Evolutionary Algorithms for Solving Multi-modal and Multi-objective Optimization Problems

Qu Boyang

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University

in fulfillment of the requirements for the degree of

Doctor of Philosophy

2011

Abstract

In artificial intelligence, evolutionary algorithms (EAs) have shown to be effective and robust in solving difficult optimization problems. EAs are generic population-based metaheuristic optimization algorithms. The mechanisms used in EAs are inspired by biological evolution: reproduction, mutation, recombination, and selection. The development of EAs can be classified into two categories: single objective and multi-objective optimization.

In this thesis, both single objective and multi-objective evolutionary algorithms have been studied. For single objective optimization, various niching techniques are integrated with differential evolution (DE) and particle swarm optimization (PSO) for multi-modal optimization. Multi-modal optimization deals with optimization tasks that involve finding all or most of the global/local peaks in one single run. EAs in their original forms are usually designed for locating one single global solution. To promote and maintain formation of multiple stable subpopulations within a single population, we introduced a neighborhood mutation technique to enhance DE with ability of handling multi-modal problems. We also proposed a locally informed PSO to tackle multi-modal optimization. Beside these, several existing niching techniques from the literature were modified and improved by us.

For multi-objective evolutionary algorithms, we proposed a summation of normalized objective **v**alues and diversified **s**election (SNOV-DS) method to replace the classical non-domination sorting. The process of classical non-domination sorting is complex and time consuming. By use of the proposed method, not only the simulation speed is increased, but also the performance of the algorithm is improved. We also introduced an ensemble of constraint handling methods (ECHM) to solve constrained multi-objective optimization problems, where each constraint handling method had its own population. ECHM allows different constraint handling methods to generate offspring and exchange

information. In this way, the offspring produced by the most suitable constraint handling method will survive and be set as parents for next generation.

Lastly, we applied the proposed algorithm to solve environmental/economic power dispatch problem. We demonstrated the superior performance of the proposed algorithm over other similar evolutionary algorithms reported in literature.

Acknowledgments

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

First of all, I am heartily thankful to my supervisor, Dr. Ponnuthurai Nagaratnam Suganthan, whose encouragement, guidance and support from the initial to the final level enabled me to develop an understanding of the subject. 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 would like to thank my collaborators Dr. Zhang Qingfu, Dr. Swagatam Das and Dr. Bijaya Ketan Panigrahi for their help, discussions and suggestions during our collaborative work. I also would like to show my gratitude to Dr. Li Xiaodong, Dr. Ofer M. Shir, Dr. Mike Preuss, and Dr. Pan Quanke for their help and discussion during my Ph.D study.

I am especially grateful to ex-lab members Dr. Liang Jing, Dr. Ashish Anand and others for their help and suggestions during the initial period of my research. It is my pleasure to work and share the lab with current lab members Rammohan, Zhao Shizheng, Zhang Rui, Zhong Weiwei, Zhou Hongming, Liu Fan and others. I thank lab staff members Mr Tan Peng Chye and Mr Pua for providing all technical support and help.

I express my sincere gratitude to my friends Jiao Hong, Wei Han, Yang Wentao and many others who provided me with their constant support over the past years. I also thank all the people who have directly or indirectly helped in the completion of the thesis.

I am grateful to Nanyang Technological University for providing me the research scholarship and excellent research facilities.

Last but not least, I would like to thank my dearest parents, especially my father who passed away in the third year of my Ph.D study. Their encouragement put me through these years towards the completion of the work. Without their moral support, the completion of this study would not be possible. My gratitude for them is beyond words.

Table of Contents

Summary

In artificial intelligence, evolutionary algorithms (EAs) have shown to be effective and robust in solving difficult optimization problems. EAs are generic population-based metaheuristic optimization algorithms. The mechanisms used in EAs are inspired by biological evolution: reproduction, mutation, recombination, and selection. The development of EAs can be classified into two categories: single objective and multi-objective optimization.

In this thesis, both single objective and multi-objective evolutionary algorithms have been studied. For single objective optimization, various niching techniques are integrated with differential evolution (DE) and particle swarm optimization (PSO) for multi-modal optimization. Multi-modal optimization deals with optimization tasks that involve finding all or most of the global/local peaks in one single run. EAs in their original forms are usually designed for locating one single global solution. To promote and maintain formation of multiple stable subpopulations within a single population, we introduced a neighborhood mutation technique to enhance DE with ability for handling multi-modal problems. We also proposed a locally informed PSO to tackle multi-modal optimization. Beside these, several existing niching techniques from the literature were modified and improved by us.

For multi-objective evolutionary algorithms, we proposed a summation of normalized objective **v**alues and diversified **s**election (SNOV-DS) method to replace the classical non-domination sorting. The process of classical non-domination sorting is complex and time consuming. By using the proposed method, not only the simulation speed is increased, but also the performance of the algorithm is improved. We also introduced an ensemble of constraint handling methods (ECHM) to solve constrained multi-objective optimization problems, where each constraint handling method had its own population. ECHM allows different constraint handling methods to generate offspring and exchange

information. In this way, the offspring produced by the most suitable constraint handling method will survive and be set as parents for next generation.

Lastly, we applied the proposed algorithm to solve environmental/economic power dispatch problem. We demonstrated the superior performance of the proposed algorithm over other similar evolutionary algorithms reported in literature.

List of Figures

List of Tables

Chapter 1

Introduction

Optimization occurs frequently in real-world scenarios. It forms an important part of our daily life. Engineers optimize the parameters of their designs. Manufacturers seek for maximum efficiency in their production processes. Investors aim for portfolios with the maximum return and the minimum risk exposure. Optimization is a procedure of finding and comparing solutions until no better solution can be found. Solutions are termed either good or bad in terms of objective, a quantitative measure of the performance of the system under study. The objective could be cost of fabrication, product reliability, potential energy, efficiency of a process which is controlled by certain characteristics of the system, called variables [24]. Depending on the number of objectives, an optimization problem can be classified as single objective or multi-objective.

Classical optimization techniques make use of differential calculus to locate the optimum solution. In order for classical optimization to work effectively, the function has to be differentiable twice with respect to the design variables, and the derivatives have to be continuous. This requirement limits the scope of classical methods in real-world applications as the practical problems are characterized by chaotic disturbances, randomness and complex non-linear dynamics which may be neither continuous nor differentiable. In recent decades, interests have shifted to evolutionary algorithms (EAs) to solve optimization problems due to their ability to handle non-continuous and/or non-differentiable functions. One of the most striking differences from classical optimization techniques is that EAs use a population of solutions in each iteration, instead of a single solution [1]. All the population members are expected to converge to the global optimum solution if the problem has a single optimum. However, EAs also can be used to capture multiple peaks if an optimization problem has multiple optimal solutions. Another attractive feature of EAs is that they only require the objective function values, while

properties such as differentiability and continuity are not necessary. Compared with classical optimization algorithms, EAs are more effective in solving complex optimization problems [2].

In past few decades, various EAs have been proposed. Due to different requirements and increase of complexity in real application, better and more stable EAs will always be needed. This thesis mainly investigates two popular EAs known as Differential Evolution (DE) and Particle Swarm Optimizer (PSO).

1.1 Motivation

Evolutionary algorithms (EAs) are a subset of evolutionary computation that involves techniques inspired by biological evolution such as reproduction, mutation, recombination, natural selection and survival of the fittest. Computational models of evolutionary processes are used by EAs as the key elements in design and implementation of computer-based problem solving system. Numerous EAs such as Genetic Algorithms (GAs) [3], Evolutionary Programming (EP) [4], [5], and Evolution Strategies (ES) [6] have been proposed to solve optimization problems. Differential Evolution (DE) and Particle Swarm Optimizer (PSO) are two relatively new techniques that have been proposed to solve complex optimization problems.

DE is a population-based nonlinear continuous global optimization algorithm. It was first reported by R. Storn and K. V. Price in 1995 [7]. Although the concept of DE is simple, it has been proven to be very competitive in solving optimization problems [8]-[10]. Different from other EAs, DE modifies individuals by using differences of randomly sampled pairs of individual vectors from the population. DE works through a simple cycle of stages which is presented in Fig. 1-1-1[21].

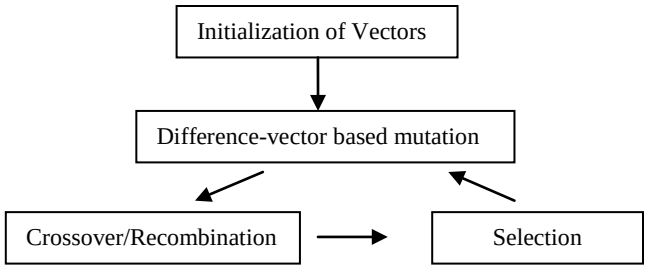


Fig 1-1-1 Main stages of DE algorithm

The Particle Swarm Optimizer (PSO) [11], [12] is based on swarm intelligence that inspired by the choreography of bird flocks, animal herds and fish schools. Similar to other evolutionary algorithms, PSO is also a population based search technique. The particular characteristic of PSO is that no mutation or crossover operators are involved in the updating process. In PSO, there are two types of swarms known as *explorer-swarm* and *memory-swarm*. The *explorer-swarm* which contains all the current particles is responsible for searching of new points. The *memory-swarm* contains all the personal bests, which can be viewed as a way of maintaining better points found in the search space so far [73].

Although DE and PSO were originally designed for single objective global optimization, many practical problems involve either multiple global/local optima or multiple objectives. The classical DE and PSO fail when solving these kinds of problems. Motivated by these observations, the target of this research is to construct new and efficient algorithms based on DE and PSO to improve their ability of solving multi-modal and multi-objective optimization problems.

1.2 Objectives

The main objectives of this thesis are:

- To develop novel DE algorithms with niching ability to solve multi-modal optimization problems.
- To develop efficient niching PSO algorithms to solve multi-modal optimization problems.

- To improve some existing DE and PSO based niching algorithms.
- To develop novel DE and efficient algorithms to solve multi-objective optimization problems.
- To develop novel and efficient constraint-handling techniques to solve multi-objective constrained optimization problems.
- To apply the proposed algorithms in solving real-world problem.

1.3 Major Contributions of the Thesis

The contributions of this thesis can be summarized as follows:

- A Dynamic Grouping Crowding Differential Evolution (DGCDE) with an ensemble of parameters is proposed. In this algorithm, the population is dynamically regrouped into three equal subpopulations every few generations. Each of the subpopulations is assigned a set of parameters. In this way the algorithm is able to make use of different sets of DE control parameters and exchange information between groups. The proposed algorithm is able to generate more stable niching behavior over the classical Crowding Differential Evolution (CDE).
- A modified Species-based DE (SDE) with a self-adaptive radius is introduced to overcome the difficulty of selecting the proper radius and improve the performance of SDE. The original SDE suffers from a serious problem that is its performance is subjected heavily to the niching parameters known as the radius. With the proposed method, the radius will be self-adjusted through the evolution process and enhance the niching ability of the algorithm.
- A neighborhood based differential evolution is proposed to limit the mutation within the Euclidean neighborhood. The neighborhood mutation is able to maintain the multiple peaks found during the evolution and evolve toward the respective global/local optimum, which is effective in solving multi-modal optimization problems.

- A distance based locally informed particle swarm optimization (LIPS) is presented, which eliminates the need to specify any niching parameters and enhance the fine search ability of PSO. Instead of using global best, LIPS uses several local bests to lead the particle. LIPS can operate as a stable niching algorithm by using the information provided by its neighborhoods.
- The lack of fine searching ability of the original niching PSO curtails the ability to locate most local or global optima. A local search technique is introduced and incorporated with some existing PSO based multi-modal optimization algorithms to enhance their fine search ability.
- A Summation of Normalized Objective Values and Diversified **S**election (SNOV-DS) method is proposed and integrated with multi-objective differential evolution (MODE) and multi-objective evolutionary programming (MOEP) to improve the performance and the speed of the parent selection process.
- An ensemble of constraint handling methods (ECHM) is proposed to tackle constrained multi-objective optimization problems. The algorithm makes use of different constraint handling methods and exchanges information between different populations.
- The proposed multi-objective algorithm is applied to Environmental/ Economic dispatch (EED) problem.

1.4 Organization of the Thesis

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

Chapter 2 presents an introduction of the evolutionary algorithms, multi-modal optimization and multi-objective optimization. A detailed literature review of Differential Evolution and Particle Swarm Optimizer used in the Thesis is also presented in this section.

Chapter 3 and 4 introduce the proposed multi-modal optimization algorithms. In chapter 3, Dynamic Grouping Crowding Differential Evolution (DGCDE),

modified SDE with a self-adaptive radius and neighborhood based DE are presented, while in chapter 4, distance based locally informed particle swarm optimization (LIPS) and niching PSO with local search technique are presented. Note that this thesis presented 5 niching algorithms that were proposed in different periods. Some early algorithms such as DGCDE, MSDE and PSOLS are relatively simple compared to the recently proposed neighborhood DE and LIPS. To keep the thesis relatively consistent, we used classical test functions for the simple algorithms (DGCDE, MSDE and PSOLS) and both classical and composition test functions for the two more powerful algorithms (neighborhood DE and LIPS) in this thesis.

Chapter 5 presents the multi-objective optimization developed in this work including multi-objective evolutionary algorithms based on the summation of normalized objectives with diversity selection (SNOV-DS) and constrained multi-objective optimization algorithm with ensemble of constraint handling.

Chapter 6 presents the application of the proposed multi-objective evolutionary algorithms. The MOEA with SNOV-DS is applied on environmental economic dispatch problem to balance the cost and pollution.

Finally, the conclusions and recommendations for future research work are addressed in Chapter 7.

Chapter 2

Background and Literature Review

This chapter introduces the theory of optimization, evolutionary algorithms, multi-modal optimization and multi-objective optimization. A detailed description of the Differential Evolution (DE) and Particle Swarm Optimizer (PSO) is also presented in this chapter.

2.1 Optimization

Optimization refers to choosing of the best element from some set of available alternatives. The process involves tuning of a set of variables known as the parameter space to minimize or maximize the given objective functions. Fig 2-1-1 shows the optimization (maximization) of a paraboloid function. *x* and *y* are the parameter space while *z* is the objective. The point at the top of the paraboloid is the optimum.

