's method [151].

15) Composition function 1 (CF1) in [119]

The f_{15} (CF1) is composed using ten sphere functions. The global optimum is easy to find once the global basin is found.

16) Composition function 5 (CF5) in [119]:

The f_{16} (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 landscape maps of these two composition functions are illustrated in Fig. 3-9.

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

(a) *Composition Function 1 (CF1)*

(b) *Composition function 5 (CF5)*

Fig. 3-9 *The Landscape Maps of Group D Problems*

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

The global optimum $\mathbf{x}^*$, the corresponding fitness value $f(\mathbf{x}^*)$, the search ranges $[\mathbf{X}_{\min}, \mathbf{X}_{\max}]$ and the initialization range of each function are given in Table 3-2. Biased initializations are used for the functions whose global optimum is at the centre of the search range.

Table 3-2 *Global Optimum, Search Ranges and Initialization Ranges of the Test Functions*

f	$\mathbf{x}^*$	$f(\mathbf{x}^*)$	Search Range	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 range	0	$[-5, 5]^D$	$[-5, 5]^D$
f_{16}	Predefined rand number distributed in the search range	0	$[-5, 5]^D$	$[-5, 5]^D$

3.3 Experimental Results and Discussions

3.3.1 Parameters Settings for the Involved PSO Algorithms

Experiments were conducted to compare nine PSO algorithms including the proposed CLPSO algorithm on the 16 test problems with 10 dimensions and 30 dimensions. The algorithms and parameters settings are listed below:

- PSO with Inertia Weight (PSO-w) [25]
- PSO with Constriction Factor (PSO-cf) [43]

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

- Local Version of PSO with Inertia Weight (PSO-w-local)
- Local Version of PSO with Constriction Factor (PSO-cf-local) [51]
- Unified Particle Swarm Optimization (UPSO) [56]
- Fully Informed Particle Swarm (FIPS) [54]
- Fitness-Distance-Ratio Based Particle Swarm Optimization (FDR-PSO) [41]
- Cooperative Particle Swarm Optimization (CPSO-H) [68]
- Comprehensive Learning Particle Swarm Optimizer (CLPSO)

Among these PSO local versions, PSO_w_local and PSO_cf_local were chosen as these versions yielded the best results [43] with von Neumann neighbourhoods where neighbours above, below, and one each side on a two-dimensional lattice were connected. Fully informed particle swarm (FIPS) with U-Ring topology that achieved the highest success rate [54] is chosen. When solving the 10-*D* problems, the population size is set at 10 and the maximum fitness evaluations (FEs) is set at 30,000. When solving the 30-*D* problems the population size is set at 40 and the maximum fitness evaluation (FEs) is set at 200,000. All experiments were run 30 times. The mean values and standard deviation of the results are presented.

When solving real world problems, usually the fitness calculation accounts for the most time as the PSO is highly computation efficient. Hence, the algorithm-related computation times of these algorithms are not compared in this paper. Further, the main difference between the CLPSO and the original PSO is the modified velocity updating equation, which has been made simpler in the CPSO, the complexity of the new algorithm is similar to the original PSO. In the experiments, a serial implementation is used, while it is easy to be modified to a parallel implementation. With a parallel form, the performance is likely to be not affected much while computational efficiency improves.

3.3.2 Experimental Results and Discussions

i. Results for the 10-*D* Problems

Table 3-3 presents the means and variances of the 30 runs of the nine algorithms on the sixteen test functions with *D* = 10. The best results out of the nine algorithms are shown in bold. Statistical test was performed to determine whether the results obtained by CLPSO are statistically different from the results generated by other algorithms. The nonparametric Wilcoxon rank sum tests [117] was done between

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

the CLPSO’s result and the best result achieved by the other eight PSO versions for each problem because of smaller sample sizes and therefore limited information on distribution. The h values presented in the last row of Table 3-3 and Table 3-4 are the results of the nonparametric Wilcoxon rank sum tests. An h value of 1 indicates that the performances of the two algorithms are statistically different with 95% certainty, whereas h value of 0 implies that the performances are not statistically different. Fig. 3-10 presents the convergence characteristics in terms of the best fitness value of the median run of each algorithm for each test function.

Table 3-3 Results for 10-D Problems

Alg.		PSO-w	PSO-cf	PSO-w-local	PSO-cf-local	UPS0	FDR	FIPS	CPSO-H	CLPSO	h
Func.	Group A	7.96e-51 ± 3.56e-50	9.84e-105 ± 4.21e-104	2.13e-35 ± 6.17e-35	1.37e-79 ± 5.60e-79	9.84e-118 ± 3.56e-117	2.21e-90 ± 9.88e-90	3.15e-30 ± 4.56e-30	4.98e-45 ± 1.00e-44	5.15e-29 ± 2.16e-28	1
	2	3.08e+0 ± 7.69e-1	6.98e-1 ± 1.46e+0	3.92e+0 ± 1.19e+0	8.60e-1 ± 1.56e+0	1.40e+0 ± 1.88e+0	8.67e-1 ± 1.63e+0	2.78e+0 ± 2.26e-1	1.53e+0 ± 1.70e+0	2.46e+0 ± 1.70e+0	1
Group B	3	1.58e-14 ± 1.60e-14	9.18e-1 ± 1.01e+0	6.04e-15 ± 1.67e-15	5.78e-2 ± 2.58e-1	1.33e+0 ± 1.48e+0	3.18e-14 ± 6.40e-14	3.75e-15 ± 2.13e-14	1.49e-14 ± 6.97e-15	4.32e-14 ± 2.55e-14	1
	4	9.69e-2 ± 5.01e-2	1.19e-1 ± 7.11e-2	7.80e-2 ± 3.79e-2	2.80e-2 ± 6.34e-2	1.04e-1 ± 7.10e-2	9.24e-2 ± 5.61e-2	1.31e-1 ± 9.32e-2	4.07e-2 ± 2.80e-2	4.56e-3 ± 4.81e-3	1
	5	2.28e-3 ± 7.04e-3	6.69e-1 ± 7.17e-1	1.41e-6 ± 6.31e-6	7.85e-2 ± 5.16e-2	1.14e+0 ± 1.17e+0	3.01e-3 ± 7.20e-3	2.02e-3 ± 6.40e-3	1.07e-15 ± 1.67e-15	0 ± 0	1
	6	5.82e+0 ± 2.96e+0	1.25e+1 ± 5.17e+0	3.88e+0 ± 2.30e+0	9.05e+0 ± 3.48e+0	1.17e+1 ± 6.11e+0	7.51e+0 ± 3.05e+0	2.12e+0 ± 1.33e+0	0 ± 0	0 ± 0	0
	7	4.05e+0 ± 2.58e+0	1.20e+1 ± 4.99e+0	4.77e+0 ± 2.84e+0	5.95e+0 ± 2.60e+0	5.85e+0 ± 3.15e+0	3.35e+0 ± 2.01e+0	4.35e+0 ± 2.80e+0	2.00e-1 ± 4.10e-1	0 ± 0	1
	8	3.20e+2 ± 1.85e+2	9.87e+2 ± 2.76e+2	3.26e+2 ± 1.32e+2	8.78e+2 ± 2.93e+2	1.08e+3 ± 2.68e+2	8.51e+2 ± 2.76e+2	7.10e+1 ± 1.50e+2	2.13e+2 ± 1.41e+2	0 ± 0	1
	9	2.80e-1 ± 5.86e-1	1.19e+0 ± 1.13e+0	6.39e-15 ± 3.18e-15	2.56e-1 ± 5.33e-1	1.00e+0 ± 9.27e-1	1.40e-1 ± 4.38e-1	2.25e-15 ± 1.54e-15	1.36e+0 ± 8.85e-1	3.56e-5 ± 1.57e-4	1
	10	1.64e-1 ± 9.40e-2	1.38e-1 ± 1.07e-1	8.04e-2 ± 4.46e-2	7.90e-2 ± 5.55e-2	7.76e-2 ± 6.40e-2	1.44e-1 ± 7.84e-2	1.70e-1 ± 1.26e-1	1.20e-1 ± 8.07e-2	4.50e-2 ± 3.08e-2	1
Group C	11	6.66e-1 ± 7.12e-1	2.17e+0 ± 1.30e+0	2.14e-1 ± 3.65e-1	1.20e+0 ± 1.22e+0	2.61e+0 ± 9.48e-1	3.34e-1 ± 3.90e-1	5.93e-14 ± 1.86e-13	4.35e+0 ± 1.35e+0	3.72e-10 ± 4.40e-10	1
	12	9.90e+0 ± 3.76e+0	1.44e+1 ± 6.04e+0	9.25e+0 ± 2.74e+0	1.35e+1 ± 6.81e+0	1.52e+1 ± 5.25e+0	9.25e+0 ± 2.50e+0	1.20e+1 ± 6.22e+0	2.67e+1 ± 1.06e+1	5.97e+0 ± 2.88e+0	1
	13	1.02e+1 ± 3.58e+0	1.53e+1 ± 6.38e+0	1.09e+1 ± 4.08e+0	1.07e+1 ± 2.81e+0	1.47e+1 ± 6.53e+0	1.07e+1 ± 3.86e+0	8.84e+0 ± 3.27e+0	1.90e+1 ± 9.05e+0	5.44e+0 ± 1.39e+0	1
	14	5.69e+2 ± 2.16e+2	1.19e+3 ± 4.23e+2	4.72e+2 ± 3.07e+2	9.09e+2 ± 3.25e+2	1.27e+3 ± 2.29e+2	1.07e+3 ± 2.23e+2	2.89e+2 ± 2.00e+2	9.67e+2 ± 3.67e+2	1.14e+2 ± 1.28e+2	1
	15	1.20e+2 ± 8.94e+1	1.60e+2 ± 1.64e+2	4.00e+1 ± 5.98e+1	9.00e+1 ± 8.52e+1	8.00e+1 ± 8.34e+1	1.00e+2 ± 9.73e+1	6.00e+1 ± 5.16e+1	1.65e+2 ± 1.42e+2	1.64e+1 ± 3.63e+1	1
	16	1.38e+2 ± 1.80e+2	2.31e+2 ± 1.93e+2	1.53e+2 ± 1.53e+2	1.34e+2 ± 1.71e+2	1.79e+2 ± 1.56e+2	1.53e+2 ± 2.01e+2	4.21e+1 ± 6.37e+1	2.46e+2 ± 2.18e+2	1.98e+1 ± 2.93e+1	1

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

(a) Sphere function

(b) Rosenbrock's function

(c) Ackley's function

(d) Griewank's function

(e) Weierstrass function

(f) Rastrigin's function

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

(g) Non-Continuous Rastrigin's function

(h) Schwefel's function

(i) Rotated Ackley's function

(j) Rotated Griewanks's function

(k) Rotated Weierstrass function

(l) Rotated Rastrigin's function

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

(m) Rotated Non-Continuous
Rastrigin's function

(n) Rotated Schwefel's function

(o) Composition function 1 (CF1)

(p) Composition function 5 (CF5)

Fig. 3-10 The Median Convergence Characteristics of 10-D Test Functions

From the results, we observe that for the Group A unimodal problems, since CLPSO has a large potential search space, it could not converge as fast as the original PSO. CLPSO achieved better results on all three multimodal groups than the original PSO. CLPSO surpasses all other algorithms on functions 4, 5, 7, 8, 10, 12, 13, 14, 15 and 16, and especially significantly improves the results on functions 7 and 8. According to the results of *t*-tests, these results are different from the second best results. The CLPSO achieved the same best result as the CPSO-H on function 6 and they both are much better than the other variants on this problem. The fully informed particle swarm (FIPS) also performs well on multi-modal problems. The FIPS performed better than the CLPSO on functions 3, 9 and 11. However, the CLPSO performs better on more complex problems when the other algorithms miss the global optimum basin. The Schwefel's function is a good example, as it traps all other algorithms in local optima. The CLPSO successfully avoids falling into the

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

deep local optimum which is far from the global optimum. On the two composition functions with randomly distributed local and global optima, CLPSO performs the best.

Comparing the results and the convergence graphs, among these nine PSO algorithms, FDR-PSO has good local search ability and converges fast. PSO with inertia weight (PSO-w) and PSO with constriction factor (PSO-cf) are two global versions where the whole population is the neighbourhood. PSO with constriction factor converges faster than the PSO with inertia weight. But, PSO with inertia weight performs better on multimodal problems. UPSO combines global PSO and local PSO together to yield a balanced performance between the global and the local versions. PSO with inertia weight (PSO-w-local), PSO with constriction factor (PSO-cf-local) and the fully informed particle swarm (FIPS) with a U-Ring topology are all local versions. They all perform better on multimodal problems than the global versions. Among the three, the FIPS yields a comparatively better performance. CPSO-H presents good performance on some unrotated multimodal problems and converges faster when compared to CLPSO. However, its performance is seriously affected after rotation. Although CLPSO's performance is also affected by the rotation, it still performs the best on four rotated problems. It can be observed that all PSO variants failed on the rotated Schwefel's function, as it becomes much harder to solve after applying rotation.

ii. Results for the 30-D Problems

The experiments conducted on 10-D problems are repeated on the 30-D problems and the results are presented in Table 3-4. As the convergence graphs are similar to the 10-D problems, they are not presented. From the results in Table 3-4, we can observe that the algorithms achieved similar ranking as in the 10-D problems. CLPSO surpasses all other algorithms on functions 3, 4, 7, 8, 10, 12, 13, 14, 15 and 16, and especially significantly improves the results on functions 7 and 8. All 30-D functions become more difficult than their 10-D counterparts and the results are not as good as 10-D cases, although we increased the maximum number of FEs from 30,000 to 200,000. Better results were achieved on Griewank's function, since this problem is known to become easier as the number of dimensions increases [118]. The results of Composition functions are not affected much since we use the same number of sub-functions with the same fixed local optima values [119].

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

Table 3-4 Results for 30-D Problems

Func.	Alg.	PSO-w	PSO-cf	PSO-w-local	PSO-cf-local	UPSO	FDR	FIPS	CPSO-H	CLPSO	h
Group A	1	9.78e-30 ± 2.50e-29	5.88e-100 ± 5.40e-100	5.35e-100 ± 4.41e-13	7.70e-54 ± 1.59e-53	4.17e-87 ± 3.15e-87	4.88e-102 ± 1.53e-101	2.69e-12 ± 6.84e-13	1.16e-113 ± 2.92e-113	4.46e-14 ± 1.73e-14	1
	2	2.93e+1 ± 2.51e+1	1.11e+1 ± 1.81e+0	2.39e+1 ± 3.07e+0	1.71e+1 ± 9.16e-1	1.51e+1 ± 8.14e-1	5.39e+0 ± 1.76e+0	2.45e+1 ± 2.19e-1	7.08e+0 ± 8.01e+0	2.10e+1 ± 2.98e+0	1
Group B	3	3.94e-14 ± 1.12e+0	1.12e+0 ± 8.65e-1	9.10e-8 ± 8.11e-8	5.33e-15 ± 1.87e-15	1.22e-15 ± 3.16e-15	2.84e-14 ± 4.10e-15	4.81e-7 ± 9.17e-8	4.93e-14 ± 1.10e-14	0 ± 0	1
	4	8.13e-3 ± 7.16e-3	2.06e-2 ± 1.90e-2	5.91e-3 ± 6.69e-3	5.91e-3 ± 8.70e-3	1.66e-3 ± 3.07e-3	1.01e-2 ± 1.23e-2	1.16e-6 ± 1.87e-6	3.63e-2 ± 3.60e-2	3.14e-10 ± 4.64e-10	1
	5	1.30e-4 ± 3.30e-4	4.10e+0 ± 2.20e+0	4.94e-3 ± 1.40e-2	1.16e-1 ± 2.79e-1	9.60e+0 ± 3.78e+0	7.49e-3 ± 1.14e-2	1.54e-1 ± 1.48e-1	7.82e-15 ± 8.50e-15	3.45e-7 ± 1.94e-7	1
	6	2.90e+1 ± 7.70e+0	5.62e+1 ± 9.76e+0	2.72e+1 ± 7.58e+0	4.53e+1 ± 1.17e+1	6.59e+1 ± 1.22e+1	2.84e+1 ± 8.71e+0	7.30e+1 ± 1.24e+1	0 ± 0	4.85e-10 ± 3.63e-10	1
	7	2.97e+1 ± 1.39e+1	2.85e+1 ± 1.14e+1	2.08e+1 ± 4.94e+0	1.54e+1 ± 1.67e+1	6.34e+1 ± 1.24e+1	1.44e+1 ± 6.28e+0	6.08e+1 ± 8.35e+0	1.00e-1 ± 3.16e-1	4.36e-10 ± 2.44e-10	1
	8	1.10e+3 ± 2.56e+2	3.78e+3 ± 6.02e+2	1.53e+3 ± 3.00e+2	3.78e+3 ± 5.37e+2	4.84e+3 ± 4.76e+2	3.61e+3 ± 3.06e+2	2.05e+3 ± 9.58e+2	1.08e+3 ± 2.59e+2	1.27e-12 ± 8.79e-13	1
	9	1.71e+0 ± 4.38e-1	1.66e+0 ± 1.10e+0	5.70e-1 ± 7.60e-1	1.78e-1 ± 5.62e-1	2.94e-1 ± 6.71e-1	3.59e-1 ± 5.93e-1	5.23e-7 ± 1.42e-7	2.10e+0 ± 3.84e-1	3.43e-4 ± 1.91e-4	1
	10	1.77e-2 ± 1.53e-2	8.62e-3 ± 8.86e-3	1.35e-2 ± 1.12e-2	1.30e-2 ± 1.06e-2	1.48e-3 ± 3.12e-3	9.60e-3 ± 1.24e-2	6.92e-4 ± 2.18e-3	5.54e-2 ± 3.97e-2	7.04e-10 ± 1.25e-11	1
Group C	11	7.00e+0 ± 1.98e+0	8.48e+0 ± 2.54e+0	5.96e+0 ± 2.09e+0	5.95e+0 ± 2.95e+0	1.85e+1 ± 3.37e+0	2.50e+0 ± 1.46e+0	9.52e-2 ± 9.53e-2	1.43e+1 ± 3.53e+0	3.07e+0 ± 1.61e+0	1
	12	6.87e+1 ± 2.05e+1	7.13e+1 ± 1.66e+1	4.10e+1 ± 7.93e+0	4.66e+1 ± 1.05e+1	7.07e+1 ± 1.70e+1	4.44e+1 ± 1.37e+1	7.41e+1 ± 2.79e+1	1.01e+2 ± 2.21e+1	3.46e+1 ± 4.59e+0	1
	13	6.32e+1 ± 1.79e+1	7.88e+1 ± 1.88e+1	5.67e+1 ± 1.36e+1	4.93e+1 ± 1.11e+1	7.74e+1 ± 1.40e+1	4.36e+1± 8.96e+0	7.58e+1 ± 1.92e+1	8.80e+1± 2.59e+1	3.77e+1 ± 5.56e+0	1
	14	2.67e+3 ± 7.03e+2	3.57e+3 ± 9.08e+2	2.60e+3 ± 5.11e+2	3.89e+3 ± 9.42e+2	5.60e+3 ± 6.50e+2	3.78e+3 ± 7.59e+2	2.60e+3 ± 8.49e+2	3.64e+3 ± 7.41e+2	1.70e+3 ± 1.86e+2	1
	15	1.00e+2 ± 1.33e+2	7.00e+1 ± 1.33e+2	2.00e+1 ± 3.16e+1	2.00e+1 ± 6.32e+1	1.00e+1 ± 2.51e+1	1.00e+1 ± 3.16e+1	1.31e-2 ± 2.92e-2	1.30e+2 ± 1.64e+2	7.50e-5 ± 1.85e-4	1
	16	2.20e+1 ± 3.34e+1	1.03e+2 ± 1.86e+2	1.08e+1 ± 5.79e+0	6.06e+1 ± 1.24e+2	3.23e+1 ± 3.65e+1	1.00e+1 ± 7.52e+0	1.04e+1 ± 4.63e+0	7.83e+1 ± 1.60e+2	7.86e+0 ± 3.64e+0	1

iii. Discussion

By analyzing the results of the CLPSO on 10-D and 30-D problems, one may conclude that the CLPSO does not perform the best for unimodal and simple multi-modal problems in Group A. According to the “No Free Lunch” theorem [120], “any elevated performance over one class of problems is offset by performance over another class”. There is a cost for better performance on multimodal problems and the cost is the slow convergence on unimodal problems. Therefore, we may not expect the best performance on all classes of problems; as the proposed CLPSO focuses on improving the PSO’s performance on multimodal problems.

The CLPSO achieves the best results on most complex multimodal problems in Groups B to D, especially on Group B - unrotated multimodal problems. This implies that the CLPSO is more effective in solving problems with less linkage. This property is due to the PSO’s dimension-wise updating rule as well as CLPSO’s

CHAPTER 3 COMPREHENSIVE LEARNING PARTICLE SWARM OPTIMIZER

learning of different dimensions from different exemplars. On more complex asymmetrical landscapes in Group D, CLPSO performs better when compared with the other algorithms. With the new updating rule, different dimensions may learn from different exemplars. Due to this, the CLPSO explores a larger search space than the original PSO. The larger search space is not achieved randomly. Instead, it is based on the historical search experience. Because of this, the CLPSO performs comparably to or better than many PSO variants on most of the multimodal problems experimented in this section.

3.4 Conclusion

This chapter presents a Comprehensive Learning PSO (CLPSO) employing a novel learning strategy where other particles' previous best positions are exemplars to be learned from by any particle and each dimension of a particle can potentially learn from a different exemplar. The new strategy makes the particles have more exemplars to learn from and a larger potential space to fly. From the analysis and experiments, we observe that this learning strategy enables the CLPSO to make use of the information in swarm more effectively to generate better quality solutions frequently when compared to eight other PSO variants. In the experiments part, by analyzing the properties of the existing test functions, composition test function which is constructed using several basic functions and can be conveniently controlled by the users is proposed. Based on the results of the nine algorithms on the sixteen chosen test problems belonging to four classes, we can conclude that CLPSO significantly improves the PSO's performance and gives the best performance compared to eight other PSO variants on most multimodal problems irrespective of whether they are unrotated or rotated

Although the CLPSO is not the best choice for solving unimodal problems, when solving real world problems, we do not frequently know the shape of the fitness landscape. Hence, it is advisable to use an algorithm that performs well on multimodal problems since such an algorithm can also solve unimodal problems. Another attractive property of the CLPSO is that it does not introduce further complex operations to the original PSO. The only difference from the original PSO is the velocity update equation. The CLPSO is also simple and easy to implement like the original PSO.

Chapter 4

Dynamic Multi-Swarm Particle Swarm Optimizer for Single Objective Optimization

As it was discussed in Chapter 2, the local version of particle swarm optimizer has better global search ability. Instead of learning from the global best, particles learn from their neighborhood best in the local version of PSO, the swarm has a slower convergence speed and has less chance to be trapped in local optima. Different topologies have been proposed and tested by the researchers. In this chapter, a novel random topology structure and an adaptive self-learning strategy are introduced and the dynamic multi-swarm particle swarm optimizer based on the new topology structure is described. After combining with a local search method, dynamic multi-swarm particle swarm optimizer with local search (DMS-L-PSO) is tested using the standard benchmark functions proposed in CEC'05. The comparison with several excellent evolutionary algorithms is also presented.

4.1 Dynamic Multi-Swarm Particle Swarm Optimizer (DMS-PSO)

4.1.1 Periodically Changed Neighborhood Structure

The dynamic multi-swarm particle swarm optimizer (DMS-PSO) is constructed based on the local version of PSO and a periodically changed neighborhood structure is used. This new neighborhood structure has two important characters:

i. Small Sized Swarms

In contrast with other evolutionary algorithms that prefer larger population, PSO needs a comparatively smaller population size. Especially for simple problems, a population with three to five particles can achieve satisfactory results. PSO with

CHAPTER 4 DYNAMIC MULTI-SWARM PARTICLE SWARM OPTIMIZER

small neighborhoods performs better on complex problems. Kennedy [50] claimed that PSO with small neighborhoods might perform better on complex problems while PSO with large neighborhood would perform better for simple problems. Hence, in the new version, small neighborhoods are used. In order to slow down the population’s convergence velocity and increase diversity, we divide the population into small sized swarms in the DMS-PSO. Each swarm uses its own members to search for better area in the search space.

ii. Random Regrouping Schedule

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 convergence property. In this case, if we keep the neighborhood structures unchanged, then there will be no information exchange among the swarms, and it will be a co-evolutionary PSO with these swarms searching in parallel. In order to avoid this situation, a randomized regrouping schedule is introduced. 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 good information obtained by each swarm is exchanged among the swarms and the diversity of the population is increased simultaneously. The new neighborhood structure has more freedom when compared with the classical neighborhood structure. It is not surprising that it performs better on complex multimodal problems.

Fig. 4-1 DMS-PSO's Search

CHAPTER 4 DYNAMIC MULTI-SWARM PARTICLE SWARM OPTIMIZER

In DMS-PSO, we use three swarms with three particles in each swarm to show the regrouping schedule. 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 a nearby local optimum. Then the whole population is regrouped into new sub-swarms. The new swarms begin their search. This process is continued until a stop criterion is satisfied. With the random regrouping schedule, particles from different sub-swarms are grouped in a new configuration so that each small swarm search space is enlarged and better solutions are possible to be found by the new small swarms.

The neighborhood topology structure for DMS-PSO is different from the existing topology structures. On the one hand, unlike some fixed topology structures, its neighborhood structure is periodically changed. On the other hand, unlike the dynamic neighborhood used in [53] and Fitness-Distance-Ratio based PSO in [41], no distance guide is involved in the DMS-PSO. This novel structure increases the freedom of the search, and makes the particles have higher chance to escape from the local optima.

This periodically changed neighborhood structure can also be grouped into multi-swarm PSOs since it employs more than one sub-swarm to do the search work. This advantage can be made use of when it is used to solve optimization problems and multi-objective optimization problem. The details will be presented in Chapter 5 and 6.

4.1.2 Adaptive Self-Learning Strategy

In the comprehensive learning particle swarm optimizer, each dimension of a particle can learn from its own **pbest** and the **pbest** of other particles. It has been shown that it can improve the performance of the algorithm when we assign some dimensions to learn from its own best positions and some dimensions to learn from other exemplars. Thus, a similar self-learning strategy is also introduced in dynamic multi-swarm particle swarm optimizer.

This self-learning strategy is similar to the crossover phase in the differential evolution (DE) algorithm [121]. In DE, after the mutation phase, the “binomial” crossover operation is applied to each pair of the generated mutant vector V_i and the corresponding target vector X_i to generate a trial vector:

CHAPTER 4 DYNAMIC MULTI-SWARM PARTICLE SWARM OPTIMIZER

$$U_i^d = \begin{cases} V_i^d, & \text{if } (rand_i^d[0, 1] \leq CR) \text{ or } (d = d_{rand}) \\ X_i^d, & \text{otherwise} \end{cases}, \quad d = 1, 2, \dots, D \quad (4-1)$$

where CR is a user-specified crossover constant in the range $[0, 1)$ and d_{rand} is a randomly chosen integer in the range $[1, D]$ to ensure that the trial vector U_i will differ from its corresponding target vector X_i by at least one parameter.

In DMS-PSO, CR is replaced by learning probability Pc . Each Particle has a corresponding Pc_i . Every R generation, $keep_id_i$ is decided by Pc_i . If this random number is larger than or equal to Pc_i , this dimension will be set at the value of its own **pbest**, $keep_id_i^d$ is set to 1 and otherwise $keep_id_i^d$ is set to 0 and it will learn from the **lbest** and its own **pbest** as the PSO with constriction coefficient:

If $keep_id_i^d = 0$

$$V_i^d \leftarrow 0.729 * V_i^d + 1.49445 * rand1_i^d * (pbest_i^d - X_i^d) + 1.49445 * rand2_i^d * (lbest_k^d - X_i^d)$$

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

$$X_i^d \leftarrow X_i^d + V_i^d$$

Otherwise

$$X_i^d \leftarrow pbest_i^d$$

(4-2)

In CLPSO, Pc_i is predefined for each particle. While in DMS-PSO, adaptive Pc is employed. We accumulate the previous learning experience within a certain generational interval so as to dynamically adapt the value of Pc to a suitable range. We assume Pc normally distributed in a range with mean $\overline{Pc}$ and standard deviation 0.1. Initially, $\overline{Pc}$ is set at 0.5 and different Pc values conforming this normal distribution are generated for each individual in the current population. During every generation, the Pc values associated with the particles which find new **pbest** are recorded. These Pc values for all individuals remain the same until the sub-swarms are regrouped. When the sub-swarms are regrouped, the mean of normal distribution of Pc is recalculated according to all the recorded Pc values corresponding to successful trial vectors during this period. With this new normal distribution's mean and the standard deviation 0.1, a new set of Pc values is generated. As a result, the proper Pc value range for the current problem can be learned to suit the particular problem. Note that the record of the successful Pc values will be emptied once the normal distribution's mean is recalculated to avoid the possible inappropriate long-term accumulation effects.

CHAPTER 4 DYNAMIC MULTI-SWARM PARTICLE SWARM OPTIMIZER

Fig. 4-2 Sub-Flowchart 1 for DMS-L-PSO (Regrouping Schedule and Adaptive Pc)

4.1.3 Local Search Phase

In order to achieve better results on multi-modal problems, DMS-PSO is designed to make the particles have a large diversity, and consequently the convergence speed will be slow. 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 local search phase is added.

Every L generations, the **pbests** of five randomly chosen particles will be used as the starting points and the BFGS Quasi-Newton method is employed to do the local search. Then the five refined solutions will be obtained. We calculate the Euclidean distance to all the **pbests** for each refined solution and replace the nearest ones with the refined solutions. In the end of the search, every $5 \cdot L$ generations, the best solution achieved so far is refined using the BFGS Quasi-Newton method.

An illustration for the local search phase for a swarm of 10 particles is given in Fig. 4-3. Five **pbests** $pbest_2$, $pbest_4$, $pbest_6$, $pbest_8$ and $pbest_{10}$ are randomly chosen as the start points for the local search and 3 local optima x_1^* , x_2^* and x_3^*

CHAPTER 4 DYNAMIC MULTI-SWARM PARTICLE SWARM OPTIMIZER

are achieved after the local search. The nearest three **pbests** **pbest₁**, **pbest₄**, **pbest₈** are replaced by **x₁^{*}**, **x₂^{*}** and **x₃^{*}** respectively.

Fig. 4-3 Illustration of Local Search Phase for a Population with 10 Particles

Quasi-Newton methods build up curvature information in each generation to formulate a quadratic model problem as below:

$$\min \mathbf{x}^T \mathbf{H} \mathbf{x} + \mathbf{c}^T \mathbf{x} + b \quad (4-3)$$

Here **H** is the Hessian matrix, a positive definite symmetric matrix, **c** is a constant vector and **b** is a constant. If **x^{*}** is the optimal solution of this model, **x^{*}** will satisfy

$$\nabla f(\mathbf{x}^*) = \mathbf{H} \mathbf{x}^* + \mathbf{c} = 0 \quad (4-4)$$

Then

$$\mathbf{x} = -\mathbf{H}^{-1} \mathbf{c} \quad (4-5)$$

Quasi-Newton methods approximate **H** using the observed behavior of **f(x)** and **∇f(x)** to build up curvature information with an updating method to avoid calculating **H** numerically. An effective formula of Broyden [122], Fletcher [123], Goldfarb [124] and Shanno [125] (BFGS) is employed in the BFGS Quasi-Newton method.

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{\mathbf{q}_k \mathbf{q}_k^T}{\mathbf{q}_k^T \mathbf{s}_k} - \frac{\mathbf{H}_k^T \mathbf{s}_k^T \mathbf{s}_k \mathbf{H}_k}{\mathbf{s}_k^T \mathbf{H}_k \mathbf{s}_k} \quad (4-6)$$

where

$$\begin{aligned} \mathbf{s}_k &= \mathbf{x}_{k+1} - \mathbf{x}_k \\ \mathbf{q}_k &= \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k) \end{aligned}$$

H₀ is set to the identity matrix **I**. The inverse Hessian **H⁻¹** can also be

CHAPTER 4 DYNAMIC MULTI-SWARM PARTICLE SWARM OPTIMIZER

approximated using DFP formula of Davidon [126], Fletcher and Powell [127]. At each generation, a line search is performed in the direction

$$\mathbf{d} = -\mathbf{H}_k^{-1} \cdot \nabla f(\mathbf{x}_k) \tag{4-7}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d} \tag{4-8}$$

To simplify, the function "fminunc" in Matlab 6.5 is employed to realize the BFGS Quasi-Newton local search. Assuming the gradient information is not available for all problems, in all experiments in this thesis no gradient information is supplied to the Quasi-Newton method.

Fig. 4-4 Sub-Flowchart 2 for DMS-L-PSO (Local Search Phase)

4.1.4 Convergence Phase

Every generation, the distances among the **pbest** of all particles are checked, if the maximum distance for each dimension is smaller than a predefined value or some other convergence criteria are satisfied (e.g. 90 percent of maximum fitness evaluations have been used), the convergence phase is started. In the convergence phase, all particles form a single swarm to become a global PSO version. The flowchart of the convergence phase is presented in Fig. 4-5.

CHAPTER 4 DYNAMIC MULTI-SWARM PARTICLE SWARM OPTIMIZER

Fig. 4-5 Sub-Flowchart 3 for DMS-L-PSO (Convergence Phase)

CHAPTER 4 DYNAMIC MULTI-SWARM PARTICLE SWARM OPTIMIZER

4.1.5 DMS-PSO with Local Search (DMS-L-PSO)

The flowchart of DMS-L-PSO is given in Fig. 4-6.

Fig. 4-6 The Flowchart of DMS-L-PSO

CHAPTER 4 DYNAMIC MULTI-SWARM PARTICLE SWARM OPTIMIZER

From the description above, DMS-L-PSO is composed of PSO with constriction coefficient, dynamic sub-swarm topology, adaptive learning strategy and local search phase. In order to show the action of each part, PSO with constriction coefficient, PSO with constriction coefficient with only the dynamic neighborhood topology (DMS-PSO), DMS-PSO with the self-adaptive P_c and the complete DMS-PSO with the self-adaptive P_c and local search (DMS-L-PSO) are tested on the 10- D multimodal Rastrigin's problem which has a huge number of local optima with a population of 15 particles for 10 runs. The convergence maps for the median run for the four PSO versions are plotted in Fig. 4-7. From the figure, it can be observed that the regrouping dynamic multi-swarm topology improves the performance of PSO with constriction coefficient but the global optimum is not found by both algorithms. With the adaptive self-learning strategy, the global optimum region of Rastrigin's problem is found and the solution is refined slowly over the generations. With the DMS-L-PSO, the global optimum is found immediately after the global optimum region is found though the region is not found as fast as the DMS-PSO with only self-learning strategy since the local search consumed some fitness evaluations when the global optimum region has not been found.

Fig. 4-7 Convergence Maps of PSO, DMS-PSO, DMS-PSO with Adaptive P_c , DMS-L-PSO on 10- D Rastrigin's Problem

CHAPTER 4 DYNAMIC MULTI-SWARM PARTICLE SWARM OPTIMIZER

4.2 Experiments

With the reasons described in Chapter 3, a set of standard test suite has been provided in CEC'05 [128] (The formulas and properties of the test functions are listed in Appendix A). Researchers can test their algorithms with the same test functions and under the same comparison criteria. The codes in C, Matlab and Java can be downloaded from <http://www.ntu.edu.sg/home/EPNSugan/>. The DMS-L-PSO was tested on this set of benchmark functions and the results are presented in this section. The comparison results of DMS-L-PSO with some other algorithms will be provided.

4.2.1 Evaluation Criteria and Parameters Setting

Problems: 25 minimization problems

Dimensions: $D = 10, 30$

Runs / problem: 25

Max_FES: $10000 * D$ (Max_FES_10D = 100000; Max_FES_30D = 300000; Max_FES_50D = 500000)

Initialization: Uniform random initialization within the search space, except for problems 7 and 25, for which initialization ranges are specified.

The same initializations are used for the comparison pairs (problems 1, 2, 3 & 4, problems 9 & 10, problems 15, 16 & 17, problems 18, 19 & 20, problems 21, 22 & 23, problems 24 & 25).

Global Optimum: All problems, except 7 and 25, have the global optimum within the given bounds and there is no need to perform search outside of the given bounds for these problems. 7 & 25 are exceptions without a search range and with the global optimum outside of the specified initialization range.

Termination: Terminate before reaching Max_FES if the error in the function value is 10^{-8} or less.

Ter_Err: 10^{-8} (termination error value)

Successful Run: A run during which the algorithm achieves the fixed accuracy level within the Max_FES for the particular dimension.

Success Rate = (# of successful runs according to the table above) / total runs

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

CHAPTER 4 DYNAMIC MULTI-SWARM PARTICLE SWARM OPTIMIZER

successful runs)

Experiments are conducted on all the twenty-five 10-*D* and 30-*D* problems. To solve these test functions, number of swarms is set at 20, each swarm’s population size is 3. Hence, the population size is 60. $\omega = 0.729$, $c_1 = c_2 = 1.49445$, $R = 10$, $L = 100$, $L_FEs = 200$. *Max_FEs* is set at 100,000 for 10-*D* and 300,000 for 30-*D*. *Vmax* restricts particles’ velocities, where **Vmax** is equal to 50% of the search range.