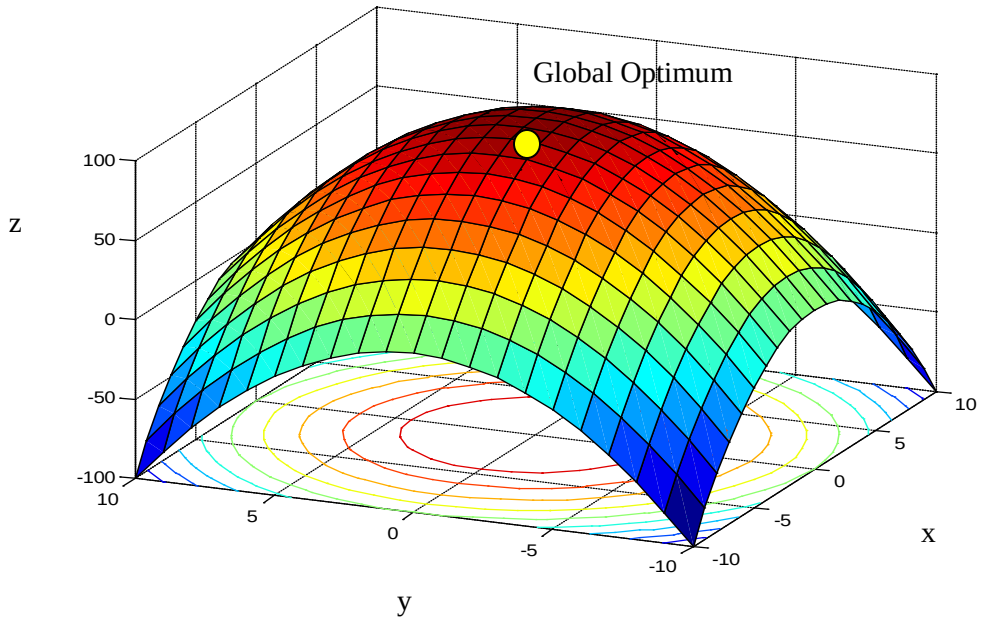


Fig 2-1-1 Optimization of a paraboloid

Formally an optimization problem can be formulated as follows:

$$\text{Maximize/Minimize } f_i(\mathbf{x}), i = 1,...,M,\ \mathbf{x} = [x_1, x_2, ..., x_D] \qquad (2-1-1)$$

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

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

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

where $f_i(\mathbf{x})$ is called "objective function", $g_j(\boldsymbol{x})$ and $h_k(\boldsymbol{x})$ are inequality constraint and equality constraint functions respectively. $\boldsymbol{x} = (x_1, x_2, \ldots, x_D)$ are *D*-dimensional decision vector. 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. Optimization problems with only bounds constraints, $\mathbf{x} \in [\mathbf{X}_{\min}, \mathbf{X}_{\max}]^D$, are regarded as unconstrained problems. Constraints divide the search space into two divisions – feasible and infeasible regions. Individuals that satisfy all of the constraints are called feasible individuals while individuals that do not satisfy at least one of the constraints are called infeasible individuals.

For some optimization problems, it is possible to have both global/local peaks. The definitions of global/local peaks (minimum) are shown as below. Note that a maximization problem can be transformed into a minimization problem by multiplying -1.

Definition 2.2.1 (Global minimum): Given a function $f : M \subseteq \mathbb{R}^n \to \mathbb{R}, M \neq \Phi$, for $\mathbf{x}^* \in M$ the value $f^* := f(\mathbf{x}^*) > -\infty$ is called a global minimum, *if*

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

Then, **x** is a ***global minimum*** point. The problem of determining a global minimum point is called the ***global minimization*** problem [24].

Definition 2.2.2 (Local minimum): For $\hat{\mathbf{x}} \in M$ the value $\hat{f} := f(\hat{\mathbf{x}})$ is called a ***local minimum***, *if* $\exists \varepsilon \in \mathbb{R}, \varepsilon > 0 : \forall \mathbf{x} \in M : \|\mathbf{x} - \hat{\mathbf{x}}\| < \varepsilon \Rightarrow \hat{f} \le f(\mathbf{x})$ In other

words, an ε-environment $U_{\varepsilon}(\hat{\mathbf{x}}) = \{\mathbf{x} \in M \mid \|\mathbf{x} - \hat{\mathbf{x}}\| < \varepsilon\}$ exists such that $\hat{f}$ is the smallest feasible objective function value within this environment [24].

In order to give a clearer view of global/local optima, Fig. 2-1-2 is presented which shows two local optima and one global optimum.

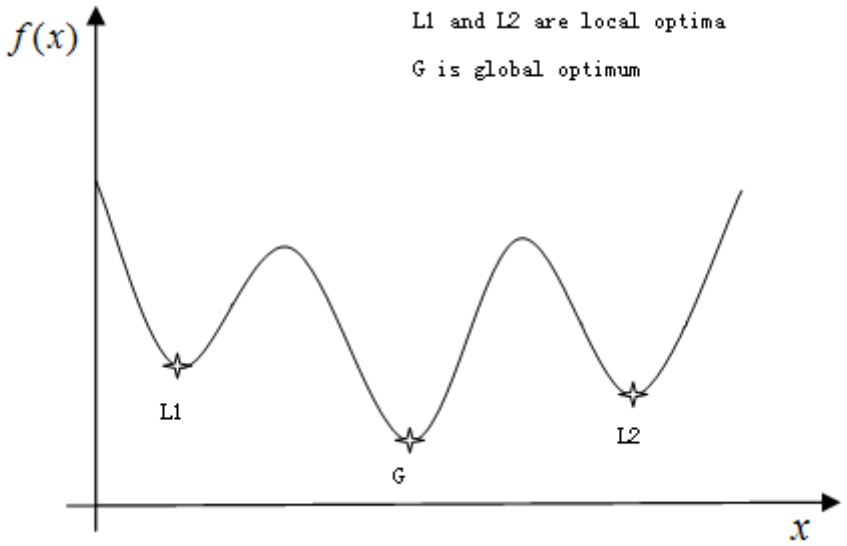


Fig. 2-1-2 Illustration of global optimum and local optima

2.2 Evolutionary Algorithms

Evolutionary Algorithms (EAs) inspired by Darwinian Theory of evolution, are a class of stochastic search techniques applicable to a wide range of problems in engineering and other real-world optimization. EAs are developed based on the natural selection and survival of the fittest in the biological world. The optimization is realized through the process including reproduction, mutation, recombination, natural selection and survival of the fittest.

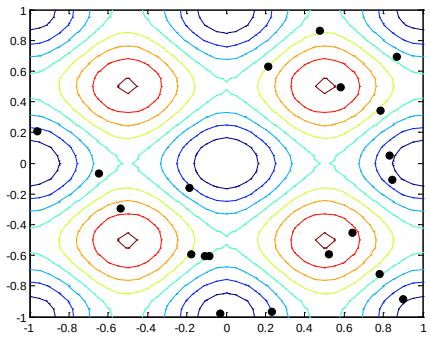
EAs deal with a population of individuals where each individual represents an instantiation of the solution to the underlying problem. The individuals are

evaluated using the objective/fitness function defined by the problem. Generally, EAs start with a randomly generated population of individuals, which communicate and exchange information to produce new individuals and evolve increasingly fitter individuals to a particular environment. The steps of a typical EA are summarized in Table 2-2-1.

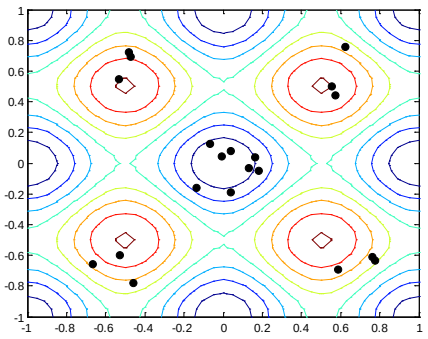
Table 2-2-1 The general steps of a typical EA

Step 1	Randomly generate the initial population to a given problem in a predefined search space.
Step 2	Evaluate the individuals using the fitness function
Step 3	Generate the offspring through the crossover and mutation operations.
Step 4	Evaluate the offspring using the fitness function
Step 5	Compare the newly generated offspring with the parents and keep the ones with higher fitness.
Step 6	Stop if the termination criteria are met, otherwise go to step 3.

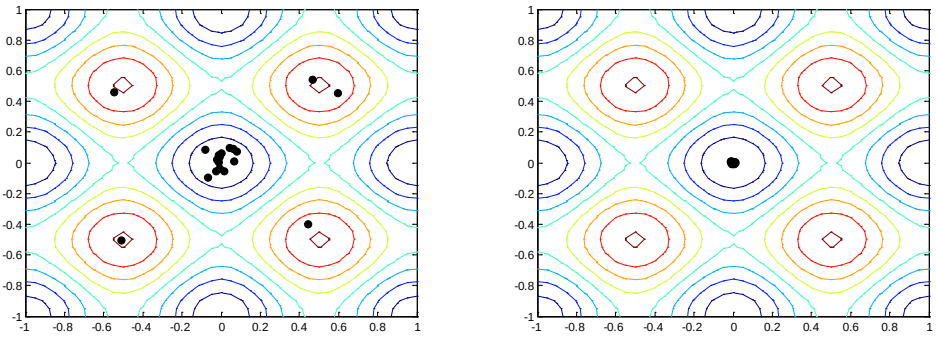
For complex optimization problems, EAs have a higher chance to avoid getting stuck into a local optimum than classical optimization methods. Hence, EAs can potentially generate better and more stable results. The illustration of search behavior of EAs which is taken from [63] is shown in Fig. 2-2-1.



(a) Step 1



(b) Step 2



(c) Step 3 (d) Step 4

Fig. 2-2-1 The search behavior of EAs [63]

The origins of EAs can be traced back to 1954 [13]. In 1957, a number of papers were published on simulation of artificial selection of organisms [14]. Then in 1960s several evolutionary methodologies had been proposed, such as genetic algorithms [15], evolutionary programming [5], and evolution strategies [17]. GAs usually use selection operator first to select good individuals, and then apply the recombination and mutation operators on these individuals to create a potential better set of solutions. On the other hand, ES and EP prefer using the recombination and mutation operator first to create a set of solutions and then use the selection operator to choose a set of good solutions [18]. The classical GAs [3], [15] consider each solution as a chromosome, represented by a bit string of fixed length. In GA, two solutions known as the parents combine to produce an offspring. Unlike GAs, EP [5] does not use any crossover operator and relies on asexual reproduction. In EP, only one parent is used in the creation of an offspring. For ES [17], either sexual (only one parent used in the creation of an offspring) or panmictic (several parents are used in creation of an offspring) can be used. The characteristics of these three classical EAs are summarized in Table 2-2-2 [20].

Besides these EAs, two recently proposed algorithms, Differential Evolution (DE) and Particle Swarm Optimizer (PSO) have attracted many researchers' interests. Both of these algorithms are population-based optimization algorithms. Although the concepts of these algorithms are simple, they are recognized as two of the most

effective optimization algorithms in literature. The details of these two EAs are presented in the following sections.

Table 2-2-2 Characteristics of different EAs [18]

	GA	EP	ES
Representation	Binary / Real-valued	Real-valued / continuous	Real-valued / continuous
Adaptability of the variation operator	None	Adaptive / Self-adaptive	Self-adaptive
Reproduction	Sexual	Asexual	sexual / Panmictic
Mutation	Bit-inversion	Gaussian / Cauchy	Gaussian / Cauchy
Recombination	uniform crossover	None	Discrete / Intermediate
Selection	Probabilistic / based on preservation	Probabilistic / extinctive	Deterministic / extinctive / based on preservation

2.3 Differential Evolution

Differential evolution (DE) was originally presented by R. Storn and K. V. Price in 1995 [7]. It has been proven to be one of the most competitive EAs [7]-[9], [21]. Similar to other EAs, DE is also a population-based stochastic global optimization technique. The optimization is realized through the following steps: initialization, mutation, recombination and selection of parents for next generation [21].

- Initialization

  DE is a parallel direct search method that searches for a global optimum point in a *D*-dimensional real parameter space. The algorithm starts with a randomly initiated population of *NP D*-dimensional real-valued parameter vectors. Each vector is considered as a candidate solution to the

optimization problem. The vector can be represented as $\mathrm{X}_{i,G} = \left\{x_{i,G}^{1},...,x_{i,G}^{D}\right\}, i = 1,...,NP$ where G represents the generation count. The vector is generally initialized using the equation:

$$x_{i,0}^{j} = x_{\min}^{j} + rand^{j}(0,1).(x_{\max}^{j} - x_{\min}^{j}) \qquad j = 1, 2, ..., D \qquad (2-3-1)$$

where j represents the j^{th} component of the i^{th} individual. $rand^{j}(0,1)$ represents a uniformly distributed random variable within the range [0,1].