4.2.2 Results of DMS-L-PSO

For each function, the DMS-L-PSO is run 25 times. Best functions error values achieved when $FES = 1e+3$, $FES = 1e+4$, $FES = 1e+5$, best error values achieved within the *Max_FEs*, Success Performance and Successful FEs for the 25 10-*D* test functions are presented in Table 4-1 to Table 4-5. Success here means achieving the desired error accuracy within the permitted number of FEs. The predefined tolerance values for the 25 test functions are $1e-6$ for functions 1-5, $1e-2$ for functions 6-14 and $1e-1$ for functions 15-25. Success Performance and Successful FEs for the twenty 30-*D* test functions are listed in Table 4-6 to Table 4-10.

Table 4-1 Best Functions Error Values Achieved When $FES = 1e+3$, $FES = 1e+4$, $FES = 1e+5$ for 10-*D* Functions 1-6

Func		1	2	3	4	5	6
FES							
1e+3	1 st (Min)	4.7606e+2	2.3241e+3	4.3890e+6	4.2381e+3	5.4461e+3	3.3333e+6
	7 th	9.0182e+2	4.1208e+3	9.3089e+6	5.5652e+3	7.5985e+3	1.6372e+7
	13 th (Median)	9.0400e+2	5.6379e+3	1.4851e+7	7.4766e+3	9.4222e+3	2.4678e+7
	19 th	1.4312e+3	6.6237e+3	2.5634e+7	9.2706e+3	9.6468e+3	4.7023e+7
	25 th (Max)	1.7879e+3	8.0709e+3	5.4493e+7	1.0960e+4	1.2218e+4	7.0993e+7
	Mean	1.1615e+3	5.3185e+3	1.6602e+7	7.3842e+3	9.4859e+3	3.2707e+7
	Std	3.2957e+2	1.5985e+3	1.0989e+7	1.7334e+3	1.5735e+3	1.9804e+7
1e+4	1 st (Min)	1.6194e-3	3.5235e+1	3.5891e+5	9.8776e+1	5.5436e+0	1.2633e+1
	7 th	2.4831e-3	5.3496e+1	4.5978e+5	1.8418e+2	3.4302e+1	3.2958e+1
	13 th (Median)	3.6850e-3	8.6247e+1	7.6777e+5	2.2116e+2	4.7204e+1	5.3564e+1
	19 th	5.5294e-3	1.0630e+2	9.8356e+5	2.7329e+2	5.2831e+1	9.1480e+1
	25 th (Max)	1.0206e-2	1.7472e+2	1.4654e+6	5.3099e+2	1.1267e+2	2.2946e+2
	Mean	4.7927e-3	8.4143e+1	7.5234e+5	2.3916e+2	5.0527e+1	7.4363e+1
	Std	2.1106e-3	4.3660e+1	3.2522e+5	8.8845e+1	2.4202e+1	5.4506e+1
1e+5	1 st (Min)	0	0	1.3977e-9	9.5871e-7	1.1250e-8	5.1036e-11
	7 th	0	5.3951e-14	5.3204e-9	5.4357e-5	1.5020e-7	1.3090e-9
	13 th (Median)	0	5.5363e-14	6.1627e-9	9.6432e-4	4.4277e-7	2.5229e-9
	19 th	0	9.5431e-14	7.8253e-9	2.1679e-3	8.2050e-7	5.0117e-9
	25 th (Max)	0	6.4026e-13	1.1240e-8	7.2618e-3	9.6243e-7	1.5958e-6
	Mean	0	1.2925e-13	5.7581e-9	1.5115e-3	6.0857e-7	5.8344e-8
	Std	0	1.3859e-13	2.2538e-9	1.7024e-3	2.0012e-7	2.5859e-7

CHAPTER 4 DYNAMIC MULTI-SWARM PARTICLE SWARM OPTIMIZER

Table 4-2 Best Functions Error Values Achieved When $FES = 1e+3$, $FES = 1e+4$, $FES = 1e+5$ for 10-D Functions 7-12

Func		7	8	9	10	11	12
FES							
1e+3	1 st (Min)	7.6292e+2	2.0301e+1	3.3796e+1	5.1843e+1	7.5823e+0	5.7017e+3
	7 th	1.0108e+3	2.0421e+1	4.0043e+1	5.7315e+1	7.8636e+0	1.3207e+4
	13 th (Median)	1.3205e+3	2.0691e+1	5.2046e+1	6.0987e+1	8.6866e+0	2.1996e+4
	19 th	1.3338e+3	2.0771e+1	5.6042e+1	7.0999e+1	9.7703e+0	2.6490e+4
	25 th (Max)	1.3760e+3	2.0812e+1	5.6270e+1	7.1588e+1	1.0631e+1	4.9658e+4
	Mean	1.2661e+3	2.0610e+1	4.9442e+1	6.2255e+1	9.3420e+0	2.2051e+4
	Std	1.8898e+2	1.1975e-1	7.7496e+0	8.7235e+0	9.0867e-1	9.7744e+3
1e+4	1 st (Min)	4.6019e-1	2.0013e+1	3.4055e+0	1.0321e+1	4.8650e+0	1.3845e+2
	7 th	6.6474e-1	2.0285e+1	5.0127e+0	1.8747e+1	5.7083e+0	4.4510e+2
	13 th (Median)	7.1225e-1	2.0297e+1	6.9235e+0	1.9209e+1	6.6029e+0	4.5243e+2
	19 th	7.6636e-1	2.0385e+1	8.0263e+0	2.4066e+1	6.7230e+0	1.3857e+3
	25 th (Max)	1.2426e+0	2.0486e+1	9.7490e+0	2.4427e+1	6.8491e+0	4.2058e+3
	Mean	7.7460e-1	2.0205e+1	6.8201e+0	1.9460e+1	5.5310e+0	1.0997e+3
	Std	1.6947e-1	8.6521e-2	1.2340e+0	4.4173e+0	5.3919e-1	1.0934e+3
1e+5	1 st (Min)	4.2919e-9	2.0000e+1	0	1.8780e+0	2.5311e+0	0
	7 th	2.0822e-2	2.0000e+1	0	2.6633e+0	3.9723e+0	0
	13 th (Median)	3.6701e-2	2.0000e+1	3.3796e+1	5.1843e+1	7.5823e+0	2.9762e-11
	19 th	6.5003e-2	2.0000e+1	4.0043e+1	5.7315e+1	7.8636e+0	3.8412e-10
	25 th (Max)	1.2053e-1	2.0000e+1	5.2046e+1	6.0987e+1	8.6866e+0	4.9274e-9
	Mean	4.4308e-2	2.0000e+1	5.6042e+1	7.0999e+1	9.7703e+0	2.9481e-11
	Std	3.0019e-2	4.5621e-9	5.6270e+1	7.1588e+1	1.0631e+1	1.9234e-10

Table 4-3 Best Functions Error Values Achieved When $FES = 1e+3$, $FES = 1e+4$, $FES = 1e+5$ for 10-D Functions 13-18

Func		13	14	15	16	17	18
FES							
1e+3	1 st (Min)	3.3767e+0	2.4335e+0	3.5805e+2	2.2146e+2	2.1878e+2	1.0121e+3
	7 th	4.2115e+0	3.2694e+0	3.8020e+2	2.2685e+2	2.5791e+2	1.0547e+3
	13 th (Median)	5.0618e+0	3.3170e+0	4.7105e+2	2.4439e+2	2.6046e+2	1.0719e+3
	19 th	5.9807e+0	3.8328e+0	5.6420e+2	2.5340e+2	2.8675e+2	1.0785e+3
	25 th (Max)	6.0583e+0	4.0128e+0	5.6433e+2	2.7466e+2	3.6976e+2	1.1612e+3
	Mean	5.1021e+0	3.6934e+0	4.6951e+2	2.4349e+2	2.8093e+2	1.1038e+3
	Std	6.7722e-1	1.8724e-1	6.6535e+1	2.3532e+1	3.9337e+1	2.7463e+1
1e+4	1 st (Min)	5.7747e-1	2.0323e+0	6.6807e+1	1.0196e+2	1.1684e+2	7.8435e+2
	7 th	1.1862e+0	2.7207e+0	9.3148e+1	1.0288e+2	1.2538e+2	8.0786e+2
	13 th (Median)	1.2460e+0	2.7868e+0	1.0326e+2	1.2440e+2	1.2997e+2	9.2833e+2
	19 th	1.2899e+0	2.8921e+0	1.2741e+2	1.3710e+2	1.3327e+2	9.7175e+2
	25 th (Max)	1.4859e+0	3.0387e+0	2.4211e+2	1.4062e+2	1.5920e+2	9.8970e+2
	Mean	1.1498e+0	2.7558e+0	1.0796e+2	1.0697e+2	1.2779e+2	8.8621e+2
	Std	2.8044e-1	2.3343e-1	3.7068e+1	1.2453e+1	1.0685e+1	7.6968e+1
1e+5	1 st (Min)	2.2524e-1	1.1661e+0	0	4.4761e+1	9.3894e+1	3.0000e+2
	7 th	2.8964e-1	1.5995e+0	0	7.7593e+1	9.4937e+1	8.0000e+2
	13 th (Median)	3.4220e-1	2.0055e+0	0	8.2394e+1	9.8583e+1	8.0000e+2
	19 th	3.4930e-1	2.0492e+0	0	8.5729e+1	1.0568e+2	8.2482e+2
	25 th (Max)	3.8759e-1	2.3752e+0	3.5114e+1	9.1622e+1	1.0736e+2	9.7261e+2
	Mean	3.4395e-1	2.0975e+0	3.8182e+0	8.5228e+1	9.7366e+1	7.4752e+2
	Std	4.4917e-2	2.7601e-1	1.0409e+1	8.8668e+0	3.3463e+0	1.7642e+2

CHAPTER 4 DYNAMIC MULTI-SWARM PARTICLE SWARM OPTIMIZER

Table 4-4 Best Functions Error Values Achieved When $FES = 1e+3$, $FES = 1e+4$, $FES = 1e+5$ for 10-D Functions 19-25

Func		19	20	21	22	23	24	25
FES								
1e+3	1 st (Min)	1.0307e+3	1.0627e+3	1.0585e+3	8.6796e+2	9.6612e+2	1.0517e+3	1.6469e+3
	7 th	1.0777e+3	1.0948e+3	1.1951e+3	9.2381e+2	1.2135e+3	1.1959e+3	1.7022e+3
	13 th (Median)	1.0936e+3	1.1078e+3	1.2489e+3	9.3786e+2	1.2803e+3	1.2155e+3	1.7075e+3
	19 th	1.1286e+3	1.1304e+3	1.2525e+3	9.9294e+2	1.2984e+3	1.2758e+3	1.7486e+3
	25 th (Max)	1.1507e+3	1.1885e+3	1.2661e+3	1.0830e+3	1.3068e+3	1.3179e+3	1.8569e+3
	Mean	1.0719e+3	1.1246e+3	1.2269e+3	9.6595e+2	1.2131e+3	1.2271e+3	1.7363e+3
	Std	2.6606e+1	2.3807e+1	4.2009e+1	4.8204e+1	7.2802e+1	6.3340e+1	4.8476e+1
1e+4	1 st (Min)	6.0864e+2	8.0233e+2	4.8400e+2	3.8336e+2	5.5891e+2	2.0001e+2	2.1249e+2
	7 th	8.1271e+2	8.2316e+2	7.6740e+2	7.4448e+2	7.1691e+2	2.0004e+2	1.0817e+3
	13 th (Median)	8.5730e+2	9.2282e+2	8.6753e+2	7.4505e+2	9.4648e+2	2.0019e+2	1.1417e+3
	19 th	9.4632e+2	9.4889e+2	9.3727e+2	7.6005e+2	9.6578e+2	2.0031e+2	1.2507e+3
	25 th (Max)	9.9401e+2	9.9227e+2	1.0093e+3	7.9087e+2	1.0927e+3	4.9102e+2	1.3422e+3
	Mean	8.7275e+2	8.7316e+2	8.2430e+2	7.4461e+2	8.3155e+2	2.2836e+2	1.0057e+3
	Std	8.5551e+1	8.0647e+1	1.3659e+2	7.5782e+1	1.7784e+2	1.0182e+2	3.5474e+2
1e+5	1 st (Min)	3.0000e+2	8.0000e+2	3.0000e+2	3.0000e+2	5.3651e+2	2.0000e+2	2.0000e+2
	7 th	8.0000e+2	8.0000e+2	3.0000e+2	7.1261e+2	5.4830e+2	2.0000e+2	2.0000e+2
	13 th (Median)	8.0000e+2	8.0000e+2	5.0000e+2	7.3615e+2	7.1647e+2	2.0000e+2	4.0032e+2
	19 th	8.0000e+2	8.0000e+2	8.0000e+2	7.3615e+2	8.9834e+2	2.0000e+2	4.0761e+2
	25 th (Max)	8.0000e+2	9.1263e+2	8.0000e+2	7.5652e+2	9.5921e+2	5.0000e+2	9.0012e+2
	Mean	6.8652e+2	8.0625e+2	5.3600e+2	6.7162e+2	7.2789e+2	2.1200e+2	3.5480e+2
	Std	1.9263e+2	34826e+1	2.1772e+2	1.3512e+2	1.6345e+2	6.0000e+1	1.4862e+2

Table 4-5 Successful FES & Success Performance for 10-D

Func	1st(Min)	7th	13th (Median)	19th	25th (Max)	Mean	Std	Success Rate	Success Performance
1	4785	5346	6398	8036	8328	6982	1412	100%	6982
2	11286	11320	11386	11491	12213	11140	113.63	100%	11140
3	10782	11853	12353	12450	12788	11663	141.04	100%	11663
4	89072	-	-	-	-	-	-	4%	2226800
5	72319	86842	87379	89772	98812	86452	8721	100%	86452
6	25026	49252	52885	64136	76693	53132	12951	100%	53132
7	50062	-	-	-	-	-	-	16%	531295
8	-	-	-	-	-	-	-	0%	-
9	23659	24931	35568	38490	46004	34297	8776.7	100%	34297
10	-	-	-	-	-	-	-	0%	-
11	-	-	-	-	-	-	-	0%	-
12	12047	18924	24681	36679	-	-	-	80%	28974
13	-	-	-	-	-	-	-	0%	-
14	-	-	-	-	-	-	-	0%	-
15	13565	27863	41997	66544	-	-	-	84%	51274
16	-	-	-	-	-	-	-	0%	-
17	-	-	-	-	-	-	-	0%	-
18	-	-	-	-	-	-	-	0%	-
19	-	-	-	-	-	-	-	0%	-
20	-	-	-	-	-	-	-	0%	-
21	-	-	-	-	-	-	-	0%	-
22	-	-	-	-	-	-	-	0%	-
23	-	-	-	-	-	-	-	0%	-
24	-	-	-	-	-	-	-	0%	-
25	-	-	-	-	-	-	-	0%	-

CHAPTER 4 DYNAMIC MULTI-SWARM PARTICLE SWARM OPTIMIZER

Table 4-6 Best Functions Error Values Achieved When $FES = 1e+3$, $FES = 1e+4$, $FES = 1e+5$, $FES = 3e+5$ for 30-D Functions 1-6

Func FES		1	2	3	4	5	6
1e+3	1st(Min)	2.1729e+4	3.2222e+4	2.5931e+8	3.4840e+4	2.1090e+4	3.2507e+9
	7th	2.3469e+4	4.4451e+4	4.4193e+8	5.0262e+4	2.1395e+4	6.1612e+9
	13th(Median)	2.7145e+4	4.6754e+4	5.0638e+8	6.4526e+4	2.4561e+4	6.8639e+9
	19th	2.7888e+4	5.3740e+4	5.0989e+8	6.8111e+4	2.5169e+4	9.7089e+9
	25th (Max)	2.9910e+4	6.5930e+4	7.4406e+8	8.7190e+4	2.6012e+4	1.0876e+10
	Mean	2.6274e+4	4.4477e+4	4.8993e+8	5.4043e+4	2.3575e+4	6.9369e+9
1e+4	Std	3.4141e+3	9.7501e+3	1.0806e+8	1.1841e+4	1.4064e+3	2.0295e+9
	1st(Min)	1.3204e-11	1.3808e+3	7.2509e+6	1.4291e+4	4.6653e+3	2.8960e+5
	7th	1.6717e-11	1.6492e+3	1.4898e+7	1.8458e+4	5.1405e+3	5.3266e+5
	13th(Median)	2.4709e-11	1.8946e+3	1.5831e+7	1.9613e+4	6.1097e+3	6.7066e+5
	19th	3.1725e-11	2.6194e+3	1.6679e+7	2.3675e+4	6.1569e+3	9.8958e+5
	25th (Max)	1.3830e-10	2.8715e+3	2.1460e+7	2.5009e+4	7.9362e+3	2.8960e+6
1e+5	Mean	3.9346e-11	1.8576e+3	1.6360e+7	2.1104e+4	5.8683e+3	7.3386e+5
	Std	3.6064e-11	4.4004e+2	2.9842e+6	3.5697e+3	6.7350e+2	3.5529e+5
	1st(Min)	0	7.3391e-7	2.5029e+5	3.8744e+3	2.7500e+3	6.4626e+0
	7th	0	1.3218e-6	5.4528e+5	5.0079e+3	2.7876e+3	2.0150e+1
	13th(Median)	0	2.0653e-6	7.2171e+5	5.9741e+3	2.8654e+3	6.9898e+1
	19th	0	2.6809e-6	8.2179e+5	6.7598e+3	3.2355e+3	9.9147e+1
3e+5	25th (Max)	4.7236e-14	5.4879e-6	1.0427e+6	7.5498e+3	3.2422e+3	1.3585e+2
	Mean	1.0773e-14	2.1915e-6	7.5996e+5	5.8692e+3	3.0995e+3	6.5721e+1
	Std	2.0940e-14	1.1641e-6	2.2436e+5	1.0131e+3	1.8028e+2	5.2804e+1
	1st(Min)	0	4.0692e-8	6.2213e-8	1.7217e+3	1.1993e+2	2.7001e-8
	7th	0	6.8478e-8	9.1830e-8	2.0168e+3	1.8822e+3	6.3028e-8
	13th(Median)	0	8.9051e-8	2.3769e-7	2.0405e+3	1.9815e+3	1.4178e-7

Table 4-7 Best Functions Error Values Achieved When $FES = 1e+3$, $FES = 1e+4$, $FES = 1e+5$, $FES = 3e+5$ for 30-D Functions 7-12

Func FES		7	8	9	10	11	12
1e+3	1st(Min)	6.5364e+3	1.6739e+1	2.3459e+2	3.1415e+2	3.2545e+1	4.4294e+5
	7th	6.5400e+3	1.7172e+1	2.5962e+2	3.5099e+2	3.3975e+1	7.0954e+5
	13th(Median)	6.7476e+3	1.7483e+1	2.7963e+2	3.6084e+2	3.5814e+1	8.7443e+5
	19th	7.4726e+3	1.8160e+1	2.8027e+2	4.4151e+2	3.7097e+1	9.1481e+5
	25th (Max)	8.8767e+3	2.0124e+1	3.5027e+2	4.5736e+2	3.7518e+1	9.2709e+5
	Mean	7.6035e+3	1.8653e+1	2.7195e+2	4.4249e+2	3.5616e+1	7.5287e+5
1e+4	Std	5.0167e+2	6.7116e-2	1.8104e+1	2.6385e+1	1.4597e+0	1.3413e+5
	1st(Min)	4.9334e+0	1.5917e+1	2.6141e+1	4.2388e+1	2.7446e+1	9.4231e+3
	7th	9.2606e+0	1.6999e+1	3.3414e+1	6.0206e+1	2.8555e+1	2.1921e+4
	13th(Median)	1.0202e+1	1.7331e+1	4.4773e+1	7.4549e+1	2.9052e+1	2.6790e+4
	19th	1.3053e+1	1.7732e+1	5.2300e+1	7.7214e+1	3.1470e+1	3.4128e+4
	25th (Max)	1.7354e+1	1.9164e+1	6.2698e+1	9.1816e+1	3.5115e+1	4.9968e+4
1e+5	Mean	1.0756e+1	1.8321e+1	3.9602e+1	7.5635e+1	3.4503e+1	2.5277e+4
	Std	3.3913e+0	7.4226e-2	7.4136e+0	1.2209e+1	1.2478e+0	1.1907e+4
	1st(Min)	4.0620e-7	1.5877e+1	9.9643e+0	2.3505e+1	2.1838e+1	3.4596e-2
	7th	2.0026e-6	1.6402e+1	1.8871e+1	3.7320e+1	2.3587e+1	1.3915e+2
	13th(Median)	8.0639e-3	1.6985e+1	2.2402e+1	3.9938e+1	2.3938e+1	4.3216e+2
	19th	8.9202e-3	1.7446e+1	2.4587e+1	4.2443e+1	2.6550e+1	1.1728e+3
3e+5	25th (Max)	1.1105e-2	1.8965e+1	2.7240e+1	5.5469e+1	2.7759e+1	4.3950e+3
	Mean	5.3041e-3	1.5141e+1	2.0712e+1	3.4216e+1	2.3179e+1	8.6385e+2
	Std	3.7872e-3	2.0657e-2	3.3129e+0	6.9524e+0	1.7465e+0	1.0742e+3
	1st(Min)	7.3120e-13	1.5594e+1	2.6823e-13	2.0802e+1	2.0240e+1	1.2941e-6
	7th	1.3994e-6	1.6068e+1	2.1826e-9	2.6995e+1	2.0428e+1	2.4993e+0
	13th(Median)	7.6607e-3	1.6494e+1	4.2198e-7	3.3170e+1	2.3727e+1	7.5441e+1

CHAPTER 4 DYNAMIC MULTI-SWARM PARTICLE SWARM OPTIMIZER

Table 4-8 Best Functions Error Values Achieved When $FES = 1e+3$, $FES = 1e+4$, $FES = 1e+5$, $FES = 3e+5$ for 30-D Functions 13-18

Func		13	14	15	16	17	18
FES							
1e+3	1st(Min)	2.7280e+1	1.1453e+1	6.4526e+2	5.0944e+2	4.7804e+2	1.1569e+3
	7th	3.3637e+1	1.1470e+1	7.4512e+2	5.5336e+2	7.5462e+2	1.1998e+3
	13th(Median)	3.8838e+1	1.2718e+1	8.3027e+2	6.0648e+2	8.4002e+2	1.2139e+3
	19th	3.9071e+1	1.3219e+1	8.4861e+2	6.5156e+2	8.8366e+2	1.2535e+3
	25th (Max)	4.1717e+1	1.3350e+1	9.1498e+2	7.4874e+2	1.0932e+3	1.3021e+3
	Mean	3.7251e+1	1.0821e+1	8.4115e+2	5.7643e+2	7.7041e+2	1.2162e+3
	Std	3.7595e+0	1.2797e-1	5.6448e+1	7.4049e+1	9.7439e+1	3.7263e+1
1e+4	1st(Min)	4.7178e+0	9.7791e+0	3.8387e+2	1.7354e+2	1.9660e+2	9.2849e+2
	7th	6.3283e+0	1.0376e+1	3.9146e+2	2.0278e+2	2.5937e+2	9.3543e+2
	13th(Median)	6.3564e+0	1.0454e+1	4.0371e+2	2.3678e+2	2.6126e+2	9.4313e+2
	19th	7.2596e+0	1.1034e+1	4.2282e+2	2.3921e+2	3.5882e+2	9.4412e+2
	25th (Max)	7.4513e+0	1.1543e+1	4.9576e+2	4.2510e+2	4.1833e+2	9.4858e+2
	Mean	6.9592e+0	1.1011e+1	4.3333e+2	2.2483e+2	3.1049e+2	9.3907e+2
	Std	8.0384e-1	2.9314e-1	3.1058e+1	8.4760e+1	6.3835e+1	4.3321e+0
1e+5	1st(Min)	1.6350e+0	9.5380e+0	2.3760e+2	7.0740e+1	1.2795e+2	9.0951e+2
	7th	2.1690e+0	9.8614e+0	2.7586e+2	8.4493e+1	1.5603e+2	9.1182e+2
	13th(Median)	2.5532e+0	1.0425e+1	3.4044e+2	9.9406e+1	1.6451e+2	9.1284e+2
	19th	3.1235e+0	1.0689e+1	3.6412e+2	1.2661e+2	1.8049e+2	9.1691e+2
	25th (Max)	3.7843e+0	1.1418e+1	4.4467e+2	4.1451e+2	3.7585e+2	9.1707e+2
	Mean	2.4683e+0	1.0399e+1	3.2854e+2	1.4416e+2	1.6733e+2	9.1114e+2
	Std	5.3493e-1	2.5418e-1	4.3508e+1	9.0415e+1	6.5105e+1	1.1274e+0
3e+5	1st(Min)	1.0452e+0	8.4284e+0	2.3110e+2	4.3453e+1	8.3846e+1	9.0672e+2
	7th	1.7188e+0	9.0145e+0	2.4079e+2	4.8817e+1	9.4989e+1	9.0754e+2
	13th(Median)	2.1228e+0	9.6691e+0	2.5388e+2	5.7569e+1	9.7643e+1	9.0956e+2
	19th	2.2089e+0	1.0651e+1	3.2344e+2	9.0089e+1	1.3591e+2	9.1174e+2
	25th (Max)	2.9695e+0	1.1400e+1	3.4841e+2	3.8210e+2	3.6621e+2	9.1285e+2
	Mean	1.9804e+0	9.0267e+0	2.8262e+2	1.1315e+2	1.0903e+2	9.1057e+2
	Std	4.1919e-1	3.1115e-1	4.5045e+1	9.7081e+1	6.4994e+1	1.4712e+0

Table 4-9 Best Functions Error Values Achieved When $FES = 1e+3$, $FES = 1e+4$, $FES = 1e+5$, $FES = 3e+5$ for 30-D Functions 19-25

Func.		19	20	21	22	23	24	25
FES								
1e+3	1st(Min)	1.1586e+3	1.1723e+3	1.2734e+3	1.2220e+3	1.2625e+3	1.3370e+3	1.7354e+3
	7th	1.1888e+3	1.2099e+3	1.3023e+3	1.3640e+3	1.2935e+3	1.3710e+3	1.8010e+3
	13th(Median)	1.2072e+3	1.2219e+3	1.3132e+3	1.3926e+3	1.3051e+3	1.3724e+3	1.8104e+3
	19th	1.2406e+3	1.2413e+3	1.3286e+3	1.3998e+3	1.3175e+3	1.3771e+3	1.8370e+3
	25th (Max)	1.2771e+3	1.2830e+3	1.3397e+3	1.5506e+3	1.3309e+3	1.4063e+3	1.8603e+3
	Mean	1.2176e+3	1.2305e+3	1.3093e+3	1.3868e+3	1.3090e+3	1.3699e+3	1.8116e+3
	Std	3.3511e+1	2.4579e+1	1.7219e+1	6.5122e+1	1.7324e+1	1.6311e+1	2.6727e+1
1e+4	1st(Min)	9.3364e+2	9.2808e+2	5.2446e+2	1.0021e+3	5.6105e+2	2.8555e+2	1.3422e+3
	7th	9.3948e+2	9.3877e+2	5.5038e+2	1.0194e+3	5.7594e+2	3.3739e+2	1.3725e+3
	13th(Median)	9.4021e+2	9.4198e+2	5.5935e+2	1.0343e+3	5.8992e+2	3.5564e+2	1.3790e+3
	19th	9.4236e+2	9.4569e+2	5.7846e+2	1.0612e+3	6.1218e+2	3.8107e+2	1.3938e+3
	25th (Max)	9.4714e+2	9.4781e+2	9.5193e+2	1.0920e+3	8.5641e+2	5.2815e+2	1.4363e+3
	Mean	9.3841e+2	9.3906e+2	5.8827e+2	1.0404e+3	6.0065e+2	3.7120e+2	1.3875e+3
	Std	3.3608e+0	4.3186e+0	1.0389e+2	2.5013e+1	5.8778e+1	6.1691e+1	2.2026e+1
1e+5	1st(Min)	9.0280e+2	8.0000e+2	5.0000e+2	8.9955e+2	5.3416e+2	2.0000e+2	2.0000e+2
	7th	9.0737e+2	9.0642e+2	5.0000e+2	9.2900e+2	5.3416e+2	2.0000e+2	2.0002e+2
	13th(Median)	9.0738e+2	9.0725e+2	5.0000e+2	9.3219e+2	5.3416e+2	2.0000e+2	2.0005e+2
	19th	9.1190e+2	9.1000e+2	5.0000e+2	9.4423e+2	5.3416e+2	2.0000e+2	6.5124e+2
	25th (Max)	9.1318e+2	9.1356e+2	5.0000e+2	9.5392e+2	5.3416e+2	2.0000e+2	1.0512e+3
	Mean	9.0841e+2	9.0263e+2	5.0000e+2	9.3344e+2	5.3416e+2	2.0000e+2	4.5812e+2
	Std	1.4093e+0	3.1337e+1	1.2941e-9	1.2166e+1	4.2751e-5	4.1284e-11	2.4124e+2
3e+5	1st(Min)	9.0195e+2	8.0000e+2	5.0000e+2	8.9221e+2	5.3416e+2	2.0000e+2	2.0000e+2
	7th	9.0275e+2	9.0601e+2	5.0000e+2	9.0849e+2	5.3416e+2	2.0000e+2	2.0000e+2
	13th(Median)	9.0582e+2	9.0652e+2	5.0000e+2	9.1773e+2	5.3416e+2	2.0000e+2	2.0000e+2
	19th	9.0717e+2	9.0733e+2	5.0000e+2	9.2036e+2	5.3416e+2	2.0000e+2	2.0000e+2
	25th (Max)	9.1030e+2	9.1186e+2	5.0000e+2	9.2552e+2	5.3416e+2	2.0000e+2	2.0000e+2
	Mean	9.0285e+2	8.9857e+2	5.0000e+2	9.1290e+2	5.3416e+2	2.0000e+2	2.0000e+2
	Std	1.3338e+0	3.0575e+1	7.5128e-12	7.8210e+0	1.0841e-5	1.2941e-13	3.1851e-6

CHAPTER 4 DYNAMIC MULTI-SWARM PARTICLE SWARM OPTIMIZER

Table 4-10 Successful FES & Success Performance for 30-D

Func	1st(Min)	7th	13th (Median)	19th	25th (Max)	Mean	Std	Success Rate	Success Performance
1	4907	4912	4913	4927	4961	4889	70.553	100%	4889
2	89965	106841	119260	132062	154062	123681	17120	100%	123681
3	246383	267136	275357	288627	-	-	-	88%	335812
4	-	-	-	-	-	-	-	0%	-
5	-	-	-	-	-	-	-	0%	-
6	271237	277638	279159	285451	-	-	-	96%	322140
7	40774	49567	59329	61035	-	-	-	96%	59620
8	-	-	-	-	-	-	-	0%	-
9	199212	219483	230481	259288	278422	208402	12084	100%	208402
10	-	-	-	-	-	-	-	0%	-
11	-	-	-	-	-	-	-	0%	-
12	113545	-	-	-	-	-	-	20%	1582661
13	-	-	-	-	-	-	-	0%	-
14	-	-	-	-	-	-	-	0%	-
15	-	-	-	-	-	-	-	0%	-
16	-	-	-	-	-	-	-	0%	-
17	-	-	-	-	-	-	-	0%	-
18	-	-	-	-	-	-	-	0%	-
19	-	-	-	-	-	-	-	0%	-
20	-	-	-	-	-	-	-	0%	-
21	-	-	-	-	-	-	-	0%	-
22	-	-	-	-	-	-	-	0%	-
23	-	-	-	-	-	-	-	0%	-
24	-	-	-	-	-	-	-	0%	-
25	-	-	-	-	-	-	-	0%	-

The first five functions are unimodal functions, function 1 is Shifted Sphere Function, function 2 is shifted Schwefel’s problem 1.2, and the function 3 is shifted rotated high conditioned elliptic function. These three functions have different condition numbers which make function 3 to be harder than function 2 and function 2 to be harder than function 1. From the results, we could observe that DMS-L-PSO achieves a better result for function 1 than function 2; a better result for function 2 than function 3. Function 4 is shifted Schwefel’s problem 1.2 with noise in fitness. With the noise in the fitness, it disturbs the search process. DMS-L-PSO succeeds once for 10-*D* case but only achieves 1e-2 error for 30-*D* which does not meet the 1e-6 criterion. Function 5 is Schwefel’s problem 2.6 with global optimum on bounds. DMS-L-PSO achieves 100% success rate on this problem for 10-*D* but only gets 1e+2 error for 30-*D*.

Functions 6-25 are multimodal problems. Function 6 is shifted Rosenbrock’s Function, a problem with unimodal and multimodal characteristics and an algorithm with good local search ability can achieve good results on Rosenbrock’s. DMS-L-PSO achieves 100% and 96% success rate on it for 10-*D* and 30-*D* owing to Quasi-Newton’s good local search ability. Function 7 is shifted rotated Griewank’s function without bounds, only the initialization range is given and the search range

CHAPTER 4 DYNAMIC MULTI-SWARM PARTICLE SWARM OPTIMIZER

is $[-\infty, +\infty]^D$. Griewank's function is more difficult with low dimension to achieve the global optimum. DMS-L-PSO only achieves 16% success rate on this problem for 10- D . Function 8 is shifted rotated Ackley's function with global optimum on bounds, which has a very narrow global basin and half of the dimensions of this basin are on the bounds. Hence, the search is almost like seeking a needle in a haystack. DMS-L-PSO fails on this problem in all 25 runs. Functions 9 and 10 are shifted Rastrigin's function and shifted rotated Rastrigin's function respectively. Both have a huge number of local optima. For 10- D DMS-L-PSO hits the global optimum in all 25 runs for function 9, but the results are not that good for function 10 owing to the rotation. Function 11 is shifted rotated Weierstrass function and the results are not very good. Function 12 is Schwefel's problem and most time the global optimum is found, but when DMS-L-PSO fails, it falls into a local optimum with very bad fitness value. The success rate for this problem is 100% and 20% for 10- D and 30- D respectively. Functions 13 and 14 are extended functions, and functions 15-25 are eleven novel composition functions. They all built up with basic functions. They give a big challenge to any search algorithm. DMS-L-PSO only achieves 88% success rate on function 15 for 10- D and fails on all others.

4.2.3 Algorithm Cost

In order to show how the relationship of the algorithm cost with the dimension, the algorithm cost is calculated in the following way:

- i) evaluate the computing time just for Function 3. For 200000 evaluations of a certain dimension D , it gives T_1 ;
- ii) the complete computing time for the algorithm with 200000 evaluations of the same D dimensional benchmark function 3 is T_2 .

We execute step b 5 times and get 5 T_2 values. $\hat{T}_2 = \text{Mean}(T_2)$. The complexity of the algorithm is reflected by: $\hat{T}_2$, T_1 , and $(\hat{T}_2 - T_1)/T_1$. In step b, we execute the complete algorithm 5 times to accommodate variations in execution time due adaptive nature of some algorithms.

CHAPTER 4 DYNAMIC MULTI-SWARM PARTICLE SWARM OPTIMIZER

Table 4-11 Algorithm Cost

	T_1	$\hat{T}_2$	$(\hat{T}_2 - T_1)/T_1$
$D = 10$	31.8713	69.8241	1.1908
$D = 30$	37.9274	78.8724	1.0796
$D = 50$	48.7241	97.8412	1.0081

In order to get a clearer idea about how the algorithm cost change with the dimension, a more detailed figure is given in Fig. 4-8. In this figure, the values of $(\hat{T}_2 - T_1)/T_1$ for 2, 3, 5, 10, 20, 30, 40 and 50 dimensions are plotted. From Table 4-11 and Fig. 4-8, it is observed the algorithm cost of DMS-L-PSO does not increase with increase in dimension.

Fig. 4-8 Algorithm Cost Change with Increase in Dimension.

4.2.4 Comparison

i. Algorithms:

Algorithms involved in the comparison:

- ✧ BLX-GL50 [129]:
Hybrid Real-Coded Genetic Algorithms with Female and Male Differentiation
- ✧ BLX-MA [130]:
Adaptive Local Search Parameters for Real-Coded Memetic Algorithms
- ✧ CoEVO [131]:
Mutation Step Co-evolution
- ✧ DE [132]:
Differential Evolution

CHAPTER 4 DYNAMIC MULTI-SWARM PARTICLE SWARM OPTIMIZER

✧ DMS-L-PSO:

Dynamic Multi-Swarm Particle Swarm Optimizer with Local Search

✧ EDA [133]:

Estimation of Distribution Algorithm

✧ G-CMA-ES [134]:

A restart Covariance Matrix Adaptation Evolution Strategy with increasing population size

✧ K-PCX [135]:

A Population-based, Steady-State real-parameter optimization algorithm with parent-centric recombination operator, apolynomial mutation operator and a niched-selection operation.