- Mutation

A mutant vector $\mathbf{V}_{i,G}$ is generated by the parent vector (target vector) $\mathbf{X}_{i,G}$ after the initialization. In literature, various strategies have been proposed for mutation operation. The most frequently used are listed as below [10], [18], [22], [23]:

"DE/best/1":

$$\mathrm{V}_{i,G} = \mathrm{X}_{best,G} + F.(\mathrm{X}_{r_1^i,G} - \mathrm{X}_{r_2^i,G}) \qquad (2-3-2)$$

"DE/best/2":

$$\mathrm{V}_{i,G} = \mathrm{X}_{best,G} + F.(\mathrm{X}_{r_1^i,G} - \mathrm{X}_{r_2^i,G}) + F.(\mathrm{X}_{r_3^i,G} - \mathrm{X}_{r_4^i,G}) \qquad (2-3-3)$$

"DE/rand/1":

$$\mathrm{V}_{i,G} = \mathrm{X}_{r_1^i,G} + F.(\mathrm{X}_{r_2^i,G} - \mathrm{X}_{r_3^i,G}) \qquad (2-3-4)$$

"DE/rand/2":

$$\mathrm{V}_{i,G} = \mathrm{X}_{r_1^i,G} + F.(\mathrm{X}_{r_2^i,G} - \mathrm{X}_{r_3^i,G}) + F.(\mathrm{X}_{r_4^i,G} - \mathrm{X}_{r_5^i,G}) \qquad (2-3-5)$$

"DE/rand-to-best/1"

$$\left.\begin{array}{l} \mathrm{V}_{i,G} = \mathrm{X}_{i,G} + F.(\mathrm{X}_{best,G} - \mathrm{X}_{i,G}) \\ + F.(\mathrm{X}_{r_1^i,G} - \mathrm{X}_{r_{2,G}^i}) \end{array}\right\} \qquad (2-3-6)$$

"DE/target-to-best/1"

"DE/rand-to-best/2"

$$\left.\begin{array}{l} V_{i,G} = X_{i,G} + F.(X_{best,G} - X_{i,G}) + \\ F.(X_{r_1^i,G} - X_{r_2^i,G} + X_{r_3^i,G} - X_{r_4^i,G}) \end{array}\right\} \quad (2\text{-}3\text{-}7)$$

"DE/target-to-best/2"

"DE/current-to-rand/1":

$$V_{i,G} = X_{i,G} + F.(X_{r_1^i,G} - X_{i,G}) + F.(X_{r_2^i,G} - X_{r_3^i,G}) \quad (2\text{-}3\text{-}8)$$

The indices $r_1^i, r_2^i, r_3^i, r_4^i, r_5^i$ are mutually exclusive integers randomly generated within the range $[1, NP]$, which are also different from the index *i*. These indices are randomly generated for each mutant vector. The scaling factor *F* is a control parameter for scaling the difference vector which is in the range of (0, 1+) [22]. $\mathbf{X}_{best,G}$ is the best individual vector with the best fitness value in the population at generation *G*. In fig. 2-3-1, the process of generating a difference vector is illustrated on a 2-D parameter space.

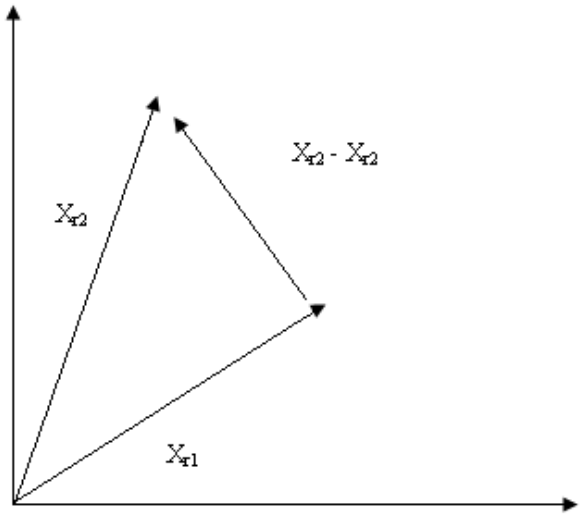


Fig. 2-3-1 Illustration of a difference vector generation in 2-D parametric space

- Crossover

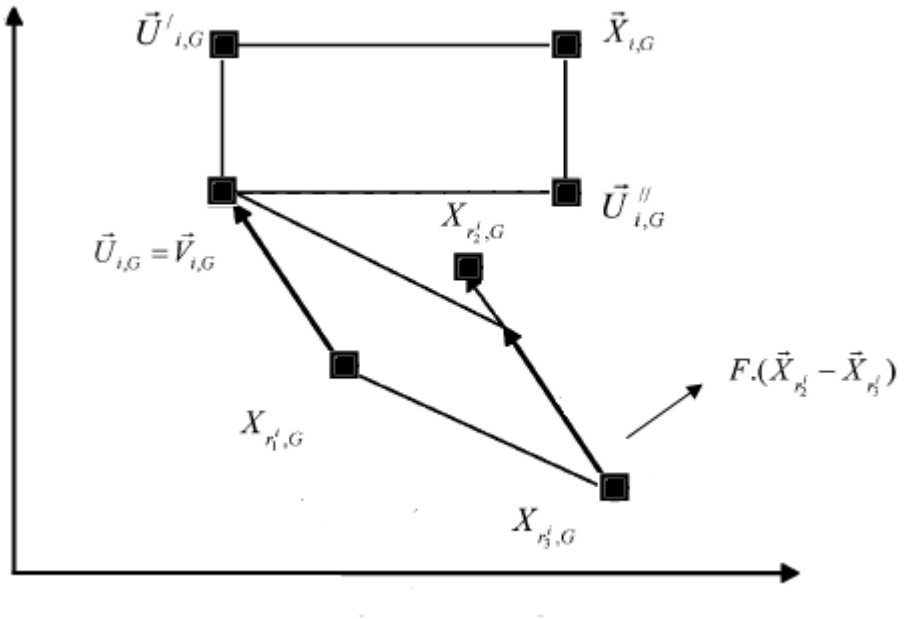


Fig. 2-3-2 Different possible trial vectors formed due to uniform crossover [21]

After the mutation, crossover operation is applied to each pair of the target vector $\mathbf{X}_{i,G}$ and its corresponding mutant vector $\mathbf{V}_{i,G}$ to produce a trial vector: $\mathrm{U}_{i,G} = \left\{u_{i,G}^1, u_{i,G}^2, ..., u_{i,G}^D\right\}$. The crossover takes components from both the parent and the mutant vector that speeds the convergence by a constant factor [22]. Generally *binomial* (or uniform) and *exponential* (or two-point modulo) are used as the crossover operators. The binomial crossover operator is defined as follow [9]:

$$u_{i,G}^j = \begin{cases} v_{i,G}^j & \text{if } (rand^j(0,1) \le CR) \text{ or } (j = j_{rand}) \\ x_{i,G}^j & \text{otherwise} \end{cases} \qquad j = 1,\ 2,\ \ldots,\ D \quad (2-3-9)$$

where $CR \in [0, 1]$ is a user-specified constant that controls the fraction of parameter values copied to the trial vector from the mutant vector and known as the crossover rate. j_{rand} is a randomly chosen integer in the range [1, D]. The binomial crossover operator copies the j^{th} parameter of the mutant vector $\mathbf{V}_{i,G}$ to the corresponding element in the trial vector $\mathbf{U}_{i,G}$

if $rand_j[0,1) \le CR$ or $j = j_{rand}$. Otherwise, it is copied from the corresponding target vector $\mathbf{X}_{i,G}$. The condition $j = j_{rand}$ is introduced to ensure that the trial vector $\mathbf{U}_{i,G}$ will differ from its corresponding target vector $\mathbf{X}_{i,G}$ by at least one parameter [18], [24]. The possible trial vectors due to uniform crossover in *2D* search space are illustrated in fig. 2-3-2 [21].

In exponential crossover [25], an integer $n \in [1, D]$ is chosen randomly, which acts as a starting point in the target vector, from where the crossover or exchange of components with the mutant vector starts. $L \in [1, D]$ denotes the number of components that are contributed by the mutant vector to the target vector. The integer L is drawn from [1, D] depending on the value of *CR*. The exponential crossover can be outlined as [18]:

$$u_{i,G}^{j} = \begin{cases} v_{i,G}^{j}, \text{ for } j = \langle n \rangle_D, \langle n+1 \rangle_D, ..., \langle n+L-1 \rangle_D \\ x_{i,G}^{j}, \text{for all other } j \in [1, D] \end{cases} \qquad (2-3-10)$$

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

- Selection

  In order to keep the population size constant over generations, the selection step is needed to determine whether the target or the trial vector survives to the next generation. In selection, the fitness value of each trial vector $f(\mathbf{U}_{i,G})$ is compared with its corresponding target vector's fitness $f(\mathbf{X}_{i,G})$. If the trial vector has a better fitness than that of the corresponding target, the trial vector will replace the target vector. Otherwise, the target vector will remain in the population for the next generation and the trial vector is discarded. The selection operation can be expressed as follows:

$$\mathbf{X}_{i,G+1} = \begin{cases} \mathbf{U}_{i,G}, & \text{if } f(\mathbf{U}_{i,G}) \le f(\mathbf{X}_{i,G}) \\ \mathbf{X}_{i,G}, & \text{otherwise} \end{cases} \qquad (2-3-11)$$

The Pseudo-code of DE algorithm is presented as follow [18], [21], [24]:

Table 2-3-1: The algorithmic description of DE [21]

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

Step 2 WHILE stopping criterion is not satisfied

DO

Step 2.1 **Mutation step**

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

FOR i = 1 to NP

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

END FOR

Step 2.2 **Crossover step**

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

a) Binomial crossover

FOR i = 1 to NP

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

FOR j = 1 to D

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

END FOR

END FOR

b) Exponential crossover

FOR i = 1 to NP

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

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

DO

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

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

$L=L+1$

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

END FOR

Step 2.3 **Selection step**

//Selection

FOR i =1 to NP

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

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

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

END IF

END IF

END FOR

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

Step 3 **END WHILE**

The performance of DE is greatly related to the techniques selected in the main steps and the control parameters. Inappropriate choice of mutation strategy and parameters may lead to premature convergence, stagnation or wastage of computational resources [22], [26]-[28]. The following subsections present the discussion on the control parameters used in DE [18], [21].

- Population size

  To obtain a global optimum while avoiding premature convergence, the population size (*NP*) of *10D* (D being the dimensionality of the problem) is initially suggested in [10]. Other different population sizes such as 5*D* to 10*D*, 3*D* to 8*D* are also used to balance the speed of convergence with the diversity of the population. Generally, a larger population will result in a better diversity which gives a higher probability of finding a global optimum. However, a larger population also implies a slower convergence rate which requires more function evaluations and resources to reach the global optimum.

- Scaling factor

  The scaling factor is a positive number used to scale the difference vector in DE. A good initial choice of *F* is 0.5. The effective range of *F* is usually between 0.4 and 1 [10]. According to [29], values of *F* smaller than 0.4 and greater than 1.0 are occasionally effective. Generally, a value *F* of 1.2 is considered as the upper limit for scaling factor. A larger *F* increases the probability of escaping from a local optimum and prevents the premature

convergence [30]. On the other hand, a larger *F* decreases the convergence speed.

- Crossover rate

  The crossover rate (*CR*) controls the percentage of the parameters in expectation are changed in a population member. Generally, the value of *CR* should be between zero and one. A small value of *CR* means a small fraction of parameters are mutated in each generation and the stepwise movement tends to be orthogonal to the current coordinate axes [21]. In contrast, a high value of *CR* causes most of the directions of the mutant vector to be inherited, thus prohibiting the generation of axis orthogonal steps. For separable problems, *CR* in the range of (0, 0.2) is able to generate a good performance. While for multi-modal and parameter dependent problems, a large CR value in the range of (0.9, 1) is more suitable to generate a good result.

Besides static *F* and *CR* values, many works have been presented to use either dynamic or adaptive control parameters [23], [31]-[33]. In [31], a fitness-based adaptation has been proposed. Although the crossover rate is fixed to 0.5, the value of scaling factor is adaptively updated at each generation using [18]:

$$F = \begin{cases} \max\left\{ l_{\min}, 1 - \left| \dfrac{f_{\max}}{f_{\min}} \right| \right\} & \text{if} \quad \left| \dfrac{f_{\max}}{f_{\min}} \right| < 1 \\ \max\left\{ l_{\min}, 1 - \left| \dfrac{f_{\min}}{f_{\max}} \right| \right\} & \text{otherwise,} \end{cases} \qquad (2-3-12)$$

where $l_{\min} = 0.4$ is the lower bound of *f*, $f_{\min}$ and $f_{\max}$ are the minimum and maximum objective function values over the individuals of the populations, obtained in a generation. In 2009, *Qin et al.* [23] introduced a SaDE algorithm, in which both the strategies and their associated control parameters are gradually self-adapted by using their previous experiences. In [32], Brest *et al*. proposed a self-adaptation method, in which control parameters *F* and *CR* were encoded into the individuals and were adjusted by introducing two new parameters τ_1 and τ_2. The

initial *F* and *CR* values are set to 0.5 and 0.9 respectively. The control parameters for the following generations are computed as:

$$F_{i,G+1} = \begin{cases} F_l + rand_1 * F_u \text{ if } rand_2 < \tau_1 \\ F_{i,G} \quad \text{otherwise} \end{cases} \qquad (2-3-13)$$

$$CR_{i,G+1} = \begin{cases} rand_3 \text{ if } rand_4 < \tau_2 \\ CR_{i,G} \quad \text{otherwise} \end{cases} \qquad (2-3-14)$$

where F_l and F_u are the lower and upper limits of scaling factor. In [32], $\tau_1 = \tau_2 = 0.1$ while the new *F* takes a value from [0.1, 0.9] and the new *CR* takes a value from [0, 1].

Due to the simplicity and outstanding performance of DE, it has been successfully applied in many areas. Table 2-3-2 lists a few significant applications in various fields [18], [21].

Table 2-3-2 Applications of DE

Areas	Application	References
Control systems and robotics	System identification	[34], [35]
	Robot motion planning and navigation	[36], [37]
Electrical power systems	Economic dispatch	[38]– [40]
	Power filter, power system	[41]
Electromagnetism and microwave engineering	Electromagnetic imaging	[42]
	Antenna array design	[43], [44]
Bioinformatics	Gene regulatory networks	[45], [46]
	Protein folding	[47]
Chemical engineering	Chemical process synthesis and design	[48]
	Parameter estimation of chemical process	[49]
Pattern recognition and image processing	Data clustering	[50]
	Image watermarking	[51]
Artificial neural networks	Training of feed-forward ANNs	[52]
	Training of B-Spline neural networks	[53]

Signal Processing	Digital filter design	[54], [55]
	Fractional order signal processing	[56]
Others	Layout synthesis for MEMS	[57]
	Engineering design	[58]
	Manufacturing optimization	[59]
	Optoelectronics	[60]

2.4 Particle Swarm Optimization

Particle swarm optimization (PSO) is one of the recent evolutionary algorithms based on swarm intelligence. It simulates the behaviors of swarms such as fish schools, bird flocks. The initial idea of PSO was introduced by Kennedy and Eberhart to produce computational intelligence by exploiting simple analogues of social interaction, rather than purely individual cognitive abilities. The first paper on PSO emerged in 1995 [11] and soon it was recognized as one of the most powerful optimization methods in EA world.

In PSO, solutions of the objective function are represented by a number of simple entities known as the particles. Each particle then moves through the search space by combining some aspect of the history of its own best (best-fitness) with those of one or more members of the swarm, with some random perturbations. Each of the particles is formed using three *D*-dimensional vectors: current position, personal best position and the velocity. The current position and personal best position are known as the *explorer-swarm* and *memory-swarm* respectively. The *explorer-swarm* is responsible for searching new points while the *memory-swarm* can be viewed as a way of maintaining better points found in the search space so far [73]. In each generation, the current position is evaluated by the objective function. If it is fitter (better fitness value) than its previous position, the coordinates are stored in the personal best position vector. The current position for current generation is obtained by adding the velocity vector to the current position of previous generation and the algorithm operates by adjusting the velocity which can effectively be seen as a step size [61].

The particles' position and velocity update of original PSO algorithm can be described as below:

$$V_i^d \leftarrow V_i^d + c_1 * rand1_i^d * (pbest_i^d - X_i^d) + c_2 * rand2_i^d * (gbest^d - X_i^d) \quad (2-4-1)$$

$$X_i^d \leftarrow X_i^d + V_i^d \quad (2-4-2$$

Where $rand1_i^d$ and $rand2_i^d$ are two random numbers in the range [0, 1]. $\mathbf{X}_i = (X_i^1, X_i^2, ..., X_i^D)$ represents the position of the *ith* particle; $\mathbf{pbest}_i = (pbest_i^1, pbest_i^2, ..., pbest_i^D)$ represents the best previous position (the position giving the best fitness value) of the *ith* particle; $\mathbf{gbest} = (gbest^1, gbest^2, ..., gbest^D)$ represents the best previous position of the population; $\mathbf{V}_i = (V_i^1, V_i^2, ..., V_i^D)$ represents the rate of the position change (velocity) for particle *i*. c_1 and c_2 are the acceleration constants, which represent the magnitude of stochastic forces that pull in the direction of personal best *pbest* and global best *gbest* positions. The update of velocity (Eqn. 2-4-1) consists of three parts: momentum part, cognitive part and social part. The momentum part prevents the abrupt change of velocity and improves the global search ability of the particle. The cognitive part represents how the particles learn from their own flying experience while the "social" part represents how the particles learn from the group's flying experience. The process for implementing the original PSO is shown in Table 2-4-1.

Table 2-4-1 The steps of the original PSO

Step 1	Initialize the population of particles with random positions and velocities on the *D* dimensional search space
Step 2	Evaluate the particles using the defined objective function and initialize *pbest* and *gbest*.
Step 3	For *i=1* to *NP* (population size)
Step 4	Update the particle's velocity using Eqn. (2-4-1)
Step 5	Update the particle's position using Eqn. (2-4-1)
Step 6	Set the variable within the bounds if it exceeds the bounds.

Step 7	Evaluate the new particle's position.
Step 8	Update the *pbest* and *gbest* if the new position is fitter
Step 9	Endfor
Step 10	Stop if the criterion is met, otherwise go to step 2.

The performance of PSO is dependent on the parameters selected. The following subsections present some brief discussion on parameters used in PSO.

- Population size

  Similar to other EAs, PSO is a population based search algorithm. Size of the population is the prime parameter to ensure good performance of PSO. The population size is often set empirically on the basis of the dimensionality and perceived difficulty of a problem. Values in the range of 20-50 are quite common [61] [67]

- Maximum velocity

  It is important to impose a limitation ***Vmax*** on the velocity for the convergence of the PSO algorithm. ***Vmax*** chops off the oscillations of the particle, so that some hopefully satisfactory compromise can be reached. The optimal value of ***Vmax*** is problem-specific and it is difficult to select a suitable ***Vmax*** if no pre-knowledge is available. Large ***Vmax*** allows particles to have the potential to fly far past good solution areas while a small ***Vmax*** causes particles potentially to be trapped into local optima.

- Inertia weight

  To reduce the importance of the maximum velocity, the modification of the PSO's velocity update equation was proposed by Shi and Eberhart [62]. A new parameter is introduced into the original velocity update equation which can be represented as:

$$V_i^d \leftarrow \omega * V_i^d + c_1 * rand1_i^d * (pbest_i^d - X_i^d) + c_2 * rand2_i^d * (gbest^d - X_i^d) \qquad (2-4-3)$$

  where the new parameter ω is termed as the inertia weight. The inertia weight is able to balance between the global and local search capabilities. A large inertia weight facilitates global, while a small inertia weight facilitates

local search [63]. The inertia weight was further studied in [64] and a linearly decreasing inertia weight was presented by Shi and Eberhart [65] in 1998. A large inertia weight is used in the initial search period to ensure that the particles have better global search ability and avoid falling into local minima. With the increase of function evaluations, a small inertia weight at the end of search process is used to refine the best solution found so far. Another effective strategy is to use an inertia weight with a random component, rather than time-decreasing [66].

- Acceleration constant and constriction coefficients

  The acceleration constant represents the mean stiffness of the springs pulling a particle, which is an important parameter for velocity update. In early PSO research, the value of acceleration constant is selected as $c_1 = c_2 = 2$. In [66], Kennedy revealed that the trajectories of nonstochastic one-dimensional particles contained interesting regularities when c_1+c_2 is between 0 and 4. To control the convergence of the particle and prevent explosion, Clerc [68], [69] introduced a constriction coefficient. The velocity update of incorporating the constriction coefficient is shown in the following equations:

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

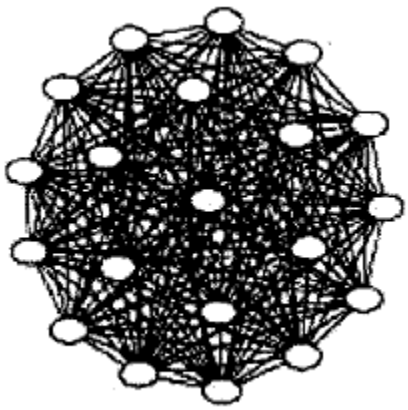
  with $\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}$ where $\varphi = c_1 + c_2,\ \varphi > 4$ $\qquad (2-4-5)$

  φ is commonly set to 4.1 and the constant multiplier χ is approximately 0.729. If we replace χ with the inertia weight ω and make c_1 and c_2 meet the condition $\varphi = c_1 + c_2$, $\varphi > 4$. Note that the PSO algorithm with the constriction factor can be considered as a special case of the PSO with inertia weight.

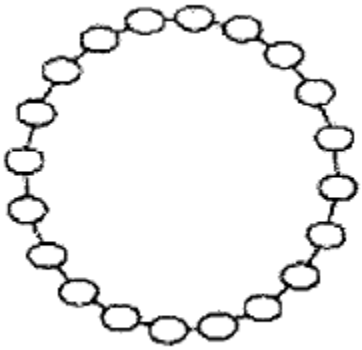
Depending on the method of selecting leading particles, PSO algorithms can be divided into global and local versions. In the global version of PSO, the trajectory

of each particle's search is influenced by the best member of the entire population. 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 global version of PSO is also known as the *gbest* topology while the *lbest* topology is the local version of PSO. Compared to *gbest* topology, *lbest* topology generally converges more slowly. However, it is also less vulnerable to the attraction of local optima [70], [71]. Different from *gbes*t topology, *lbest* topology has many different structures. Some of the commonly used structures are listed as below [71] and these topology structures taken from [71] are also shown in Fig. 2-4-1 [71].

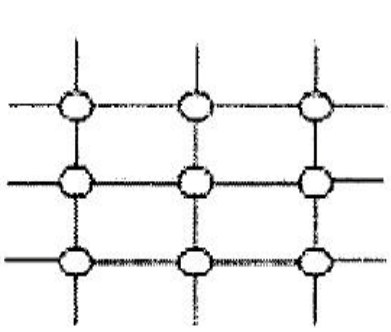
- All: *gbest* topology which treats the entire population as the individual's neighborhood.
- Ring: where adjacent members of the population array comprises the neighborhood.
- von Neumann: neighbors above, below, and on each side on a two-dimensional lattice are connected.
- Star: one central node influences, and is influenced by, all other members of the population.
- Pyramid: a three-dimensional wire-frame triangle.



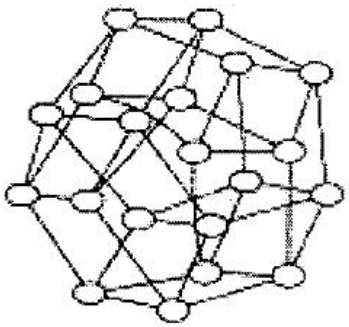
(a) All (*gbest*)



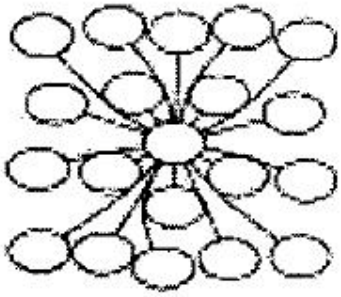
(b) Ring



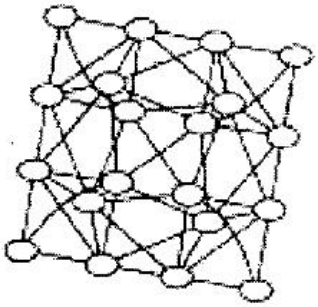
(c) von Neumann (flattened out)



(d) von Neumann (wrapped)



(c) Star structure



(d) Pyramid structure

Fig. 2-4-1 Commonly used topology structures of PSO [71]

The topology structures mentioned above are static. In 1999, Suganthan [72] introduced a dynamically adjusted neighborhood which is adjusted according to the distances between particles and a predefined criterion. The local version PSO is dynamically transformed into global version along the running process. It is effective to begin the search with an *lbest* ring lattice and slowly increase the neighborhood size, as lbest topology seems better for exploring the search space while *gbest* topology converges faster. Peram et al. [73] developed a new PSO topology using a weighted Euclidean distance in identifying the interaction partner for a particle. For each particle, the algorithm indentifies its neighborhood based on the highest “fitness distance ratio” (FDR). FDR is the ratio of the difference between the target particle’s fitness and neighbor’s fitness to the distance between them in the search space on that dimension [61]. In 2005, Liang and Suganthan [74] randomly grouped subpopulation of size *n* and occasionally randomized all the

connections. A parameter-free particle swarm system called TRIBES is designed by Clerc [75]. Mendes and Kennedy [76] proposed a fully informed particle swarm (FIPS) 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.

As PSO algorithm is simple in concept, easy to implement and computationally efficient, it has been applied in many areas. Some of the important applications are listed in Table 2-4-2. More details can be found in [77].

Table 2-4-2. Applications of PSO

Areas	Applications	References
Antennas	The optimal control and design of phrased	[78]-[80]
	Broadband antenna design and modelling	[81], [82]
Biomedical	Human tremor analysis for the diagnosis of	[83]
	Biometrics	[84]
Control	PI and PID controllers	[85]
	Power plants and systems control	[86], [87]
Design	Filter design	[88]
	Antenna design	[89]
Distribution networks	Network reconfiguration and expansion	[90]
	Distributed generation	[91]
Electronics and electromagnetics	FPGA-based temperature control	[92]
	Electromagnetic shape design	[93]
Image and video	Inversion of ocean color reflectance	[94]

	Image registration	[95]
Neural networks	Neural networks control for nonlinear	[96]
	Design of radial basis function networks	[97]
Power systems and plants	Power system performance optimization	[98]
	Hybrid power generation systems	[99]
Scheduling	Flow shop scheduling	[100]
	Scheduling in battery energy storage systems	[101]

2.5 Multi-modal Optimization

Multi-modal functions refer to the problems that have multiple global/local optima. The main objective to use EAs for multi-modal optimization is to locate and maintain multiple solutions in the search space. A multi-modal function with a number of maxima and minima with unequal peak heights is shown in Fig. 2-5-1.

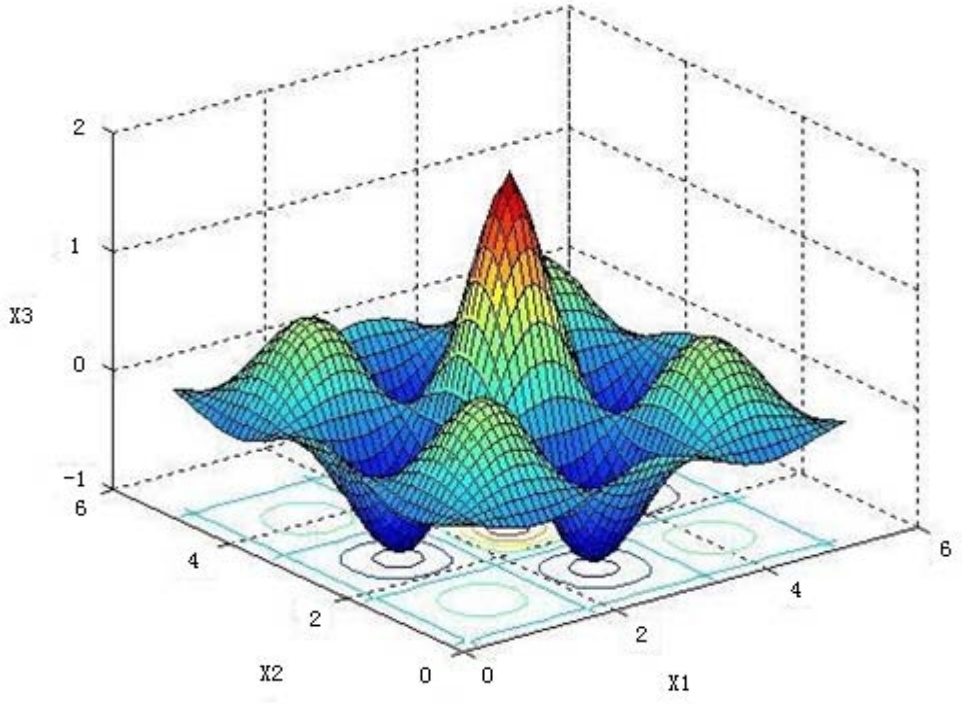


Fig. 2-5-1 A multi-modal function

Multi-modal function optimization has been studied over decades in the research region of Evolutionary Computation, as some practical optimization problems often require finding all global optima (or multiple optima including global and some local optima) instead of just a single optimum. When optimizing complex problem with many local optima, EAs suffer from either premature or slow convergence. To overcome the difficulties imposed by multiple local optima, hybrid EAs incorporating local search are developed. Various concepts have been proposed to realize multi-modal function optimization, such as niching [102], crowding population [103] techniques etc. Basically to implement multi-modal optimization in Evolutionary Computation, ideas of niching technique [102] have been incorporated which helps to maintain the population diversity.