✧ L-CMA-ES [136]:

A restart local search Covariance Matrix Adaptation Evolution Strategy

✧ L-SaDE [137]:

Self-adaptive Differential Evolution algorithm with Local Search

✧ SPC-PNX [138]:

A steady-state real-paramter GA with PNX crossover operator

ii. Comparison Results for 10-D

In the comparison, only the problems for which at least one algorithm succeeded once are considered. Thus for 10-D, only problems 1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12 and 15 are considered. The success rates of the 11 algorithms on these 12 problems are listed in Table 4-12. The empirical distribution (success rate vs SP/SP_{best}) over all successful functions for 10-D is presented in Fig. 4-9 and the detailed values are listed in Table 4-13. SP here means the Success Performance for each problem. $SP = \text{mean (FES for successful runs)} * (\# \text{ of total runs}) / (\# \text{ of successful runs})$. SP_{best} is the minimal FES of all algorithms for each problem.

CHAPTER 4 DYNAMIC MULTI-SWARM PARTICLE SWARM OPTIMIZER

Fig. 4-9 Empirical Distribution over all Functions for 10-D.

Table 4-12 Success Rates of the 11 Algorithms for 10-D

Alg. Func.	BLX-GL	BLX-MA	CoEVO	DE	DMS	EDA	G-CMA	K-PCX	L-CMA	L-SaDE	SPC-PNX
1	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
2	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
3	0%	0%	100%	80%	100%	92%	100%	0%	100%	64%	0%
4	100%	0.96	100%	100%	4%	100%	100%	84%	28%	96%	100%
5	100%	0%	0%	100%	100%	100%	100%	0%	100%	0%	100%
6	100%	0%	0%	96%	100%	88%	100%	40%	100%	100%	0%
7	36%	0%	0%	6%	16%	4%	100%	20%	100%	24%	4%
9	12%	72%	0%	44%	100%	0%	76%	96%	0%	100%	4%
10	0%	0%	0%	0%	0%	0%	92%	88%	0%	0%	0%
11	0%	0%	0%	48%	0%	12%	24%	0%	0%	0%	0%
12	52%	0%	0%	76%	80%	40%	88%	0%	48%	100%	0%
15	0%	20%	0%	4%	84%	0%	0%	0%	0%	92%	0%

CHAPTER 4 DYNAMIC MULTI-SWARM PARTICLE SWARM OPTIMIZER

Table 4-13 Normalized Success Performance (SP/SPbest) for 10-D

Algorithms		BLX-GL	BLX-MA	CoEVO	DE	DMS	EDA	G-CMA	K-PCX	L-CMA	L-SaDE	SPC-PNX
Func	SR SP _{best}	50%	32.33%	33.33%	62.83%	65.33%	53%	81.67%	44%	56.33%	64.67%	34%
1	1610	11.52	7.2906	14.142	18.267	4.3366	6.1863	1	9.6329	1.0807	6.2894	4.1771
2	2380	17.052	15.377	11.501	19.457	4.6807	4.4538	1	14.348	1.0966	4.3013	13.03
3	6500	-	-	6.7126	17.695	1.7943	2.4462	1	-	1.0523	8.0471	-
4	2900	14.332	25.705	16.336	18.059	767.86	3.9655	1	19.588	66.552	15.724	10.591
5	5850	4.8126	-	-	6.9651	14.778	4.2906	1	-	1.0017	-	6.8819
6	9130	5.7032	-	-	5.1915	5.8195	7.4699	1.1829	23.722	1	5.3425	-
7	4670	12.374	-	-	256.96	113.77	404.71	1	38.048	1.1777	36.824	386.15
9	17048	10.004	5.7188	-	10.371	2.0118	-	4.4404	2.8875	-	1	64.189
10	54991	-	-	-	-	-	-	1.182	1	-	-	-
11	188522	-	-	-	1	-	2.9174	1.3951	-	-	-	-
12	28974	3.4206	-	-	2.4817	1	1.2218	1.1286	-	3.2615	1.1021	-
15	33165	-	8.3252	-	74.175	1.546	-	-	-	-	1	-

iii. Comparison Results for 30-D

Only problems 1, 2, 3, 4, 5, 6, 7, 9, 10, 11 and 12 have been solved at least once for all algorithms thus for 30-D case, only these 11 problems are considered in the comparison. The success rates of all the eleven algorithms on the successful problems are listed in Table 4-14. The empirical distribution over all successful functions for 30-D is presented in Fig. 4-10 and the detailed values are listed in Table 4-15.

Table 4-14 Success Rates of the 11 Algorithms for 30-D

Alg. Func.	BLX-GL	BLX-MA	CoEVO	DE	DMS	EDA	G-CMA	K-PCX	L-CMA	L-SaDE	SPC-PNX
1	100%	100%	12%	100%	100%	100%	100%	100%	100%	100%	100%
2	100%	0%	32%	0%	100%	100%	100%	0%	100%	96%	88%
3	0%	0%	0%	0%	88%	100%	100%	0%	100%	0%	0%
4	0%	0%	0%	0%	0%	100%	40%	0%	0%	52%	76%
5	0%	0%	0%	0%	0%	0%	100%	0%	100%	0%	0%
6	100%	0%	0%	0%	96%	0%	100%	0%	100%	0%	4%
7	100%	0%	44%	88%	96%	100%	100%	44%	100%	80%	64%
9	0%	36%	0%	0%	100%	0%	36%	72%	0%	100%	0%
10	0%	0%	0%	0%	0%	0%	12%	56%	0%	0%	0%
11	0%	0%	0%	0%	0%	0%	4%	0%	0%	0%	0%
12	0%	0%	0%	0%	20%	0%	32%	0%	0%	0%	0%

CHAPTER 4 DYNAMIC MULTI-SWARM PARTICLE SWARM OPTIMIZER

Fig. 4-10 Empirical Distribution over all Functions for 30-D

Table 4-15 Normalized Success Performance (SP/SP_{best}) for 30-D

Algorithms		BLX-GL	BLX-MA	CoEVO	DE	DMS	EDA	G-CMA	K-PCX	L-CMA	L-SaDE	SPC-PNX
Func.	<div>SR SP_{best}</div>	36.36%	12.36%	8%	17.09%	54.55%	45.45%	65.82%	24.73%	54.55%	38.91%	30.18%
1	1475.3	39.423	21.472	932.97	93.912	3.3139	101.67	3.0502	1	3.24	13.715	20.556
2	13000	12.274	-	64.365	-	9.5139	12.308	1	-	1.0462	11.448	24.258
3	42700	-	-	-	-	7.8644	5.0351	1	-	1.0164	-	Inf
4	59000	-	-	-	-	-	3.3729	1	-	-	-	6.1583
5	65900	-	-	-	-	-	-	1	-	-	-	Inf
6	60000	3.5854	-	-	-	5.369	-	1	-	-	-	86.755
7	6110	10.073	-	93.169	32.655	9.7578	21.44	1	100.94	1.1457	22.057	60.66
9	98934	-	-	-	-	2.1065	-	7.9851	3.3533	-	1	-
10	448940	-	-	-	-	-	-	5.3905	1	-	-	-
11	4980000	-	-	-	-	-	-	1	-	-	-	-
12	225000	-	-	-	-	7.034	-	1	-	-	-	-

From the comparison results of the 11 algorithms on the 25 problems for 10-D and 30-D, DMS-L-PSO achieved 65.33% and 54.55% success rate for all successful functions for 10-D and 30-D respectively and the best algorithm

CHAPTER 4 DYNAMIC MULTI-SWARM PARTICLE SWARM OPTIMIZER

G-CMA-ES achieved 81.67% and 65.82% respectively. Though not the best one, DMS-L-PSO is highly competitive algorithm among so many popular good algorithms, especially on 30- D problems.

4.3 Conclusion

In this chapter, a Dynamic Multi-Swarm Optimizer with local search was presented. This algorithm is constructed based on the local version of PSO and employs a periodically changing neighborhood topology. The population is divided into several sub-swarms and the sub-swarms are regrouped every R generations. In this way, the information obtained in the search process can be exchanged in the sub-swarms and lead the sub-swarms fly to the good region.

In order to increase the diversity of the particles and emphasize the action of the historical information of each particle, an adaptive self-learning strategy is introduced. The self-learning strategy can be seen as a variant of the comprehensive learning strategy. With the self-adaptive learning probability P_c , the particle can randomly decide which dimension employs the information of the **pbest** and which dimension follows **lbest**.

With the dynamic multi-swarm structure and self-learning strategy, the DMS-PSO possesses a good diversity and good global search ability. In order to improve the local search ability and speed up the convergence after finding the global region, the BFGS Quasi-Newton local search method is combined in the DMS-PSO. Every L generations, some **pbests** will be randomly chosen as the starting points to start the local search. The **pbests** nearest to the refined solution will be replaced.

Experiments were conducted on the 25 standard benchmark functions proposed in CEC'05. The comparison with ten other good evolutionary algorithms was presented and the results showed though not the best algorithm, DMS-L-PSO performs well. Success rates of 65.33% and 54.55% were achieved for all successful functions for 10- D and 30- D respectively.

In the next two chapters, DMS-PSO will be extended to solve constrained problems and multi-objective problems.

Chapter 5

Dynamic Multi-Swarm Particle Swarm Optimizer for Constrained Optimization

In this Chapter, the Dynamic Multi-Swarm Particle Swarm Optimizer with local search (DMS-L-PSO) which was designed to solve single objective bound constrained problems in Chapter 4 will be extended to solve single objective optimization problems with general constraints. The constrained optimization is briefly introduced and followed by a review of the constraint-handling methods. The DMS-C-PSO is described and discussed. Experiments are conducted on the standard benchmark functions proposed in CEC'06 and comparison results with other evolutionary algorithms are presented.

5.1 Constrained Optimization

Optimization of constrained problems is an important area in the optimization field since most optimization problems have constraints of different types due to the physical, geometric and other limitations. In general, the constrained problems can be transformed into the following form:

$$\begin{aligned} \text{Minimize} \quad & f(\mathbf{x}), \quad \mathbf{x} = [x_1, x_2, \dots, x_D], \mathbf{x} \in [\mathbf{X}_{\min}, \mathbf{X}_{\max}]^D \quad (5-1) \\ \text{subject to:} \quad & g_j(\mathbf{x}) \leq 0, j = 1, \dots, J \\ & h_k(\mathbf{x}) = 0, k = 1, \dots, K \end{aligned}$$

If we denote with $\mathcal{F}$ the *feasible region* and $\mathcal{S}$ the whole search space, $\mathbf{x} \in \mathcal{F}$ if $\mathbf{x} \in \mathcal{S}$ and all constraints are satisfied. In this case, $\mathbf{x}$ is called a *feasible solution*. At a point $\mathbf{x}$, we call the constraints $g_j(\mathbf{x})$ which satisfy $g_j(\mathbf{x}) = 0$ as *active constraints* at $\mathbf{x}$. Equality constraints $h_k(\mathbf{x})$ are active at all feasible solutions.

Usually equality constraints are transformed into inequality constraints of the

CHAPTER 5 DMS-PSO FOR CONSTRAINED OPTIMIZATION

form

$$|h_k(\mathbf{x})| - \varepsilon \leq 0, \text{ for } k = 1, \dots, K \quad (5-2)$$

Then the formula of the constrained problem can be simplified to:

$$\begin{aligned} \text{Minimize} \quad & f(\mathbf{x}), \quad \mathbf{x} = [x_1, x_2, \dots, x_D], \mathbf{x} \in [\mathbf{X}_{\min}, \mathbf{X}_{\max}]^D \\ \text{subject to:} \quad & G_i(\mathbf{x}) \leq 0, i = 1, \dots, m, \text{ and } m = J + K \end{aligned} \quad (5-3)$$

here $G_i(\mathbf{x}) = g_i(\mathbf{x}), i = 1, \dots, J$ and $G_i(\mathbf{x}) = |h_k(\mathbf{x})| - \varepsilon, i = J + 1, \dots, J + K$

A solution is regarded as feasible if $G_i(x) \leq 0$, for $j = 1, \dots, J + K$. In this thesis, ε is set to 10^{-4} .

Additional to the difficulties existing in the single objective optimization problem with only bound constraints, a major difficulty for constrained optimization is to find feasible regions. The feasible space can be composed by functions with different properties and can be discrete. Since many evolutionary algorithms are developed as unconstrained search techniques, when they are used to solve constrained problems, an additional mechanism is required to be incorporated into the fitness function or the evolution strategies to guide the search direction. Thus a variety of constraint-handling techniques have been proposed to be combined with the existing unconstrained optimization algorithms to solve the constrained problems.

5.2 Constraint-Handling Techniques

The constraint-handling techniques can be classed into the following three categories:

5.2.1 Penalty Functions:

The penalty functions, which were proposed in the 1940s [142] and later expanded by many researchers, use the amount of constraint violation to penalize the infeasible solutions and to favor feasible solutions as following:

$$\text{fitness}(\mathbf{x}) = f(\mathbf{x}) + P(\mathbf{x}) \quad (5-4)$$

There are different ways to design the penalty function $P(\mathbf{x})$:

i. Static Penalties

The penalty coefficients are predefined. For example, in [141], some levels of

CHAPTER 5 DMS-PSO FOR CONSTRAINED OPTIMIZATION

violation are predefined and penalty coefficients are pre-chosen for each level. The weakness of this approach is that too many parameters need to be adjusted and predefined. Sometimes a proper penalty coefficient is difficult to locate due to the complexity of the constraint functions.

ii. Dynamic Penalty

In the dynamic penalty method, the penalties are dynamically increased with the generation. The penalty is small in the beginning of the search but large in the end to guide the population into the feasible region. The penalty function proposed in [143] by Joines and Houch and the annealing penalty proposed by Michalewicz and Attia in [144] are in this category.

iii. Adaptive Penalty

Since penalty coefficients are difficult to fix, some researchers proposed adaptive penalty with which the penalty can be dynamically modified according to feedback from the search process [145][146][149].

Although penalty function is the most popular and simple approach, it has some limitations [150]. In order to overcome these limitations, many alternative methods are proposed

5.2.2 Superiority of Feasible Solutions

i. Start with a Population of Feasible Individuals

The GENOCOP (Genetic algorithm for Numerical Optimization for Constrained problems) [147] needs a feasible starting point to start the search. In [148], the search was started with a group of feasible solutions and only feasible solutions in the search process were kept. Obviously, the weakness of this kind of method is that the user must find a way to find the initial feasible population. For the constrained problems which have only small feasible range, the feasible solution is difficult to find. And it is also a challenge to this kind of method when there are many discrete feasible ranges.

ii. Feasible Favored comparing criterion

For two solution $\mathbf{x}_1$ and $\mathbf{x}_2$, $\mathbf{x}_1 > \mathbf{x}_2$ if

- a. $\mathbf{x}_1$ and $\mathbf{x}_2$ are infeasible, $V(\mathbf{x}_1) < V(\mathbf{x}_2)$
- b. $\mathbf{x}_1$ is feasible and $\mathbf{x}_2$ is infeasible

CHAPTER 5 DMS-PSO FOR CONSTRAINED OPTIMIZATION

c. $\mathbf{x}_1$ and $\mathbf{x}_2$ are feasible, $f(\mathbf{x}_1) < f(\mathbf{x}_2)$

Here $V(\mathbf{x})$ is the overall violation function which is used to evaluate the violation

of the constraints. $V(\mathbf{x}) = \sum_{i=1, G_i(\mathbf{x}) > 0}^m G_i(\mathbf{x})$ and $V(\mathbf{x}) = \max(\max(G_i(\mathbf{x})), 0)$ are

two commonly employed formats. $V(\mathbf{x})$ can also be replaced by other rank objective. If the feasible favored comparison criterion is used as selection operator, when the whole population is feasible, all individuals will be kept feasible in the search process and all infeasible offspring solutions will be rejected.

This kind of comparison criteria has been used by many researchers in their works in the past few years [151][152][153]. Some researchers extended it with a generation decreasing ε to make the comparison criteria dynamically changed [154][155][156]. With the generation decreasing ε , the feasible region is first enlarged and then shrinks to the real feasible region to the feasible favor constraint and guide the population into the region near the real feasible region.

iii. Specially Designed Operators

The GENOCOP [147] used specialized operators to transform infeasible individuals to feasible individuals. In [157], through designing a proper transactions method to map a decoded solution to a feasible solution, the original problems were transformed into other problems which are easier to solve.

5.2.3 Separation of Objective and Constraints

i. Stochastic Ranking

Different from the methods which always consider feasible solutions better, in Stochastic Ranking [158], there is probability P_f which decides whether to compare the individuals using objective function or constraint functions. It balances the influence of the penalty of the objective and the penalty function in determining the rank of the population.

ii. Co-evolution Methods

In HCVEGA [159][140], the population was split into $m+1$ subpopulations of equal size. One subpopulation handles the objective function and the other m subpopulations use one constraint function as its fitness function, chosen from the objective function and constrained functions. In this way, the whole population

CHAPTER 5 DMS-PSO FOR CONSTRAINED OPTIMIZATION

moves to the feasible region with low objective function value for minimization problem.

iii. Multi-Objective Optimization Techniques

Constrained optimization problem has the same property as the multi-objective optimization in that they all have more than one function to be considered. Then it is natural to use the multi-objective techniques to solve constrained problems. The objective and the constraints are divided into two objectives to form a bi-objective problem in [160]. The Pareto ranking which is always used in multi-objective optimization is used in [161]. Pareto dominance selection is proposed by Coello in [162]. A comparison of some multiobjective-based constraint-handling methods in EA can be found in [163].

5.3 DMS-L-PSO with a New Constraint-Handling

Mechanism (DMS-C-PSO)

5.3.1 The Novel Constraint-Handling Mechanism

As other evolutionary algorithm, particle swarm optimizer is first designed to solve unconstrained problems. And then some researchers extended it to solve constrained problems. However, this field has not attracted enough attention. The current approaches just combine the existing constraint-handling methods with PSO. In [73], the search was started with a group of feasible solutions and only feasible solutions in the search process were kept; [74] employed a simple penalty function; [75] compared the preservation of feasible solutions method and penalty function method; [76] used subpopulation and use MOPSO in each subpopulation; [78] preferred the feasible solution to the infeasible solution and sorted the feasible solutions according to the objective function while it sorted the infeasible solutions according to the sum of constraints violation.

The original DMS-PSO is designed to handle unconstrained optimization problems, thus in order to handle constrained problems, a constraint-handling mechanism is required to guide the swarm to search for feasible region. Based on its multi-swarm property, it is natural to let the sub-swarms take charge of different tasks. The constraint-handling approach used in HCVEGA is a good reference,

CHAPTER 5 DMS-PSO FOR CONSTRAINED OPTIMIZATION

where each subpopulation tries to evolve along one single constraint or the objective function and each individual in a subpopulation is allowed to mate with any other in any subpopulation. In this way, it is expected to have population of feasible individuals with high fitness values, but it has a main drawback that the number of subpopulations needs to increase linearly with the number of constraints of the problem. In order to overcome this weakness, in the novel constraint-handling mechanism, the objective and constraints are assigned to the sub-swarms adaptively according to the difficulties of the constraints.

Suppose that there are m constraints, the population is divided into n sub-swarms with 3 members in each sub-swarm and the population size is ps ($ps = n*3$). n is a positive integer and ' $n = m$ ' is not required.

$$\text{Define} \quad a > b = \begin{cases} 1 & \text{if } a > b \\ 0 & \text{if } a \leq b \end{cases} \quad (5-5)$$

$$p_i = \frac{\sum_{j=1}^{ps} (G_i(\mathbf{x}_j) > 0)}{ps}, \quad i = 1, 2, \dots, m \quad (5-6)$$

$$fp = 1 - \bar{\mathbf{p}}, \quad \mathbf{p} = [p_1, p_2, \dots, p_m] \quad (5-7)$$

$$gp_i = p_i / m \quad (5-8)$$

$$\text{thus} \quad fp + \sum_{i=1}^m gp_i = 1 \quad (5-9)$$

For each sub-swarm, the first step is using roulette selection according to fp and gp_i to assign the objective function or a single constraint as its target. If sub-swarm i is assigned to improve constraint j , set $obj(i) = j$ and if sub-swarm i is assigned to improve the objective function, set $obj(i) = 0$. Since gp_i reflects the difficulty of the constraint i , the more difficult constraints will have more sub-swarms search for it. The second step is assigning swarm member for this sub-swarm. Sort the unassigned particles according to $obj(i)$, and assign the best and $sn-1$ worst particles to sub-swarm i . While comparing two particles i and j , the following comparison criteria is used:

- i. If $obj(i) = obj(j) = k$ (particle i and j handling the same constraint k), particle i wins if

CHAPTER 5 DMS-PSO FOR CONSTRAINED OPTIMIZATION

$$\begin{aligned}
& g_k(\mathbf{x}_i) < g_k(\mathbf{x}_j) \text{ and } g_k(\mathbf{x}_j) > 0 \\
& \text{or } V(\mathbf{x}_i) < V(\mathbf{x}_j) \& g_k(\mathbf{x}_i), g_k(\mathbf{x}_j) \leq 0 \\
& \text{or } f(\mathbf{x}_i) < f(\mathbf{x}_j) \& V(\mathbf{x}_i) = V(\mathbf{x}_j)
\end{aligned} \tag{5-10}$$

ii. If $\text{obj}(i) = \text{obj}(j) = 0$ (particle i and j handling $f(\mathbf{x})$), or $\text{obj}(i) \neq \text{obj}(j)$ (i and j handling different objectives), particle i wins if

$$\begin{aligned}
& V(\mathbf{x}_i) < V(\mathbf{x}_j) \\
& \text{or } f(\mathbf{x}_i) < f(\mathbf{x}_j) \& V(\mathbf{x}_i) = V(\mathbf{x}_j)
\end{aligned} \tag{5-11}$$

Here
$$V(\mathbf{x}) = \sum_{i=1}^m (\text{weight}_i \cdot g_i(\mathbf{x})) \tag{5-12}$$

$$\text{weight}_i = \frac{1 / g_{i\max}}{\sum_{i=1}^m (1 / g_{i\max})}, \quad i = 1, 2, \dots, m \tag{5-13}$$

The effect of **weight** is to balance the impacts of different constraints. It is experimentally proven that **weight** is very useful when the differences are too huge among the constraints. $g_{i\max}$ is the estimated maximum value of the constraint i and is updated periodically, thus the estimated $g_{i\max}$ becomes more and more accurate.

In this way, the constraints which are more difficult will have more sub-swarms work for it, while the easier ones will have less or even no sub-swarm working for it. And the search will focus on finding feasible solutions and then concentrate on improving the objective function. There will be more sub-swarms evolve along the fitness increasing direction if more constraints are satisfied. Every R generations, when the sub-swarms are regrouped, the objectives are reassigned for each sub-swarm. An illustration of the search behavior of a particle is given in Fig. 5-1. A particle i is first in a sub-swarm of which the objective is evolve to decrease $G_1(\mathbf{x})$, after that in the regrouping schedule, it is grouped into sub-swarm of which the objective is $G_2(\mathbf{x})$, then $G_3(\mathbf{x})$, then $G_1(\mathbf{x})$ again. A little by a little, the particle flies to the feasible region and it gets a new objective, optimizing the objective function $f(\mathbf{x})$.

CHAPTER 5 DMS-PSO FOR CONSTRAINED OPTIMIZATION

Fig. 5-1 Illustration of the Search Behavior of a Particle *i*

5.3.2 Find the Relevant Dimensions for Each Objective

Sometimes the objective and constraints functions concerns different variables. For example, for a five dimensional constrained problem, $f(\mathbf{x}) = f(\mathbf{x}_1, \mathbf{x}_2)$, $G_1(\mathbf{x}) = G_1(\mathbf{x}_1, \mathbf{x}_3)$, $G_2(\mathbf{x}) = G_2(\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4)$, $G_3(\mathbf{x}) = G_3(\mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_5)$. In this case, the relevant dimensions for the objective function $f(\mathbf{x})$ is [1, 2], and the relevant dimensions for $G_1(\mathbf{x})$, $G_2(\mathbf{x})$ and $G_3(\mathbf{x})$ are [1, 3], [2, 3, 4], [1, 3, 5] respectively. The relevant matrix for this problem will be:

$$\mathbf{M} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Under such condition, moving all dimensions when the sub-swarm serves for only one objective is not necessary and unreasonable. Thus, in DMS-L-PSO for constrained problems, only the relevant dimensions are considered in the evolution.

In the initialization step, we randomly generate a vector $\mathbf{x}$ within the search range. For each dimension, we fix the other dimensions except the concerned

CHAPTER 5 DMS-PSO FOR CONSTRAINED OPTIMIZATION

dimension, generate five random values within the search range for this dimension. Thus, five different solutions are obtained which are different in only one dimension. Calculate $f(\mathbf{x})$, $G_i(\mathbf{x}), i=1, \dots, m$ for these five solutions. For an objective, if the five solutions have different results, then this dimension is the relevant dimension for that objective. In this way, estimated relevant dimensions will be obtained and this method will consume $5*m*D$ fitness evaluations. According to the experiments done by the author, this method is efficient and reliable.

With the obtained relevant matrix, in the position updating step, only the velocities and positions of the relevant dimensions for the assigned objective will be updated, while the dimensions other than the relevant dimensions maintain the old values.

5.3.3 Local Search Phase

In the local search phase, Sequential Quadratic Programming (SQP) method is employed. At each major generation, a Quadratic Programming (QP) sub-problem is solved. An active set strategy is used in this method. The Hessian matrix is estimated using the updating strategy with the BFGS formula, the same as the BFGS Quasi-Newton method employed in the Chapter 4. And in each generation, the line search with the merit function is used. The details will not be discussed in this thesis. In the experiment part, "*fmincon*" function in Matlab 6.5 is used.

5.3.4 DMS-L-PSO with a New Constraint-Handling Mechanism

With the novel constraint-handling mechanism, relevant dimensions and the SQP local search, DMS-L-PSO is extended to solve constrained optimization problem. The new algorithm is denoted as DMS-C-PSO to distinguish the two algorithms. The whole flow of DMS-C-PSO is described as below:

n : number of swarms	ns : Each swarm's population size, $ns = 3$
ps : population size, $ps = n * ns$	R : Regrouping period
L : local refining period	FEs : fitness evaluations used.
L_FEs : Max FEs using in the local search	
Max_FEs : Max fitness evaluations, stop criterion	

CHAPTER 5 DMS-PSO FOR CONSTRAINED OPTIMIZATION

Step 1: Initialization -

Initialize ps particles (position $\mathbf{X}$ and velocity $\mathbf{V}$), calculate $f(\mathbf{X})$, $G_j(\mathbf{X})$ ($j = 1, 2, \dots, m$) for each particle. Set $\mathbf{pbest} = \mathbf{X}$ and initialize g_{imax} for each constraint. Find the relevant dimensions for each objective. Set $FES = ps + 5 * m * D$.

Step 2: Divide the population into sub-swarms -

Update g_{imax} for each constraint and calculate **weight** according to Eqn. (5-13).

Calculate fp and gp according to Eqn. (5-7) and Eqn. (5-9).

For each sub-swarm i , assign $obj(i)$ using roulette selection according to fp and gp . Then sorting the unassigned particles according to $obj(i)$ and assigning the best and $sn-1$ worst particles to sub-swarm i .

Update $\mathbf{Pc}$ as DMS-L-PSO, the only difference is making sure that there is at least one dimension of the relevant dimension will not learn from itself other than randomly assign one dimension in the D dimension to learn from the local best. Set $\mathbf{Pc}_s = \emptyset$.

Step 3: Update the particles -

For each particle, find its local best $\mathbf{lbest}_i$ and update the position $\mathbf{X}_i$ and velocity $\mathbf{V}_i$ the same as DMS-L-PSO. If the particle flies out of the search range, assign the dimensions which overstep the bounds in a random position between the old position and bounds.

Calculate $f(\mathbf{X}_i)$, $g_j(\mathbf{X}_i)$ ($j = 1, 2, \dots, m$), update $fitcount$. Compare $\mathbf{X}_i$ and $\mathbf{pbest}_i$ using Eqn. (5-10) and Eqn. (5-11) introduced in Section 5.3.1. If $\mathbf{X}_i$ wins, update $\mathbf{pbest}_i$ and add Pc_i into the $\mathbf{Pc}$ archive $\mathbf{Pc}_s$. And execute the same comparison for $\mathbf{X}_i$ and $\mathbf{lbest}_k$.

Step 4: Regroup -

Every R generations, go to Step 2.

Step 5: Local Search Phase -

Every L generations, randomly choose 5 particles' $\mathbf{pbest}$ and start local search with SQP method using these solutions as start points and L_FES as the fitness evaluations. Update FES . Compare the result $\mathbf{x}$ with the start point $\mathbf{pbest}_i$. If $\mathbf{x}$ wins, update that $\mathbf{pbest}_i$.

Step 6: If $FES \leq 0.7 * Max_FES$, go to Step 3. Otherwise go to Step 7.

CHAPTER 5 DMS-PSO FOR CONSTRAINED OPTIMIZATION

Step 7: Global Search Phase:

Set $n = 1$, $ns = ps$, and $obj = 0$, $Pc_i = 0$ for all particles, continue search using one swarm. Every L generations, start local search using **gbest** as start points using $5*L_FES$ as the max fitness evaluations. Update FES . Stop search if $FES \geq Max_FES$.

5.4 Experiments

Experiments are conducted on the 24 constraint real parameter problems provided in [139]. The formulas of the test problems are listed in Appendix B. The codes can be downloaded from <http://www.ntu.edu.sg/home/EPNSugan/>.

5.4.1 Evaluation Criteria and Parameters Setting

Problems: 23 minimization problems with constraints (g20 is excluded since there is no feasible solution for g20.)

Runs / problem: 25

Max_FES: 500, 000

Initialization: Uniform random initialization within the search space,

Feasible Run: A run during which at least one feasible solution is found in Max_FES.

Successful Run: A run during which the algorithm finds a feasible solution x satisfying $(f(x) - f(x^*)) \leq 0.0001$.

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

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

Success Performance = mean (FES for successful runs) * (# of total runs) / (# of successful runs)

Parameters Setting for DMS-C-PSO:

a) All parameters to be adjusted:

ω , c_1 , c_2 , **Vmax**, n , R , L , L_FES

b) Corresponding dynamic ranges:

All parameters are fixed except n .

c) Guidelines on how to adjust the parameters

n can be set small for simple problems and big for complex problems.

CHAPTER 5 DMS-PSO FOR CONSTRAINED OPTIMIZATION

d) Actual parameter values used.

$\omega = 0.729, \quad c_1 = c_2 = 1.49445; \quad \mathbf{Vmax} = 0.5 * (\mathbf{Xmax} - \mathbf{Xmin})$

$n = 20; \quad R = 100; \quad L = 500$

$L_{FES} = 1,000; \quad Max_{FES} = 500,000$

5.4.2 Results of DMS-C-PSO

The best functions error values achieved when FEs = 5e+3, FEs = 5e+4, FEs = 5e+5 for the 24 test functions are presented in Table 5-1 to Table 5-4. Success Rate and Successful Performance are listed in Table 5-5.

Table 5-1 Error Values Achieved When FEs = 5e+3, FEs = 5e+4, FEs = 5e+5 for Problems 1-6

FES	Func.	g01	g02	g03	g04	g05	g06
5e+3	Best	1.7561e+1(0)	2.5311e-1(0)	2.5721e-1 (0)	5.9512e+1(0)	9.5724e+1(2)	5.8572e+1(0)
	Median	8.7261e+0(2)	3.5324e-1(0)	3.9824e-1 (0)	9.5960e+1(0)	1.2988e+2(3)	2.3912e+2(0)
	Worst	1.2847e+0(5)	6.5243e-1(0)	5.2841e-1 (0)	3.5341e+2(0)	1.8341e+2(3)	8.9534e+2(0)
	c^1	1 2 2	0 0 0	0 0 0	0 0 0	2 3 3	0 0 0
	$\bar{v}^2$	2.1823e-1	0	0	0	9.5024e+0	0
	Mean	8.6254e+0	4.1612e-1	3.5821e-1	1.0024e+2	1.1352e+2	2.5759e+2
	Std	2.9741e+0	3.8512e-2	1.0184e-1	1.4024e+2	2.0951e+2	3.1502e+2
5e+4	Best	1.1234e-11(0)	9.5224e-1(0)	0(0)	1.4953e-9 (0)	8.0485e-9 (0)	1.0585e-8 (0)
	Median	1.9512e-11(0)	1.2852e-2(0)	0(0)	2.9672e-9 (0)	7.9502e-8 (0)	1.2857e-7 (0)
	Worst	3.9512e-11(0)	4.5212e-2(0)	0(0)	3.5479e-7 (0)	2.5821e-7 (0)	3.9581e-7 (0)
	c	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0
	$\bar{v}$	0	0	0	0	0	0
	Mean	2.5661e-11	2.0521e-2	0	2.9673e-9	7.3905e-8	1.29582e-7
	Std	3.5235e-12	1.1352e-2	0	3.5965e-8	1.5760e-8	2.4871e-8
5e+5	Best	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)
	Median	0(0)	0 (0)	0(0)	0(0)	0(0)	0(0)
	Worst	0(0)	1.8352e-2(0)	0(0)	0(0)	0 (0)	0(0)
	c	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0
	$\bar{v}$	0	0	0	0	0	0
	Mean	0	1.7451e-3	0	0	0	0
	Std	0	3.9512e-3	0	0	0	0

c^1 is the number of violated constraints at the median solution: the sequence of three numbers indicates the number of violations (including inequality and equalities) by more than 1.0, more than 0.01 and more than 0.0001 respectively.

$\bar{v}^2$ is the mean value of the violations of all constraints at the median solution. The numbers in the parenthesis after the fitness value of the best, median, worst solution are the number of constraints which cannot satisfy feasibility condition at the best, median and worst solutions respectively. Due to the accuracy of the given optimal value $f(\mathbf{x}^*)$, when $(f(\mathbf{x}) - f(\mathbf{x}^*)) < 1e-10$ the final errors are reported as 0.