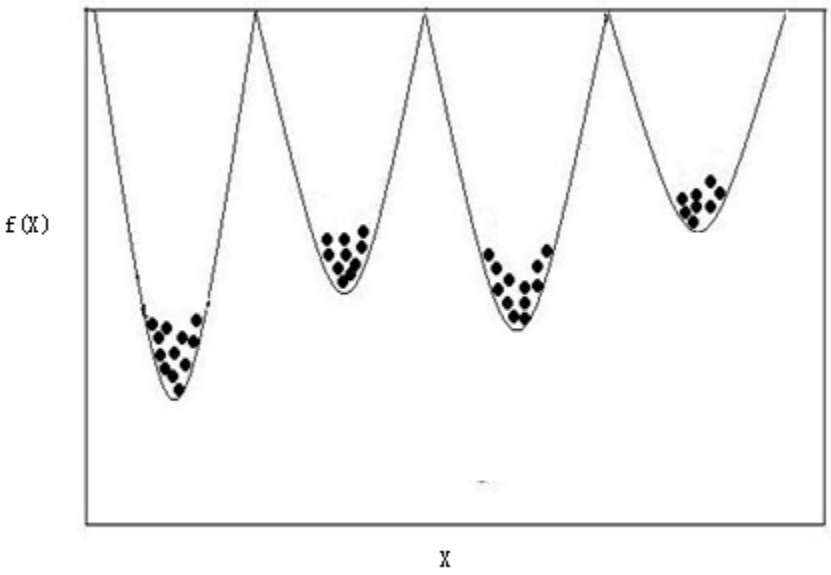


Fig. 2-5-2 Multiple optima captured by sub-populations

A good multi-modal algorithm would not dedicate all of the population individual to the same search space region, rather it would use only those individuals required to optimize the identified optima of the search space, while other individuals are exploring other areas, i.e., multi-modal techniques force molecular diversity [5]. To accomplish this goal, multi-modal techniques known as the niching are needed to

measure differences between explored regions, and then structure the search space. Niching is a generic term which is adopted from biological science. It refers to the method of finding and preserving multiple stable groups of solutions, so as to prevent the population convergence to a single solution. A niching method can be incorporated into a standard EA to promote and maintain formation of multiple stable subpopulations within a single population, with an aim to locate multiple global optimal or suboptimal solutions. Fig. 2-5-2 shows how the multiple optima are captured by the sub-populations of a niching algorithm.

Niching algorithms can be categorized based on the way that niches are located. Three categories are identified [102]:

- *Sequential niching* (or *Temporal niching*) locates niches over time. Find multiple niches iteratively/temporarily.
- *Parallel niching* (or *Spatial niching*) locates all niches in parallel. Find and maintain multiple niches simultaneously in a single population.
- *Quasi-sequential niching* locates niches sequentially and the search for new niches continues while the found niches are maintained in parallel.

Parallel niching (PN) method follows the *parallel hillclimbing* [104] approach which is similar to the binary search technique. While handling the real world vector, the hillclimbing method starts with a large step size and each population element continues to hillclimb until they can no longer improve. Then we divide the step size into half of the previous and carry out the hillclimbing approach. This process continues till it reaches a predefined smallest possible step size. The sequential niching (SN) [104] method is practically an extension of iterating GA's that maintains the best solution of each run off-line. To avoid converging to particular optima more than once whenever SN found an optimum, it depresses the search space at all points within some radius known as niche radius [104] of the solution. A further categorization of niching algorithms can be made according to speciation behaviour [102]:

- Sympatric speciation: individuals from the population form subpopulation of those that exist in the same search space, but evolve to exploit different resources.
- Allopatric speciation: there is no inter population communication and subpopulation can be developed only through perturbation from available genetic information (mutation).
- Parapatric speciation: here the inter population communication may not be encouraged, where development of new species is evolved as a result of segregated species sharing a common boundary.

In following subsections, various niching methods available in the literature are briefly presented.

- Crowding

  The crowding method introduced by De Jong in 1975 [103] allows competition for limited resources among similar individuals in the population. Hence, the competition is within each niche. This approach will maintain the diversity of the whole population. To determine similarity in crowding method, a distance metric should be utilized, either genotypic or phenotypic. In genotypic distance sharing, the distance function 'd' is simply the Hamming distance between two strings (where strings are binary coded). In phenotypic distance sharing, the distance function 'd' is defined using problem-specific knowledge of the phenotype. The most common choice for a phenotypic distance function is Euclidian distance. In crowding, the number of individuals assembling about a peak is largely determined by the size of that peak's basin of attraction under crossover. A parameter *CF* called crowding factor is used to control the size of the sample. CF is generally set to 2 or 3. Because of this low CF values, replacement errors are the main problem for crowding [105], [106].
- Deterministic Crowding

  Deterministic crowding [106],[107],[108] is a modification of the technique first introduced by De Jong [103] to maintain diverse population,

eliminate parameter requirements, reduce replacement error and restore selection pressure. The algorithm first chooses two parents from the population randomly and performs crossover and mutation to generate two offspring. Then the children replace the nearest parent if they are of greater fitness.

- Probabilistic Crowding

  To modify this deterministic nature of the algorithm and thus provide a restorative pressure in such cases, Mengshoel [109] proposed probabilistic crowding. In this case, a probabilistic replacement rule was proposed that permitted individuals with higher fitness to win over individuals with lower fitness in proportion to their fitness. This allowed a restorative pressure and prevented the loss of niches of lower fitness. Basically, this algorithm is deterministic crowding with a probabilistic replacement operator. In this probabilistic crowding, two similar individuals X and Y compete in a probabilistic tournament where the probability of X winning the tournament can be summarized as:

$$p(X) = \frac{f(X)}{f(X) + f(Y)} \qquad (2-5-1)$$

  where f is the fitness function used to determine the fitness of the individuals.

- Restricted Tournament Selection (RTS)

  Restricted Tournament Selection, introduced by Harick [110], [111] is a modified tournament selection for multi-modal optimization. RTS scheme allows the GAs to choose which individual will be replaced. As in deterministic crowding, RTS randomly selects two parents from the population and yields two offspring by applying crossover and mutation operators. After that, for each offspring, the algorithm chooses a random sample of '*w*' (window size analogous to CF in Crowding) individuals from the population and determines which one is the nearest to the offspring, by applying the similarity distance measure either Euclidian (for real variables) or Hamming (for binary coded variables). The nearest member within the

'*w*' individuals will compete with the offspring to determine who has higher fitness. If the offspring wins, it is allowed to enter the population by replacing its opponent. This type of tournament will restrict an element of the population from competing with others that are too dissimilar from it.

- Fitness Sharing

  The fitness sharing was introduced by Holland [3] and extended by Goldberg and Richardson [16]. The concept is to divide the population into different subgroups according to the similarity of the individuals. An individual must share its information with other individuals within the same niche. The shared fitness for i^{th} individual can be represented by using following equation:

$$f_{shared}(i) = \frac{f_{original}(i)}{\sum_{j=1}^{N} sh(d_{ij})} \qquad (2-5-2)$$

  where the sharing function is calculated as $sh(d_{ij}) = \begin{cases} 1-(\frac{d_{ij}}{\sigma_{share}})^{\alpha}, & \text{if } d_{ij} < \sigma_{share} \\ 0, & \text{otherwise} \end{cases}$. d_{ij} is the distance between individuals i and j, σ_{share} is the sharing radius, N is the population size and α is a constant called sharing level.

- Clearing

  Clearing [19], [112] is another widely used niching method. Different from fitness sharing, clearing removes the bad individuals and keeps only the best individual (or a few top individuals) within each niche. The algorithm first sorts the population in descending order according to the fitness values. Then it picks one individual at a time from the top and removes all the individuals with worse fitness than the selected one within the specified clearing radius. This step will be repeated until all the individuals in the

population are either selected or removed. Clearing eliminates similar individuals and maintains the diversity among the selected individuals.

- Speciation

  The idea of speciation is commonly used in multi-modal optimization [9], [18], [19]. This method also depends on a radius parameter r_s , which measures the Euclidean distance from the center of a species to its boundary. The center of a species is called the species seed. Each of the species is built around the dominating species' seed. All individuals that fall within the radius of the species seed are identified as the same species. In this way, the whole population is classified into different groups according to their similarity.

- Clustering

  In [113], [114] Yin proposed a niching scheme using a clustering methodology incorporated into niching methods to help form the niches and avoid the need for estimation of σ_{share} needed in sharing technique. The fitness is calculated based on the distance d_{ic} between the individual and its niche centroid. The formation of the niches is based on the adaptive Macqueen's K-mean algorithm. The algorithm begins with a fixed number (*k*) of seed points taken as the best *k* individuals. Using a minimum allowable distance $d_{\min}$ between niche centroids, a few clusters are formed from the seed points. The remaining population members are then added to these existing clusters, or are used to form new clusters based on $d_{\min}$ and $d_{\max}$ .These computations are performed in each generation.

  The final fitness of an individual is calculated by the relation:

$$F_i = \frac{f_i}{n_c\left(1-\left(d_{ic}/2d_{\max}\right)^{\alpha}\right)} \qquad (2-5-3)$$

where n_c is the number of individuals in the niche containing the individual i, $d_{\max}$ is the maximum distance allowed between an individual and its niche centroid, and α is a constant.

The complexities of the above mentioned niching methods are listed in Table 2-5-1.

Table 2-5-1 Complexities of the niching algorithms

	Algorithm	**Complexity**
1	Clearing	O(c*N*)
2	Deterministic Crowding	O(*N*)
3	Probabilistic Crowding	O(*N*)
4	Sharing	$O(N^2)$
5	RTS	$O(N \times w)$
6	Clustering	$O(C \times N_c \times N)$

2.6 Multi-objective Optimization

Many real-world search and optimization problems naturally involve more than one objective that conflicting with each other. As there is no single solution for these problems, our aim is to find Pareto optimal trade-off solutions representing the best possible compromises among all objectives. This means that any improvement in a Pareto optimal point in one objective must lead to deterioration in at least one other objective. Fig. 2-6-1 [1] shows a multi-objective problem involved in buying an automobile car. Cost is considered as one objective which should be minimized while the comfort level is another objective in decision-making process which should be maximized. Point 1 and 2 are known as the two

extreme cases for this problem. Any point (such as A, B, C) between these two represents certain trade-off solutions among the two objectives.

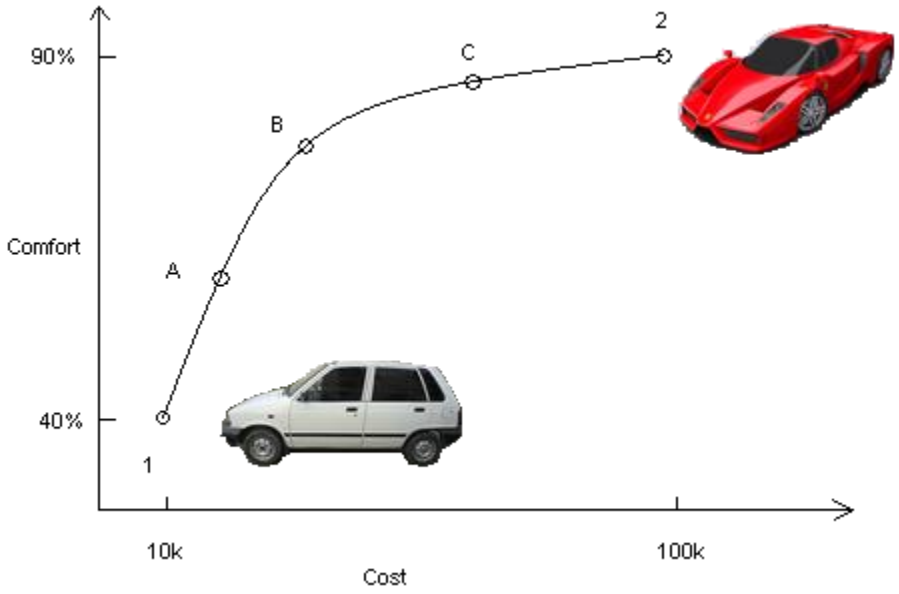


Fig. 2-6-1 Illustration of car-buying decision-making problem

A multi-objective optimization problems can be formally formulated using the following mathematical expression.

$$\text{Min imize:} \quad \mathbf{y} = f(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})) \qquad (2-6-1)$$

$$\text{subject to:} \quad g_j(\mathbf{x}) \le 0,\ j = 1, \ldots, J$$

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

$$\mathbf{x} \in [\mathbf{Xmin}, \mathbf{Xmax}]$$

$\mathbf{x}$ is the decision vector, and $\mathbf{y}$ is the objective vector. Different from the single objective optimization, there are two spaces to be considered. One is the decision space, which denoted as $\mathbf{X}$; while the other one is called the objective space, denoted as $\mathbf{Y}$.

To solve a multi-objective optimization problem, the domination concept is commonly used. A solution $\boldsymbol{x}_1$ is said to dominate another solution $\boldsymbol{x}_2$ if both conditions specified below are satisfied [1]:

- The solution $\boldsymbol{x}_1$ is no worse than $\boldsymbol{x}_2$ in all objectives.
- The solution $\boldsymbol{x}_1$ is strictly better than $\boldsymbol{x}_2$ in at least one objective.

If any of the above conditions is violated, the solution $\boldsymbol{x}_1$ does not dominate the solution $\boldsymbol{x}_2$. A non-dominated set **P** is a number of solutions that are not dominated by any of other solutions [1]. When the set **P** is the entire search space, it is called the Pareto-optimal set. Fig. 2-6-2 that taken from [115] illustrates the definition of Pareto-optimal set/front (the front which includes points p, q and r).