CHAPTER 5 DMS-PSO FOR CONSTRAINED OPTIMIZATION

Table 5-2 Error Values Achieved When $FES = 5e+3$, $FES = 5e+4$, $FES = 5e+5$ for Problems 7-12

Func.		g07			g08			g09			g10			g11			g12		
FES		g07			g08			g09			g10			g11			g12		
5e+3	Best	1.2947e+1(0)			0(0)			2.4857e+0(0)			3.5958e+3(0)			1.9274e-6 (0)			1.9475e-8(0)		
	Median	1.8576e+2(0)			9.0751e-7 (0)			1.8724e+1(0)			2.5857e+3(1)			4.5972e-5 (0)			3.4957e-4(0)		
	Worst	9.6736e+2(2)			1.2841e-3 (0)			1.2752e+2(0)			2.4058e+1(2)			2.5988e-3 (0)			2.4856e-2(0)		
	c	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
	$\bar{v}$	0			0			0			1.5579e-2			0			0		
	Mean	1.5987e+2			2.6883e-4			2.9475e+1			6.9441e+3			9.5834e-5			2.4971e-3		
	Std	1.9057e+2			9.6731e-4			4.5958e+1			3.7619e+3			2.8573e-3			3.5865e-3		
5e+4	Best	0(0)			0(0)			0(0)			8.4093e-8 (0)			0 (0)			0(0)		
	Median	0(0)			0(0)			0(0)			1.1546e-7 (0)			1.2947e-7 (0)			0(0)		
	Worst	0 (0)			0 (0)			0 (0)			1.4053e-7 (0)			8.5974e-5 (0)			0(0)		
	c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	$\bar{v}$	0			0			0			0			0			0		
	Mean	0			0			0			1.1714e-7			9.8461e-6			0		
	Std	0			0			0			1.4908e-8			9.4582e-6			0		
5e+5	Best	0(0)			0(0)			0(0)			0(0)			0(0)			0(0)		
	Median	0(0)			0(0)			0(0)			1.0124e-8 (0)			0(0)			0(0)		
	Worst	0 (0)			0 (0)			0 (0)			3.4056e-8 (0)			0 (0)			0(0)		
	c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	$\bar{v}$	0			0			0			0			0			0		
	Mean	0			0			0			1.1471e-8			0			0		
	Std	0			0			0			8.8141e-9			0			0		

Table 5-3 Error Values Achieved When $FES = 5e+3$, $FES = 5e+4$, $FES = 5e+5$ for Problems 13-18

Func.		g13			g14			g15			g16			g17			g18		
FES		g13			g14			g15			g16			g17			g18		
5e+3	Best	3.4987e-1(2)			-4.2947e+1(3)			2.9480e+0(1)			1.2947e-1(0)			3.081e+2(3)			1.2948e+0(9)		
	Median	2.9572e-1(3)			-1.2947e+2(3)			-3.8841e-2(2)			2.3947e-1(0)			1.2842e+2(4)			2.0812e+0(10)		
	Worst	9.5921e-1(3)			-2.987e+2(3)			2.0394e+0(2)			3.9712e-1(1)			2.4029e+2(4)			-2.1021e+0(9)		
	c	0	2	3	3	3	3	0	2	2	0	0	0	3	4	4	9	10	10
	$\bar{v}$	2.4957e-1			3.0891e+0			1.5864e-1			0			3.1021e+1			5.2012e+0		
	Mean	2.3875e-1			-1.4888e+2			2.0582e+0			2.1000e-1			1.0284e+2			5.1510e-1		
	Std	4.2974ee-1			3.9899e+1			1.6041e+0			1.2746e-1			8.0192e+1			2.3077e+0		
5e+4	Best	0(0)			0(0)			0(0)			2.3947e-11 (0)			7.4058e+1(0)			0(0)		
	Median	2.4871e-6(0)			0(0)			0(0)			3.3737e-8(0)			7.4058e+1(0)			0(0)		
	Worst	2.8460e-2(0)			0 (0)			3.4857e-6 (0)			1.8409e-2(0)			7.4058e+1(0)			1.009e-1(0)		
	c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	$\bar{v}$	0			0			0			0			0			0		
	Mean	4.2864e-3			0			1.2946e-7			3.9690e-3			7.4058e+1			1.2211e-2		
	Std	2.3871e-2			0			2.9925e-6			6.1081e-3			0			3.0579e-2		
5e+5	Best	0(0)			0(0)			0(0)			0(0)			7.4058e+1(0)			0(0)		
	Median	0(0)			0(0)			0(0)			0(0)			7.4058e+1(0)			0(0)		
	Worst	1.9374e-6 (0)			0 (0)			0 (0)			0 (0)			7.4058e+1(0)			0 (0)		
	c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	$\bar{v}$	0			0			0			0			0			0		
	Mean	9.9742e-8			0			0			0			7.4058e+1			0		
	Std	1.0241e-6			0			0			0			0			0		

CHAPTER 5 DMS-PSO FOR CONSTRAINED OPTIMIZATION

Table 5-4 Error Values Achieved When $FES = 5e+3$, $FES = 5e+4$, $FES = 5e+5$ for Problems 19-24

FES \ Func.		g19	g21	g22	g23	g24
5e+3	Best	9.2084e+1(0)	1.2048e+2(5)	6.0284e+3(18)	2.4124e+2(4)	2.2948e-4 (0)
	Median	2.3018e+2(0)	8.3985e+2(5)	7.9987e+3(19)	-2.4041e+2(6)	9.8712e-3 (0)
	Worst	4.2084e+2(0)	6.0812e+2(5)	1.1204e+4(19)	2.3314e+2(6)	2.3947e-2 (0)
	c	0 0 0	2 5 5	18 19 19	2 5 6	0 0 0
	$\bar{v}$	0	1.8982e+0	4.1028e+7	2.4919e+0	0
	Mean	3.3084e+2	3.2081e+2	9.9721e+3	-1.0414e+2	9.7591e-3
	Std	6.9274e+1	2.4058e+2	4.9251e+3	4.2301e+2	1.2982e-2
5e+4	Best	8.9912e-9(0)	1.2084e-7(0)	6.2941e+0 (5)	1.2094e-3(0)	0(0)
	Median	2.2914e-8(0)	2.5972e-3(0)	3.4018e+3(17)	9.4194e-3(0)	0(0)
	Worst	4.2010e-6(0)	2.3085e+1(0)	3.3021e+3(20)	9.9080e-2(0)	0 (0)
	c	0 0 0	0 0 0	3 7 17	0 0 0	0 0 0
	$\bar{v}$	0	0	1.2391e+1	0	0
	Mean	8.9274e-8	6.9274e+0	2.9913e+3	9.3218e-3	0
	Std	4.0812e-7	1.9813e+1	6.4141e+3	1.3141e-2	0
5e+5	Best	0(0)	9.9274e-7(0)	3.0492e-5 (0)	0 (0)	0(0)
	Median	0(0)	3.2048e-7(0)	1.2200e+2 (0)	9.2974e-9 (0)	0(0)
	Worst	0 (0)	2.4817e-5(0)	1.2200e+2 (0)	1.2491e-5 (0)	0 (0)
	c	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0
	$\bar{v}$	0	0	2.0481e+0	0	0
	Mean	0	5.2048e-7	1.1031e+2	4.4182e-7	0
	Std	0	2.9679e-6	3.2041e+1	3.5817e-5	0

Table 5-5 Number of FES to Achieve the Fixed Accuracy Level ($(f(\mathbf{x}) - f(\mathbf{x}^*)) \leq 0.0001$), Success Rate, Feasible Rate and Success Performance

Func.	Best	Median	Worst	Mean	Std	Feasible Rate	Success Rate	Success Performance
g01	22370	25643	46100	33250	10361	100%	100%	33250
g02	52565	86205	-	-	-	100%	96%	88201
g03	23429	24726	26389	24760	451.17	100%	100%	24760
g04	24955	24642	25191	25162	207.65	100%	100%	25162
g05	27406	28463	30315	28140	541.7	100%	100%	28140
g06	25362	26841	27787	27396	436.8	100%	100%	27396
g07	24308	26083	27205	26351	429.69	100%	100%	26351
g08	1557	3883	7720	3919.4	1650.1	100%	100%	3919.4
g09	28630	29370	28741	28808	133.21	100%	100%	28808
g10	23890	25101	25536	25086	530.36	100%	100%	25086
g11	953	12913	28660	14395	11766	100%	100%	14395
g12	786	6695	8937	5310.7	2818.3	100%	100%	5310.7
g13	27896	29058	227583	40282	42387	100%	100%	40282
g14	21420	23889	49362	24474	5332	100%	100%	24474
g15	27465	28949	28926	28727	346.08	100%	100%	28727
g16	26122	27562	108041	53378	31902	100%	100%	53378
g17	-	-	-	-	-	100%	0%	-
g18	27132	27978	89347	31737	14210	100%	100%	31737
g19	20190	21569	23651	20810	884.59	100%	100%	20810
g21	26078	144142	256269	139228	69729	100%	100%	139228
g22	-	-	-	-	-	100%	0%	-
g23	51734	191425	487621	204741	97241	100%	100%	204741
g24	13290	18138	25784	18750	3382.1	100%	100%	18750

From the results, we can observe that among those 24 test functions, g08, g11 and g12 are comparatively easy, and the success performances of DMS-C-PSO are under 5000 while g02, g13, g17, g20 and g21 are comparatively difficult. DMS-C-PSO found feasible solutions for all problems and achieved 100% success rate for most problems except g2, g17, and g22. From the constraint violation convergence graphs, DMS-C-PSO found feasible solutions efficiently except for g20 and g22.

CHAPTER 5 DMS-PSO FOR CONSTRAINED OPTIMIZATION

Comparing with the previous results provided in [140], it is obvious that the novel constraint handling mechanism performs very well on finding feasible regions. g2 and g17 are complex multimodal cases. The local optima mislead the particles to search along the false directions. Thus on these problems, a larger diversity is favored, increasing diversity can improve the results but will affect the convergence speed for other problems. g22 is an interesting problem and a feasible solution can be found only when the algorithm employs an elitist strategy. It is why DMS-C-PSO always found good solution at the end of the search, when all the particles are grouped into one swarm and fly to gbest. But even the feasible region is found, it is hard to find a solution good enough to satisfy the accuracy. Thus DMS-C-PSO succeeded only once for g22 in all 25 runs.

5.4.3 Comparison:

1) Algorithms:

Ten algorithms involved in the comparison:

- ✧ DE [153] :
Differential Evolution
- ✧ DMS-C-PSO :
Dynamic Multi-Swarm Particle Swarm Optimizer with the New Constraint-Handling Mechanism
- ✧ ε _DE [155] :
 ε Constrained Differential Evolution with Gradient-Based Mutation and Feasible Elites
- ✧ GDE[164] :
Generalized Differential Evolution
- ✧ jDE-2 [152] :
Self-adaptive Differential Evolution
- ✧ MDE [165]
Modified Differential Evolution
- ✧ MPDE [166]
Multi-Populated Differential Evolution Algorithm
- ✧ PCX [167]
A Population-Based, Parent Centric Procedure

CHAPTER 5 DMS-PSO FOR CONSTRAINED OPTIMIZATION

- ✧ PESO+ [156]
Particle Evolutionary Swarm Optimization Plus
- ✧ SaDE [168]
Self-adaptive Differential Evolution Algorithm

2) Empirical Distribution of Normalized Success Performance

Only the results where at least one algorithm was successful at least once are used.
(g20 and g22 excluded).

Table 5-6 Success Rate

Alg. Func.	DE	DMS	e-DE	GDE	jDE-2	MDE	MPDE	PCX	PESO+	SaDE
1	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
2	84%	96%	100%	72%	92%	16%	92%	64%	56%	84%
3	0%	100%	100%	4%	0%	100%	84%	100%	100%	96%
4	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
5	100%	100%	100%	92%	68%	100%	100%	100%	100%	100%
6	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
7	100%	100%	100%	100%	100%	100%	100%	100%	96%	100%
8	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
9	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
10	100%	100%	100%	100%	100%	100%	100%	100%	16%	100%
11	100%	100%	100%	100%	96%	100%	96%	100%	100%	100%
12	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
13	32%	100%	100%	40%	0%	100%	48%	100%	100%	100%
14	100%	100%	100%	96%	100%	100%	100%	100%	0%	80%
15	100%	100%	100%	96%	96%	100%	100%	100%	100%	100%
16	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
17	20%	0%	100%	16%	4%	100%	28%	100%	0%	4%
18	100%	100%	100%	76%	100%	100%	100%	100%	92%	92%
19	100%	100%	100%	88%	100%	0%	100%	100%	0%	100%
21	60%	100%	100%	60%	92%	100%	68%	100%	0%	60%
23	0%	100%	100%	40%	92%	100%	100%	100%	0%	8%
24	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

CHAPTER 5 DMS-PSO FOR CONSTRAINED OPTIMIZATION

Fig. 5-2 Empirical Distribution over All Functions

Table 5-7 Normalized Success Performance

Func.	Algorithms	DE	DMS	ϵ _DE	GDE	jDE-2	MDE	MPDE	PCX	PESO+	SaDE
	SR SP _{best}	81.64%	95.27%	100%	80.91%	83.64%	91.64%	91.64%	98.36%	70.91%	91.09%
1	25115	1.3304	1.3239	2.3615	1.6133	2.0062	3.0011	1.7292	2.1981	4.0427	1
2	88201	1.5292	1	1.6987	1.6957	1.6542	1.0909	3.4577	1.4501	4.6807	2.0844
3	24760	Inf	1	3.6109	144.47	Inf	1.817	1.0041	1.411	18.2	12.074
4	15281	1.0461	1.6466	1.7156	1	2.6653	2.7198	1.3666	2.0279	5.2271	1.643
5	21306	5.0256	1.3208	4.5729	9.0821	20.972	1	10.16	4.4478	21.227	3.4263
6	5202	1.3731	5.2664	1.4189	1.2501	5.6686	1	2.0327	6.5015	10.863	2.4118
7	26351	3.5594	1	2.8197	4.7056	4.8478	7.3698	2.1783	4.4447	13.938	1.0488
8	918	1.183	4.2695	1.2407	1.6002	3.5251	1	1.6498	3.0784	6.671	1.4412
9	16152	1.5976	1.7836	1.4315	1.8716	3.4001	1	1.3029	2.8806	6.0391	1.3278
10	25086	4.7523	1	4.1949	3.2928	5.826	6.5439	1.9385	3.5489	112.76	1.7606
11	3000	4.46	4.7983	5.4733	2.82	17.976	1	7.7854	12.896	150.03	8.3703
12	1308	3.9021	4.0602	3.1529	2.4075	4.8593	1	3.2401	6.8502	6.1835	1.9694
13	21732	38.207	1.8536	1.5985	38.688	Inf	1	34.169	2.4726	20.726	1.1581
14	24474	2.7877	1	4.6351	9.4029	3.9979	11.916	1.7453	2.4204	Inf	1.8387
15	10458	5.5429	2.7469	8.0528	7.1605	23.081	1	19.141	4.488	43.039	2.5818
16	8730	1.3278	6.1143	1.4875	1.5148	3.6306	1	1.4963	3.4817	5.6174	1.7123
17	26364	50.389	Inf	3.7498	81.489	426.06	1	27.742	5.1627	Inf	474.13
18	28261	2.8151	1.123	2.0931	16.987	3.6963	3.6617	1.5585	2.4779	8.2431	1
19	20810	8.5165	1	17.124	11.066	9.6036	Inf	5.6835	6.2314	Inf	2.5067
21	38217	4.2571	3.6431	3.5362	15.162	3.3103	2.9455	5.4703	1	Inf	4.2958
23	1.2955	Inf	1.5804	1.5497	8.2081	2.7592	2.7821	1.6261	1.29	Inf	1
24	1794	1.6856	10.452	1.6455	1.7051	5.6834	1	2.4204	6.4916	11.137	2.5775

CHAPTER 5 DMS-PSO FOR CONSTRAINED OPTIMIZATION

From the comparison, it can be observed ϵ_{DE} is the best one algorithm out of the ten algorithms. DMS-C-PSO is one of the first three good algorithms and an average success rate 95.27% is achieved.

5.5 Conclusion

This chapter extended the DMS-L-PSO to solve the single objective optimization problem with constraints. A so called DMS-C-PSO was constructed based on DMS-L-PSO and a novel constraint-handling mechanism. The new constraint-handling method assigns the sub-swarms with different objectives and the more difficult constraint functions will have more sub-swarms searching for feasible solutions. In this way, constraint functions and the objective function are optimized at the same time and the particles will converge to feasible region with better objective function value. The novel constraint-handling mechanism can be combined in any multi-swarm algorithms. In the local search phase, the Sequential Quadratic Programming (SQP) is employed to improve the local search ability of the proposed algorithm.

Experiments were conducted on the standard benchmark functions set proposed in CEC'06. The results are compared with the results of other nine evolutionary algorithms which participated in the competition of this special session. The comparison results show DMS-C-PSO is one of the first three algorithms and a success rate 95.27% is achieved. 100% feasible rate showed the efficiency of the proposed constraint-handling mechanism.

Chapter 6

Dynamic Multi-Swarm Particle Swarm Optimizer for Multi-Objective Optimization

In the previous two chapters, Dynamic Multi-Swarm Particle Swarm Optimizer (DMS-PSO) for single objective optimization problems with only bound constraints and with general constraints have been discussed. In this Chapter, DMS-PSO will be extended to solve Multi-objective optimization problems with constraints. A brief review on the multi-objective PSOs is presented. Then, DMS-MO-PSO is introduced. Through analysis, novel **pbest** and **lbest** updating criteria which are more suitable for MO-PSOs are proposed. By combining the external archive and the novel updating criteria, excellent performance is achieved by DMS-MO-PSO on eight benchmark test functions.

6.1 Multi-objective Optimization

We always encounter multi-objective problems with conflicting objectives. For example, in an automatic air-conditioning system, we have two objectives: keeping a comfort temperature and saving the energy. Then we must find a way to balance these two objectives. This is just a simple example. In fact, there are many more complex problems in engineering, business and so on. Thus, we need to find different tradeoffs and choose a proper one or more for those problems. We can use the following formula to mathematically express the multi-objective optimization problems:

$$\text{Minimize } \mathbf{y} = f(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_m(\mathbf{x})), \mathbf{x} \in [\mathbf{Xmin}, \mathbf{Xmax}] \quad (6-1)$$

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

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

$\mathbf{x}$ is the decision vector, $\mathbf{y}$ is the objective vector. Different from the single objective optimization, there are two spaces to be considered. One is the decision

CHAPTER 6 DMS-PSO FOR MULTI-OBJECTIVE OPTIMIZATION

space, we denoted it as $\mathbf{X}$; another one is called the objective space, we denoted it as $\mathbf{Y}$.

Definition 1: Pareto Dominance [169]

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

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

Definition 2: Non-dominated Set [169]

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 [169]

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

Like global and local optimal solutions in the case of single-objective optimization, there could be global and local Pareto-optimal sets in multi-objective optimization.

The objective of multi-objective optimization is to find a set of solutions which can express the Pareto-optimal set well, thus there are two goals for the optimization:

- i) Convergence to the Pareto-optimal set
- ii) Diversity of solutions in the Pareto-optimal set

6.2 Multi-objective Particle Swarm Optimization

Development of evolutionary algorithms to solve multi-objective optimization problems has attracted much interest recently and a number of multi-objective evolutionary algorithms have been suggested. 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

CHAPTER 6 DMS-PSO FOR MULTI-OBJECTIVE OPTIMIZATION

algorithm employs a unique combination of specific techniques to achieve these goals.

The main advantage of evolutionary algorithms (EAs) in solving multi-objective optimization problems is their ability to find multiple Pareto-optimal solutions in one single run. [169] As particle swarm optimizers (PSO) also have this ability, recently there are several proposals to extend PSO to solve multi-objective problems.

Ray and Liew [82] combined Pareto dominance and concepts of evolutionary techniques with the particle swarm. The approach uses crowding to maintain diversity and Pareto ranks to handle constraints. Better performing particles are recorded into a set of leaders based on non-dominated rank and the remaining particles move towards a leader randomly selected from the leaders. Leaders with fewer individuals around them have a high probability of being selected. Parsopoulos and Vrahatis [84] introduced two methods that extend the PSO to be able to handle multi-objective problems. They were a weighted aggregation approach and Vector evaluated PSO. In Hu and Eberhart [85], a dynamic neighborhood and a new **pbest** updating strategy were proposed. In each generation, the neighborhood best is dynamically chosen according to a particle's distance to the other particles. The approach is then further improved by adding a secondary population, called extended memory, to store global Pareto optimal solutions to reduce computation time [170]. Although these approaches have been shown to find multiple non-dominated solutions on many test problems, researchers realized the need of introducing elitism as evidenced in many recent successful MOEAs. Further, recently more researchers are interested in incorporating an external archive into MOPSO to enhance the convergence properties.

Fieldsend and Singh [86] used dominated tree archive to select the global best individual based on a concept of closeness to members in the non-dominated set, and maintained a set of previous best solutions for each particle. Turbulence is incorporated to improve the performance of the multiobjective PSO. This approach uses an unbounded archive. However, some researchers bound the archive size to reduce the complexity of archive updating [87][88][90][91].

Mostaghim and Teich [92] proposed a sigma method in MOPSO for finding the best local guides for each particle in order to converge fast to the Pareto-optimal front with good diversity. In another paper [88], the same authors use ϵ -dominance

CHAPTER 6 DMS-PSO FOR MULTI-OBJECTIVE OPTIMIZATION

to fix the archive size and compared ε -dominance to the clustering techniques. They used an initial archive instead of an empty archive for MOPSO.

Li [93] extended the PSO to multi-objective problems with the non-dominated sorting concept of NSGA-II [94] and constructed the so called Non-dominated Storing Particle Swarm Optimizer (NSPSO). Instead of a single comparison between a particle's personal best and current position, it executes non-domination comparison among all particles' personal bests and their offspring in the entire population. Later, the author proposed a maxmin PSO for multi-objective optimization, which uses a fitness function derived from the maxmin strategy to determine Pareto-domination. One advantage is that no additional clustering or niching technique is needed since the maxmin fitness function provided the domination information and diversity information. Both algorithms showed competitive performance with the real-coded NSGA-II [172].

Bartz-Beielstein *et al.* [90] proposed DOPS that integrates the archiving technique into particle swarm optimization. They also analyzed several modifications and extensions of the archiving techniques. Coello and Lechuga [95] proposed MOPSO with an external repository and with an adaptive grid similar to PAES. This approach selects a global best based on roulette wheel selection of a hypercube. Coello *et al.* [91] also incorporated a special mutation operator to enhance the exploratory capabilities. Another improved version (called AMOPSO) is presented by Pulido and Coello [96], in which a clustering technique is used to divide the population of particles into several swarms in order to maintain a better distribution of solutions.

6.3 Dynamic Multi-Swarm Particle Swarm Optimization for Multi-Objective Optimization Problems

6.3.1 External Archive and Non-dominated Sorting

DMS-PSO has been introduced in Chapter 4. When it is extended to solve multi-objective problems, an external archive is added to keep a historical record of the non-dominated solutions obtained during the search process. The maximum size

CHAPTER 6 DMS-PSO FOR MULTI-OBJECTIVE OPTIMIZATION

of the archive $N_{\max}$ is predefined. The technique of updating the external archive is similar to the NSGAII [172] and the schematic is presented in Fig. 6-1.

Fig. 6-1 External Archive Updating

i. Non-dominated Sorting:

After adding the new solutions to the external archive, non-dominated sorting is performed on the external archive. We first find the best non-dominated solutions, non-dominated solutions of level 1, in the whole external archived population. We then find the next best non-dominated solutions, non-dominated solutions of level 2, in the remaining members of the external archive. In this way, we find all pareto fronts with different levels: $F_1, F_2, \dots$.

ii. Crowding Distance Sorting:

If the external archived population reaches its maximum size $N_{\max}$, set new external archive $P = \emptyset$, perform $P = P \cup F_i$ until $|P| + |F_{i+1}| = N_{\max}$. We sort the solutions in F_{i+1} according to one objective as in Fig. 6-2, assign a large distance to the boundary solution and calculate the summation of the distances from the nearest two solutions in other members of F_{i+1} . We include the most widely spread $N_{\max} - |P|$ solutions in P .

CHAPTER 6 DMS-PSO FOR MULTI-OBJECTIVE OPTIMIZATION

Fig. 6-2 The Crowding Distance Calculation

6.3.2 Choose Local Best for Each Sub-Swarm

Fig. 6-3 Illustration of Choosing Local Best for Each Sub-Swarm
(Sorting according to $f_2(\mathbf{x})$)

Different from the DMS-PSO for single objective optimization problem, in DMS-MO-PSO, a novel **lbest** selection method is employed. If $|\mathbf{F}_1| \geq n$, where n is the number of the sub-swarms, **lbest** are chosen from $\mathbf{F}_1$, the best non-dominated solutions set in the external archive. After the sub-swarms are regrouped, we sort the external archive based on one objective function (randomly chosen) then divide the external archive to n parts equally according to that objective as Fig. 6-3. For each sub-swarm, we randomly choose one member from one of the corresponding

CHAPTER 6 DMS-PSO FOR MULTI-OBJECTIVE OPTIMIZATION

part. Thus, each part has one sub-swarm searching for it. If there is no solution in the corresponding part, select one nearest non-dominated solution from either of the neighboring parts. If $|\mathbf{F}_1| < n$, the members of $\mathbf{F}_1$ are assigned to the first n sub-swarms as their **lbest**, then non-dominated sorting is done among **pbests**, and the non-dominated ones are randomly chosen as **lbest** for the remaining sub-swarms. This method can maintain the diversity of the population to obtain an external archive with good diversity.

6.3.3 Update **pbest** and **lbest**

An important characteristic of PSO is that it has **pbest** and **lbest** (or **gbest** in the global version) to record the historical information of the particles. **pbest** and **lbest** guide the search of the particles. Thus, it is important to decide how to update them. In single objective optimization, the answer is straightforward and **pbest** and **lbest** will be updated if better solutions are found. A better solution here means a solution which has a larger fitness value. But, in the multi-objective optimization world, the answer is not straightforward.

In PSO variants for multi-objective optimization, there exist five updating methods:

- i. **pbest** is replaced if **X** dominates **pbest**, otherwise if **X** is mutually non-dominating with **pbest**, **pbest** has 50% probability to be replaced [95].
- ii. **pbest** is replaced if **X** dominates **pbest**, otherwise if **X** is mutually non-dominating with **pbest** [89].
- iii. **pbest** is replaced only if **X** dominates **pbest** [85].
- iv. **pbest** is replaced if **X** dominates any pareto optimal solution in the current generation [87].
- v. A set of mutually non-dominating **pbests** is maintained for each **X** [86].

Obviously, **X** is better than **pbest** if **X** dominates **pbest**. But should we update **pbest** if **X** is not dominated by **pbest**? If we do not accept the non-dominated solution, in the end of evolutionary search, when the Pareto front has been found, the updating will seldom happen. But, should we accept the non-dominated solution? From Fig. 6-4, we can observe that each time **pbest** is updated by a non-dominated solution, obviously the **pbest** moves away from the true Pareto Front. Though this is just an extreme example, this can happen in the

CHAPTER 6 DMS-PSO FOR MULTI-OBJECTIVE OPTIMIZATION

search process. Thus, updating **pbest** or **lbest** when the new solution is not dominated by **pbest** or **lbest** is not a good idea. Sometime, it may lead the particle to fly in the wrong direction. It is because no information of external archive which contains the historical information of the whole search process is considered in the comparison.

Fig. 6-4 Illustration of an Extreme Example for **pbest** Updating if **pbest** is Updated When X Is Not Dominated by **pbest**

Therefore, a new comparison criterion is proposed here. Best non-dominated solutions external archive will be used in the updating and in order to reduce the computation complexity, a reference front **RF** is used instead of all member of $\mathbf{F}_1$. The maximal size of the reference front N_{ref} is predefined.

If $|\mathbf{F}_1| \leq N_{\text{ref}}$, $\mathbf{RF} = \mathbf{F}_1$;

If $|\mathbf{F}_1| > N_{\text{ref}}$, put the boundary solutions of in **RF**, we then randomly chose $N_{\text{ref}} - 2$ solutions from $\mathbf{F}_1$ except the boundary solutions.

With the new comparison criterion, **pbest** or **lbest** is updated when

- i. X dominates **pbest** or **lbest** , or
- ii. X is not dominated by any member of the reference front **RF**.

Four possible scenarios of **pbest** updating (**lbest** updating is the same) are shown in Fig. 6-5.

- i. X_1 dominates pbest_1 , so pbest_1 will be updated;
- ii. X_2 is not dominated by pbest_2 and X_2 is not dominated by any member of **RF**, so pbest_2 will be updated;

CHAPTER 6 DMS-PSO FOR MULTI-OBJECTIVE OPTIMIZATION

- iii. X_3 is not dominated by $\mathbf{pbest}_3$ and X_3 is dominated by at least one member of $\mathbf{RF}$, so $\mathbf{pbest}_3$ will not be updated;
- iv. X_4 is dominated by $\mathbf{pbest}_4$, so $\mathbf{pbest}_4$ will not be updated;

Fig. 6-5 An Illustration About *pbest* Updating

Since constraints are frequently associated with real-world optimization problems, we also use constrained-domination to handle constraints [172]. A solution i is said to constrained-dominate a solution j , if any of the following conditions is true.

- i. Solution i is feasible and solution j is not.
- ii. Solution i and j are both infeasible, but solution i has a smaller overall constraint violation.
- iii. Solutions i and j are feasible and solution i dominates solution j .

According to this constrained-domination principle, DMS-MO-PSO can deal with constrained problems without changing the modularity or computational complexity.

6.3.4 Convergence Phase

Different from the DMS-L-PSO for single objective optimization problems, in DMS-MO-PSO, in the convergence phase, the sub-swarms will not be grouped into one big swarm since the goals of multi-objective optimization are convergence to the Pareto Front and maintaining the diversity of the solutions. For the sake of improving the convergence to the Pareto Front, we set $Pc_i = 1$ for all particles to

CHAPTER 6 DMS-PSO FOR MULTI-OBJECTIVE OPTIMIZATION

stop the self learning and speed up the convergence when $|F_i| > 0.5N_{\max}$. The local search phase is removed since the objective of DMS-MO-PSO is to find a set of solutions not one solution. The convergence phase also plays the role of local search.

The flowchart of DMS-MO-PSO is given in Fig. 6-6:

Fig. 6-6 The Flowchart of the DMS-MO-PSO

CHAPTER 6 DMS-PSO FOR MULTI-OBJECTIVE OPTIMIZATION

6.4 Experiments

6.4.1 Performance Measures

In order to measure the performance of MOEAs quantitatively, we need some performance metrics to evaluate and compare the algorithms. Three different metrics are employed to evaluate the performance of an MOEA:

Convergence Metric (γ):

This metric finds an average distance between non-dominated solutions found and the actual Pareto-optimal front, as follows:

$$\gamma = \frac{\sum_{i=1}^N d_i}{N} \quad (6-2)$$

where N is the number of non-dominated solutions obtained with an algorithm and d_i is the Euclidean distance (in objective space) between the each of the non-dominated solutions and the nearest member of the actual Pareto optimal front. A smaller value of γ demonstrates a better convergence performance.

Spread Metric (Δ):

Deb *et al.* [172] proposed such a metric to measure the spread in solutions obtained by an algorithm. This metric is defined as

$$\Delta = \frac{\sum_{m=1}^M d_m^e + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{\sum_{m=1}^M d_m^e + (N - 1)\bar{d}} \quad (6-3)$$

Here, the parameters d_m^e are the Euclidean distance between the extreme solutions of Pareto optimal front and the boundary solutions of the obtained non-dominated set corresponding to m^{th} objective function. The parameter d_i is the Euclidean distance between neighboring solutions in the obtained non-dominated solutions set and $\bar{d}$ is the mean value of these distances. Δ is zero for an ideal distribution when $d_m^e = 0$ and all d_i equal to $\bar{d}$. Smaller the value of Δ , the better the diversity of the non-dominated set is.

CHAPTER 6 DMS-PSO FOR MULTI-OBJECTIVE OPTIMIZATION

ε -indicator (ε):

Zitzler *et al.* [173] recommended the unary and binary ε -indicator to show the factor by which an approximation set is worse than another with respect to all objectives. The ε value can be calculated as

$$\varepsilon(\mathbf{A}, \mathbf{B}) = \max_{y^2 \in \mathbf{B}} \min_{y^1 \in \mathbf{A}} \max_{1 \leq i \leq m} \frac{y_i^1}{y_i^2} \quad (6-4)$$

$\mathbf{A}$, $\mathbf{B}$ are two non-dominated set achieved by two different MO algorithms, $\mathbf{A}$ can be considered better than $\mathbf{B}$ when $\varepsilon(\mathbf{A}, \mathbf{B}) \cong 1$ and $\varepsilon(\mathbf{B}, \mathbf{A}) > 1$.

Replace $\mathbf{B}$ with the true pareto front $\mathbf{PT}$, the binary ε -indicator becomes the unary ε -indicator and $\varepsilon(\mathbf{A}, \mathbf{PT})$ should be as small as possible.

$$\varepsilon(\mathbf{A}, \mathbf{PT}) = \max_{y^2 \in \mathbf{PT}} \min_{y^1 \in \mathbf{A}} \max_{1 \leq i \leq m} \frac{y_i^1}{y_i^2} \quad (6-5)$$

6.4.2 Experimental Settings

In the simulations, eight test problems are chosen from the standard MOEA literature. The problems are defined in Appendix C. All MOEAs are run for a maximum of 25000 fitness function evaluations (FES). For real-coded NSGA-II, we use a population size of 100, crossover probability of 0.9 and mutation probability of $1/n$, where n is the number of decision variables, distribution indexes for crossover and mutation operators as $\eta_c = 20$ and $\eta_m = 20$ as presented in [172]. The population obtained at the end of 250 generations is used to calculate the performance metrics. PAES uses a depth of four and an archive size of 100. MOPSO uses a population size of 50, a repository size of 100 and 30 divisions for the adaptive grid with mutation as presented in [171]. DMS-MO-PSO uses the following parameter values: 10 sub-swarms with 3 particles in each sub-swarm, archive size $N_{\max} = 100$, the size of referent front $N_{\text{ref}} = 10$, regrouping period $R = 5$, the initial $P_c_{\text{mean}} = 0.1$. For these three approaches, we use all members in the archive after 25000 FES to calculate the performance metrics.

CHAPTER 6 DMS-PSO FOR MULTI-OBJECTIVE OPTIMIZATION

6.4.3 Experimental Results

Table 6-1, Table 6-2 and Table 6-3 show the means and variances of the convergence, diversity metrics and unary ε values obtained using the four algorithms NSGA-II, PAES, MOPSO and DMS-MO-PSO by repeatedly running 30 times on each problem. The best mean result on each problem is emphasized in boldface. The nonparametric Wilcoxon rank sum tests are conducted between the DMS-MO-PSO's result and the best result achieved by the other three MOEAs for each problem. The h values are presented in the last rows. An h value of 1 indicates that the performances of the two algorithms are statistically different with 95% certainty, whereas h value of 0 implies that the performances are not statistically different.

Table 6-1 Convergence Metric (γ) Comparison of the Four Algorithms

Algorithms		SCH	FON	KUR	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
NSGA-II	mean	0.0043	0.0021	0.0324	0.0674	0.1897	0.6211	5.1219	3.1209
	std	0.0004	0.0002	0.1074	0.0246	0.0615	0.0329	2.2526	0.3413
PAES	mean	0.0045	0.0360	1.0955	0.0006	0.0005	0.0745	3.5097	7.5964
	std	0.0004	0.1315	2.1724	0.0003	0.0003	0.0034	1.0967	0.8102
MOPSO	mean	0.0044	0.0013	0.0252	0.0189	0.0162	0.0267	5.6413	0.7501
	std	0.0004	0.0001	0.0041	0.0032	0.0099	0.0069	2.7814	0.4208
DMS	mean	0.0044	0.0012	0.0162	0.0018	0.0016	0.0018	0.0018	0.0039
	std	0.0003	0.0001	0.0015	0.0003	0.0004	0.0004	0.0008	0.0012
h		0	1	1	1	1	1	1	1

Table 6-2 Diversity Metric (Δ) Comparison of the Four Algorithms

Algorithms		SCH	FON	KUR	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
NSGA-II	mean	0.2823	0.4470	0.7680	0.5401	0.9482	0.7958	0.9445	0.9639
	std	0.0269	0.0335	0.0533	0.0454	0.1344	0.0148	0.0590	0.0212
PAES	mean	0.7416	0.5533	0.6896	1.1197	1.1539	0.9707	1.0108	0.8900
	std	0.0427	0.1276	0.0840	0.1561	0.1849	0.0331	0.1671	0.0815
MOPSO	mean	0.7618	0.5861	0.7849	0.5980	0.6810	0.7203	0.9720	1.0054
	std	0.0517	0.0413	0.0961	0.0470	0.2006	0.0322	0.0310	0.0741
DMS	mean	0.1723	0.1371	0.2665	0.1615	0.1835	0.5015	0.1647	0.1573
	std	0.0110	0.0142	0.0125	0.0109	0.1547	0.0155	0.0124	0.0129
h		1	1	1	1	1	1	1	1

CHAPTER 6 DMS-PSO FOR MULTI-OBJECTIVE OPTIMIZATION

Table 6-3 Unary ϵ Value Comparison of the Four Algorithms

Algorithms		SCH	FON	KUR	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
NSGA-II	mean	1.0210	1.0086	4.8849	1.0788	2.0780	2.1363	6.5837	4.9280
	std	0.0048	0.0016	0.7690	0.0422	0.3089	0.1283	2.3495	0.3336
PAES	mean	1.0397	1.0762	2.3494	1.5231	1.6001	1.5478	4.3087	6.9855
	std	0.0078	0.2511	2.5352	0.2767	0.2970	0.0009	0.9363	0.8056
MOPSO	mean	1.0479	1.0133	1.2372	1.0215	1.2963	1.0841	7.2443	2.4276
	std	0.0139	0.0042	0.1290	0.0025	0.4416	0.0168	2.7363	0.4585
DMS	mean	1.0141	1.0046	1.0307	1.0059	1.0380	1.0076	1.0059	1.0075
	std	0.0012	0.0005	0.0150	0.0004	0.1817	0.0027	0.0007	0.0023
h		1	1	1	1	1	1	1	1

DMS-MO-PSO is able to converge better than the other three algorithms except on ZDT1 and ZDT2, where PAES yielded better convergence measure. But according to the diversity metric, it is observed PAES achieves a bad diversity for these two problems. With respect to the diversity measure and unary ϵ value, DMS-MO-PSO outperforms the other algorithms in all test problems. According to the statistical significance test, DMS-MO-PSO achieved the better performance on most problems for these three metrics. In the following, we will discuss the performance of the four approaches on each test problem.

Test problem SCH is the simplest among the nine problems with only a single variable. All the four algorithms perform well on this problem, and almost get the same convergence measure and unary ϵ values. However, DMS-MO-PSO performs better than other three algorithms with respect to the diversity measure. The ϵ values of binary ϵ -indicator for each problem are listed in Table 6-4 and the Pareto fronts generated by the four algorithms of the median run on this problem are plotted in Fig. 6-7.

CHAPTER 6 DMS-PSO FOR MULTI-OBJECTIVE OPTIMIZATION

Fig. 6-7 Pareto Fronts Generated by the Four Algorithms on SCH

Table 6-4 Binary ε Values for Problem SCH

Algorithm	NSGA-II	PAES	MOPSO	DMS	PF
NSGA-II	1	1.0178	1.0173	1.0189	1.021
PAES	1.037	1	1.0348	1.0364	1.0397
MOPSO	1.0448	1.0426	1	1.0439	1.0479
DMS	1.0124	1.0129	1.0126	1	1.0141
PF	1.0058	1.0059	1.006	1.0062	1

The FON is a two-objective optimization problem with three variables. The Pareto optimal front is a single non-convex curve. Fig. 6-8 shows that DMS-MO-PSO effectively finds a well spread solution set along the front. The ε values of binary ε -indicator for each problem are listed in Table 6-5. It is observed that the performances of the four algorithms are similar.

CHAPTER 6 DMS-PSO FOR MULTI-OBJECTIVE OPTIMIZATION

Fig. 6-8 Pareto Fronts Generated by the Four Algorithms on FON

Table 6-5 Binary ε Values for Problem FON

Algorithm	NSGA-II	PAES	MOPSO	DMS	PF
NSGA-II	1	0.99425	1.0073	1.0077	1.0086
PAES	1.075	1	1.0736	1.075	1.0762
MOPSO	1.0129	0.99786	1	1.0126	1.0133
DMS	1.0034	0.99136	1.004	1	1.0046
PF	1.0011	0.98877	1.0012	1.0013	1

The KUR problem has three disconnected Pareto-optimal regions, which may cause difficulty in finding non-dominated solutions in all regions. DMS-MO-PSO performs well as shown in Fig. 6-9(d), obtaining non-dominated solutions in all regions.

CHAPTER 6 DMS-PSO FOR MULTI-OBJECTIVE OPTIMIZATION

Fig. 6-9 Pareto Fronts Generated by the Four Algorithms on KUR

Table 6-6 Binary ϵ Values for Problem KUR

Algorithm	NSGA-II	PAES	MOPSO	DMS	PF
NSGA-II	1	3.8105	4.3423	4.8382	4.8849
PAES	1.4799	1	2.021	2.2957	2.3494
MOPSO	1.0321	1.0334	1	1.2138	1.2372
DMS	1.0095	0.9266	1.0117	1	1.0307
PF	1.0028	0.92025	1.0041	1.004	1

ZDT1 is probably the easiest of all of the ZDT problems, the only difficulty an MOEA may face in this problem is the large number of variables. Though PAES achieves a small convergence metric, it achieves a bad diversity and unary ϵ value. Consider the three metrics and the binary ϵ -indicator, DMS-MO-PSO gives the best performance.

CHAPTER 6 DMS-PSO FOR MULTI-OBJECTIVE OPTIMIZATION

Fig. 6-10 Pareto Fronts Generated by the Four Algorithms on ZDT1

Table 6-7 Binary ε Values for Problem ZDT1

Algorithm	NSGA-II	PAES	MOPSO	DMS	PF
NSGA-II	1	1.0653	1.0598	1.0767	1.0788
PAES	1.4827	1	1.5121	1.5213	1.5231
MOPSO	1.0003	1.0184	1	1.0197	1.0215
DMS	1.0006	1.0046	1.0006	1	1.0059
PF	1	1.0014	1	1.001	1

Non-dominated solutions obtained in DMS-MO-PSO on ZDT2 are shown in Fig. 6-11. DMS-MO-PSO found a better spread with a smaller convergence metric than the others.

CHAPTER 6 DMS-PSO FOR MULTI-OBJECTIVE OPTIMIZATION

Fig. 6-11 Pareto Fronts Generated by the Four Algorithms on ZDT2

Table 6-8 Binary ε Values for Problem ZDT2

Algorithm	NSGA-II	PAES	MOPSO	DMS	PF
NSGA-II	1	1.6748	1.7617	2.0358	2.078
PAES	1.2631	1	1.4613	1.5934	1.6001
MOPSO	1.0153	1.1489	1	1.2918	1.2963
DMS	0.9972	1.0301	1.0319	1	1.038
PF	0.99676	1.0013	1	1.0015	1

The Pareto optimal front of ZDT3 is made up of five disjoint curves. Large values of γ and γ obtained by NSGA-II in Table 6-1 and Table 6-2 demonstrate that this approach could not converge to the Pareto optimal front with diverse distributions. Although the front obtained in MOPSO, shown in Fig. 6-12 (c), almost converges to the true front, it could not perform as well as DMS-MO-PSO, which produces the non-dominated solutions well converged and spread out over the entire

CHAPTER 6 DMS-PSO FOR MULTI-OBJECTIVE OPTIMIZATION

front, as shown in Fig. 6-12 (d).

Fig. 6-12 Pareto Fronts Generated by the Four Algorithms on ZDT3

Table 6-9 Binary ε Values for Problem ZDT3

Algorithm	NSGA-II	PAES	MOPSO	DMS	PF
NSGA-II	1	1.3875	1.9713	2.1206	2.1363
PAES	1.0029	1	1.428	1.5364	1.5478
MOPSO	0.99999	1.0674	1	1.0761	1.0841
DMS	1.0007	1.0545	1.0007	1	1.0076
PF	0.99999	1.0526	1	1.0001	1

The problem ZDT4 has 21^9 different local Pareto-optimal fronts in the search space. Because of the hurdles caused by a large number of local Pareto-optimal fronts, only DMS-MO-PSO could converge to the global front. The Pareto fronts generated by the four algorithms are plotted in Fig. 6-13. Though NSGA-II cannot converge to the global front with the current parameters, when a different

CHAPTER 6 DMS-PSO FOR MULTI-OBJECTIVE OPTIMIZATION

distribution index $\eta_m = 10$ is used, the global Pareto-optimal front can be obtained on ZDT4 [172]. A mean convergence metric 0.039853 and a mean diversity metric 0.540475 can be achieved by NSGAII with new parameter. However, it is still worse than the DMS-MO-PSO, even the same parameters used for all the problems.

Fig. 6-13 Pareto Fronts Generated by the Four Algorithms on ZDT4

Table 6-10 Binary ϵ Values for Problem ZDT4

Algorithm	NSGA-II	PAES	MOPSO	DMS	PF
NSGA-II	1	1.6094	1.3547	6.5711	6.5837
PAES	1.0779	1	1.0955	4.3006	4.3087
MOPSO	1.4579	1.829	1	7.2311	7.2443
DMS	1.0006	1.0008	1.0008	1	1.0059
PF	0.99985	1	1	1.0014	1

CHAPTER 6 DMS-PSO FOR MULTI-OBJECTIVE OPTIMIZATION

The problem ZDT6 is another hard problem. The adverse density of solutions across the Pareto-optimal front, together with the non-convex nature of the front, makes it difficult for many multi-objective optimization algorithms to maintain a well-distributed non-dominated set and converge to the true Pareto-optimal front. We could observe that MOPSO, NSGA-II and PAES could not converge to the true Pareto front of ZDT6, while DMS-MO-PSO performs well by converging to the true front with a good spread of solutions along the front as presented in Fig. 6-4. As shown in Table 6-1, Table 6-2, Table 6-3 and Table 6-11, the average values of γ , Δ and ε obtained by DMS-MO-PSO on problem ZDT6 are much better than the corresponding performance metrics obtained by the other algorithms.

Fig. 6-14 Pareto Fronts Generated by the Four Algorithms on ZDT6

CHAPTER 6 DMS-PSO FOR MULTI-OBJECTIVE OPTIMIZATION

Table 6-11 Binary ϵ Values for Problem ZDT6

Algorithm	NSGA-II	PAES	MOPSO	DMS	PF
NSGA-II	1	1.6094	1.3547	6.5711	6.5837
PAES	1.0779	1	1.0955	4.3006	4.3087
MOPSO	1.4579	1.829	1	7.2311	7.2443
DMS	1.0006	1.0008	1.0008	1	1.0059
PF	0.99985	1	1	1.0014	1

By comparing all four algorithms on eight test functions, it is observed that DMS-MO-PSO achieves fairly good results on all the eight functions. Except SCH problem, on which all four algorithms obtained good results, DMS-MO-PSO performs better than the other three algorithms especially on the complex problems ZDT4 and ZDT6.

About the computational complexity, DMS-MO-PSO has the similar complexity with NSGAII since the main time-consuming schedule for these two algorithms is updating the external archive and DMS-MO-PSO used the similar technique as NSGAII. Through it is a little slower than MOPSO and PAES, comparing the results, DMS-MO-PSO has much better performance. And in many real problems, the most time-consuming part is the cost functions calculations. In such a case, the computational complexity of these four algorithms can all be omitted.

6.5 Conclusion

This chapter extended the DMS-PSO to solve multi-objective optimization problems with an external archive and a novel **pbest** and **lbest** updating strategy. From the analysis, it is observed that updating **pbest** or **lbest** just because the new solution is not dominated by the **pbest** or **lbest** is not enough. Sometimes it will lead the particles to fly in a wrong direction. By judging the domination relationship of the new solution and iteratively updating the reference front, the **pbest** and **lbest** are updated in a more reasonable way. Non-dominated sorting and crowding distance sorting also help the DMS-MO-PSO to have a better external archive.

We evaluated the proposed approach on eight test problems currently adopted in the literature. The proposed DMS-MO-PSO significantly outperforms other three

CHAPTER 6 DMS-PSO FOR MULTI-OBJECTIVE OPTIMIZATION

representative multi-objective evolutionary algorithms, mainly on larger dimensional problems. It also demonstrates a good performance when solving a multimodal problem, ZDT4. Although NSGA-II has no external archive, it combines the parent and offspring populations, which has the same effect as external archive to avoid missing the non-dominated solutions. DMS-MO-PSO, MOPSO and PAES all incorporate external archive. To improve diversity of the non-dominated solutions, DMS-MO-PSO and NSGA-II use crowding distance. PAES and MOPSO use adaptive grids. But, DMS-MO-PSO performs the best among these four algorithms, which demonstrates that the self-learning strategy employed in DMS-MO-PSO is as effective as in single-objective optimization when dealing with multi-objective optimization problem. In brief, DMS-MO-PSO is an effective multi-objective evolutionary algorithm capable of converging to the true Pareto optimal front and maintaining a good diversity along the Pareto front.

Chapter 7

Application - Detection of the Bragg Wavelength of Each FBG for an FBG Sensor Network using DMS-PSO

This chapter applied DMS-PSO to detect the Bragg wavelength of the FBGs for an FBG sensor network and a tree search structure is proposed to enhance global search ability resulting in reduced computation time and increased accuracy for networks with a large number of FBGs.

7.1 FBG Sensor Network

A key issue in Wavelength Division Multiplexed (WDM) Fiber Bragg Grating (FBG) sensor network is the accurate detection of the Bragg wavelength of each FBG within the network. A popular scheme for wavelength detection is the so-called Conventional Peak Detection (CPD) technique where a Tunable Optical Filter (TOF) is used to scan through the working range of the FBG spectrums and detect the peak (Bragg) wavelength corresponding to each FBG [174]. The CPD technique is however not much applicable when the spectrums of the FBGs within the network are partially or fully overlapped [175]. The overlapping spectrums would cause crosstalk among the sensors and then introduce errors in Bragg wavelength detections. This limits the system performance in terms of either the number of sensors or the measurement range of the sensors within the network.

Optimization techniques are suitable to be employed to solve this kind of problem. Considering this is a complex multimodal problem, the classic gradient search method cannot perform well, while the Evolutionary Algorithm (EA) which is capable of solving the complex nonlinear optimization problem can be a better choice. By the use of EA, the Bragg wavelength detection error and the computational time could be reduced. A binary Genetic Algorithm (GA) has been

CHAPTER 7 APPLICATION

used for the determination of the Bragg wavelengths [176] and the results showed that this technique was capable of quickly determining the Bragg wavelengths even when the spectrums of the FBGs within the network were partially or completely overlapped. However, because of the limitations of the simple binary GA, the performance could not be improved further when the number of sensors is increased. In this paper, a novel dynamic multi-swarm particle swarm optimizer (DMS-PSO), which has better global search ability, is employed instead of the simple binary GA in order to improve the performance for more FBGs network and reduce the computational time.

The basic principles of applying the EA technique to FBG sensor network may be explained as follows referring to the FBG network as shown in Fig. 7-1.

Fig. 7-1 Schematic Diagram for *N*-FBG Network: (OSA: Optical Spectrum Analyzer; FBG: Fiber Bragg Grating; PC: Personal Computer)

Assume that $g_i(\lambda)$ ($0 \leq g_i(\lambda) \leq 1; i = 1, 2, \dots, N$) are the spectral shapes of the N FBGs. The measured spectrum from say an optical spectrum analyzer (OSA) may be expressed as

$$R(\lambda) = \sum_{i=1}^N R_{i0} g_i(\lambda - \lambda_{Bi}) + \text{Noise}(\lambda) \quad (7-1)$$

where λ_{Bi} , R_i are the Bragg wavelength and peak reflectivity of the i^{th} FBG. $\text{Noise}(\lambda)$ is a random spectral fluctuation accounting for various noises occurring in the system. From the original reflection spectrums of these N FBGs, combined spectrum can be constructed as follows

CHAPTER 7 APPLICATION

$$R_v(\lambda, \mathbf{s}) = \sum_{i=1}^N R_i g_i(\lambda - s_i) \quad \mathbf{s} = \{s_1, s_2, \dots, s_N\} \quad (7-2)$$

By varying s_i , variable spectrums are constructed that cover all possible combinations of $R_i g_i(\lambda)$. The variance between actual measured spectrums is given in Eqn. (7-1) and the artificially constructed spectrum is given in Eqn. (7-2) [177].

$$g(\mathbf{s}) = \int_0^\infty [R(\lambda) - R_v(\lambda, \mathbf{s})]^2 d\lambda, \quad \mathbf{s} = \{s_1, s_2, \dots, s_N\} \quad (7-3)$$

is minimized when $s_i = \lambda_{Bi}$. The values of s_i corresponding to the minimum variance are therefore respectively the Bragg wavelengths of FBG_i. Hence, this is a minimization problem:

$$\text{Min} \quad \{g(\mathbf{s})\} = \int_0^\infty [R(\lambda) - R_v(\lambda, \mathbf{s})]^2 d\lambda, \quad \mathbf{s} = \{s_1, s_2, \dots, s_N\} \quad (7-4)$$

After discretization, it becomes

$$\text{Min} \quad \{g(\mathbf{s})\} = \sum_{j=1}^L [R(\lambda_j) - R_v(\lambda_j, \mathbf{s})]^2, \quad \mathbf{s} = \{s_1, s_2, \dots, s_N\} \quad (7-5)$$

Based on the aforementioned principle, wavelength detection accuracy of a few pico-meters can be achieved by ‘scanning’ s_i ($i = 1, 2, \dots, N$) within specified range at sufficiently fine steps. If the range is set to 1531.0 nm~1532.0 nm and is sampled in 1000 points, the accuracy of 1pm is obtained and the number of calculation cycles of Eqn. (7-5) by conventional ‘scanning’ method is 1000^N , so that the computational time is very long and makes it difficult to be applied in some practical situations. This minimization problem is a complex multimodal problem which has trapped near the search bounds. Using an EA which has good global search ability to solve this minimization problem is an attractive approach.

7.2 Improving the Performance of an FBG Sensor

Network using Dynamic Multi-Swarm PSO

7.2.1 Simulation Results

DMS-PSO is employed to solve this problem. The schedule is the same with DMS-L-PSO except no special local search is added in since there is noise in this

CHAPTER 7 APPLICATION

minimization problem and the usual local search method will not help much. Thus the details will be repeated here.

The schematic diagram of a WDM N -FBGs sensor network used in the following simulations is shown in Fig. 7-1. Light from a broadband source (BBS) is coupled from one arm of a $2 \times N$ optical fiber coupler with a coupling ratio of $1/N$ to a WDM N -FBGs sensor network. The reflectivities (R_i) and the Bragg wavelengths (λ_{Bi}) are both different for all FBGs where subscript $i = 1, 2, \dots, N$. The reflected light from all FBGs is coupled back into the other arm of the same coupler and the combined spectrum is detected by an optical spectrum analyzer (OSA) where the span width of the OSA covers the whole spectral ranges of all FBGs. The OSA samples the spectrum into k points and passes the sampled data to a personal computer (PC) for further analyze.

Assume the spectrums of the FBGs are Gaussian shape [178] and their full-width at half-maximum (FWHM) are $\Delta\lambda$. The peak reflectivities and/or the shapes of the FBGs in the WDM network should be different, so the reflectivity of the i^{th} FBG in the WDM N -FBGs sensor network is set to be $(i \times R_N)/N$ and the separation of the Bragg wavelengths between the adjacent FBGs ($\lambda_{Bi} - \lambda_{B(i-1)}$) is ξ pm ($2 \leq i \leq N$). For example, in a 10-FBGs sensor network, the reflectivity of the FBG (R_i) is $(i \times R_{10})/10$ where R_{10} is equal to 100%. The Bragg wavelength of the FBG (λ_{Bi}) is $\lambda_{B10} - \xi \times (10 - i)$ where λ_{B10} and ξ are 1531.50 nm and 1 pm respectively. Moreover, the FWHM of all FBGs are assumed to be 0.2 nm. The span width of the OSA is set to be 1nm (1531.0 nm-1532.0 nm) and is sampled by 1000 points. The individual and combined uncontaminated spectrum of the 10-FBGs sensor network from the OSA is shown in Fig. 7-2. The detection for each Bragg wavelength is nearly impossible when the 10 FBGs are partially overlapped. Therefore, the DMS-PSO algorithm is used to detect for each Bragg wavelength on this partially overlapping case.

Fig. 7-2 The Spectrums of the 10-FBGs Sensor Network from the OSA when λ_{B10} and ξ are 1531.50 nm and 1 pm Respectively.

Except the simple binary GA, a good evolution strategy CMA-ES which uses covariance matrix adaptation is also tested in the simulation with DMS-PSO. The simulation is conducted with different number of sensors and with different signal-to-noise ratio (SNR) conditions. The white noise is added to each FBG for testing the ability of the EAs to detect the Bragg wavelength in a noisy environment. For each specified number of FBGs and SNR, the simulation is repeated for 10 times. The Bragg wavelength detection for each case is accomplished with binary GA, CMA-ES and DMS-PSO algorithms. For all the three algorithms, the max fitness evaluation times (Max_FEs) is set to 50,000. The parameters of GA [175] were chosen: population size is 100, the number of bits per coefficient 10, crossover possibility $P_c = 1$, mutation possibility $P_m = 0.1$. The default parameters are used in CMA-ES. Population size is 30 and the regrouping period R is set to 10 for DMS-PSO. The algorithms will be terminated when the best result, which is achieved so far, has not been improved within 200 generations. The mean values of the root-mean-square (RMS) values of the Bragg wavelength detection errors of the 10 runs are used to evaluate the performance of algorithms. A P4 3G, 1024MB personal computer is used in this simulation. The results are plotted in Fig. 7-3 and the computational cost of GA and DMS-PSO are compared in Fig. 7-4.

CHAPTER 7 APPLICATION

Fig. 7-3 The Mean RMS Values of the Wavelength Detection Error Due to GA
CMA-ES and DMS-PSO ($\xi = 0.1\text{ pm}$)

Fig. 7-4 Comparison of the Computational Cost of GA, CMA-ES and DMS-PSO
($\xi = 0.1\text{ pm}$)

From the results, DMS-PSO achieves the zero error for all cases except for 10 FBGs with SNR at 1 while the simple GA and CMS-ES only achieve zero error for two sensors case. For 10 sensors with SNR at 1, the mean of RMS values of the wavelength detection error for the DMS-PSO is 1.20 pm, while the mean error of the

CHAPTER 7 APPLICATION

GA and CMS-ES are 298.77 pm and 245.35 pm respectively. Moreover, the computational cost by DMS-PSO is similar with CMS-ES and is about 5~10 times lower than that of binary GA. For 10 FBGs case, when SNR is equal to 1, the computation time is 146.24 s, 37.21 s and 29.83 s for GA and DMS-PSO respectively. Additionally, Quasi-Newton method is also tested under the same condition for 10 sensors with SNR at 1, the mean of RMS values of the wavelength detection error for Quasi-Newton method is 871.24 pm. Comparatively, evolutionary algorithms achieved much better results than the gradient search method and the DMS-PSO performs better with less number of fitness evaluations than the other two algorithms.

In order to test the algorithms' performance in the overlapped situation, ξ is set to be 0 pm and then repeat the simulation. The results are presented in Fig. 7-5. Except that the mean RMS values are 0.16 pm for 10 FBGs case when SNRs are equal to 1, zero errors are achieved by DMS-PSO for the other cases. While the results of the simple GA and CMA-ES are much worse. From the results shown in Fig. 7-3 and Fig. 7-5, it is observed that irrespective of partial overlapped or totally overlapped, DMS-PSO performs well even with a large number of FBGs in the WDM network.

Fig. 7-5 The Mean RMS Values of the Wavelength Detection Error Due to the Simple GA, CMA-ES and DMS-PSO for Overlapping Situation (Accuracy: 1 pm)

Fig. 7-8 Measured Bragg Wavelength vs. Applied Strain
(one step corresponds to 33 με)

7.3 Improving the Performance of an FBG Sensor
Network using Tree Search Dynamic Multi-Swarm
PSO

By minimizing Eqn.(7-5), wavelength detection accuracy of a few pico-meters is achieved by ‘scanning’ $s_i (i = 1, 2, \dots, N)$ within specified range in fine steps. If the range is 1531.0 nm ~1532.0 nm and is sampled by 10000 points, 0.1 pm accuracy is obtained. The number of calculation cycles of Eqn.(7-5) by conventional ‘scanning’ method is 10000N. Hence, the computation time is too long and makes it unsuitable in practical situations. A Tree Search structure is developed to solve this special problem.

The fitness evaluated using Eqn.(7-5) has a computation complexity of $O(L)$, where L is the number of sample points. If the span width of the OSA is 1 nm, the number of sample points needed for the accuracies of 1 pm and 0.1 pm are 1000 and 10000 respectively. The computation complexity of 0.1 pm accuracy is 10 times that

CHAPTER 7 APPLICATION

of 1 pm. Thus, the tree search method is proposed to search the large search space with less time consuming low accuracy fitness function to obtain an initial solution, and then shrink the search range with the obtained initial solution as the center and use a higher accuracy fitness function to refine the result around the initial solution. The original DMS-PSO is extended to Tree Search Dynamic Multi-Swarm PSO (TS-DMS-PSO). For an M -layers tree search DMS-PSO, Eqn.(7-5) is divided into M sub-objective equations. In the sub objective equation of each layer, different L value is used. In each layer, DMS-PSO is used to search for the optimal solution to the corresponding sub objective Eqn. (7-5) becomes

$$\text{Min}\{g(s, L_m)\} = \sum_{j=1}^{L_m} [R(\lambda_j) - R_v(\lambda_j, s)]^2, \quad s = \{s_1, s_2, \dots, s_N\} \quad (7-6)$$

where $m = 1, 2, \dots, M$ and $L_1 < L_2 < \dots < L_M$

If the original search range for the problem for layer i is $[S_i^{\min}, S_i^{\max}]$ and the coarse result achieved for layer i is S_i , the new ensued search range for layer $i+1$ is $[\max(S_i^{\min}, S_i - (S_i^{\max} - S_i^{\min})/10), \min(S_i^{\max}, S_i + (S_i^{\max} - S_i^{\min})/10)]$.

7.3.1 Simulation Results

The parameters for the FBG network are the same as section 7.2.1 except the span width of OSA is sampled by 10000 points and the corresponding sample resolution is 0.1 pm. The simulation is also conducted with different number of sensors and different signal-to-noise ratios (SNR). For each specified number of FBGs and SNR, the simulation is repeated 20 times. We used the simple GA, DMS-PSO and TS-DMS-PSO to detect each Bragg wavelengths. The population size is 30. The maximum number of fitness evaluations (Max_FEs) is 60,000. The algorithms are terminated when the best result did not improve for 200 generations or Max_FEs is completed. Other parameters of the algorithms are: GA- 14 bits per coefficient, crossover probability $P_c = 1$, mutation probability $P_m = 0.1$; DMS-PSO: 10 sub-swarms, $R = 10$; TS-DMS-PSO: 10 sub-swarms, $R = 10$, $M = 2$, $L_1 = 1000$, $L_2 = 10000$.

The mean values of the root-mean-square (RMS) values of the Bragg wavelength detection errors of the 20 runs are used to evaluate the algorithms. The results and the computation costs are compared in Fig. 7-9 and Fig. 7-10. From the results, we observe that DMS-PSO and TS-DMS-PSO presents smaller errors than the simple

CHAPTER 7 APPLICATION

GA. TS-DMS-PSO performs better when the number of FBGs and the noise level increase. If the number of sensors is equal to 10 and SNR is 1, the mean RMS values of the wavelength detection error for the 20 runs achieved by the simple GA, DMS-PSO and TS-DMS-PSO are 301.82 pm, 71.92 pm and 7.22 pm respectively. However, TS-DMS-PSO requires the lowest computation time of 76.23 compared to 1026.84s and 578.21s.

Fig. 7-9 The Mean RMS Detection Errors for Partially Overlapped Case
($\xi = 0.1 \text{ pm}$)

Fig. 7-10 The Computation Costs for Partially Overlapped Case ($\xi=0.1 \text{ pm}$)

CHAPTER 7 APPLICATION

To test the algorithms when the sensors are completely overlapped ξ is set to 0.0 pm and the simulation is repeated. The results of 20 runs are plotted in Fig. 7-11 and Fig. 7-12. When SNR is 1, for 10 FBGs, the mean RMS values of the wavelength detection error achieved by the GA, DMS-PSO and TS-DMS-PSO are 370.18 pm, 65.24 pm and 0.00 pm respectively. The computation times are 1079.92 s, 592.41 s and 89.24 s.

Fig. 7-11 The Mean RMS Detection Errors for Overlapped Case ($\xi = 0.0\text{pm}$)

Fig. 7-12 The Computation Costs for Overlapped Case ($\xi = 0.0\text{ pm}$)

Simulations on 2-10 FBGs have shown TS-DMS-PSO yielding the best results at

the lowest computation cost when the sensors are partially or completely overlapped, especially for large number sensors with high noise.

7.4 Conclusion

In this chapter, DMS-PSO is employed to determine the Bragg wavelengths of the sensors in an FBG sensor network. Simulations and experiment have shown that the DMS-PSO can quickly and accurately determine the Bragg wavelengths of the sensors, when the spectrums of the FBGs within the network are partially or completely overlapped. When the number of sensors is 10 and SNR is equal to 1, the mean RMS values of the wavelength detection error achieved by DMS-PSO for 10 runs is 1.20 pm and 1.60 pm for partially or completely overlapped cases respectively in the simulation. In the experiment, when one sensor was shifted and another is fixed, the detection accuracies for the two sensors were 1.60 pm and 1.68 pm. The limitation of the CPD technique is overcome. Comparisons between GA, CMA-ES and DMS-PSO show a better search ability with higher accuracy and less computation cost of the DMS-PSO.

In order to reduce the computation cost when requiring high accuracy, a novel TS-DMS-PSO is developed. A specially designed tree search structure is combined in the DMS-PSO. Simulations show that TS-DMS-PSO can accurately determine the Bragg wavelengths of the sensors, when the spectrums of the FBGs are partially or completely overlapped. When SNR is 1, for 10 partially/completely overlapped FBGs, the mean RMS values of wavelength detection errors due to TS-DMS-PSO are 7.22 pm/0.00 pm in 76.23 s/89.24 s. With a two-layer, TS-DMS-PSO consumes only 10%-20% computation time and achieves better results than GA and the original DMS-PSO.

Chapter 8

Conclusions and Recommendations

This chapter summarizes the contributions of this thesis, followed by some recommendations for future research.

8.1 Conclusions

Particle Swarm Optimizer, as a comparably new member in the EA world, attracts the interest of the author. The author surveyed on the origin and development of this interesting algorithm. Inspired by the works done by other researchers, comparing the strengths and weaknesses of different hybrids, the author proposed two advanced PSO algorithms to solve different kind of optimization problems. The novel strategies for Particle Swarm Optimizer and new PSO algorithms for single optimization without constraints, constrained optimization and multi-objective optimization are introduced and investigated in this thesis. The brief introduction on optimization, and reviews on evolutionary algorithms and Particle Swarm Optimizer were presented in Chapter 2.

One of the two proposed algorithms, Comprehensive Learning Particle Swarm Optimizer (CLPSO), employing a comprehensive learning strategy, was presented in Chapter 3. The novel comprehensive learning strategy makes the particles learn from themselves and other particles. With the new updating strategy, different dimensions of a particle can learn from different exemplars. The learning probability P_c controls the probability that a dimension of the particle should learn from other particles. Each particle is assigned with a different P_c . The exemplars will be changed if the particle stops improving for some generations. From the analysis, it was observed that the new strategy gave the particles larger potential search space. Thus, the new CLPSO algorithm has better global search ability. Experiments were conducted on CLPSO and other eight PSO variants on a set of carefully selected benchmark functions including the composition functions which

CHAPTER 8 CONCLUSIONS AND RECOMMENDATIONS

was also introduced in this Chapter 3. The composition functions are constructed using several basic functions and the users can control the properties through changing the basic functions or changing the parameters in the composition function. The results showed that CLPSO performs better than the other PSO variants on most test functions especially on the complex multimodal problems due to its excellent global search ability.

Another PSO variant, Dynamic Multi-Swarm Particle Swarm Optimizer with Local search (DMS-L-PSO) for single objective optimization with bound constraints, was introduced in Chapter 4. This algorithm was composed of the local version of PSO, a dynamic multi-swarm neighborhood structure, a self-learning strategy and a local search phase. The population of DMS-L-PSO was divided into sub-swarms and the sub-swarms are always regrouped randomly every R generations. The information was transferred within this periodically changing neighborhood topology. The novel structure increased the diversity of the particles and discouraged the premature convergence. Enlightened by the idea of comprehensive learning strategy, an adaptive self-learning strategy was introduced, with which some dimensions would employ the historical information kept in **pbest** while some dimensions do the usual updating. The learning probability P_c self-adjusted in the search process. In order to improve the local search capability, Quasi-Newton local search method is combined in the algorithms. Experiments were conducted on the benchmark functions defined in CEC'05 and the comparison results with other evolutionary algorithms are presented to show the performance of the DMS-L-PSO.

After the introduction of how to employ the DMS-L-PSO algorithm in single objective optimization problem with only bounds, this algorithm was extended to DMS-C-PSO to solve the single objective optimization problems with general constraints in Chapter 5. DMS-C-PSO used a novel constraint-handling mechanism. Each sub-swarm of DMS-C-PSO is assigned an objective randomly from the objective function and the constraint functions. The difficult constraints would have a high probability to let the sub-swarms work for it. In another word, there will be more sub-swarms searching using the more difficult constraint function as the objective for feasible solutions. The whole swarm would fly to the better feasible region. The Quasi-Newton method in DMS-L-PSO is replaced by the SQP local search method for constrained problems. Experiments were

CHAPTER 8 CONCLUSIONS AND RECOMMENDATIONS

conducted on the standard benchmark function proposed in CEC'06 and a 100% feasible rate and 95.27% success rate were achieved by the proposed algorithm.

In Chapter 6, the DMS-PSO was extended to be a DMS-MO-PSO algorithm to solve multi-objective problems. The external archive was used to store the non-dominated solutions found in the search. Non-dominated sorting and crowding distance sorting were used to reject the unwanted solutions when the maximal length of the external archive is exceeded. The **pbest** and **lbest** updating strategy was different from the DMS-PSO for single objective optimization and other multi-objective PSOs. A new solution would be accepted as the new **pbest** (**lbest**) only when the new solution dominates the **pbest** (**lbest**) or the new solution is not dominated by any member in the reference front. The members of the reference front are chosen from the best non-dominated solutions in the external archive. Every R generations, when the sub-swarms are regrouped, an **lbest** is chosen from the best non-dominated solutions in the external archive. No local search method is used in the DMS-MO-PSO. Experiments were conducted on eight benchmark functions and the results show that DMS-MO-PSO is the best when compared with three other algorithms. Comparison was based on convergence to the true Pareto optimal front and diversity of generated solutions along the Pareto front.

In Chapter 7, an FBG sensor network was introduced and DMS-PSO was applied to determine the Bragg wavelengths of the sensors. This problem is a multimodal problem with noise and can only be solved using an algorithm with good global search ability. By combining with a tree search structure which is specially designed for this special problem, the DMS-PSO can locate the Bragg wavelengths of the sensors for 2-10 FBGs quickly and accurately.

8.2 Recommendations for Further Research

i. Extending CLPSO to Solve Constrained Problems

The CLPSO introduced in this thesis can only be applied on single objective problems with bound constraints. It can be extended to solve constrained problems by combining with a constraint-handling method similar to the constraint-handling method used in DMS-C-PSO. For CLPSO, the objective can be assigned to each particle rather than the sub-swarm in the DMS-C-PSO. Each particle evolves itself

CHAPTER 8 CONCLUSIONS AND RECOMMENDATIONS

according to its objective at one time and the objectives are reassigned every few generations.

ii. Extending DMS-PSO as an Omni-Optimizer

DMS-PSO has been introduced in this thesis to solve different types of problems. We can use DMS-L-PSO to solve single objective optimization problems with bound constraints, use DMS-C-PSO to solve constrained problems and use DMS-MO-PSO to solve multi-objective problems. We need to choose one DMS-PSO variant for each type. Thus, extending DMS-PSO to a multi-objective, multi-optima optimizer as the omni optimizer proposed by Deb and Tiwari in [182] would be very interesting and valuable. The new DMS-PSO should automatically adjust its behavior when it is applied to different types of optimization problems, such as single-objective uni-optimal problems, single-objective multi-optima problems, multi-objective uni-optimal problems and multi-objective multi-optimal problems.

iii. Modifying the Constraint-Handling Mechanism in DMS-C-PSO

In DMS-C-PSO, each sub-swarm evolves along one objective and when the constraints are too much the diversity of the sub-swarms will be too big and the convergence will slow down. Thus combining some constraints together would be a good idea. In this way, a sub-swarm can improve some constraints at one time, and then improve some other constraints subsequently. This new mechanism is expected to be better than the current one.

iv. Extending CLPSO or DMS-PSO for Binary or Discrete Problems

This thesis covers only the real parameter optimization part and no algorithms have been designed for binary or discrete problems. Extending the comprehensive learning strategy and the adaptive self-learning strategy for solving binary or discrete problems thereby resulting novel CLPSO or DMS-PSO variants to solve these problems would be an interesting topic.

v. Providing a Standard Set of Benchmark Functions for MO Problems

Standard benchmark functions sets for single objective optimization with bounds constraints and single objective optimization with constraints have been proposed

CHAPTER 8 CONCLUSIONS AND RECOMMENDATIONS

in CEC'05 and CEC'06 respectively. Selecting a set of benchmark functions from the existing multi-objective test functions and constructing some new test functions to form a standard set of multi-objective test functions are meaningful and necessary.

AUTHOR'S PUBLICATIONS

Journal Papers

1. J. J. Liang, C. C. Chan, P. N. Suganthan & V. L. Huang, "Wavelength detection in FBG sensor network using tree search DMS-PSO," *IEEE Photonics Technology Letters*, vol. 18(12), pp. 1305 - 1307, June 2006
2. V.L. Huang, P. N. Suganthan and J. J. Liang, "Comprehensive Learning Particle Swarm Optimizer for Solving Multiobjective Optimization Problems," *International Journal of Intelligent Systems*, vol. 21(2), pp. 209-226, 2006.
3. J. J. Liang, P. N. Suganthan, A. K. Qin and S. Baska, "Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions," *IEEE Transactions on Evolutionary Computation*, vol. 10(3), pp. 281-295 June 2006.
4. J. J. Liang, S. Baskar, P. N. Suganthan and A. K. Qin, "Performance Evaluation of Multiagent Genetic Algorithm," *Natural Computing*, vol. 5(1) , pp. 83-96, March 2006.
5. S. Baskar, A. Alphones, P. N. Suganthan and J. J. Liang, "Design of Yagi-Uda Antennas Using Particle Swarm Optimization with new learning strategy," *IEE Proceedings on Antenna and Propagation*, vol. 152(5), pp. 340 - 346, Oct. 2005.

Technical Reports

1. V. L. Huang, A. K. Qin, K. Deb, E. Zitzler, P. N. Suganthan, J. J. Liang, M. Preuss and S. Huband, "Problem Definitions for Performance Assessment of Multi-objective Optimization Algorithms", Special Session on Constrained Real-Parameter Optimization, *Technical Report*, Nanyang Technological University, Singapore, 2007.
2. 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 CEC 2005 Special Session on Real-Parameter Optimization," *Technical Report*, Nanyang Technological University, Singapore, May 2005 and KanGAL Report #2005005, IIT Kanpur, India.

3. J. J. Liang, Thomas Philip Runarsson, Efren Mezura-Montes, Maurice Clerc, P. N. Suganthan, Carlos A. Coello Coello & K. Deb, "Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization," *Technical Report*, Nanyang Technological University, Singapore, March 2006.

Conference Papers

1. Zhao, S.Z. Liang, J.J. Suganthan, P.N. Tasgetiren, M.F. Dynamic multi-swarm particle swarm optimizer with local search for Large Scale Global Optimization," in *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 3845-3852, 2008.
2. J. J. Liang & P. N. Suganthan, "Adaptive Comprehensive Learning Particle Swarm Optimizer with History Learning," in *Lecture Notes in Computer Science(LNCS)*, vol. 4247, in *Proceedings of The 6th International Conference on Simulated Evolution and Learning*, 2006.
3. J. J. Liang & P. N. Suganthan, "Dynamic Multi-Swarm Particle Swarm Optimizer with a Novel Constrain-Handling Mechanism," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC2006)*, pp. 9-16, 2006.
4. J. J. Liang, C. C. Chan, V. L. Huang, & P. N. Suganthan, "Improving the performance of a FBG sensor network using a novel dynamic multi-swarm particle swarm optimizer algorithm", in *Proceedings of SPIE Symposium on Optics East*, vol. 5998, pp. 191-197, October 2005.
5. J. J. Liang and P. N. Suganthan, "Dynamic Multi-Swarm Particle Swarm Optimizer with Local Search," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC 2005)*, vol. 1, pp.522 - 528, Sept. 2005
6. J. J. Liang, and P. N. Suganthan, "Dynamic Multi-Swarm Particle Swarm Optimizer," in *Proceedings of IEEE International Swarm Intelligence Symposium (SIS 2005)*, pp. 124-129, 2005.
7. J. J. Liang, P. N. Suganthan and K. Deb, "Novel composition test functions for numerical global optimization," in *Proceedings of IEEE International Swarm Intelligence Symposium (SIS 2005)*, pp. 68-75, 2005.
8. J. J. Liang, A. K. Qin, P. N. Suganthan and S. Baskar, "Evaluation of Comprehensive Learning Particle Swarm Optimizer," in *2004 11th Int. Conf.*

AUTHOR'S PUBLICATIONSS

- on Neural Information Processing (ICONIP 2004)*, vol. 3316, pp. 230-235, Nov., 2004.
9. J. J. Liang, A. K. Qin, P. N. Suganthan and S. Baskar, "Particle Swarm Optimization Algorithms with Novel Learning Strategies," in *2004 Int. Conf. on Systems, Man and Cybernetics (SMC 2004)*, vol. 4, pp. 3659 - 3664, Oct. 2004.
 10. K. Qin, P. N. Suganthan and J. J. Liang "A new generalized LVQ algorithm via harmonic to minimum distance measure transition," in *IEEE Int. Conf. on systems, man and cybernetics 2004 (SMC 2004)*, vol. 5, pp. 4821 - 4825, Oct. 2004.