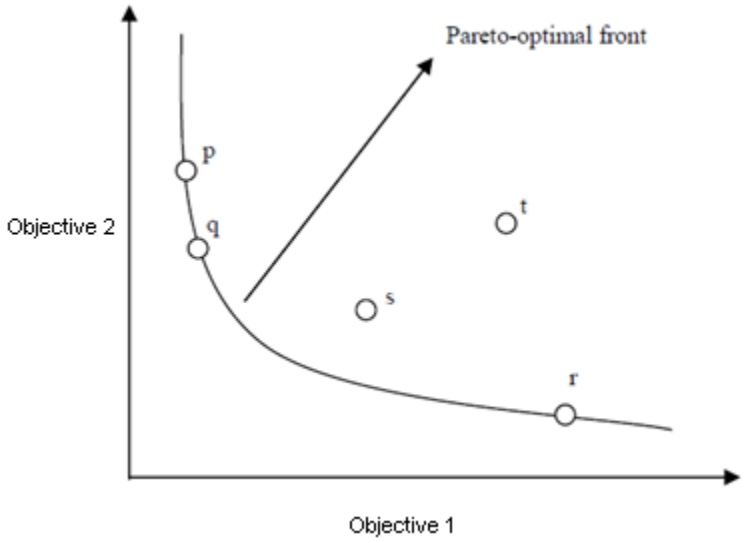


Fig. 2-6-2 Illustration of Pareto optimal front [115]

The target 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:

- Convergence to the Pareto-optimal set.
- Diversity of solutions in the Pareto-optimal set.

In literature, many classical methods are used to handle multi-objective optimization problems. The classical method is only able to find one single optimal solution in one simulation. Due to various advantages, interest has shifted to EAs in recent years. Since an EA works with a population of solutions, a population of Pareto-optimal solutions can be captured in one single simulation run. It eliminates repetitive use of single-objective optimization method for finding one different Pareto-optimal solution in each run [1]. It is able to preserve a diverse set of multiple non-dominated solutions through the evolvement of the population.

The first real implementation of a multi-objective evolutionary algorithm can be traced back to early 1980s. In 1984, David Schaffer [116] modified the simple tripartite genetic algorithm with selection, crossover and mutation by performing independent selection cycles according to each objective [1]. The main advantage of this algorithm is its simplicity. However, when the optimal front is concave, certain points will not be found through the optimization procedure [115]. Kursawe [117] proposed a vector-optimized evolution strategy (VOES) for multi-objective optimization in 1990. The algorithm is modified from a basic single objective self-adaptive evolution strategy. It performs a domination check to retain non-dominated solutions and certain niching mechanism to eliminate crowded solutions [1].

Among all the multi-objective evolutionary algorithms proposed, the elitist multi-objective genetic algorithm (NSGA-II) [128], [129] is one of the most famous and commonly used algorithms. NSGAII uses both elite-preservation strategy and explicit diversity-preserving mechanisms. Fig 2-6-3 which is taken from [137] shows the procedure of NSGAII algorithm. $\mathbf{P_t}$ is the parent population, while $\mathbf{Q_t}$ is the offspring population produced by $\mathbf{P_t}$. In, NSGAII, the two populations are combined together to form a combined population $\mathbf{R_t}$ of size 2*NP* (population size). Then the non-dominated sorting is performed on the combined populations to assign the front numbers (F_1, F_2, F_3 and so on. Note that it is only an example to illustrate the sorting and selection process and not necessarily means only three fronts are considered for every problem). Lastly, the crowding distance is applied

to select the *NP* number of populations for next generations. Due to use of the crowding comparison procedure during the population reduction phase, the diversity of the population is maintained which makes NSGAII effective in solving multi-objective optimization problems.

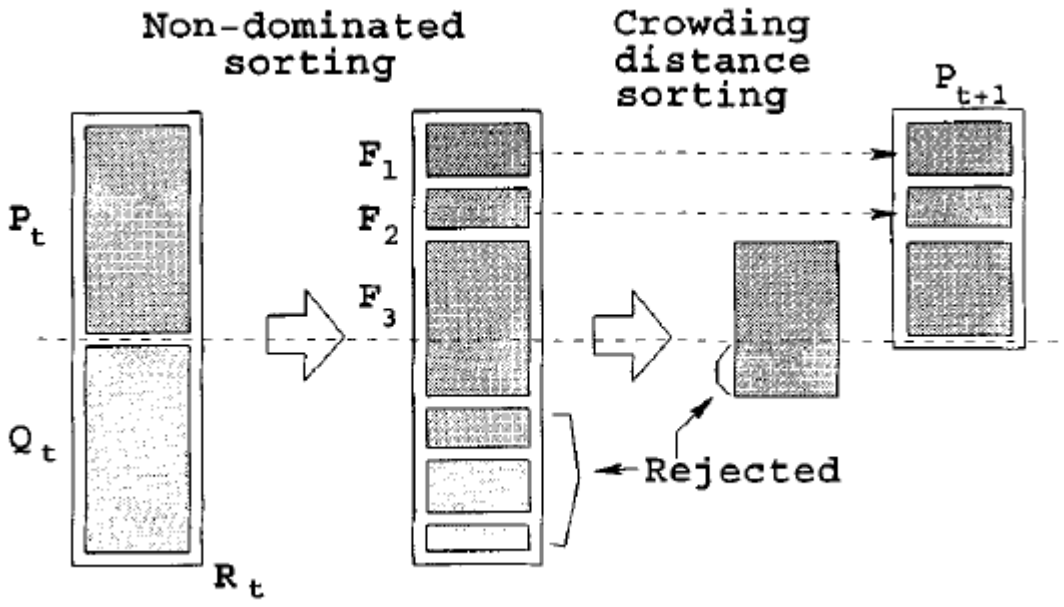


Fig. 2-6-3 The procedure of NSGAII [137]

More recently, Li and Zhang [142] introduced a multi-objective evolutionary algorithm based on decomposition (MOEA/D). The algorithm decomposes a multi-objective problem into a set of scalar objective optimization problems. Unlike traditional scalarization methods, MOEA/D solves the set of sub-problems simultaneously in a single run. The experimental results suggested that MOEA/D is very promising in solving multi-objective optimization problems.

There are many other multi-objective evolutionary algorithms proposed in the past few decades. C. M. Fonseca and P. J. Fleming proposed a rank-based fitness assignment method which allowed direct intervention of an external decision maker for multiple objective genetic algorithm (MOGA) [118], [119]. J. Horn et al. presented a niched Pareto genetic algorithm (NAGA) for multi-objective optimization which applied a niching pressure to spread its population out along

the Pareto optimal tradeoff surface in [120]. Strength Pareto evolutionary algorithm (SPEA) was introduced by E. Zitzler and L. Thiele in [121], [122]. In this work, the authors combined several features of previous multiobjective evolutionary algorithms in a unique manner. In [123] [124], K. Deb introduced a Non-dominated sorting genetic algorithm (NSGA) which can maintain stable and uniform reproductive potential across non-dominated individuals. And later it was improved and modified to a fast and elitist multi-objective genetic algorithm, NSGA-II [137]. P. Hajela et al. constructed a weight-based genetic algorithm (WBGA) and applied it in multi-objective optimal designs of structural systems with a mix of continuous, integer and discrete design variables [125], [126]. In [127], a spatial predator-prey evolution strategy (PPES) using the predator-prey model from ecology to approximate the shape of the Pareto-optimal set of multi-objective optimization problems was presented. Hajime KITA et al. [130] used the concepts of the entropy and the temperature in the selection operation and constructed the so-called thermodyanamical genetic algorithm (TDGM) for multi-objective optimization through combining the Pareto-based ranking technique. Distance based Pareto genetic algorithm (DBPGA) was introduced by A. Osyczka and S. Kundu in [131]. V. L. Huang et al. integrated a Pareto dominance concept into comprehensive learning particle swarm optimizer which was originally designed for single objective optimization problems to solve multi-objective optimization problems (MOCLPSO) [132]. K. Zielinski and R. Laur suggested an adaptive parameter setting method for a multi-objective particle swarm optimization algorithm (MO_PSO) [133]. Multi-objective Optimization based on self-adaptive differential evolution algorithm (MOSaDE) which is also generalized from the single-objective optimization algorithm was proposed in [135]. L-Y. Tseng and C. Chen proposed multiple trajectory search (MTS) which used multiple agents to search the solution space concurrently for multi-objective optimization [136] .

Chapter 3

Differential Evolution Based Niching Algorithms for Multi-modal Optimization

This chapter presents a short survery on some existing DE based niching algorithms for multi-modal optimization. The proposed DE based niching algorithms are also presented in this chapter.

3.1 Existing DE Based Niching Algorithms

As one of the most powerful evolutionary algorithms, DE has been incorporated with various niching techniques to solve multi-modal problems. In this section, some of the popular approaches are introduced.

3.1.1 Sharing DE and Crowding DE

One of early DE niching work was proposed by Thomsen [145], who imported the fitness sharing concept to DE. The sharing DE (SHDE) modified the fitness for each individual and the bad individuals are purged. The drawbacks of SHDE are the requirement to define the niching parameter and high computational complexity.

Thomsen [145] also introduced a crowding-based DE (CDE) in the same work to overcome the problems of SHDE. In CDE, the competition is between the offspring and its most similar (measured by parameter-space Euclidean distance, the smaller the more similar) member of the whole population to maintain the diversity of the population. Thomsen shows that CDE outperforms SHDE in locating and maintaining multiple optima. Although the idea of CDE is simple, it has been proven to be one of the most effective DE-based niching algorithms. The steps of CDE are summarized in Table 3-1-1.

Table 3-1-1 The Crowding DE (CDE)

Step 1	Randomly generate *NP* initial trial solutions.
Step 2	For i = 1 to *NP* Produce an offspring u_i using the standard DE. Calculate the Euclidean distance of u_i to the other individuals in the DE population. Compare the fitness of u_i with the most similar individual (measured in Euclidean distance) and replace it if u_i has a better fitness value. Endfor
Step 3	Stop if a termination criterion is satisfied. Otherwise go to Step 2.

3.1.2 Speciation-based DE

Species-based DE (SDE) [146] is another commonly used DE niching algorithm. The concept is the same as speciation described in Chapter 2.5. Different niches are formed around the species seed. The mutation is carried out within each species. The details of SDE are presented in Table 3-1-2.

Table 3-1-2. The Species-based DE (SDE)

Step 1	Randomly generate *NP* number of initial trial solutions.
Step 2	Sort all individuals in descending order of their fitness values.
Step 3	Determine the species seeds for the current population: The most-fit individual will be set as the first species seed. Then all individuals are checked in turn from most-fit to least-fit against the species seeds found so far. If an individual does not fall in the radius of any seeds, it will be indentified as another species seed.
Step 4	For each species, execute a global DE variant: 4.1 If a species has less than *m* individuals, then randomly generate new individuals within the radius of the species seed.

	4.2	If the child's fitness is the same as its species seed, replace this child with a randomly generated new individual.
Step 5	Keep only the *NP* fitter individuals from the combined population.	
Step 6	Stop if a termination criterion is satisfied. Otherwise go to Step 4.	

3.1.3 Multi-population DE

A multipopulation DE (MDE) was proposed by Zaharie [147] in 2004 to handle multi-modal problems. The algorithm initially divides the search space into a number of non-overlapping subdomains, from which the subpopulations are initialized. For each of the subpopulations, the DE algorithm is run independently. In the search process, the subpopulations are not restricted to the subdomains used in the initialization and the number of the subdomains increases periodically.

Zaharie [148] also added crowding to the multipopulation concept to produce a multipopulation crowding DE (MCDE). The subpopulations are initialized the same way as in MDE, but for each subpopulation, the CDE is applied.

3.1.4 Other algorithms

Besides the above mentioned DE niching algorithms, various approaches for hybridizing DE with local search methods to increase the convergence speed have been proposed in literature. Guo and Wang [149] applied the local search to the best located point after a preset number of iterations. After that the worst half of the population is reinitialized and the search process is repeated. Chiou and Wang [150] introduced an acceleration operator to DE which speeds up the local search if no improvement is observed between the generations. The algorithm also uses a migration operator to generate a new population, when the diversity drops too low.

Tirronen [151], [152] proposed several memetic differential evolution techniques. The methods monitor the diversity of the population. If the population diversity decreases below certain threshold, the algorithms periodically activate different local search methods.

3.2 Dynamic Grouping Crowding Differential Evolution with an Ensemble of Parameters for Multi-modal Optimization

In this section, dynamic grouping crowding differential evolution (DGCDE) with an ensemble of parameters is introduced. In this algorithm, the population is dynamically regrouped into 3 equal subpopulations every few generations. Each of the subpopulations is assigned a set of parameters. The algorithm is tested on 14 classical benchmark multi-modal optimization problems and compared with the crowding differential evolution (Crowding DE) in literature. As shown in the experimental results, the proposed algorithm outperforms the Crowding DE with all three different parameter settings on the benchmark problems.

3.2.1 DGCDE with ensemble of parameters

Although CDE has been shown to be effective in solving multi-modal optimization problems, the original CDE only uses one single set of DE parameters. As we know, the performance of DE is closely related to the control parameters. According to “No free lunch” theorem [153], it is unlikely to find one parameter that can outperform all other parameters, since different parameter is suitable for different problems. Different search stage also requires different parameter in order to reach the best performance. Motivated by this observation, a dynamic grouping crowding differential evolution with ensemble of parameter is proposed to handle multi-modal problems.

In DGCDE, the algorithm starts with a single population (same as CDE). This population will be randomly divided into three subpopulations, while each of the subpopulation is assigned with a set of control parameters. The subpopulation runs as a separated DE for *m* generations. After every *m* generations, the subpopulation is re-combined and randomly regrouped again. In this way, the algorithm is able to make use of three different parameter sets and keep the offspring that generated by the most suitable parameters. The flowchart and algorithm are shown in Fig. 3-2-1

and Table 3-2-1 respectively. The three sets of control parameters for three different subpopulations are chosen as F_1 =0.9, CR_1 =0.3; F_2 =0.5, CR_2 =0.1; F_2 =0.5, CR_2 =0.3. Note that these parameters are commonly used in DE algorithms which have been shown to be effective in solving different types of problems [23]. Using these parameters is likely to generate an algorithm that is suitable for a large variety of problems.