BIBLIOGRAPHY

- [1] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the Sixth International Symposium on Micromachine and Human Science*, Nagoya, Japan, pp. 39-43, 1995.
- [2] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks*, Piscataway, NJ, pp. 1942-1948, 1995.
- [3] W. C. Davidon, "Variable metric algorithms for minimization," *Argonne National Lab. Report*, 1959
- [4] R. Fletcher and M. J. D. Powell, "A rapidly convergent descent method for minimization," *Computer Journal*, vol. 6, pp. 163-168, 1963.
- [5] C. G. Broyden, "The Convergence of a Class of Double-Rank Minimization Algorithms", Parts 1 and 2, *Journal of the Institute of Mathematics and Its Applications*, vol. 6, pp. 76-90, 1970 and vol. 6, pp. 222-231, 1970.
- [6] R. Fletcher, "A new approach to variable metric methods," *Computer Journal*, vol. 13, pp. 317-322, 1970.
- [7] D. Goldfarb, "A family of variable metric methods derived by variational means," *Mathematics of Computation*, vol. 24, pp. 23-26, 1970.
- [8] D.F. Shanno, "Conditioning of quasi-Newton methods for function minimization," *Mathematical Center, Amsterdam*, Holland, 1970.
- [9] Barricelli, Nils Aall, "Esempi numerici di processi di evoluzione," *Methodos*, pp. 45-68, 1954.
- [10] Fraser, Alex, "Simulation of genetic systems by automatic digital computers. I. Introduction". *Aust. J. Biol. Sci.* vol. 10, pp. 484-491, 1957.
- [11] H. J. Bremermann, "The evolution of intelligence. The nervous system as a model of its environment", *Technical report*, no.1, Dept. Mathematics, Univ. Washington, Seattle, July, 1958.
- [12] J. H. Holland, "Outline for a logical theory of adaptive systems," *Journal of the Association for Computing Machinery*, vol. 3, pp. 297-314, 1962.
- [13] J. H. Holland, *Adaptation in natural and artificial systems*. The University of Michigan Press, Ann Arbor, MI, 1975.

BIBLIOGRAPHY

- [14] L. J. Fogel, "Toward inductive inference automata," In *Proceedings of the International Federation for Information Processing Congress*, pp. 395-399, 1962.
- [15] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence through Simulated Evolution*, Wiley, New York, 1966.
- [16] I. Rechenberg, "Cybernetic solution path of an experimental problem," *Royal Aircraft Establishment*, Library translation No. 1122, Farnborough, Hants., UK, August 1965.
- [17] I. Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, Stuttgart, 1973.
- [18] I. Rechenberg, "Evolutionsstrategie", *Werkstatt Bionik und Evolutionstechnik*, Frommann- Holzboog, vol. 1, Stuttgart, 1994.
- [19] H.-P. Schwefel, Kybernetische, "Evolution als Strategie der experimentellen Forschung in der Stromungstechnik. Diplomarbeit," Technische Universität Berlin, 1965.
- [20] H.-P. Schwefel. "Naimerische Optimierung von Computer-Modellen mittels der Evolutaonsstratrgcc," in *Interdisciplinary Systems Research*. Birkhauser, Basel, vol. 26, 1977.
- [21] J. R. Koza, "Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems," *Stanford University Computer Science Department technical report STAN-CS-90-1314*, 1990.
- [22] J. R. Koza, "Genetic Programming: On the Programming of Computers by Means of Natural Selection", *MIT Press*, 1992.
- [23] C. M. Bishop, "Neural Networks for Pattern Recognition," chapter 7, pp. 253-294. Oxford University Press, 1995.
- [24] J. Kennedy, R. C. Eberhart and Y. Shi, *Swarm Intelligence*, Morgan Kaydmann, 2001.
- [25] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1998)*, pp. 69-73, 1998.
- [26] H. Y. Fan and Y. Shi, "Study on Vmax of particle swarm optimization," in *Proceedings of the Workshop on Particle Swarm Optimization*, IN, 2001.
- [27] S. Yang and M. Wang, and L. Jiao. "A quantum particle swarm

BIBLIOGRAPHY

- optimization", in *Proceedings of IEEE Congress on Evolutionary Computation*, vol. 1. pp. 320-324, June 2004.
- [28] J. S. Vesterstroem, J. Riget, and T. Krink. "Division of label in particle swarm optimisation", in *Proceedings of IEEE Congress on Evolutionary Computation (CEC 2002)*, vol. 2, pp. 1570-1575, May 2002.
- [29] T. Krink, J. S. Vesterstroem, and J. Riget, "Particle swarm optimisation with spatial particle extension," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC 2002)*, vol. 2, pp. 1474 - 1479, 2002.
- [30] D. N. Wike, "Analysis of the particle swarm optimization algorithm," *Master of Engineering thesis*, Dept of Mechanical and Aeronautical Engineering., University of Pretoria, South Africa, 2005.
- [31] Y. Shi and R. C. Eberhart, " Fuzzy adaptive particle swarm optimization," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC 2001)*, vol. 1, pp. 101-106, 2001.
- [32] P. J. Angeline, "Using selection to improve particle swarm optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC1998)*, pp. 84-89, 1998.
- [33] P. J. Angeline, "Evolutionary optimization versus particle swarm optimization: philosophy and performance differences," in *Evolutionary Programming VII: Proceedings of the Seventh Annual Conference on Evolutionary Programming*, pp. 601-610, 1998.
- [34] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics*, pp. 4104-4109, 1997.
- [35] A. Cervantes, I. Galvan and P. Isasi, "A comparison between the Pittsburgh and Michigan approaches for the binary PSO algorithm," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2000)*, vol. 1, pp. 290 - 297, 2005.
- [36] T. O. Ting, M. V. C. Rao and C. K. Loo, "A novel approach for unit commitment problem via an effective hybrid particle swarm optimization," *IEEE Transactions on Power Systems*, vol. 21, pp. 411 - 418, Feb. 2006.
- [37] D. K. Agrafiotis and W. Cedeno, "Training feedforward neural networks using multi-phase particle swarm optimization," *Journal of Medicinal Chemistry*, pp. 1098-1107, 2002.

BIBLIOGRAGHY

- [38] C. K. Mohan and B. Al-kazemi, "Discrete particle swarm optimization," in *Proceedings. of Workshop on Particle Swarm Optimization 2001*, Indianapolis, Indiana, 2001.
- [39] R. Rastegar, M.R. Meybodi and K. Badie, "A new discrete binary particle swarm optimization based on learning automata", in *Proceedings of 2004 International Conference on Machine Learning and Applications*, pp. 456 - 462, 2004.
- [40] F. Afshinmanesh,A. Marandi and A. Rahimi-Kian, "A Novel Binary Particle Swarm Optimization Method Using Artificial Immune System," in *The International Conference on Computer as a Tool (EUROCON 2005)*, vol. 1, pp. 217 - 220, 2005.
- [41] T. Peram, K. Veeramachaneni, and C. K. Mohan, "Fitness-distance-ratio based particle swarm optimization," in *Proceedings IEEE Swarm Intelligence System (SIS 2003)*, pp. 174-181, 2003.
- [42] M. Clerc, "The swarm and the queen: towards a deterministic and adaptive particle swarm optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1999)*, pp. 1951-1957, 1999.
- [43] 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.
- [44] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proceedings. of the IEEE Congress on Evolutionary Computation (CEC 2000)*, pp. 84-88, 2000.
- [45] A. Ratnaweera, S.Halgamuge, and H. Watson, "Particle swarm optimization with self-adaptive acceleration coefficients," in *Proceedings of the 1st International Conference on Fuzzy Systems and Knowledge Discovery 2002 (FSKD 2002)*, pp. 264-268, 2003.
- [46] A. Ratnaweera, S. Halgamuge, and H. Watson, "Self-organizing hierachical particle swarm optimizer with time varying accelerating coefficients," *IEEE Transactions on Evolutionary Computation*, vol. 8(3), pp. 240-255, 2004.
- [47] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Evolutionary Programming VII: Proceedings of the Seventh Annual Conference on Evolutionary Programming*, pp. 591-600, 1998.

BIBLIOGRAPHY

- [48] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the 1999 Congress on Evolutionary Computation*, pp. 1945-1950, 1999.
- [49] R. C. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2001)*, pp. 81-86, 2001.
- [50] J. Kennedy, "Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC 1999)*, pp. 1931-1938, 1999.
- [51] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002)*, pp. 1671-1676, 2002.
- [52] P. N. Suganthan, "Particle swarm optimizer with neighborhood operator," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1999)*, pp. 1958-1962, 1999.
- [53] X. Hu and R. C. Eberhart, "Multiobjective optimization using dynamic neighborhood particle swarm optimization," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC 2002)*, pp. 1677-1681, 2002.
- [54] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler, maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8 (3), pp. 204--210, 2004.
- [55] M. Lovbjerg, T. K. Rasmussen, and T. Krink, "Hybrid particle swarm optimiser with breeding and subpopulations," in *Proceedings of the Genetic and Evolutionary Computation Conference 2001 (GECCO 2001)*, 2001.
- [56] K. E. Parsopoulos and M. N. Vrahatis, "UPSO- A Unified Particle Swarm Optimization Scheme," *Lecture Series on Computational Sciences*, pp. 868-873, 2004.
- [57] K. E. Parsopoulos and M. N. Vrahatis, "Unified particle swarm optimization for tackling operations research problems," in *Proceedings of 2005 IEEE Swarm Intelligence Symposium*, pp. 53-59, 2005.
- [58] K. E. Parsopoulos and M. N. Vrahatis, "Unified Particle Swarm Optimization in Dynamic Environments," *Lecture Series on Computational Sciences*, vol. 3449, pp. 590-599, 2005.
- [59] K. E. Parsopoulos and M. N. Vrahatis, "Unified Particle Swarm

BIBLIOGRAGHY

- Optimization for Solving Constrained Engineering Optimization Problems," *Lecture Series on Computational Sciences*, vol. 3612, pp. 582-591, 2005.
- [60] V. Miranda and N. Fonseca, "New evolutionary particle swarm algorithm (EPSO) applied to voltage/VAR control," in *Proceedings of The 14th Power Systems Computation Conference (PSCC 2002)*, pp. 87 -93, 2002.
- [61] M. Lovbjerg and T. Krink, "Extending particle swarm optimisers with self-organized criticality," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002)*, Honolulu, Hawaii USA, 2002.
- [62] T. Krink and M. Lovbjerg, "The LifeCycle model: combining particle swarm optimization, genetic algorithms and hill-climbers," in *Proceedings of Parallel Problem Solving from Nature (PPSN 2002)*, pp. 621-630, September, 2002.
- [63] T. M. Blackwel and P. J. Bentley, "Don't push me! Collision-avoiding swarms," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002)*, Honolulu, Hawaii USA, 2002.
- [64] K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimization method in multiobjective problems," in *Proceedings of the 2002 ACM Symposium on Applied Coputing (SAC 2002)*, pp. 603-607, 2002.
- [65] J. Robinson, S. Sinton, and Y. Rahmat-Samii, "Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna," in *IEEE Antennas and Propagation Society International Symposium and URSI National Radio Science Meeting*, vol.1, pp. 16-21, San Antonio, TX, 2002.
- [66] T. Hendtlass and M. A. Randall, "A survey of ant colony and particle swarm meta-heuristics and their application to discrete optimisation problems," in *Proceedings of the Inaugural Workshop on Artificial Life (AL 2001)*, pp. 15-25, 2001.
- [67] T. A. Hendtlass, "combined swarm differential evolution algorithm for optimization problems," in *Lecture Notes in Computer Science(LNCS) No.2070: Proceedings of the 14th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE 2001)*, pp. 11-18, 2001.
- [68] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol.

BIBLIOGRAGHY

- 8(3), pp. 225-239, 2004.
- [69] X. Xie, W. Zhang, and Z. Yang, "A dissipative particle swarm optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002)*, pp. 1945-1950, 2002.
 - [70] 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(6), pp. 211-224, 2004.
 - [71] C. Wei, Z. He, Y. Zhang, and W. Pei, "Swarm directions embedded in fast evolutionary programming," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002)*, vol. 2, pp. 1278-1283, 2002.
 - [72] K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimization method for constrained optimization problems," in *Proceedings of the Euro-International Symposium on Computational Intelligence*, pp. 214-220, 2002.
 - [73] X. Hu and R. C. Eberhart, "Solving constrained nonlinear optimization problems with particle swarm optimization," in *Proceedings of the Sixth World Multi conference on Systemic, Cybernetics and Informatics 2002 (SCI 2002)*, pp. 203-206, 2002.
 - [74] T. Ray and K. M. Liew, "A swarm with an effective information sharing mechanism for unconstrained and constrained single objective optimization problem," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC 2001)*, pp. 75-80, 2001.
 - [75] N. Li, F. Liu, D. Sun, and C. Huang, "Particle Swarm Optimization for Constrained Layout Optimization" in *Proceedings of the Fifth World Congress on Intelligent Control and Automation*, vol.3, pp. 2214-2218, 2004.
 - [76] J. Chunlin, "A Revised Particle Swarm Optimization Approach for Multi-Objective and Multi-Constraint Optimization," in *Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2004)*, 2004.
 - [77] G. Coath and S. K. Halgamuge, "A comparison of constraint-handling methods for the application of particle swarm optimization to constrained nonlinear optimization problems," in *Proceedings of IEEE Congress on Evolutionary Computation 2003 (CEC 2003)*, pp. 2419-2425, 2003.
 - [78] G. T. Pulido and C. A. Coello Coello, "A Constraint-Handling Mechanism for Particle Swarm Optimization," in *Proceedings of IEEE Congress on*

BIBLIOGRAPHY

- Evolutionary Computation (CEC 2004)*, vol. 2, pp. 1396-1403, June 2004.
- [79] Shi and R. A. Krohling, "Co-evolutionary particle swarm optimization to solve Min-max problems," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002)*, vol. 2, pp. 1682-1687, 2002.
 - [80] R. A. Krohling, H. Knidel, and Y. Shi, "Solving numerical equations of hydraulic problems using particle swarm optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002)*, vol.2, pp. 1688-1690, 2002.
 - [81] E. C. Laskari, K. E. Parsopoulos, and M. N. Vrahatis, "Particle swarm optimization for minimax problems," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002)*, vol.2, pp. 1576 -1581, 2002.
 - [82] J. E. Fieldsend, "Multi-objective particle swarm optimization methods," *Technical Report #419*, Department of Computer Science, University of Exeter, March 2004.
 - [83] T. Ray, and K. M. Liew, "A swarm metaphor for multiobjective design optimization," *Engineering Optimization*, vol. 34(2), pp. 141-153, 2002.
 - [84] K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimization method in multiobjective problems," in *Proceedings of the ACM Symposium on Applied Computing 2002 (SAC 2002)*, pp. 603-607, 2002.
 - [85] X. Hu and R. C. Eberhart, "Multiobjective optimization using dynamic neighborhood particle swarm optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002)*, vol. 2, pp. 1677-1681, 2002.
 - [86] J. E. Fieldsend and S. Singh, "A multi-objective algorithm based upon particle swarm optimization, an efficient data structure and turbulence," in *Proceedings of the 2002 U.K. Workshop on Computational Intelligence*, pp. 37-44, 2002.
 - [87] X. Hu, R. C. Eberhart and Y. Shi, "Particle swarm with extended memory for multiobjective optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium 2003 (SIS 2003)*, pp. 193-197, 2003.
 - [88] S. Mostaghim and J. R. Teich, "The role of e-dominance in multi-objective particle swarm optimization methods," in *Proceedings of IEEE Congress on Evolutionary Computation 2003 (CEC 2003)*, pp. 1764-1771, 2003.
 - [89] S. Mostaghim and J. R. Teich, "Strategies for finding good local guides in

BIBLIOGRAPHY

- multi-objective particle swarm optimization (mopso)," in *Proceedings of IEEE 2003 Swarm Intelligence Symposium*, pp. 26-33, 2003.
- [90] T. Bartz-Beielstein, P. Limbourg, J. Mehnen, K. Schmitt, K. E. Parsopoulos, M. N. Vrahatis, "Particle swarm optimizers for Pareto optimization with enhanced archiving techniques," in *Proceedings of 2003 Congress on Evolutionary Computation (CEC 2003)*, vol. 3, pp. 1780 - 1787, Canberra, Piscataway NJ, 2003.
- [91] Carlos A. Coello Coello, G. T. Pulido and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8(3), pp.256-279, 2004.
- [92] S. Mostaghim, and J. R. Teich, "Strategies for finding local guides in multi-objective particle swarm optimization (MOPSO)," in *Proceedings of the IEEE Swarm Intelligence Symposium 2003 (SIS 2003)*, pp. 26-33, 2003.
- [93] X. Li, "A non-dominated sorting particle swarm optimizer for multiobjective optimization," in *Lecture Notes in Computer Science (LNCS) No. 2723: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2003)*, pp. 37-48, 2003.
- [94] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A fast and elitist multi-objective genetic algorithms: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6(2), pp. 182-197, 2002.
- [95] Carlos. A. Coello Coello and M. S. Lechunga, "MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization," in *Proceedings of the 2002 Congress on Evolutionary Computation, part of the 2002 IEEE World Congress on Computational Intelligence*, pp. 2051-1056, 2002.
- [96] G. T. Pulido and C. A. Coello Coello, "Using clustering techniques to improve the performance of a multi-objective particle swarm optimizer," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2004)*, pp. 225-237, 2004.
- [97] R. Brits, A. P.Engelbrecht, and F. van den Bergh, "A niching particle swarm optimizer," in *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning 2002 (SEAL 2002)*, pp. 692-696, Singapore, 2002.
- [98] R. Brits, A. P. Engelbrecht and F. van den Bergh, "Scalability of niche PSO," in *Proceedings of the IEEE Swarm Intelligence Symposium 2003 (SIS 2003)*, pp. 228-234, 2003.

BIBLIOGRAGHY

- [99] R. C. Eberhart and Y. Shi, "Tracking and optimizing dynamic systems with particle swarms," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2001)*, pp. 94-97, 2001.
- [100] A. Carlisle and G. Dozier, "Tracking changing extrema with particle swarm optimizer," *Technical Report CSSE01-08*, Auburn University, 2001.
- [101] K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimizer in noisy and continuously changing environments," in *Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing (ASC 2001)*, pp. 289-294, Cancun, Mexico, 2001.
- [102] Carlisle and G. Dozier, "Adapting particle swarm optimization to dynamic environments," in *Proceedings of International Conference on Artificial Intelligence (ICAI 2000)*, pp. 429-434, 2000.
- [103] R. C. Eberhart and Y. Shi, "Evolving artificial neural networks," in *Proceedings of International Conference on Neural Networks and Brain*, pp. 5-13, 1998.
- [104] R. C. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2001)*, vol. 1, pp. 81-86, 2001.
- [105] M. P. wachowiak, R. Smolikova, Y. 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, 2004.
- [106] L. Messerschmidt and A. P. Engelbrecht, "Learning to play games using a PSO-based competitive learning approach," in *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning 2002 (SEAL 2002)*, pp. 444-448, 2002.
- [107] R. C. Eberhart and X. Hu, "Human tremor analysis using particle swarm optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1999)*, pp. 1927-1930, 1999.
- [108] 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 Transactions on Power Systems*, vol. 15, pp. 1232-1239, 2000.
- [109] W. C. Zhong, J. Liu, M. Z. Xue and L. C. Jiao, "A multiagent genetic

BIBLIOGRAGHY

- algorithm for global numerical optimization," *IEEE Transactions on Systems, Man and Cybernetics (Part B)*, vol. 34, pp. 1128-1141, April 2004.
- [110] Coello Coello, C. A., Toscano Pulido, G., and Salazar Lechuga, M. "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8(3), pp.256-279, June 2004.
- [111] X. Yao Y Liu and G. M. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3(2), pp. 82-102, 1999.
- [112] C-Y Lee and X. Yao, "Evolutionary Programming Using mutations Based on the Levy Probability Distribution", *IEEE Transactions on Evolutionary Computation*, vol. 8(1), pp. 1-13, February 2004.
- [113] S. C. Esquivel and C. A. Coello Coello, "On the Use of Particle Swarm Optimization with Multimodal Functions," in *Proceedings of 2003 Congress on Evolutionary Computation (CEC 2003)*, vol. 2, pp. 1130-1136, December 2003.
- [114] Y. W. Leung and Y. P. Wang, "An orthogonal genetic algorithm with quantization for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 5(1), pp. 41-53, Feb. 2001.
- [115] R. Salomon, "Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions." *BioSystems*, vol. 39, pp.263-278, 1996.
- [116] Hans-Paul Paul Schwefel, *Evolution and Optimum Seeking*, Publisher: John Wiley & Sons, Inc, 1993.
- [117] Lehmann, E. L. *Nonparametric Statistical Methods Based on Ranks*. New York: McGraw-Hill, 1975.
- [118] D. Whitley, D. Rana, J. Dzubera and E. Mathias, "Evaluating evolutionary algorithms," *Artificial Intelligence*, vol. 85, pp. 245-276, 1996.
- [119] J. J. Liang, P. N. Suganthan and K. Deb, "Novel composition test functions for numerical global optimization," in *Proceedings of Swarm Intelligence Sym.*, pp. 68- 75, June 2005.
- [120] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 67-82, 1997.
- [121] R. Storn and K. Price, "Differential evolution-A simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of*

BIBLIOGRAPHY

- Global Optimization*, vol. 11, pp. 341-359, 1997.
- [122] 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.
 - [123] Fletcher, R., "A New Approach to Variable Metric Algorithms," *Computer Journal*, vol. 13, pp 317-322, 1970.
 - [124] Goldfarb, D., "A Family of Variable Metric Updates Derived by Variational Means," *Mathematics of Computing*, vol. 24, pp 23-26, 1970.
 - [125] Shanno, D.F., "Conditioning of Quasi-Newton Methods for Function Minimization," *Mathematics of Computing*, vol. 24, pp 647-656, 1970.
 - [126] Davidon, W.C., "Variable Metric Method for Minimization," *A.E.C. Research and Development Report*, ANL-5990, 1959.
 - [127] Fletcher, R. and M.J.D. Powell, "A Rapidly Convergent Descent Method for Minimization," *Computer Journal*, Vol. 6, pp 163-168, 1963.
 - [128] 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 CEC 2005 Special Session on Real-Parameter Optimization", *Technical Report, Nanyang Technological University, Singapore*, May 2005 and *KanGAL Report #2005005*, IIT Kanpur, India.
 - [129] C. Garcia-Martinez and M. Lozano, "Hybrid real-coded genetic algorithms with female and male differentiation," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, vol.1, pp.896-903, 2005.
 - [130] D. Molina, F. Herrera and M. Lozano "Adaptive Local Search Parameters for Real-Coded Memetic Algorithms," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, vol.1, pp.888-895, 2005.
 - [131] P. Posik, "Real-Parameter Optimization Using the Mutation Step Co-evolution," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, vol.1, pp.872-879, 2005.
 - [132] J. Ronkkonen, S. Kukkonen and K.V. Price , "Real-Parameter Optimization with Differential Evolution," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, vol.1, pp.506-513, 2005.
 - [133] B. Yuan and M. Gallagher, "Experimental results for the special session on

BIBLIOGRAPHY

- real-parameter optimization at CEC 2005: a simple, continuous EDA," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC 2005)* vol.2, pp.1792-1799, 2005.
- [134] A. Auger and N. Hansen, "A restart CMA evolution strategy with increasing population size," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, vol.2, pp.1769-1776, 2005.
- [135] A. Sinha, S. Tiwari and K. Deb "A Population-Based, Steady-State Procedure for Real-Parameter Optimization," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, vol.1, pp.514-521, 2005.
- [136] A. Auger and N. Hansen, "Performance evaluation of an advanced local search evolutionary algorithm," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, vol.2, pp.1777-1784, 2005
- [137] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, vol.2, pp.1785-1791, 2005.
- [138] P. J. Ballester, J. Stephenson, J. N. Carter and K. Gallagher, "Real-Parameter Optimization Performance Study on the CEC-2005 benchmark with SPC-PNX," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, vol.1, pp.498-505, 2005.
- [139] J. J. Liang, Thomas Philip Runarsson, Efren Mezura-Montes, Maurice Clerc, P. N. Suganthan, Carlos A. Coello Coello & K. Deb, "Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization", *Technical Report, Nanyang Technological University*, Singapore, Dec. 2005.
- [140] E. Mezura-Montes and C. A. C. Coello, "A Numerical Comparison of some Multiobjective-Based Techniques to Handle Constraints in Genetic Algorithms", *Technical Report EVOCINV-0302002*, Evolutionary Computation Group at CINVESTAV, México, 2002.
- [141] A. Homaifar, S. H. Y. Lai, and X. Qi, "Constrained Optimization via Genetic Algorithms," *Simulation*, vol. 62(4), pp. 242-254, 1994.
- [142] R. Courant, "Variational methods for the solution of problems in equilibrium and vibrations,Bull," Amer. Math. Soc., vol. 49, pp. 1–23, 1943;

BIBLIOGRAGHY

- reprinted in *Int. J. Numer. Meth. Engrg.*, vol. 37, pp. 643–645, 1994.
- [143] J. Joines and C. Houck, "On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs," in *Proc. of the first IEEE Conference on Evolutionary Computation (CEC 1994)*, pp. 579-584, 1994.
 - [144] Z. Michalewicz and N. Attia, "Evolutionary Optimization of Constrained Problems," in *Proceedings of the third Annual Conference on Evolutionary Programming*, pp. 98-108, 1994.
 - [145] A. E. Smith and D. M. Tate, "Genetic Optimization Using a Penalty Function," in *Proceedings of the fifth International Conference on Genetic Algorithms*, pp. 499-503, 1993.
 - [146] A. E. Eiben, J. K. van der Hauw, and J. I. van Hermert, "Adaptive penalties for evolutionary graph coloring," in *Artificial Evolution' 97*, pp. 95-106, Berlin, 1998.
 - [147] Z. Michalewicz, "Genetic Algorithms + Data Structures = Evolution Programs," Springer-Verlag, 1992.
 - [148] X. Hu and R. C. Eberhart, "Solving constrained nonlinear optimization problems with particle swarm optimization," in *Proceedings of the Sixth World Multiconference on Systemics, Cybernetics and Informatics 2002 (SCI 2002)*, pp. 203-206, 2002.
 - [149] Carlos A. Coello Coello, "Self-Adaptation Penalties for GA-based Optimization," in *Proceedings. of the 1999 Congress on Evolutionary Computation (CEC 1999)*, vol. 1, pp. 573-580, 1999.
 - [150] Carlos A. Coello Coello. "A survey of constraint handling techniques used with evolutionary algorithms," *Technical Report Lania-RI-9904*, Laboratorio Nacional de Informtica Avanzada, 1999.
 - [151] T. Ray, "Constraint Robust Optimal Design using a Multiobjective Evolutionary Algorithm," in *Proceedings of Congress on Evolutionary Computation 2002 (CEC 2002)*, vol.1, pp. 419-424, 2002.
 - [152] J. Brest, V. Zumer and M. S. Maucec, "Self-Adaptive Differential Evolution Algorithm in Constrained Real-Parameter Optimization," in *Proceedings. of the sixth IEEE Conference on Evolutionary Computation (CEC 2006)*, pp. 215-222 2006.
 - [153] K. Zielinski and R. Laur, "Constrained Singel-Objective Optimization

BIBLIOGRAPHY

- Using Differential Evolution", in *Proceedings of the sixth IEEE Conference on Evolutionary Computation (CEC 2006)*, pp. 223-230, 2006.
- [154] T. Takahama and S. Sakai, "Constrained optimization by constrained particle swarm optimizer with ε -level control," in *Proceedings of the 4th IEEE International Workshop on Soft Computing as Transdisciplinary Science and Technology (WSTST'05)*, Muroran, Japan, pp. 1019-1029, May 2005
- [155] T. Takahama and S. Sakai, "Constrained Optimization by the ε Constrained Differential Evolution with Gradient-Based Mutation and Feasible Elites," in *Proceedings of the sixth IEEE Conference on Evolutionary Computation*, pp. 1-8, 2006.
- [156] A. E. Munoz-Zavala, A. Hernandez-Aguirre, E. R. Villa-Diharce and S. Botello-Rionda, "PESO+ for Constrained optimization," in *Proceedings of the sixth IEEE Conference on Evolutionary Computation (CEC 2006)*, pp. 231-238, 2006.
- [157] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. The Institute of Electrical and Electronic Engineers, New York, 1995.
- [158] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 4(3), pp. 284-294, September 2000.
- [159] Carlos A. Coello Coello, "Treating Constraints as Objectives for Single-Objective Evolutionary Optimization," *Engineering Optimization*, vol. 32(3), pp. 275-308, 2000.
- [160] E. Camponogara and S. N. Talukdar, "A Genetic Algorithm for Constrained and Multiobjective Optimization," in *3rd Nordic Workshop on Genetic Algorithms and Their Applications (3NWGA)*, pp. 49-62, 1997.
- [161] T. Ray, T. Kang and S. K. Chye, "An Evolutionary Algorithm for Constrained Optimization," in *Proceedings of the Genetic and Evolutionary computation Conference (FECCO'2000)*, pp. 771-777, 2000.
- [162] Carlos. A. Coello Coello, "Constraint-handling using an evolutionary multiobjective optimization technique," *Civil Engineering and Environmental Systems*, vol. 17, pp. 319-346, 2000.
- [163] E. Mezura-Montes and Carlos A. Coello Coello, "A Numerical Comparison

BIBLIOGRAGHY

- of some Multiobjective-based Techniques to Handle Constraints in Genetic Algorithms," *Technical Report EVOCINV-03-2002*, Evolutionary Computation Group at CINVESTAV-IPN, Mexico, D.F. 07300, September 2002
- [164] S. Kukkonen and J. Lampinen, "Constrained Real-Parameter Optimization with Generalized Differential Evolution," in *Proceedings of the sixth IEEE Conference on Evolutionary Computation*, pp. 207 - 214, 2006.
 - [165] E. Mezura-Montes, J. Velazquez-Reyes and Carlos A. Coello Coello, "Modified Differential Evolution for Constrained Optimization," in *Proceedings of the sixth IEEE Conference on Evolutionary Computation*, pp. 25- 2, 2006.
 - [166] M. Fatih Tasgetiren and P. N. Suganthan, "A Multi-Populated Differential Evolution Algorithm for Solving Constrained Optimization Problem," in *Proceedings of the sixth IEEE Conference on Evolutionary Computation*, pp. 33-40, 2006.
 - [167] A. Sinha, a. Srinivasan and K. Deb, "A Population-Based, Parent Centric Procedure for Constrained Real-Parameter Optimization," in *Proceedings of the sixth IEEE Conference on Evolutionary Computation*, pp. 239-245, 2006.
 - [168] V. L. Huang, A. K. Qin and P. N. Suganthan, "Self-adaptive Differential Evolution Algorithm for Constrained Real-Parameter Optimiation," in *Proceedings of the sixth IEEE Conference on Evolutionary Computation*, pp. 17-24, 2006.
 - [169] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. England : John Wiley & Sons Ltd , 2001.
 - [170] X. Hu, R. C. Eberhart and Y. Shi, "Particle swarm with extended memory for multiobjective optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium 2003 (SIS 2003)*, pp. 193-197, 2003.
 - [171] Carlos A. Coello Coello, G. T. Pulido and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization". *IEEE Transactions on Evolutionary Computation*, vol. 8(3), pp. 256-279, 2004.
 - [172] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A fast and elitist multi-objective genetic algorithms: NSGA-II," *IEEE Transactions on*

BIBLIOGRAPHY

- Evolutionary Computation*, vol. 6(2), pp. 182-197, 2002.
- [173] E. Zitzler, , L. Thiele, M. Laumanns, C. M. Fonseca, and V.G. Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *TIK-Report No. 139*, Institut für Technische Informatik und Kommunikationsnetze, Switzerland, 2002.
- [174] E. Udd, *Fiber Optic Smart Structure*, Wiley, New York, 1995.
- [175] C. Z. Shi, C. C. Chan, W.Jin, Y.B. Liao, Y. Zhou and M. S. Demokan, "Improving the performance of a FBG sensor network using a genetic algorithm," *Sensors and Actuators A* 107, pp. 57-61, 2003.
- [176] C. Z. Shi, C. C. Chan, W.Jin, Y.B. Liao, Y. Zhou and M. S. Demokan, "Improving the performance of a FBG Sensors in a WDM Network Using a Simulated Annealing Technique," *IEEE Photonics Technol. Lett.*, vol. 16, No.1, January 2004.
- [177] J. M. Gong, J. M. K. MacAlphine, C. C. Chan, W.Jin, M. Zhang, Y. B. Liao, "A novel wavelength detection technique for fiber Bragg grating sensors," *IEEE Photonics Technol. Lett.*, vol. 14 (5), pp.678-680, 2002.
- [178] M. G. Xu, H. Geiger, J. P. Dakin, "Modeling and performance analysis of a fiber Bragg grating interrogation system using an acoustoptic tunable filter", *IEEE Journal of Lightwave Technol.* vol. 14, pp. 391-396, 1996.
- [179] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multi-objective optimization", *Evolutionary Computation Journal*, vol. 3(1), pp. 1-16, 1995.
- [180] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proceedings of the First International Conference on Genetic Algorithms*, pp.93-100, 1987.
- [181] F. Kursawe, "A variant of evolution strategies for vector optimization," in *Parallel Problem Solving From Natur I (PPSN-I)*, pp. 193–197, 1990.
- [182] K. Deb and S. Tiwari. "Omni-optimizer: A Procedure for Single and Multi-objective Optimization," in Carlos A. Coello Coello and Arturo Hernández Aguirre and Eckart Zitzler (editors), *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005*, pp.47-61, March 2005.

Appendix A

Definitions of the 25 CEC'05 Test Functions

1. Unimodal Functions:

D : dimensions

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

f_bias 1*25 vector, record all the 25 function's f_bias

1) F_1 : Shifted Sphere Function

$$F_1(\mathbf{x}) = \sum_{i=1}^D z_i^2 + f_bias_1, \mathbf{z} = \mathbf{x} - \mathbf{o}, \mathbf{x} = [x_1, x_2, ..., x_D] \tag{A-1}$$

$\mathbf{x} \in [-100,100]^D$, Global optimum: $\mathbf{x}^* = \mathbf{o}$, $F_1(\mathbf{x}^*) = f_bias_1 = -450$

Properties: Unimodal; Shifted; Separable; Scalable

Fig. A-1 3-D Map for 2-D Function F_1

APPENDIX A

2) F_2 : Shifted Schwefel’s Problem 1.2

$$F_2(\mathbf{x}) = \sum_{i=1}^D \left(\sum_{j=1}^i z_j \right)^2 + f_bias_2, \quad \mathbf{z} = \mathbf{x} - \mathbf{o}, \mathbf{x} = [x_1, x_2, \dots, x_D] \tag{A-2}$$

$\mathbf{x} \in [-100, 100]^D$, Global optimum $\mathbf{x}^* = \mathbf{o}$, $F_2(\mathbf{x}^*) = f_bias_2 = -450$

Properties: Unimodal; Shifted; Non-separable; Scalable.

Fig. A-2 3-D Map for 2-D Function F_2

3) F_3 : Shifted Rotated High Conditioned Elliptic Function

$$F_3(\mathbf{x}) = \sum_{i=1}^D (10^6)^{\frac{i-1}{D-1}} z_i^2 + f_bias_3, \quad \mathbf{z} = (\mathbf{x} - \mathbf{o}) * \mathbf{M}, \mathbf{x} = [x_1, x_2, \dots, x_D] \tag{A-3}$$

$\mathbf{M}$: orthogonal matrix

$\mathbf{x} \in [-100, 100]^D$, Global optimum $\mathbf{x}^* = \mathbf{o}$, $F_3(\mathbf{x}^*) = f_bias_3 = -450$

Properties: Unimodal; Shifted; Rotated; Non-separable; Scalable.

APPENDIX A

Fig.A-3 3-D Map for 2-D Function F_3

4) F_4 : Shifted Schwefel’s Problem 1.2 with Noise in Fitness

$$F_4(\mathbf{x}) = \left(\sum_{i=1}^D \left(\sum_{j=1}^i z_j \right)^2 \right) * (1 + 0.4 |N(0,1)|) + f_bias_4, \quad \mathbf{z} = \mathbf{x} - \mathbf{o}, \mathbf{x} = [x_1, x_2, \dots, x_D]$$

(A-4)

$\mathbf{x} \in [-100,100]^D$, Global optimum $\mathbf{x}^* = \mathbf{o}$, $F_4(\mathbf{x}^*) = f_bias_4 = -450$

Properties: Unimodal; Shifted; Non-separable; Scalable; Noise in fitness.

Fig. A-4 3-D Map for 2-D Function F_4

5) F_5 : Schwefel’s Problem 2.6 with Global Optimum on Bounds

$$f(\mathbf{x}) = \max\{|x_1 + 2x_2 - 7|, |2x_1 + x_2 - 5|\}, i = 1, \dots, n, \quad \mathbf{x}^* = [1, 3], f(\mathbf{x}^*) = 0 \tag{A-5}$$

Extend to D dimensions:

APPENDIX A

$$F_5(\mathbf{x}) = \max\{|\mathbf{A}_i \mathbf{x} - \mathbf{B}_i|\} + f_bias_5, i = 1, \dots, D, \mathbf{x} = [x_1, x_2, \dots, x_D]$$

$\mathbf{A}$ is a $D \times D$ matrix, a_{ij} are integer random numbers in the range $[-500, 500]$,

$\det(\mathbf{A}) \neq 0$, $\mathbf{A}_i$ is the i^{th} row of $\mathbf{A}$.

$$\mathbf{B}_i = \mathbf{A}_i * \mathbf{o},$$

$$o_i = -100, \text{ for } i = 1, 2, \dots, \lceil D/4 \rceil$$

$$o_i \text{ are random number in the range } [-100, 100], \text{ for } i = \lceil D/4 \rceil + 1, \dots, \lfloor 3D/4 \rfloor - 1$$

$$o_i = 100, \text{ for } i = \lfloor 3D/4 \rfloor, \dots, D$$

$$\mathbf{x} \in [-100, 100]^D, \text{ Global optimum } \mathbf{x}^* = \mathbf{o}, F_5(\mathbf{x}^*) = f_bias_5 = -310$$

Properties: Unimodal; Non-separable; Scalable; If the initialization procedure initializes the population at the bounds; this problem will be solved easily.

Fig. A-5 3-D Map for 2-D Function F_5

2. Basic Multimodal Functions

6) F_6 : Shifted Rosenbrock's Function

$$F_6(\mathbf{x}) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_bias_6, \mathbf{z} = \mathbf{x} - \mathbf{o} + 1, \mathbf{x} = [x_1, x_2, \dots, x_D]$$

(A-6)

$$\mathbf{x} \in [-100, 100]^D, \text{ Global optimum } \mathbf{x}^* = \mathbf{o}, F_6(\mathbf{x}^*) = f_bias_6 = 390$$

Properties: Multi-modal; Shifted; Non-separable; Scalable; Having a very narrow valley from local optimum to global optimum.

Fig. A-6 3-D Map for 2-D Function F_6

7) F_7 : Shifted Rotated Griewank’s Function without Bounds

$$F_7(\mathbf{x}) = \sum_{i=1}^D \frac{z_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 + f_bias_7, \quad \mathbf{z} = (\mathbf{x} - \mathbf{o}) * \mathbf{M}, \mathbf{x} = [x_1, x_2, \dots, x_D]$$

(A-7)

$\mathbf{M}'$: linear transformation matrix, condition number = 3

$$\mathbf{M} = \mathbf{M}'(1 + 0.3|N(0,1)|)$$

Initialize population in $[0, 600]^D$, Global optimum $\mathbf{x}^* = \mathbf{o}$ is outside of the initialization range, $F_7(\mathbf{x}^*) = f_bias_7 = -180$.

Properties: Multi-modal; Rotated; Shifted; Non-separable; Scalable; No bounds for variables $\mathbf{x}$.

Fig. A-7 3-D Map for 2-D Function F_7

APPENDIX A

8) F_8 : Shifted Rotated Ackley's Function with Global Optimum on Bounds

$$F_8(\mathbf{x}) = -20 \exp(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D z_i^2}) - \exp(\frac{1}{D} \sum_{i=1}^D \cos(2\pi z_i)) + 20 + e + f_bias_8 \quad (\text{A-8})$$

$$\mathbf{z} = (\mathbf{x} - \mathbf{o}) * \mathbf{M}$$

$o_{2j-1} = -32, o_{2j}$ are randomly distributed in the search range, for $j = 1, 2, \dots, \lfloor D/2 \rfloor$

$\mathbf{M}$: linear transformation matrix, condition number = 100

$\mathbf{x} \in [-32, 32]^D$, Global optimum $\mathbf{x}^* = \mathbf{o}$, $F_8(\mathbf{x}^*) = f_bias_8 = -140$

Properties: Multi-modal; Rotated; Shifted; Non-separable; Scalable; A's condition number $\text{Cond}(\mathbf{A})$ increases with the number of variables as $O(D^2)$; Global optimum on the bound; If the initialization procedure initializes the population at the bounds, this problem will be solved easily.

Fig. A-8 3-D Map for 2-D Function F_8

9) F_9 : Shifted Rastrigin's Function

$$F_9(\mathbf{x}) = \sum_{i=1}^D (z_i^2 - 10 \cos(2\pi z_i) + 10) + f_bias_9, \quad \mathbf{z} = \mathbf{x} - \mathbf{o}, \quad \mathbf{x} = [x_1, x_2, \dots, x_D] \quad (\text{A-9})$$

$\mathbf{x} \in [-5, 5]^D$, Global optimum $\mathbf{x}^* = \mathbf{o}$, $F_9(\mathbf{x}^*) = f_bias_9 = -330$

Properties: Multi-modal; Shifted ;Separable; Scalable ;Local optima's number is huge.

Fig. A-9 3-D Map for 2-D Function F_9

10) F_{10} : Shifted Rotated Rastrigin’s Function

$$F_{10}(\mathbf{x}) = \sum_{i=1}^D (z_i^2 - 10 \cos(2\pi z_i) + 10) + f_bias_{10} \tag{A-10}$$

$$\mathbf{z} = (\mathbf{x} - \mathbf{o}) * \mathbf{M}, \mathbf{x} = [x_1, x_2, \dots, x_D]$$

$\mathbf{M}$: linear transformation matrix, condition number = 2

$\mathbf{x} \in [-5, 5]^D$, Global optimum $\mathbf{x}^* = \mathbf{o}$, $F_{10}(\mathbf{x}^*) = f_bias_{10} = -330$

Properties: Multi-modal; Shifted; Rotated; Non-separable; Scalable; Local optima’s number is huge.

Fig. A-10 3-D Map for 2-D Function F_{10}

APPENDIX A

11) F_{11} : Shifted Rotated Weierstrass Function

$$F_{11}(\mathbf{x}) = \sum_{i=1}^D \left(\sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k (z_i + 0.5))] \right) - D \sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k \cdot 0.5)] + f_bias_{11} \quad (\text{A-11})$$

$a = 0.5, b = 3, k_{\max} = 20, \mathbf{z} = (\mathbf{x} - \mathbf{o}) * \mathbf{M}, \mathbf{x} = [x_1, x_2, \dots, x_D]$

$\mathbf{M}$: linear transformation matrix, condition number = 5

$\mathbf{x} \in [-0.5, 0.5]^D$, Global optimum $\mathbf{x}^* = \mathbf{o}$, $F_{11}(\mathbf{x}^*) = f_bias_{11} = 90$

Properties: Multi-modal; Shifted; Rotated; Non-separable; Scalable; Continuous but differentiable only on a set of points.

Fig. A-11 3-D Map for 2-D Function F_{11}

12) F_{12} : Schwefel's Problem 2.13

$$F_{12}(\mathbf{x}) = \sum_{i=1}^D (\mathbf{A}_i - \mathbf{B}_i(\mathbf{x}))^2 + f_bias_{12}, \mathbf{x} = [x_1, x_2, \dots, x_D] \quad (\text{A-12})$$

$$\mathbf{A}_i = \sum_{j=1}^D (a_{ij} \sin \alpha_j + b_{ij} \cos \alpha_j), \mathbf{B}_i(\mathbf{x}) = \sum_{j=1}^D (a_{ij} \sin x_j + b_{ij} \cos x_j), \text{ for } i = 1, \dots, D$$

$\mathbf{A}, \mathbf{B}$ are two $D \times D$ matrix, a_{ij}, b_{ij} are integer random numbers in the range $[-100, 100]$, $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \dots, \alpha_D]$, α_j are random numbers in the range $[-\pi, \pi]$.

$\mathbf{x} \in [-\pi, \pi]^D$, Global optimum $\mathbf{x}^* = \boldsymbol{\alpha}$, $F_{12}(\mathbf{x}^*) = f_bias_{12} = -460$

Properties: Multi-modal; Shifted; Non-separable; Scalable.

Fig. A-12 3-D Map for 2-D Function F_{12}

3. Expanded Functions

Using a 2-D function $F(x, y)$ as a starting function, corresponding expanded function is:

$$EF(x_1, x_2, \dots, x_D) = F(x_1, x_2) + F(x_2, x_3) + \dots + F(x_{D-1}, x_D) + F(x_D, x_1)$$

13) F_{13} : Shifted Expanded Griewank's plus Rosenbrock's Function (F8F2)

$$\text{F8: Griewank's Function: } F8(\mathbf{x}) = \sum_{i=1}^D \frac{x_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad (\text{A-13})$$

$$\text{F2: Rosenbrock's Function: } F2(\mathbf{x}) = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2) \quad (\text{A-14})$$

$$\begin{aligned} F8F2(x_1, x_2, \dots, x_D) = & F8(F2(x_1, x_2)) + F8(F2(x_2, x_3)) + \dots + F8(F2(x_{D-1}, x_D)) \\ & + F8(F2(x_D, x_1)) \end{aligned} \quad (\text{A-15})$$

Shift to

$$\begin{aligned} F_{13}(\mathbf{x}) = & F8(F2(z_1, z_2)) + F8(F2(z_2, z_3)) + \dots + F8(F2(z_{D-1}, z_D)) \\ & + F8(F2(z_D, z_1)) + f_bias_{13} \end{aligned} \quad (\text{A-16})$$

$$\mathbf{z} = \mathbf{x} - \mathbf{o} + 1, \quad \mathbf{x} = [x_1, x_2, \dots, x_D]$$

$$\mathbf{x} \in [-5, 5]^D, \text{ Global optimum } \mathbf{x}^* = \mathbf{o}, \quad F_{13}(\mathbf{x}^*) = f_bias_{13}(13) = -130$$

Properties: Multi-modal; Shifted; Non-separable; Scalable

APPENDIX A

Fig. A-13 3-D Map for 2-D Function F_{13}

14) F_{14} : Shifted Rotated Expanded Scaffer’s F6 Function

$$F(x, y) = 0.5 + \frac{(\sin^2(\sqrt{x^2 + y^2}) - 0.5)}{(1 + 0.001(x^2 + y^2))^2} \tag{A-17}$$

Expanded to

$$F_{14}(\mathbf{x}) = EF(z_1, z_2, \dots, z_D) = F(z_1, z_2) + F(z_2, z_3) + \dots + F(z_{D-1}, z_D) + F(z_D, z_1) + f_bias_{14}$$
$$\mathbf{z} = (\mathbf{x} - \mathbf{o}) * \mathbf{M}, \mathbf{x} = [x_1, x_2, \dots, x_D] \tag{A-18}$$

$\mathbf{M}$: linear transformation matrix, condition number = 3

$\mathbf{x} \in [-100, 100]^D$, Global optimum $\mathbf{x}^* = \mathbf{o}$, $F_{14}(\mathbf{x}^*) = f_bias_{14} (14) = -300$

Properties: Multi-modal; Shifted; Non-separable; Scalable.

Fig. A-14 3-D Map for 2-D Function F_{14}

4. Composition functions

In the following composition functions,

Number of basic functions $n = 10$.

D : dimensions

$\mathbf{o}$: $n \times D$ matrix, defines $f_i(\mathbf{x})$'s global optimal positions

bias = [0, 100, 200, 300, 400, 500, 600, 700, 800, 900]. Hence, the first function $f_1(\mathbf{x})$ always the function with the global optimum.

$C = 2000$

15) F_{15} : Hybrid Composition Function 1

$f_{1-2}(\mathbf{x})$: Rastrigin's Function

$$f_i(\mathbf{x}) = \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i) + 10) \quad (\text{A-19})$$

$f_{3-4}(\mathbf{x})$: Weierstrass Function

$$f_i(\mathbf{x}) = \sum_{i=1}^D \left(\sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k (x_i + 0.5))] \right) - D \sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k \cdot 0.5)] \quad (\text{A-20})$$

$a = 0.5$, $b = 3$, $k_{\max} = 20$

$f_{5-6}(\mathbf{x})$: Griewank's Function

$$f_i(\mathbf{x}) = \sum_{i=1}^D \frac{x_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad (\text{A-21})$$

$f_{7-8}(\mathbf{x})$: Ackley's Function

$$f_i(\mathbf{x}) = -20 \exp\left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2}\right) - \exp\left(\frac{1}{D} \sum_{i=1}^D \cos(2\pi x_i)\right) + 20 + e \quad (\text{A-22})$$

$f_{9-10}(\mathbf{x})$: Sphere Function

$$f_i(\mathbf{x}) = \sum_{i=1}^D x_i^2 \quad (\text{A-23})$$

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

$\lambda = [1, 1, 10, 10, 5/60, 5/60, 5/32, 5/32, 5/100, 5/100]$

$\mathbf{M}_i$ are all identity matrices

$\mathbf{x} \in [-5, 5]^D$, Global optimum $\mathbf{x}^* = \mathbf{o}_1$, $F_{15}(\mathbf{x}^*) = f_{\text{bias}_{15}} = 120$

APPENDIX A

Properties: Multi-modal; Separable near the global optimum (Rastrigin); Scalable; A huge number of local optima; Different function’s properties are mixed together; Sphere Functions give two flat areas for the function

Please notice that these formulas are just for the basic functions, no shift or rotation is included in these expressions. $\mathbf{x}$ here is just a variable in a function.

Take f_1 as an example, when we calculate $f_1((\mathbf{x}-\mathbf{o}_1)/\lambda_1)*\mathbf{M}_1$, we need

calculate $f_1(\mathbf{z}) = \sum_{i=1}^D (z_i^2 - 10 \cos(2\pi z_i) + 10)$, $\mathbf{z} = ((\mathbf{x}-\mathbf{o}_1)/\lambda_1)*\mathbf{M}_1$.

Fig. A-15 3-D Map for 2-D Function F_{15}

16) F_{16} : Rotated Hybrid Composition Function of F_{15}

Except $\mathbf{M}_i$ are different linear transformation matrixes with condition number of 2, all other settings are the same as F_{15} .

$\mathbf{x} \in [-5,5]^D$, Global optimum $\mathbf{x}^* = \mathbf{o}_1$, $F_{16}(\mathbf{x}^*) = f_bias_{16}=120$

Properties: Multi-modal; Rotated; Non-Separable; Scalable; A huge number of local optima; Different function’s properties are mixed together; Sphere Functions give two flat areas for the function.

Fig. A-16 3-D Map for 2-D Function F_{16}

17) F_{17} : F_{16} with Noise in Fitness

Let $(F_{16} - f_{bias_{16}})$ be $G(x)$, then

$$F_{17}(x) = G(x) * (1 + 0.2|N(0,1)|) + f_{bias_{17}} \tag{A-24}$$

All settings are the same as F_{16} .

$x \in [-5,5]^D$, Global optimum $x^* = \mathbf{o}_1$, $F_{17}(x^*) = f_{bias_{17}} = 120$

Properties: Multi-modal; Rotated; Non-Separable; Scalable; A huge number of local optima; Different function's properties are mixed together; Sphere Functions give two flat areas for the function; With Gaussian noise in fitness.

Fig. A-17 3-D Map for 2-D Function F_{17}

APPENDIX A

18) F_{18} : Rotated Hybrid Composition Function $f_{1-2}(\mathbf{x})$: Ackley's Function

$$f_i(\mathbf{x}) = -20 \exp(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2}) - \exp(\frac{1}{D} \sum_{i=1}^D \cos(2\pi x_i)) + 20 + e \quad (\text{A-25})$$

 $f_{3-4}(\mathbf{x})$: Rastrigin's Function

$$f_i(\mathbf{x}) = \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i) + 10) \quad (\text{A-26})$$

 $f_{5-6}(\mathbf{x})$: Sphere Function

$$f_i(\mathbf{x}) = \sum_{i=1}^D x_i^2 \quad (\text{A-27})$$

 $f_{7-8}(\mathbf{x})$: Weierstrass Function

$$f_i(\mathbf{x}) = \sum_{i=1}^D \left(\sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k (x_i + 0.5))] \right) - D \sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k \cdot 0.5)] \quad (\text{A-28})$$

$$a = 0.5, b = 3, k_{\max} = 20$$

 $f_{9-10}(\mathbf{x})$: Griewank's Function

$$f_i(\mathbf{x}) = \sum_{i=1}^D \frac{x_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad (\text{A-29})$$

$$\sigma = [1, 2, 1.5, 1.5, 1, 1, 1.5, 1.5, 2, 2];$$

$$\lambda = [2*5/32; 5/32; 2*1; 1; 2*5/100; 5/100; 2*10; 10; 2*5/60; 5/60]$$

$\mathbf{M}_i$ are all rotation matrices. Condition numbers are [2 3 2 3 2 3 20 30 200 300]

$$\mathbf{o}_{10} = [0, 0, \dots, 0]$$

$$\mathbf{x} \in [-5, 5]^D, \text{ Global optimum } \mathbf{x}^* = \mathbf{o}_1, F_{18}(\mathbf{x}^*) = f_{\text{bias}_{18}} = 10$$

Properties: Multi-modal; Rotated; Non-Separable; Scalable; A huge number of local optima; Different function's properties are mixed together; Sphere Functions give two flat areas for the function.

A local optimum is set on the origin

Fig. A-18 3-D Map for 2-D Function F_{18}

19) F_{19} : Rotated Hybrid Composition Function with narrow basin global optimum

All settings are the same as F_{18} except

$$\sigma = [0.1, 2, 1.5, 1.5, 1, 1, 1.5, 1.5, 2, 2],,$$

$$\lambda = [0.1*5/32; 5/32; 2*1; 1; 2*5/100; 5/100; 2*10; 10; 2*5/60; 5/60]$$

$$\mathbf{x} \in [-5, 5]^D, \text{ Global optimum } \mathbf{x}^* = \mathbf{o}_1, F_{19}(\mathbf{x}^*) = f_bias_{19}(19) = 10$$

Properties: Multi-modal; Non-separable; Scalable; A huge number of local optima; Different function's properties are mixed together; Sphere Functions give two flat areas for the function; A local optimum is set on the origin; A narrow basin for the global optimum.

Fig. A-19 3-D Map for 2-D Function F_{19}

APPENDIX A

20) F_{20} : Rotated Hybrid Composition Function with Global Optimum on the Bounds

All settings are the same as F_{18} except $\sigma_{1(2j)} = 5$, for $j = 1, 2, \dots, \lfloor D/2 \rfloor$

$\mathbf{x} \in [-5, 5]^D$, Global optimum $\mathbf{x}^* = \mathbf{0}_1$, $F_{20}(\mathbf{x}^*) = f_{\text{bias}_{20}} = 10$

Properties: Multi-modal; Non-separable; Scalable; A huge number of local optima; Different function's properties are mixed together; Sphere Functions give two flat areas for the function.; A local optimum is set on the origin; Global optimum is on the bound; If the initialization procedure initializes the population at the bounds, this problem will be solved easily.

Fig. A-20 3-D Map for 2-D Function F_{20}

21) F_{21} : Rotated Hybrid Composition Function

$f_{1-2}(\mathbf{x})$: Rotated Expanded Scaffer's F6 Function

$$F(x, y) = 0.5 + \frac{(\sin^2(\sqrt{x^2 + y^2}) - 0.5)}{(1 + 0.001(x^2 + y^2))^2} \quad (\text{A-30})$$

$$f_i(\mathbf{x}) = F(x_1, x_2) + F(x_2, x_3) + \dots + F(x_{D-1}, x_D) + F(x_D, x_1) \quad (\text{A-31})$$

$f_{3-4}(\mathbf{x})$: Rastrigin's Function

$$f_i(\mathbf{x}) = \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i) + 10) \quad (\text{A-32})$$

$f_{5-6}(\mathbf{x})$: F8F2 Function

APPENDIX A

$$F8(\mathbf{x}) = \sum_{i=1}^D \frac{x_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad (\text{A-33})$$

$$F2(\mathbf{x}) = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2) \quad (\text{A-34})$$

$$f_i(\mathbf{x}) = F8(F2(x_1, x_2)) + F8(F2(x_2, x_3)) + \dots + F8(F2(x_{D-1}, x_D)) + F8(F2(x_D, x_1)) \quad (\text{A-35})$$

$f_{7-8}(\mathbf{x})$: Weierstrass Function

$$f_i(\mathbf{x}) = \sum_{i=1}^D \left(\sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k (x_i + 0.5))] \right) - D \sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k \cdot 0.5)] \quad (\text{A-36})$$

$$a = 0.5, b = 3, k_{\max} = 20$$

$f_{9-10}(\mathbf{x})$: Griewank's Function

$$f_i(\mathbf{x}) = \sum_{i=1}^D \frac{x_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad (\text{A-37})$$

$$\sigma = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2],$$

$$\lambda = [5*5/100; 5/100; 5*1; 1; 5*1; 1; 5*10; 10; 5*5/200; 5/200];$$

$\mathbf{M}_i$ are all orthogonal matrix

$$\mathbf{x} \in [-5, 5]^D, \text{ Global optimum } \mathbf{x}^* = \mathbf{o}_1, F_{21}(\mathbf{x}^*) = f_{\text{bias}_{21}} = 360$$

Properties: Multi-modal; Rotated; Non-Separable; Scalable; A huge number of local optima; Different function's properties are mixed together.

Fig. A-21 3-D Map for 2-D Function F_{21}

APPENDIX A

22) F_{22} : Rotated Hybrid Composition Function 3 with High Condition Number Matrix

All settings are the same as F_{21} except M_i 's condition numbers are [10 20 50 100 200 1000 2000 3000 4000 5000]

$\mathbf{x} \in [-5, 5]^D$, Global optimum $\mathbf{x}^* = \mathbf{o}_1$, $F_{22}(\mathbf{x}^*) = f_bias_{22} = 360$

Properties: Multi-modal; Non-separable; Scalable; A huge number of local optima; Different function's properties are mixed together; Global optimum is on the bound.

Fig. A-22 3-D Map for 2-D Function F_{22}

23) F_{23} : Non-Continuous Rotated Hybrid Composition Function

All settings are the same as F_{23} .

$$\text{Except } x_j = \begin{cases} x_j & |x_j - o_{1j}| < 1/2 \\ \text{round}(2x_j)/2 & |x_j - o_{1j}| \geq 1/2 \end{cases} \text{ for } j = 1, 2, \dots, D \quad (\text{A-38})$$

$$\text{round}(x) = \begin{cases} a-1 & \text{if } x \leq 0 \& b \geq 0.5 \\ a & \text{if } b < 0.5 \\ a+1 & \text{if } x > 0 \& b \geq 0.5 \end{cases} \quad (\text{A-39})$$

where a is x 's integral part and b is x 's decimal part

All "round" operators in this document use the same schedule.

$\mathbf{x} \in [-5, 5]^D$, Global optimum $\mathbf{x}^* = \mathbf{o}_1$, $f(\mathbf{x}^*) \approx f_bias(23) = 360$

APPENDIX A

Properties: Multi-modal; Non-separable; Scalable; A huge number of local optima; Different function's properties are mixed together; Non-continuous; Global optimum is on the bound.

Fig. A-23 3-D Map for 2-D Function F_{23}

24) F_{24} : Rotated Hybrid Composition Function

$f_1(\mathbf{x})$: Weierstrass Function

$$f_i(\mathbf{x}) = \sum_{i=1}^D \left(\sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k (x_i + 0.5))] \right) - D \sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k 0.5)] \quad (\text{A-40})$$

$$a = 0.5, b = 3, k_{\max} = 20$$

$f_2(\mathbf{x})$: Rotated Expanded Scaffer's F6 Function

$$F(x, y) = 0.5 + \frac{(\sin^2(\sqrt{x^2 + y^2}) - 0.5)}{(1 + 0.001(x^2 + y^2))^2} \quad (\text{A-41})$$

$$f_i(\mathbf{x}) = F(x_1, x_2) + F(x_2, x_3) + \dots + F(x_{D-1}, x_D) + F(x_D, x_1) \quad (\text{A-42})$$

$f_3(\mathbf{x})$: F8F2 Function

$$F8(\mathbf{x}) = \sum_{i=1}^D \frac{x_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad (\text{A-43})$$

$$F2(\mathbf{x}) = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2) \quad (\text{A-44})$$

$$f_i(\mathbf{x}) = F8(F2(x_1, x_2)) + F8(F2(x_2, x_3)) + \dots + F8(F2(x_{D-1}, x_D)) + F8(F2(x_D, x_1)) \quad (\text{A-45})$$

APPENDIX A

$f_4(\mathbf{x})$: Ackley's Function

$$f_i(\mathbf{x}) = -20 \exp(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2}) - \exp(\frac{1}{D} \sum_{i=1}^D \cos(2\pi x_i)) + 20 + e \quad (\text{A-46})$$

$f_5(\mathbf{x})$: Rastrigin's Function

$$f_i(\mathbf{x}) = \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i) + 10) \quad (\text{A-47})$$

$f_6(\mathbf{x})$: Griewank's Function

$$f_i(\mathbf{x}) = \sum_{i=1}^D \frac{x_i^2}{4000} - \prod_{i=1}^D \cos(\frac{x_i}{\sqrt{i}}) + 1 \quad (\text{A-48})$$

$f_7(\mathbf{x})$: Non-Continuous Expanded Scaffer's F6 Function

$$F(x, y) = 0.5 + \frac{(\sin^2(\sqrt{x^2 + y^2}) - 0.5)}{(1 + 0.001(x^2 + y^2))^2} \quad (\text{A-49})$$

$$f(\mathbf{x}) = F(y_1, y_2) + F(y_2, y_3) + \dots + F(y_{D-1}, y_D) + F(y_D, y_1) \quad (\text{A-50})$$

$$y_j = \begin{cases} x_j & |x_j| < 1/2 \\ \text{round}(2x_j)/2 & |x_j| \geq 1/2 \end{cases} \text{ for } j = 1, 2, \dots, D \quad (\text{A-51})$$

$f_8(\mathbf{x})$: Non-Continuous Rastrigin's Function

$$f(\mathbf{x}) = \sum_{i=1}^D (y_i^2 - 10 \cos(2\pi y_i) + 10) \quad (\text{A-52})$$

$$y_j = \begin{cases} x_j & |x_j| < 1/2 \\ \text{round}(2x_j)/2 & |x_j| \geq 1/2 \end{cases} \text{ for } j = 1, 2, \dots, D \quad (\text{A-53})$$

$f_9(\mathbf{x})$: High Conditioned Elliptic Function

$$f(\mathbf{x}) = \sum_{i=1}^D (10^6)^{\frac{i-1}{D-1}} x_i^2 \quad (\text{A-54})$$

$f_{10}(\mathbf{x})$: Sphere Function with Noise in Fitness

$$f_i(\mathbf{x}) = (\sum_{i=1}^D x_i^2) (1 + 0.1 |N(0, 1)|) \quad (\text{A-55})$$

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

$\lambda = [10; 5/20; 1; 5/32; 1; 5/100; 5/50; 1; 5/100; 5/100]$

$\mathbf{M}_i$ are all rotation matrices, condition numbers are [100 50 30 10 5 5 4 3 2 2].

$\mathbf{x} \in [-5, 5]^D$, Global optimum $\mathbf{x}^* = \mathbf{o}_1$, $F_{24}(\mathbf{x}^*) = f_bias_{24} = 260$

Properties: Multi-modal; Rotated; Non-Separable; Scalable; A huge number of local optima; Different function's properties are mixed together; Unimodal Functions give flat areas for the function.

Fig. A-24 3-D Map for 2-D Function F_{24}

25) F_{25} : Rotated Hybrid Composition Function without bounds

All settings are the same as F_{24} except no exact search range set for this test function.

5. Comparisons Pairs

1) Different Condition Numbers:

- F_1 . Shifted Rotated Sphere Function
- F_2 . Shifted Schwefel's Problem 1.2
- F_3 . Shifted Rotated High Conditioned Elliptic Function

2) Function With Noise Vs Without Noise

- Pair 1:
 - F_2 . Shifted Schwefel's Problem 1.2
 - F_4 . Shifted Schwefel's Problem 1.2 with Noise in Fitness
- Pair 2:
 - F_{16} . Rotated Hybrid Composition Function
 - F_{17} . F_{16} with Noise in Fitness

3) Function without Rotation Vs With Rotation

Pair 1:

F_9 . Shifted Rastrigin's Function

F_{10} . Shifted Rotated Rastrigin's Function

Pair 2:

F_{15} . Hybrid Composition Function 1

F_{16} . Rotated Hybrid Composition Function 1

4) Continuous Vs Non-continuous

F_{21} . Rotated Hybrid Composition Function 3

F_{23} . Non-Continuous Rotated Hybrid Composition Function 3

5) Global Optimum on Bounds Vs Global Optimum on Bounds

F_{18} . Rotated Hybrid Composition Function 2

F_{20} . Rotated Hybrid Composition Function 2 with the Global Optimum on the Bounds

6) Wide Global Optimum Basin Vs Narrow Global Optimum Basin

F_{18} . Rotated Hybrid Composition Function 2

F_{19} . Rotated Hybrid Composition Function 2 with a Narrow Basin for the Global Optimum

7) Orthogonal Matrix Vs High Condition Number Matrix

F_{21} . Rotated Hybrid Composition Function 3

F_{22} . Rotated Hybrid Composition Function 3 with High Condition Number Matrix

8) Global Optimum in the Initialization Range Vs outside of the Initialization Range

F_{24} . Rotated Hybrid Composition Function 4

F_{25} . Rotated Hybrid Composition Function 4 without Bounds

6. Similar Groups

1) Unimodal Functions

Function 1-5

2) Multi-modal Functions

Function 6-25

Single Function: Function 6-12

Expanded Function: Function 13-14

Hybrid Composition Function: Function 15-25

3) Functions with Global Optimum outside of the Initialization Range

F_7 . Shifted Rotated Griewank’s Function without Bounds

F_{25} . Rotated Hybrid Composition Function 4 without Bounds

4) Functions with Global Optimum on Bounds

F_5 . Schwefel’s Problem 2.6 with Global Optimum on Bounds

F_8 . Shifted Rotated Ackley’s Function with Global Optimum on Bounds

F_{20} . Rotated Hybrid Composition Function 2 with the Global Optimum on the Bounds

Appendix B

Definitions of the 24 CEC'06 Test Functions

g01

Minimize

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

subject to:

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

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

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

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

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

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

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

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

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

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

g02

Minimize:

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

APPENDIX B

subject to:

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

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

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

g03

Minimize

$$f(\mathbf{x}) = -(\sqrt{D})^D \prod_{i=1}^D x_i \tag{B-3}$$

subject to:

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

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

g04

Minimize:

$$f(\mathbf{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141 \tag{B-4}$$

subject to:

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

APPENDIX B

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

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

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

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

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

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

g05:

Minimize:

$$f(\mathbf{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.00002/3)x_2^3 \quad (\text{B-5})$$

subject to:

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

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

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

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

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

where $0 \leq x_1 \leq 1200$, $0 \leq x_2 \leq 1200$, $-0.55 \leq x_3 \leq 0.55$ and $-0.55 \leq x_4 \leq 0.55$.