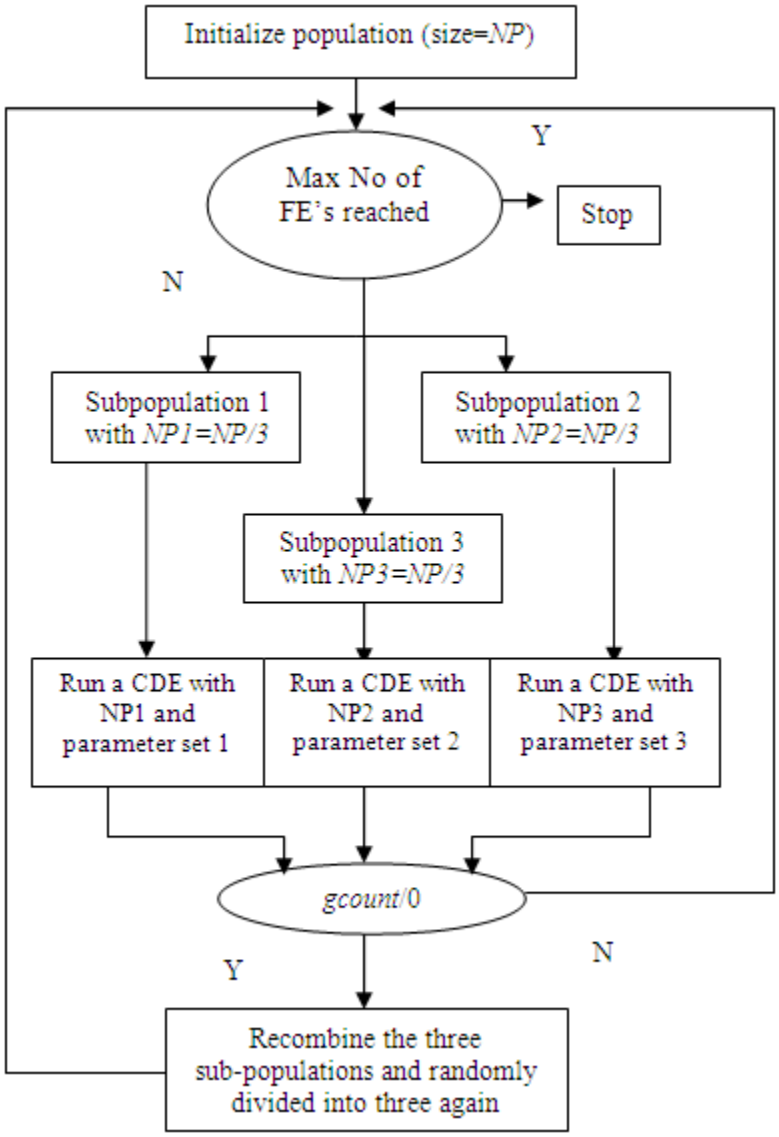


Fig. 3-2-1 The flowchart of DGCDE

Table 3-2-1 DGCDE algorithm

Step 1	Randomly initialize *NP* (population size) solutions. Set generation counter *gcount=1*.
Step 2	Randomly divide the *NP* solution into three groups (*NP/3* members for each group). Each group is assigned with a different set of control parameter (*F*, *CR*).

Step 3	If *gcount*/m=0
	Combine the subgroups and Goes to step2 to regroup the population.
	Else
	Run each group as a separated CDE.
	Endif

Step 4	Stop if the termination criteria are met otherwise go to step 3.

3.2.2 Experiment preparation

To assess the niching ability of the proposed algorithm, some frequently used multi-modal optimization benchmark test functions with different characteristics are used. The algorithms are run till the termination criteria (maximum function evaluations) has been reached or all peaks are found. This type of the termination criterion is commonly used in evolutionary algorithms. The following four DE variants were used in the experiments. Note that DGCDE is designed for locating only the global peaks.

1. DGCDE: dynamic grouping Crowding DE with ensemble of parameters.
2. CDE1: Crowding DE with *F* set to 0.9, *CR* set to 0.3.
3. CDE2: Crowding DE with *F* set to 0.5, *CR* set to 0.1.
4. CDE3: Crowding DE with *F* set to 0.5, *CR* set to 0.3.

- Test Functions

  The test functions that used in this work are listed in Appendix A. These 14 test functions are AF1 (Appendix A test function F1) to AF14 (Appendix A

test function F1). They can be grouped into four categories according to their properties and complexities.

1. *1-D Deceptive Functions:* AF1-AF3 are linear functions. However, they are considered to be deceptive, as the existing local optima can mislead the population to move to local optima instead of global optima. AF3 is more difficult than the first two functions, as the three local peaks generate additional challenge for an optimizer. These three functions are used to test the ability of an algorithm to locate global optima as well as the local optima.
2. *1-D Multi-modal Functions:* AF4 has five evenly distributed peaks, while AF5 is similar to AF4 but with the five peaks in decreasing height. AF6 is also similar to AF4; the difference is the distribution of the peaks. AF7 is like AF6, but with five peaks decrease exponentially.
3. *2-D Multi-modal Functions:* AF8 has four global optima, while AF9 has two global peaks as well as two local peaks. AF10 has 25 evenly spaced peaks of unequal heights, with one being the global peak.
4. *More Challenging Functions:* AF11 has 18 global optima in pairs, with many local optima. AF12-AF14, the inverted Vincent functions have 6^n peaks, with no local peaks. n is the problem dimension.

- Experimental Setup

  For the simulations, Matlab 7.1 is used as the programming language. The configurations of the computer are Intel Pentium® 4 CPU, 2 Gb of memory. The population size is set 60 for all test function and test variants. The maximum number of function evaluations is set to 10000 for all the test function. Therefore, the maximum number of generation will be 167. Twenty independent runs are conducted for each of the algorithms.

- Performance Measures

  To assess the performance of different algorithms, a level of accuracy (typically in the range of [0, 1]) need to be specified. This parameter used to measure how close the obtained solutions to the known global peaks are. An optimum is considered to be found if there exists a solution in the

population within the tolerated distance to that optimum. The level of accuracy used in this experiment is summarized in Table 3-2-2.

Table 3-2-2 Level of accuracy used in the experiments

Test Function	Level of accuracy	Test Function	Level of accuracy	Test Function	Level of accuracy
AF1	0.05	AF6	0.000001	AF11	0.05
AF2	0.05	AF7	0.000001	AF12	0.0001
AF3	0.05	AF8	0.0005	AF13	0.001
AF4	0.000001	AF9	0.000001	AF14	0.001
AF5	0.000001	AF10	0.00001		

When doing the comparison, following two criteria are used:

1. Success Rate (The percentage of runs in which all global peaks are successfully located)
2. Average number of optima found

For more challenging functions, if not all peaks are found in any run, the success rate will become 0. All the performances are measured over twenty runs to compensate for the random effect.

3.2.3 Experimental results

- Success rate (SR)

The success rate for all the four DE variants are recorded and presented in Table 3-2-3. The rank of each algorithm is presented in the bracket and the total ranks for all the algorithms are listed in the last row of the table. As can be revealed from the results, the proposed DGCDE always ranks number one among all the tested algorithms, especially on problems 4, 6, 7, 10 and 12. Therefore, we can conclude that DGCDE with ensemble of parameters can outperform all the three CDE with different parameter sets on the tested benchmark problems. The superior performance is due to the mixing of different sets of control parameters. Different parameters perform

differently during different search stages. If one set of parameters is used through the whole search process, the algorithm may be stuck in certain search stage which slows down the convergence. DGCDE allows the populations to switch among different sets of control parameters and improve the performance. Note that the success rate obtained is highly related to the user defined parameter level of accuracy (demonstrated in the following section).

Table 3-2-3. The success rate

Test Function	CDE1	CDE2	CDE3	DGCDE
AF1	1 (1)	1 (1)	1 (1)	**1 (1)**
AF2	1 (1)	1 (1)	1 (1)	**1 (1)**
AF3	1 (1)	1 (1)	1 (1)	**1 (1)**
AF4	0.1 (4)	0.95 (2)	0.95 (2)	**1 (1)**
AF5	0.7 (4)	1 (1)	1 (1)	**1 (1)**
AF6	0.2 (2)	0.15 (4)	0.2 (2)	**0.9 (1)**
AF7	0.55 (3)	0.65 (2)	0.35 (4)	**0.85 (1)**
AF8	0 (1)	0 (1)	0 (1)	**0 (1)**
AF9	0 (1)	0 (1)	0 (1)	**0 (1)**
AF10	0.2 (4)	0.95 (2)	0.95 (2)	**1 (1)**
AF11	0 (1)	0 (1)	0 (1)	**0 (1)**
AF12	0.1 (4)	0.3 (3)	0.4 (2)	**0.65 (1)**
AF13	0 (1)	0 (1)	0 (1)	**0 (1)**
AF14	0 (1)	0 (1)	0 (1)	**0 (1)**
Total Rank	29	22	21	**14**

- Number of optima found

  In order to give a clear view of the advantage of DGCDE, the number of optima found for each test function is shown in Table 3-2-4. The mean value is highlighted in bold face. As we can see, the same conclusion can be draw from this table. The proposed DGCDE outperforms all the CDEs that with different parameter sets, especially for the more challenging 2D inverted Shubert function.

Table 3-2-4. The number of optima found

Test Function		CDE1	CDE2	CDE3	DGCDE
AF1	Mean	**1**	**1**	**1**	**1**
	Max	1	1	1	1
	Min	1	1	1	1

	Std	0	0	0	0
	Mean	**1**	**1**	**1**	**1**
AF2	Max	1	1	1	1
	Min	1	1	1	1
	Std	0	0	0	0
	Mean	**1**	**1**	**1**	**1**
AF3	Max	1	1	1	1
	Min	1	1	1	1
	Std	0	0	0	0
	Mean	**3.6**	**4.95**	**4.95**	**5**
AF4	Max	5	5	5	5
	Min	2	4	4	5
	Std	0.82	0.22	0.22	0
	Mean	**0.7**	**1**	**1**	**1**
AF5	Max	1	1	1	1
	Min	0	1	1	1
	Std	0.47	0	0	0
	Mean	**3.6**	**3.65**	**3.95**	**4.9**
AF6	Max	5	5	5	5
	Min	1	2	3	4
	Std	1.09	0.88	0.6	0.31
	Mean	**0.55**	**0.65**	**0.35**	**0.85**
AF7	Max	1	1	1	1
	Min	0	0	0	0
	Std	0.51	0.49	0.49	0.37
	Mean	**0.15**	**0.3**	**0.1**	**0.35**
AF8	Max	1	1	1	1
	Min	0	0	0	0
	Std	0.37	0.47	0.31	0.49
	Mean	**0**	**0.05**	**0**	**0.05**
AF9	Max	0	1	0	1
	Min	0	0	0	0
	Std	0	0.22	0	0.22
	Mean	**0.2**	**0.95**	**0.95**	**1**
AF10	Max	1	1	1	1
	Min	0	0	0	1
	Std	0.41	0.22	0.22	0
	Mean	**4.65**	**5.86**	**3.2**	**11.05**
AF11	Max	9	13	8	17
	Min	1	1	0	4
	Std	2.81	2.86	2.12	2.99
	Mean	**4.95**	**5.15**	**5.2**	**5.55**
AF12	Max	6	6	6	6
	Min	4	4	3	4

	Std	0.51	0.67	0.83	0.69
	Mean	**18.47**	**18.12**	**17.83**	**18.25**
AF13	Max	25	24	23	24
	Min	13	15	16	15
	Std	2.56	2.43	2.17	2.49
	Mean	**40.65**	**40.23**	**41.05**	**40.85**
AF14	Max	45	46	47	49
	Min	35	32	36	37
	Std	3.06	3.22	2.53	2.6207

- Locate both global and local optima

  A good niching algorithm should locate not only the global optima but also the local optima. To test the ability of locating both global and local optima, test function AF5 and AF10 are used. The results are shown in Table 3-2-5, which again indicate the DGCDE performs the best out of the four variants.

Table 3-2-5. Locating global and local optima

Test Function		CDE1	CDE2	CDE3	DGCDE
	Mean	**3.8**	**5**	**5**	**5**
	Max	5	5	5	5
AF5	Min	2	5	5	5
	Std	1.15	0	0	0
	SR	0.35	1	1	1
	Mean	**6.25**	**11.55**	**13.7**	**14.05**
	Max	12	15	17	17
AF10	Min	2	9	11	10
	Std	2.84	1.39	1.89	1.67
	SR	0	0	0	0

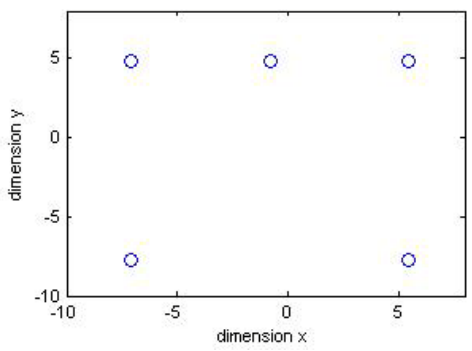
- Effect of varying population size (only DGCDE is considered)

  Population size is one of the most important parameters for the proposed niching algorithm. Different population size may generate different performance. To test how the performance varies with the population size, the 2D inverted Shubert function is used on the proposed DGCDE. For this experiment, 6 different populations (from 30 to 180) are examined. The results are shown in Table 3-2-6 and Fig. 3-2-2 plots the located optima position of median run for each population. As can be observed from the table, the average number of peaks located and the SR gradually increases as the population size increases. When the population size reaches 120 or

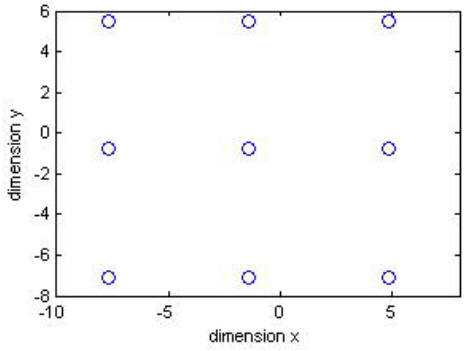
above, the algorithm can obtain 100% success rate for this test function.