The best known solution $\mathbf{x}^* = ((679.945148297028709, 1026.06697600004691, 0.118876369094410433, -0.39623348521517826))$, where $f(\mathbf{x}^*) = 5126.4967140071$.

g06:

Minimize:

$$f(\mathbf{x}) = (x_1 - 10)^3 + (x_2 - 20)^3 \quad (\text{B-6})$$

subject to:

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

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

where $13 \leq x_1 \leq 100$ and $0 \leq x_2 \leq 100$. The optimum solution is

APPENDIX B

$\mathbf{x}^* = (14.09500000000000064, 0.8429607892154795668),$ where
 $f(\mathbf{x}^*) = -6961.81387558015.$ Both constraints are active.

g07:

Minimize:

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

subject to:

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

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

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

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

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

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

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

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

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

g08

Minimize:

$$f(\mathbf{x}) = -\frac{\sin^3(2\pi x_1) \sin(2\pi x_2)}{x_1^3(x_1 + x_2)} \quad (\text{B-8})$$

subject to:

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

APPENDIX B

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

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

g09

Minimize:

$$f(\mathbf{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7 \quad (\text{B-9})$$

subject to:

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

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

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

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

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

g10

Minimize:

$$f(\mathbf{x}) = x_1 + x_2 + x_3 \quad (\text{B-10})$$

subject to:

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

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

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

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

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

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

APPENDIX B

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

g11

Minimize:

$$f(\mathbf{x}) = x_1^2 + (x_2 - 1)^2 \quad (\text{B-11})$$

subject to:

$$h(\mathbf{x}) = x_2 - x_1^2 = 0$$

where $-1 \leq x_1 \leq 1$ and $-1 \leq x_2 \leq 1$. The optimum solution is $\mathbf{x}^* = (-0.707036070037170616, 0.5000000004333606807)$ where $f(\mathbf{x}^*) = 0.7499$.

g12

Minimize:

$$f(\mathbf{x}) = -(100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2) / 100 \quad (\text{B-12})$$

subject to:

$$g(\mathbf{x}) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0$$

where $0 \leq x_i \leq 10$ ($i = 1, 2, 3$) and $p, q, r = 1, 2, \dots, 9$. The feasible region of the search space consists of 9^3 disjointed spheres. A point (x_1, x_2, x_3) is feasible if and only if there exist p, q, r such that the above inequality holds. The optimum is located at $\mathbf{x}^* = (5, 5, 5)$ where $f(\mathbf{x}^*) = -1$.

g13

Minimize:

$$f(\mathbf{x}) = e^{x_1 x_2 x_3 x_4 x_5} \quad (\text{B-13})$$

subject to:

$$h_1(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$$

APPENDIX B

$$h_2(\mathbf{x}) = x_2x_3 - 5x_4x_5 = 0$$

$$h_3(\mathbf{x}) = x_1^3 + x_2^3 + 1 = 0$$

where $-2.3 \leq x_i \leq 2.3$ ($i = 1, 2$) and $-3.2 \leq x_i \leq 3.2$ ($i = 3, 4, 5$). The optimum solution is $\mathbf{x}^* = (-1.71714224003, 1.59572124049468, 1.8272502406271, -0.763659881912867, -0.76365986736498)$ where $f(\mathbf{x}^*) = 0.053941514041898$.

g14

Minimize:

$$f(\mathbf{x}) = \sum_{i=1}^{10} x_i (c_i + \ln(x_i / \sum_{j=1}^{10} x_j)) \quad (\text{B-14})$$

subject to:

$$h_1(\mathbf{x}) = x_1 + 2x_2 + 2x_3 + x_6 + x_{10} - 2 = 0$$

$$h_2(\mathbf{x}) = x_4 + 2x_5 + x_6 + x_7 - 1 = 0$$

$$h_3(\mathbf{x}) = x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 = 0$$

where the bounds are $0 < x_i \leq 10$ ($i = 1, \dots, 10$), and $c_1 = -6.089$, $c_2 = -17.164$, $c_3 = -34.054$, $c_4 = -5.914$, $c_5 = -24.721$, $c_6 = -14.986$, $c_7 = -24.1$, $c_8 = -10.708$, $c_9 = -26.662$, $c_{10} = -22.179$. The best solution is at $\mathbf{x}^* = (0.0406684113216282, 0.147721240492452, 0.783205732104114, 0.00141433931889084, 0.485293636780388, 0.000693183051556082, 0.0274052040687766, 0.0179509660214818, 0.0373268186859717, 0.0968844604336845)$ where $f(\mathbf{x}^*) = -47.7648884594915$.

g15

Minimize:

$$f(\mathbf{x}) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3 \quad (\text{B-15})$$

subject to:

$$h_1(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2 - 25 = 0$$

$$h_2(\mathbf{x}) = 8x_1 + 14x_2 + 7x_3 - 56 = 0$$

where the bounds are $0 \leq x_i \leq 10$ ($i = 1, 2, 3$). The best known solution is at $\mathbf{x}^* = (3.51212812611795133, 0.216987510429556135, 3.55217854929179921)$

APPENDIX B

where $f(\mathbf{x}^*) = 961.715022289961$.

g16

Minimize:

$$f(\mathbf{x}) = 0.000117y_{14} + 0.1365 + 0.00002358y_{13} + 0.000001502y_{16} + 0.0321y_{12} \\ + 0.004324y_5 + 0.0001\frac{c_{15}}{c_{16}} + 37.48\frac{y_2}{c_{12}} - 0.0000005843y_{17}$$

(B-16)

subject to:

$$g_1(\mathbf{x}) = \frac{0.28}{0.72}y_5 - y_4 \leq 0$$

$$g_2(\mathbf{x}) = x_3 - 1.5x_2 \leq 0$$

$$g_3(\mathbf{x}) = 3496\frac{y_2}{c_{12}} - 1.5x_2 \leq 0$$

$$g_4(\mathbf{x}) = 110.6 + y_1 - \frac{62212}{c_{17}} \leq 0$$

$$g_5(\mathbf{x}) = 213.1 - y_1 \leq 0$$

$$g_6(\mathbf{x}) = y_1 - 405.23 \leq 0$$

$$g_7(\mathbf{x}) = 17.505 - y_2 \leq 0$$

$$g_8(\mathbf{x}) = y_2 - 1053.6667 \leq 0$$

$$g_9(\mathbf{x}) = 11.275 - y_3 \leq 0$$

$$g_{10}(\mathbf{x}) = y_3 - 35.03 \leq 0$$

$$g_{11}(\mathbf{x}) = 214.228 - y_4 \leq 0$$

$$g_{12}(\mathbf{x}) = y_4 - 665.585 \leq 0$$

$$g_{13}(\mathbf{x}) = 7.458 - y_5 \leq 0$$

$$g_{14}(\mathbf{x}) = y_5 - 584.463 \leq 0$$

$$g_{15}(\mathbf{x}) = 0.961 - y_6 \leq 0$$

$$g_{16}(\mathbf{x}) = y_6 - 265.916 \leq 0$$

$$g_{17}(\mathbf{x}) = 1.612 - y_7 \leq 0$$

$$g_{18}(\mathbf{x}) = y_7 - 7.046 \leq 0$$

$$g_{19}(\mathbf{x}) = 0.146 - y_8 \leq 0$$

APPENDIX B

$$g_{20}(\mathbf{x}) = y_8 - 0.222 \leq 0$$

$$g_{21}(\mathbf{x}) = 107.99 - y_9 \leq 0$$

$$g_{22}(\mathbf{x}) = y_9 - 273.366 \leq 0$$

$$g_{23}(\mathbf{x}) = 922.693 - y_{10} \leq 0$$

$$g_{24}(\mathbf{x}) = y_{10} - 1286.105 \leq 0$$

$$g_{25}(\mathbf{x}) = 922.832 - y_{11} \leq 0$$

$$g_{26}(\mathbf{x}) = y_{11} - 1444.046 \leq 0$$

$$g_{27}(\mathbf{x}) = 18.766 - y_{12} \leq 0$$

$$g_{28}(\mathbf{x}) = y_{12} - 537.141 \leq 0$$

$$g_{29}(\mathbf{x}) = 1072.163 - y_{13} \leq 0$$

$$g_{30}(\mathbf{x}) = y_{13} - 3247.039 \leq 0$$

$$g_{31}(\mathbf{x}) = 8961.448 - y_{14} \leq 0$$

$$g_{32}(\mathbf{x}) = y_{14} - 26844.086 \leq 0$$

$$g_{33}(\mathbf{x}) = 0.063 - y_{15} \leq 0$$

$$g_{34}(\mathbf{x}) = y_{15} - 0.386 \leq 0$$

$$g_{35}(\mathbf{x}) = 71084.33 - y_{16} \leq 0$$

$$g_{36}(\mathbf{x}) = y_{16} - 140000 \leq 0$$

$$g_{37}(\mathbf{x}) = 2802713 - y_{17} \leq 0$$

$$g_{38}(\mathbf{x}) = y_{17} - 12146108 \leq 0$$

where:

$$y_1 = x_2 + x_3 + 41.6$$

$$c_1 = 0.024x_4 - 4.62$$

$$y_2 = \frac{12.5}{c_1} + 12$$

$$c_2 = 0.0003535x_1^2 + 0.5311x_1 + 0.08705y_2x_1$$

$$c_3 = 0.052x_1 + 78 + 0.02377y_2x_1$$

$$y_3 = \frac{c_2}{c_3}$$

$$y_4 = 19y_3$$

APPENDIX B

$$c_4 = 0.04782(x_1 - y_3) + \frac{0.1956(x_1 - y_3)^2}{x_2} + 0.6376y_4 + 1.594y_3$$

$$c_5 = 100x_2$$

$$c_6 = x_1 - y_3 - y_4$$

$$y_5 = c_6c_7$$

$$y_6 = x_1 - y_5 - y_4 - y_3$$

$$y_7 = \frac{c_8}{y_1}$$

$$y_8 = \frac{c_8}{3798}$$

$$c_9 = y_7 - \frac{0.0663y_7}{y_8} - 0.3153$$

$$y_9 = \frac{96.82}{c_9} + 0.321y_1$$

$$y_{10} = 1.29y_5 + 1.258y_4 + 2.29y_3 + 1.71y_6$$

$$y_{11} = 1.71x_1 - 0.452y_4 + 0.58y_3$$

$$c_{10} = \frac{12.3}{752.3}$$

$$c_{11} = (1.75y_2)(0.995x_1)$$

$$c_{12} = 0.995y_{10} + 1998$$

$$y_{12} = c_{10}x_1 + \frac{c_{11}}{c_{12}}$$

$$y_{13} = c_{12} - 1.75y_2$$

$$y_{14} = 3623 + 64.4x_2 + 58.4x_3 + \frac{146312}{y_9 + x_5}$$

$$c_{13} = 0.995y_{10} + 60.8x_2 + 48x_4 - 0.1121y_{14} - 5095$$

$$y_{15} = \frac{y_{13}}{c_{13}}$$

$$y_{16} = 148000 - 331000y_{15} + 40y_{13} - 61y_{15}y_{13}$$

$$c_{14} = 2324y_{10} - 28740000y_2$$

$$y_{17} = 14130000 - 1328y_{10} - 531y_{11} + \frac{c_{14}}{c_{12}}$$

APPENDIX B

$$c_{15} = \frac{y_{13}}{y_{15}} - \frac{y_{13}}{0.52}$$

$$c_{16} = 1.104 - 0.72y_{15}$$

$$c_{17} = y_9 + x_5$$

and where the bounds are $704.4148 \leq x_1 \leq 906.3855$, $68.6 \leq x_2 \leq 288.88$,

$0 \leq x_3 \leq 134.75$, $193 \leq x_4 \leq 287.0966$ and $25 \leq x_5 \leq 84.1988$. The best known solution is at $\mathbf{x}^* = (705.174537070090537, 68.5999999999999943, 102.899999999999991, 282.324931593660324, 37.5841164258054832)$ where $f(\mathbf{x}^*) = -1.90515525853479$.

g17

Minimize:

$$f(\mathbf{x}) = f(x_1) + f(x_2) \quad (\text{B-17})$$

where

$$f_1(x_1) = \begin{cases} 30x_1 & 0 \leq x_1 < 300 \\ 31x_1 & 300 \leq x_1 < 400 \end{cases}$$

$$f_2(x_2) = \begin{cases} 28x_2 & 0 \leq x_2 < 100 \\ 29x_2 & 100 \leq x_2 < 200 \end{cases}$$

subject to

$$h_1(\mathbf{x}) = -x_1 + 300 - \frac{x_3x_4}{131.078} \cos(1.48477 - x_6) + \frac{0.90798x_4^2}{131.078} \cos(1.47588)$$

$$h_2(\mathbf{x}) = -x_2 - \frac{x_3x_4}{131.078} \cos(1.48477 + x_6) + \frac{0.90798x_4^2}{131.078} \cos(1.47588)$$

$$h_3(\mathbf{x}) = -x_5 - \frac{x_3x_4}{131.078} \sin(1.48477 + x_6) + \frac{0.90798x_4^2}{131.078} \sin(1.47588)$$

$$h_4(\mathbf{x}) = 200 - \frac{x_3x_4}{131.078} \sin(1.48477 + x_6) + \frac{0.90798x_3^2}{131.078} \sin(1.47588)$$

where the bounds are $0 \leq x_1 \leq 400$, $0 \leq x_2 \leq 1000$, $340 \leq x_3 \leq 420$, $340 \leq x_4 \leq 420$, $-1000 \leq x_5 \leq 1000$ and $0 \leq x_6 \leq 0.5236$. The best known solution is at $\mathbf{x}^* = (201.784467214523659, 99.99999999999999005, 383.071034852773266, 420, -10.9076584514292652, 0.0731482312084287128)$ where $f(\mathbf{x}^*) = 8853.53967480648$.

APPENDIX B

g18

Minimize:

$$f(\mathbf{x}) = -0.5(x_1x_4 - x_2x_3 + x_3x_9 - x_5x_9 + x_5x_8 - x_6x_7) \quad (\text{B-18})$$

subject to

$$g_1(\mathbf{x}) = x_3^2 + x_4^2 - 1 \leq 0$$

$$g_2(\mathbf{x}) = x_9^2 - 1 \leq 0$$

$$g_3(\mathbf{x}) = x_5^2 + x_6^2 - 1 \leq 0$$

$$g_4(\mathbf{x}) = x_1^2 + (x_2 - x_9)^2 - 1 \leq 0$$

$$g_5(\mathbf{x}) = (x_1 - x_5)^2 + (x_2 - x_6)^2 - 1 \leq 0$$

$$g_6(\mathbf{x}) = (x_1 - x_7)^2 + (x_2 - x_8)^2 - 1 \leq 0$$

$$g_7(\mathbf{x}) = (x_3 - x_5)^2 + (x_4 - x_6)^2 - 1 \leq 0$$

$$g_8(\mathbf{x}) = (x_3 - x_7)^2 + (x_4 - x_8)^2 - 1 \leq 0$$

$$g_9(\mathbf{x}) = x_7^2 + (x_8 - x_9)^2 - 1 \leq 0$$

$$g_{10}(\mathbf{x}) = x_2x_3 - x_1x_4 \leq 0$$

$$g_{11}(\mathbf{x}) = -x_3x_9 \leq 0$$

$$g_{12}(\mathbf{x}) = x_5x_9 \leq 0$$

$$g_{13}(\mathbf{x}) = x_6x_7 - x_5x_8 \leq 0$$

where the bounds are $-10 \leq x_i \leq 10$ ($i = 1, \dots, 8$) and $0 \leq x_9 \leq 20$. The best known solution is at $\mathbf{x}^* = (-0.657776192427943163, -0.153418773482438542, 0.323413871675240938, -0.946257611651304398, -0.657776194376798906, -0.753213434632691414, 0.323413874123576972, -0.346462947962331735, 0.59979466285217542)$ where $f(\mathbf{x}^*) = -0.866025403784439$.

g19

Minimize:

$$f(\mathbf{x}) = \sum_{j=1}^5 \sum_{i=1}^5 c_{ij} x_{(10+i)} x_{(10+j)} + 2 \sum_{j=1}^5 d_j x_{(10+j)}^3 - \sum_{i=1}^{10} b_i x_i \quad (\text{B-19})$$

subject to:

APPENDIX B

$$g_j(\mathbf{x}) = -2 \sum_{i=1}^5 c_{ij} x_{(10+i)} - 3d_j x_{(10+j)}^2 - e_j + \sum_{i=1}^{10} a_{ij} x_i \leq 0, j = 1, \dots, 5$$

where $\mathbf{b} = [-40, -2, -0.25, -4, -4, -1, -40, -60, 5, 1]$ and the remaining data is on Table B-1. The bounds are $0 \leq x_i \leq 10$ ($i = 1, \dots, 15$). The best known solution is at $\mathbf{x}^* = (1.66991341326291344\text{e-}17, 3.95378229282456509\text{e-}16, 3.94599045143233784, 1.06036597479721211\text{e-}16, 3.2831773458454161, 9.99999999999999822, 1.12829414671605333\text{e-}17, 1.2026194599794709\text{e-}17, 2.50706276000769697\text{e-}15, 2.24624122987970677\text{e-}15, 0.370764847417013987, 0.278456024942955571, 0.523838487672241171, 0.388620152510322781, 0.298156764974678579)$ where $f(\mathbf{x}^*) = 32.6555929502463$.

Table B-1 Data Set for Test Problem g19

<i>j</i>	1	2	3	4	5
<i>e_j</i>	-15	-27	-36	-18	-12
<i>c_{1j}</i>	30	-20	-10	32	-10
<i>c_{2j}</i>	-20	39	-6	-31	32
<i>c_{3j}</i>	-10	-6	10	-6	-10
<i>c_{4j}</i>	32	-31	-6	39	-20
<i>c_{5j}</i>	-10	32	-10	-20	30
<i>d_j</i>	4	8	10	6	2
<i>a_{1j}</i>	-16	2	0	1	0
<i>a_{2j}</i>	0	-2	0	0.4	2
<i>a_{3j}</i>	-3.5	0	2	0	0
<i>a_{4j}</i>	0	-2	0	-4	-1
<i>a_{5j}</i>	0	-9	-2	1	-2.8
<i>a_{6j}</i>	2	0	-4	0	0
<i>a_{7j}</i>	-1	-1	-1	-1	-1
<i>a_{8j}</i>	-1	-2	-3	-2	-1
<i>a_{9j}</i>	1	2	3	4	5
<i>a_{10j}</i>	1	1	1	1	1

g20

Minimize:

$$f(\mathbf{x}) = \sum_{i=1}^{24} a_i x_i \tag{B-20}$$

subject to:

$$g_i(\mathbf{x}) = \frac{(x_i + x_{(i+12)})}{\sum_{j=1}^{24} x_j + e_i} \leq 0, \quad i = 1, 2, 3$$

APPENDIX B

$$g_i(\mathbf{x}) = \frac{(x_{(i+3)} + x_{(i+15)})}{\sum_{j=1}^{24} x_j + e_i} \leq 0, \quad i = 4, 5, 6$$

$$h_i(\mathbf{x}) = \frac{x_{(i+12)}}{b_{(i+12)} \sum_{j=13}^{24} \frac{x_j}{b_j}} - \frac{c_i x_i}{40 b_i \sum_{j=1}^{12} \frac{x_j}{b_j}} = 0, \quad i = 1, \dots, 12$$

$$h_{13}(\mathbf{x}) = \sum_{i=1}^{24} x_i - 1 = 0$$

$$h_{14}(\mathbf{x}) = \sum_{i=1}^{12} \frac{x_i}{d_i} + k \sum_{i=13}^{24} \frac{x_i}{b_i} - 1.671 = 0$$

where $k = 0.7302 * 530 * 14.7 / 40$ and the data set is detailed on Table B-2. The bounds are $0 \leq x_i \leq 10$ ($i = 1, \dots, 24$). The best known solution is at $\mathbf{x}^* = (1.28582343498528086\text{e-}18, \quad 4.83460302526130664\text{e-}34, \quad 0, \quad 0, \quad 6.30459929660781851\text{e-}18, \quad 7.57192526201145068\text{e-}34, \quad 5.03350698372840437\text{e-}34, \quad 9.28268079616618064\text{e-}34, \quad 0, \quad 1.76723384525547359\text{e-}17, \quad 3.55686101822965701\text{e-}34, \quad 2.99413850083471346\text{e-}34, \quad 0.158143376337580827 \quad 2.29601774161699833\text{e-}19, \quad 1.06106938611042947\text{e-}18, \quad 1.31968344319506391\text{e-}18, \quad 0.530902525044209539, \quad 0, \quad 2.89148310257773535\text{e-}18, \quad 3.34892126180666159\text{e-}18, \quad 0, \quad 0.310999974151577319, \quad 5.41244666317833561\text{e-}05, \quad 4.84993165246959553\text{e-}16)$. This solution is a little infeasible and no feasible solution is found so far. This problem can have feasible solutions after deleting the first 6 inequality constraints.

g21

Minimize:

$$f(\mathbf{x}) = x_1 \quad (\text{B-21})$$

subject to:

$$g_1(\mathbf{x}) = -x_1 + 35x_2^{0.6} + 35x_3^{0.6} \leq 0$$

$$h_1(\mathbf{x}) = -300x_3 + 7500x_5 - 7500x_6 - 25x_4x_5 + 25x_4x_6 + x_3x_4 = 0$$

$$h_2(\mathbf{x}) = 100x_2 + 155.365x_4 + 2500x_7 - x_2x_4 - 25x_4x_7 - 15536.5 = 0$$

$$h_3(\mathbf{x}) = -x_5 + \ln(-x_4 + 900) = 0$$

$$h_4(\mathbf{x}) = -x_6 + \ln(x_4 + 300) = 0$$

APPENDIX B

$h_5(\mathbf{x}) = -x_7 + \ln(-2x_4 + 700) = 0$

where the bounds are $0 \leq x_1 \leq 1000$, $0 \leq x_2, x_3 \leq 40$, $100 \leq x_4 \leq 300$, $6.3 \leq x_5 \leq 6.7$, $5.9 \leq x_6 \leq 6.4$ and $4.5 \leq x_7 \leq 6.25$. The best known solution is at $\mathbf{x}^* =$ (193.724510070034967, 5.56944131553368433e-27, 17.3191887294084914, 100.047897801386839, 6.68445185362377892, 5.99168428444264833, 6.21451648886070451) where $f(\mathbf{x}^*) = 193.724510070035$.

Table B-2 Data Set for Test Problem g20

<i>i</i>	<i>a_i</i>	<i>b_i</i>	<i>c_i</i>	<i>d_i</i>	<i>e_i</i>
1	0.0693	44.094	123.7	31.244	0.1
2	0.0577	58.12	31.7	36.12	0.3
3	0.05	58.12	45.7	34.784	0.4
4	0.2	137.4	14.7	92.7	0.3
5	0.26	120.9	84.7	82.7	0.6
6	0.55	170.9	27.7	91.6	0.3
7	0.06	62.501	49.7	56.708	
8	0.1	84.94	7.1	82.7	
9	0.12	133.425	2.1	80.8	
10	0.18	82.507	17.7	64.517	
11	0.1	46.07	0.85	49.4	
12	0.09	60.097	0.64	49.1	
13	0.0693	44.094			
14	0.0577	58.12			
15	0.05	58.12			
16	0.2	137.4			
17	0.26	120.9			
18	0.55	170.9			
19	0.06	62.501			
20	0.1	84.94			
21	0.12	133.425			
22	0.18	82.507			
23	0.1	46.07			
24	0.09	60.097			

g22

Minimize:

$f(\mathbf{x}) = x_1$ (B-22)

subject to

$g_1(\mathbf{x}) = -x_1 + x_2^{0.6} + x_3^{0.6} + x_4^{0.6} \leq 0$

$h_1(\mathbf{x}) = x_5 - 100000x_8 + 1 \times 10^7 = 0$

APPENDIX B

$$h_2(\mathbf{x}) = x_6 + 100000x_8 - 100000x_9 = 0$$

$$h_3(\mathbf{x}) = x_7 + 100000x_9 - 5 \times 10^7 = 0$$

$$h_4(\mathbf{x}) = x_5 + 100000x_{10} - 3.3 \times 10^7 = 0$$

$$h_5(\mathbf{x}) = x_6 + 100000x_{11} - 4.4 \times 10^7 = 0$$

$$h_6(\mathbf{x}) = x_7 + 100000x_{12} - 6.6 \times 10^7 = 0$$

$$h_7(\mathbf{x}) = x_5 - 120x_2x_{13} = 0$$

$$h_8(\mathbf{x}) = x_6 - 80x_3x_{14} = 0$$

$$h_9(\mathbf{x}) = x_7 - 40x_4x_{15} = 0$$

$$h_{10}(\mathbf{x}) = x_8 - x_{11} + x_{16} = 0$$

$$h_{11}(\mathbf{x}) = x_9 - x_{12} + x_{17} = 0$$

$$h_{12}(\mathbf{x}) = -x_{18} + \ln(x_{10} - 100) = 0$$

$$h_{13}(\mathbf{x}) = -x_{19} + \ln(-x_8 + 300) = 0$$

$$h_{14}(\mathbf{x}) = -x_{20} + \ln(x_{16}) = 0$$

$$h_{15}(\mathbf{x}) = -x_{21} + \ln(-x_9 + 400) = 0$$

$$h_{16}(\mathbf{x}) = -x_{22} + \ln(x_{17}) = 0$$

$$h_{17}(\mathbf{x}) = -x_8 - x_{10} + x_{13}x_{18} - x_{13}x_{19} + 400 = 0$$

$$h_{18}(\mathbf{x}) = x_8 - x_9 - x_{11} + x_{14}x_{20} - x_{14}x_{21} + 400 = 0$$

$$h_{19}(\mathbf{x}) = x_9 - x_{12} - 4.60517x_{15} + x_{15}x_{22} + 100 = 0$$

where the bounds $0 \leq x_1 \leq 20000$, $0 \leq x_2, x_3, x_4 \leq 1 \times 10^6$, $0 \leq x_5, x_6, x_7 \leq 4 \times 10^7$, $100 \leq x_8 \leq 299.99$, $100 \leq x_9 \leq 399.99$, $100.01 \leq x_{10} \leq 300$, $100 \leq x_{11} \leq 400$, $100 \leq x_{12} \leq 600$, $0 \leq x_{13}, x_{14}, x_{15} \leq 500$, $0.01 \leq x_{16} \leq 300$, $0.01 \leq x_{17} \leq 400$, $-4.7 \leq x_{18}, x_{19}, x_{20}, x_{21}, x_{22} \leq 6.25$. The best known solutions is at $\mathbf{x}^* = (236.430975504001054, 135.82847151732463, 204.818152544824585, 6446.54654059436416, 3007540.83940215595, 4074188.65771341929, 32918270.5028952882, 130.075408394314167, 170.817294970528621, 299.924591605478554, 399.258113423595205, 330.817294971142758, 184.51831230897065, 248.64670239647424, 127.658546694545862, 269.182627528746707, 160.000016724090955, 5.29788288102680571, 5.13529735903945728, 5.59531526444068827, 5.43444479314453499,$

APPENDIX B

5.07517453535834395) where $f(\mathbf{x}^*) = 236.430975504001$.

g23

Minimize:

$$f(\mathbf{x}) = -9x_5 - 15x_8 + 6x_1 + 16x_2 + 10(x_6 + x_7) \quad (\text{B-23})$$

subject to:

$$g_1(\mathbf{x}) = x_9x_3 + 0.02x_6 - 0.025x_5 \leq 0$$

$$g_2(\mathbf{x}) = x_9x_4 + 0.02x_7 - 0.015x_8 \leq 0$$

$$h_1(\mathbf{x}) = x_1 + x_2 - x_3 - x_4 = 0$$

$$h_2(\mathbf{x}) = 0.03x_1 + 0.01x_2 - x_9(x_3 + x_4) = 0$$

$$h_3(\mathbf{x}) = x_3 + x_6 - x_5 = 0$$

$$h_4(\mathbf{x}) = x_4 + x_7 - x_8 = 0$$

where the bounds are $0 \leq x_1, x_2, x_6 \leq 300$, $0 \leq x_3, x_5, x_7 \leq 100$, $0 \leq x_4, x_8 \leq 200$ and $0.01 \leq x_9 \leq 0.03$. The best known solution is at $\mathbf{x}^* = (0.00510000000000259465, \dots, 99.99470000000000514, 9.01920162996045897\text{e-}18, 99.99990000000000535, 0.0001000000000027086086, 2.75700683389584542\text{e-}14, 99.999999999999574, 2000.0100000100000100008)$, where $f(\mathbf{x}^*) = -400.055099999999584$.

g24

Minimize:

$$f(\mathbf{x}) = -x_1 - x_2 \quad (\text{B-24})$$

subject to:

$$g_1(\mathbf{x}) = -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \leq 0$$

$$g_2(\mathbf{x}) = -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 36 \leq 0$$

where the bounds are $0 \leq x_1 \leq 3$ and $0 \leq x_2 \leq 4$. The feasible global minimum is at $\mathbf{x}^* = (2.32952019747762, 3.17849307411774)$ where $f(\mathbf{x}^*) = -5.50801327159536$. This problem has a feasible region consisting on two disconnected sub-regions.

APPENDIX B

Table B-3 Details of the 24 Test Problems

Function		n	Type of f	ρ	LI	NI	LE	NE	a
G1	Min	13	quadratic	0.0111%	9	0	0	0	6
G2	Max	20	nonlinear	99.8474%	0	2	0	0	1
G3	Max	10	polynomial	0.0000%	0	0	0	1	1
G4	Min	5	quadratic	52.1230%	0	6	0	0	2
G5	Min	4	cubic	0.0000%	2	0	0	3	3
G6	Min	2	cubic	0.0066%	0	2	0	0	2
G7	Min	10	quadratic	0.0003%	3	5	0	0	6
G8	Max	2	nonlinear	0.8560%	0	2	0	0	0
G9	Min	7	polynomial	0.5121%	0	4	0	0	2
G10	Min	8	linear	0.0010%	3	3	0	0	6
G11	Min	2	quadratic	0.0000%	0	0	0	1	1
G12	Max	3	quadratic	4.7713%	0	1	0	0	0
G13	Min	5	nonlinear	0.0000%	0	0	0	3	3
G14	Min	10	nonlinear	0.0000%	0	0	3	0	3
G15	Min	3	quadratic	0.0000%	0	0	1	1	2
G16	Max	5	nonlinear	0.0204%	4	34	0	0	4
G17	Min	6	nonlinear	0.0000%	0	0	0	4	4
G18	Max	9	quadratic	0.0000%	0	12	0	0	4
G19	Max	15	nonlinear	33.4761%	0	5	0	0	-
G20	Min	24	linear	0.0000%	0	6	2	12	-
G21	Min	7	linear	0.0000%	0	1	0	5	6
G22	Min	22	linear	0.0000%	0	1	8	11	-
G23	Min	9	linear	0.0000%	0	2	3	1	-
G24	Min	2	linear	79.6556%	0	2	0	0	2

* n is the number of decision variables, $\rho=F//S$ is the estimated ratio between the feasible region and the search space, LI is the number of linear inequality constraints, NI the number of nonlinear inequality constraints, LE is the number of linear equality constraints and NE is the number of nonlinear equality constraints. a is the number of active constraints on the optima. $\epsilon = 0.0001$ for NE

APPENDIX B

Table B-4 $f(x^*)$ and the Bounds for the 24 Problems

Function	n	$f(x^*)$	bounds
G1	13	-15	$0 \leq x_i \leq 1$ ($i=1, \dots, 9$), $0 \leq x_i \leq 100$ ($i=10, 11, 12$), $0 \leq x_{13} \leq 1$
G2	20	0.803619	$0 \leq x_i \leq 10$ ($i=1, \dots, 20$)
G3	10	1	$0 \leq x_i \leq 1$ ($i=1, \dots, 10$)
G4	5	-30665.539	$78 \leq x_1 \leq 102$, $33 \leq x_2 \leq 45$, $27 \leq x_i \leq 45$ ($i=3, 4, 5$)
G5	4	5126.4981	$0 \leq x_i \leq 1200$ ($i=1, 2$), $-0.55 \leq x_i \leq 0.55$ ($i=3, 4$)
G6	2	-6961.81381	$13 \leq x_1 \leq 100$, $0 \leq x_2 \leq 100$
G7	10	24.3062091	$-10 \leq x_i \leq 10$ ($i=1, \dots, 10$)
G8	2	0.095825	$0 \leq x_i \leq 10$ ($i=1, 2$)
G9	7	680.6300573	$-10 \leq x_i \leq 10$ ($i=1, \dots, 7$)
G10	8	7049.3307	$100 \leq x_1 \leq 10000$, $1000 \leq x_i \leq 10000$ ($i=2, 3$), $0 \leq x_i \leq 1$ ($i=4, \dots, 8$)
G11	2	0.75	$-1 \leq x_i \leq 1$ ($i=1, 2$)
G12	3	1	$0 \leq x_i \leq 10$ ($i=1, 2, 3$)
G13	5	0.0539498	$-2.3 \leq x_i \leq 2.3$ ($i=1, 2$), $-3.2 \leq x_i \leq 3.2$ ($i=3, 4, 5$)
G14	10	-47.7644	$0 \leq x_i \leq 10$ ($i=1, \dots, 10$)
G15	3	955.0351	$0 \leq x_i \leq 10$ ($i=1, \dots, 3$)
G16	5	1.9146	$704.4148 \leq x_1 \leq 906.3855$, $68.6 \leq x_2 \leq 288.88$, $0 \leq x_3 \leq 134.75$, $193 \leq x_4 \leq 287.0966$, $25 \leq x_5 \leq 84.1988$
G17	6	8877	$0 \leq x_1 \leq 400$, $0 \leq x_2 \leq 1000$, $340 \leq x_3 \leq 420$, $340 \leq x_4 \leq 420$, $-1000 \leq x_5 \leq 1000$, $0 \leq x_6 \leq 0.5236$
G18	9	1	$-10 \leq x_i \leq 10$ ($i=1, \dots, 8$), $0 \leq x_9 \leq 20$
G19	15	-	$0 \leq x_i \leq 10$ ($i=1, \dots, 15$)
G20	24	-	$0 \leq x_i \leq 10$ ($i=1, \dots, 24$)
G21	7	193.7783	$0 \leq x_1 \leq 1000$, $0 \leq x_2 \leq 40$, $0 \leq x_3 \leq 40$, $100 \leq x_4 \leq 300$, $6.3 \leq x_5 \leq 6.7$, $5.9 \leq x_6 \leq 6.4$, $4.5 \leq x_7 \leq 6.25$
G22	22	-	$0 \leq x_1 \leq 20000$, $0 \leq x_i \leq 1 \times 10^6$ ($i=2, 3, 4$), $0 \leq x_i \leq 4 \times 10^7$ ($i=5, 6, 7$), $100 \leq x_8 \leq 299.99$, $100 \leq x_9 \leq 399.99$, $100.01 \leq x_{10} \leq 300$, $100 \leq x_{11} \leq 400$, $100 \leq x_{12} \leq 600$, $0 \leq x_i \leq 500$ ($i=13, 14, 15$), $0.01 \leq x_{16} \leq 300$, $0.01 \leq x_{17} \leq 400$, $-4.7 \leq x_i \leq 6.25$ ($i=18, \dots, 22$)
G23	9	-	$0 \leq x_i \leq 300$ ($i=1, 2, 6$), $0 \leq x_i \leq 100$ ($i=3, 5, 7$), $0 \leq x_i \leq 200$ ($i=4, 8$), $0.01 \leq x_9 \leq 0.03$
G24	2	-5.5080	$0 \leq x_1 \leq 3$, $0 \leq x_2 \leq 4$

Appendix C

Multi-objective Optimization Test Functions

Test problem 1(SCH):

$$\begin{aligned} \text{Min } f_1(\mathbf{x}) &= x^2 \\ \text{Min } f_2(\mathbf{x}) &= (x-2)^2 \end{aligned} \quad (\text{C-1})$$

where $D=1$ and $x \in [-10^3, 10^3]$. The optimal solutions are $x^* \in [0, 2]$.

Schaffer's problem (SCH) has a convex Pareto front [180].

Fig. C-1 The Search Space Near the Pareto-Optimal Region for SCH

Test problem 2 (FON):

$$\begin{aligned} \text{Min } f_1(\mathbf{x}) &= 1 - \exp\left(-\sum_{i=1}^3 \left(x_i - \frac{1}{\sqrt{D}}\right)^2\right) \\ \text{Min } f_2(\mathbf{x}) &= 1 - \exp\left(-\sum_{i=1}^3 \left(x_i + \frac{1}{\sqrt{D}}\right)^2\right) \end{aligned} \quad (\text{C-2})$$

where $D=3$ and $x_i \in [-4, 4]$. The optimal solutions are $x_1^*, x_2^*, x_3^* \in [-1/\sqrt{D}, 1/\sqrt{D}]$. This two-objective problem (FON) is proposed by Fonseca and Fleming [179] and it has a nonconvex Pareto front

Fig. C-2 *The Search Space Near the Pareto-Optimal Region for FON*

Test problem 3 (KUR):

$$\begin{aligned} \text{Min } f_1(\mathbf{x}) &= \sum_{i=1}^{D-1} (-10 \exp(-0.2 \sqrt{x_i^2 + x_{i+1}^2})) \\ \text{Min } f_2(\mathbf{x}) &= \sum_{i=1}^D (|x_i|^{0.8} + 5 \sin(x_i^3)) \end{aligned} \tag{C-3}$$

where $D=3$ and $x_i \in [-5, 5]$. The most left one solution is $x_1^* = x_2^* = x_3^* = 0$. Some Pareto-optimal solutions correspond to $x_1^* = x_2^* = 0$, and some solutions correspond to $x_1^* = x_3^*$. This problem was proposed by Kursawe [181]. The KUR problem has three disconnected Pareto-optimal regions, which may cause difficulty in finding non-dominated solutions in all regions.

Fig. C-3 *The Search Space Near the Pareto-Optimal Region for KUR*

Test problem 4 (ZDT 1):

ZDT1 has a convex Pareto front

$$\begin{aligned}
 &\text{Minimize } f_1(\mathbf{x}) = x_1 \\
 &\text{Minimize } f_2(\mathbf{x}) = g(\mathbf{x})[1 - \sqrt{x_1 / g(\mathbf{x})}] \\
 &g(\mathbf{x}) = 1 + 9 \cdot (\sum_{i=2}^D x_i) / (D-1)
 \end{aligned} \tag{C-4}$$

where $D = 30$ and $x_i \in [0, 1]$. The optimal solutions are $x_1 \in [0, 1]$ and $x_i = 0, i = 2, \dots, D$. The only difficulty an MOEA may face in this problem is the large number of variables.

Fig. C-4 The Search Space Near the Pareto-Optimal Region for ZDT1

Test problem 5 (ZDT2):

$$\begin{aligned}
 &\text{Min } f_1(\mathbf{x}) = x_1 \\
 &\text{Min } f_2(\mathbf{x}) = g(\mathbf{x})[1 - (x_1 / g(\mathbf{x}))^2] \\
 &g(\mathbf{x}) = 1 + 9(\sum_{i=2}^D x_i) / (D-1)
 \end{aligned} \tag{C-5}$$

where $D = 30$ and $x_i \in [0, 1]$. The optimal solutions are $x_1 \in [0, 1]$ and $x_i = 0, i = 2, \dots, n$. This problem has a nonconvex Pareto-optimal front.

APPENDIX C

Fig. C-5 The Search Space Near the Pareto-Optimal Region for ZDT2

Test problem 6 (ZDT3):

$$\begin{aligned} \text{Min } f_1(\mathbf{x}) &= x_1 \\ \text{Min } f_2(\mathbf{x}) &= g(\mathbf{x})[1 - \sqrt{x_1 / g(\mathbf{x})} - \frac{x_1}{g(\mathbf{x})} \sin(10\pi x_1)] \\ g(\mathbf{x}) &= 1 + 9(\sum_{i=2}^D x_i) / (D - 1) \end{aligned} \tag{C-6}$$

where $D = 30$ and $x_i \in [0, 1]$. The optimal solutions are $x_1 \in [0, 1]$ and $x_i = 0, i = 2, \dots, D$. ZDT3 problem has several disconnected Pareto-optimal fronts.

Fig. C-6 The Search Space Near the Pareto-Optimal Region for ZDT3

Test problem 7 (ZDT4):

$$\begin{aligned}
 \text{Min } f_1(\mathbf{x}) &= x_1 \\
 \text{Min } f_2(\mathbf{x}) &= g(\mathbf{x}) \left[1 - \sqrt{f_1(\mathbf{x}) / g(\mathbf{x})} \right] \\
 g(\mathbf{x}) &= 1 + 10(D-1) + \sum_{i=2}^D (x_i^2 - 10 \cos(4\pi x_i))
 \end{aligned} \tag{C-7}$$

where $D = 10$, $x_1 \in [0, 1]$ and $x_i \in [-5, 5]$ ($i = 2, \dots, D$). The optimal solutions are $x_1 \in [0, 1]$, $x_i = 0$, $i = 2, \dots, D$. There exists 21^9 local Pareto-optimal solutions, each corresponding to $x_1 \in [0, 1]$, $x_i = 0.5m$, $i = 2, \dots, D$, where m is any integer in $[-10, 10]$. These local optimal solutions construct 100 distinct Pareto-optimal fronts in the objective space.

Fig. C-7 The Search Space Near the Pareto-Optimal Region for ZDT4

Test problem 8 (ZDT6):

$$\begin{aligned}
 \text{Min } f_1(\mathbf{x}) &= 1 - \exp(-4x_1) \sin^6(6\pi x_1) \\
 \text{Min } f_2(\mathbf{x}) &= g(\mathbf{x}) \left[1 - (f_1(\mathbf{x}) / g(\mathbf{x}))^2 \right] \\
 g(\mathbf{x}) &= 1 + 9 \left[\left(\sum_{i=2}^D x_i \right) / (D-1) \right]^{0.25}
 \end{aligned} \tag{C-8}$$

where $D = 10$ and $x_i \in [0, 1]$. The optimal solutions are $x_1 \in [0, 1]$, $x_i = 0$, $i = 2, \dots, D$.

ZDT6 is another hard problem. This problem has non-convex and non-uniformly

APPENDIX C

spaced Pareto-optimal fronts. The adverse density of solutions across the Pareto-optimal front, together with the non-convex nature of the front, makes it difficult for many multi-objective optimization algorithms to maintain a well-distributed non-dominated set and converge to the true Pareto-optimal front.

Fig. C-8 *The Search Space Near the Pareto-Optimal Region for ZDT6*