Table 3-2-6. The effect of population size (AF11)

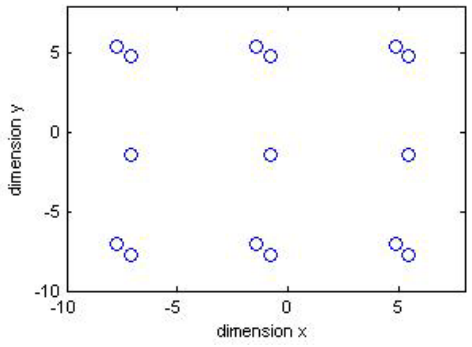
NP=30	Mean	**4.7**	NP=120	Mean	**18**
	Max	6		Max	18
	Min	4		Min	18
	Std	0.92		Std	0
	SR	0		SR	1
NP=60	Mean	**10.3**	NP=150	Mean	**18**
	Max	18		Max	18
	Min	8		Min	18
	Std	3.32		Std	0
	SR	0.15		SR	1
NP=90	Mean	**16.36**	NP=180	Mean	**18**
	Max	18		Max	18
	Min	11		Min	18
	Std	2.48		Std	0
	SR	0.65		SR	1



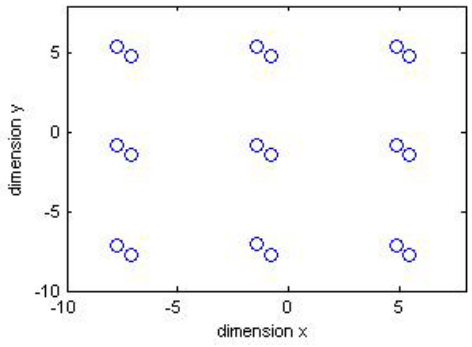
(a) *NP*=30



(b) *NP*=60



(a) *NP*=90



(b) *NP*=120, *NP*=150, *NP*=180

Fig. 3-2-2 The effect of population size plot

- Comparisons with other parameter sets

  DGCDE is further compared with other CDEs with different sets of parameters. The results are presented in Table 3-2-7. The parameter settings are listed in Table 3-2-7. The same conclusion can be drawn from these results.

Table 3-2-7. Comparisons with other parameter sets

Test Function	CDE4	CDE5	CDE6	CDE7	CDE8	DGCDE
F/CR	0.9/0.1	0.2/0.2	0.6/0.4	0.4/0.6	0.1/0.9	-
AF1	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	**1 (1)**
AF2	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	**1 (1)**
AF3	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	**1 (1)**
AF4	0.28 (4)	0.35 (3)	0 (6)	0.25 (5)	0.95 (2)	**1 (1)**
AF5	0.72 (3)	0.7 (4)	0.3 (6)	0.4 (5)	0.95 (2)	**1 (1)**
AF6	0.28 (4)	0.35 (3)	0.1 (6)	0.2 (5)	0.7 (2)	**0.9 (1)**
AF7	0.6 (4)	0.65 (3)	0.4 (6)	0.7 (2)	0.5 (5)	**0.85 (1)**
AF8	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	**0 (1)**
AF9	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	**0 (1)**
AF10	0.52 (4)	0.9 (2)	0.3 (5)	0.6 (3)	0.25 (6)	**1 (1)**
AF11	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	**0 (1)**
AF12	0.3 (4)	0.6 (2)	0.25 (5)	0.25 (5)	0.5 (3)	**0.65 (1)**
AF13	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	**0 (1)**
AF14	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	**0 (1)**
Total Rank	31	25	42	**33**	**32**	**14**

- Effect of varying level of accuracy

  As mentioned in previous section, the parameter *level of accuracy* can affect the results of the algorithm. To demonstrate this more clearly, the plot (Fig. 3-2-3) on level of accuracy vs success rate is generated based on test function 12 (Inverted Vincent function). From the plot we can see that the success rate is increased by decreasing the level of accuracy.

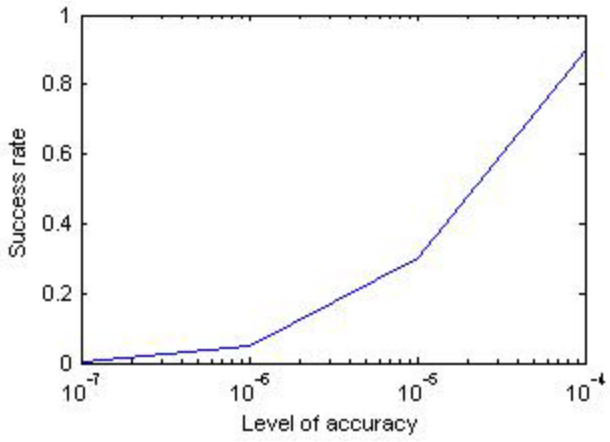


Fig. 3-2-3 The effect of varying level of accuracy

3.2.4 Conclusion

In this work, a dynamic grouping crowding differential evolution (DGCDE) with ensemble of parameter algorithm for multi-modal optimization is introduced to overcome the difficulty of choosing DE parameter set. The proposed algorithm is compared with the original Crowding DE with different parameter settings on a set of commonly used multi-modal benchmark test problems. As we can see from the result, the proposed algorithm outperforms the Crowding DE with single parameter set for all the test problems.

3.3 Modified Species-based Differential Evolution with Self-adaptive Radius for Multi-modal Optimization

Species-based differential evolution (SDE) is one of the recent algorithms that use the notion of speciation for solving multi-modal optimization problems. In this work, a modified SDE with a self-adaptive radius is proposed to overcome the difficulty of selecting the proper radius and improve the performance of SDE. The presented algorithm was tested on a set of classical benchmark multi-modal optimization problems and compared with the original SDE and several other

niching algorithms in literature. As shown in the experimental results, the proposed algorithm outperforms these algorithms on the benchmark problems.

3.3.1 Modified SDE with self-adaptive radius

- SDE with self-adaptive radius

Most existing niching methods suffer from a serious problem - their performance is subjected heavily to some niching parameters which are often difficult to set by a user [154]. For instance, the performance of the original SDE is highly related to the niching radius. To overcome this problem, a self-adaptive radius method is proposed and combined with the original SDE. The steps of the proposed self-adaptive radius method are shown in Table 3-3-1.

Table 3-3-1. The steps of the proposed self-adaptive radius

Input	Sorted population in decreasing fitness value order
Step 1	Take the first individual (best individual in fitness value) as the reference point and find the farthest individual (Euclidean Distance) to the reference point in the current population. The distance between these two individual is set as the radius range.
Step 2	Calculate the expected number of niches. (Assume 10 members per niche. 10 is a preferred number per niche. The actual number can be different. The expected number of niches is equal to the population size (*NP*) divided by 10. Note the maximum expected number of niches is limited to 20, as large number of niches will affect the performance of the niching algorithm.
Step 3	The radius used in this generation is equal to the radius range divided by the expected number of niches.
Output	Niching radius for current population.

- Modified SDE with self-adaptive radius

To further improve the performance of proposed algorithm, some modifications have been done in the selection steps of the original SDE. In original SDE, child is competed with its parent and the winner will survive for the next generation. However, parent-child selection will reduce the diversity of the species of current population. In modified SDE, crowding concept is applied to each of the species. Each offspring is compared with the nearest (Euclidean Distance) member within its species and the winner will be kept for the next generation. The modified SDE is able to keep a higher diversity and generate a better performance.

3.3.2 Experiments and results

- Experimental setup

To assess the performance of the proposed algorithm, 14 frequently used multi-modal optimization benchmark test functions (AF1-AF14) with different characteristics are examined. The following different multi-modal optimization algorithms are examined and compared in the experiments. Note that MSDESA is designed to locate both local/global optima.

1. CDE [145]: Crowding differential evolution.
2. SCMA-ES [155]: Self-adaptive niching CMA-ES
3. R2pso [154]: An *lbest* PSO with a ring topology, each member interacts with only its immediate member to its right.
4. SDE [146]: Speciation-based DE.
5. SDESA: Speciation-based DE with self-adaptive radius.
6. MSDESA: Modified speciation-based DE with self-adaptive radius.

The level of accuracy, population size as well as the maximum number of function evaluations are listed in Table 3-3-2. 25 independent runs are conducted for each of the algorithms to remove the random effect. Success rate and average number of optima found are used as the criteria to access the performance. The DE parameters are listed as below: *F=0.9, CR=0.1*.

Table 3-3-2. Level of accuracy used in this experiment

Test Function	Level of	Population size	No. of FEs
AF1	0.05	50	10000
AF2	0.05	50	10000
AF3	0.05	50	10000
AF4	0.000001	50	10000
AF5	0.000001	50	10000
AF6	0.000001	50	10000
AF7	0.000001	50	10000
AF8	0.0005	50	10000
AF9	0.000001	50	10000
AF10	0.00001	500	100,000
AF11	0.05	250	100000
AF12	0.0001	100	20000
AF13	0.001	500	200000
AF14	0.001	1000	400000

- Experimental results

  The success rates for SDE, SDESA and MSDESA are recorded and presented in Table 3-3-3. The rank of each algorithm is presented in the bracket. As can be seen from the table, the proposed SDESA performs better than the original SDE, while the proposed MSDESA performs the best among the compared SDE versions. MSDESA generates a much smaller total rank than the original SDE. The better performance is due to the self-adaptive radius, which can vary during the evolution process according to the range of the current population. What is more, the modified selection process further improved the diversity of the population and makes the algorithm more suitable to locate multiple peaks. Therefore, we can state that MSDESA outperforms the original SDE on the tested benchmark problems. The niching radius used for the original SDE in this experiment is listed in the second column of the table. Beside this, the number of optima found for each test function is shown in Table 3-3-4. The best algorithm is highlighted in bold face.

Table 3-3-3. The success rate

Test	Rs	SDE	SDESA	MSDESA
AF1	0.5	84 (3)	**100 (1)**	**100 (1)**
AF2	0.5	68 (3)	**100 (1)**	**100 (1)**
AF3	0.5	4 (2)	4 (2)	**24 (1)**
AF4	0.01	72 (3)	**100 (1)**	**100 (1)**
AF5	0.01	0 (1)	0 (1)	**0 (1)**
AF6	0.01	60 (2)	56 (3)	**100 (1)**
AF7	0.01	**100 (1)**	**100 (1)**	**100 (1)**
AF8	0.5	72 (3)	**100 (1)**	**100 (1)**
AF9	0.5	**100 (1)**	**100 (1)**	**100 (1)**
AF10	0.5	0 (2)	0 (2)	**72 (1)**
AF11	0.5	0 (3)	96 (2)	**100 (1)**
AF12	0.2	48 (3)	68 (2)	**96 (1)**
AF13	0.2	0 (2)	0 (2)	**4 (1)**
AF14	0.2	0 (2)	0 (2)	**0 (1)**
Total Rank		31	22	**14**

Table 3-3-4. The number of optima found

Test	Rs		SDE	SDESA	MSDESA
AF1	0.5	Min	1	2	2
		Max	2	2	2
		Mean	1.84	**2**	**2**
		Std	0.3742	0	0
AF2	0.5	Min	1	2	2
		Max	2	2	2
		Mean	1.68	**2**	**2**
		Std	0.4761	0	0
AF3	0.5	Min	2	2	3
		Max	5	5	5
		Mean	3.04	3.04	**3.52**
		Std	0.7895	0.7895	0.8718
AF4	0.01	Min	4	5	5
		Max	5	5	5
		Mean	4.72	**5**	**5**
		Std	0.4583	0	0
AF5	0.01	Min	1	1	2
		Max	3	4	4
		Mean	1.52	2.88	**3**
		Std	0.5859	0.781	0.6455
AF6	0.01	Min	4	4	5
		Max	5	5	5
		Mean	4.6	4.56	**5**
		Std	0.5	0.5066	0
AF7	0.01	Min	1	1	1
		Max	1	1	1
		Mean	**1**	**1**	**1**

		Std	0	0	0
		Min	3	4	4
AF8	0.5	Max	4	4	4
		Mean	3.72	**4**	**4**
		Std	0.4583	0	0
		Min	2	2	2
AF9	0.5	Max	2	2	2
		Mean	**2**	**2**	**2**
		Std	0	0	0
		Min	1	1	18
AF10	0.5	Max	2	18	25
		Mean	1.32	6.36	**23.84**
		Std	0.4761	5.016	2.3036
		Min	9	17	18
AF11	0.5	Max	17	18	18
		Mean	12.36	17.96	**18**
		Std	1.9122	0.2	0
		Min	3	4	5
AF12	0.2	Max	6	6	6
		Mean	4.88	5.44	**5.96**
		Std	1.1299	0.8699	0.2
		Min	20	21	24
AF13	0.2	Max	27	27	36
		Mean	22.84	23.64	**27.28**
		Std	1.8859	1.5242	2.3544
		Min	42	57	80
AF14	0.2	Max	60	73	104
		Mean	50.5597	64.36	**86.24**
		Std	4.0734	4.872	5.372

The proposed algorithm is also compared with a number of niching methods in literature. The results of success rate are shown in Table 3-3-5. As can been seen, MSDESA performs the best out of the compared algorithms.

Table 3-3-5 The success rate of different niching algorithms

Test	CDE	SCMA-	r2pso	SDE	SDESA	MSDESA
AF1	**100 (1)**	**100 (1)**	72 (6)	84 (5)	**100 (1)**	**100 (1)**
AF2	**100 (1)**	**100 (1)**	56 (6)	68 (5)	**100 (1)**	**100 (1)**
AF3	**44 (1)**	0 (5)	0 (5)	4 (3)	4 (3)	24 (2)
AF4	28 (5)	0 (6)	92 (3)	72 (4)	**100 (1)**	**100 (1)**
AF5	**48 (1)**	0 (2)	0 (2)	0 (2)	0 (2)	0 (2)
AF6	28 (5)	0 (6)	88 (2)	60 (3)	56 (4)	**100 (1)**
AF7	60 (6)	96 (5)	**100 (1)**	**100 (1)**	**100 (1)**	**100 (1)**
AF8	0 (6)	44 (4)	28 (5)	72 (4)	**100 (1)**	**100 (1)**

AF9	0 (6)	**100 (1)**	56 (5)	**100 (1)**	**100 (1)**	**100 (1)**
AF10	0 (3)	0 (3)	60 (2)	0 (3)	0 (3)	**72 (1)**
AF11	72 (3)	0 (5)	4 (4)	0 (5)	96 (2)	**100 (1)**
AF12	56 (4)	0 (6)	68 (2)	48 (5)	68 (2)	**96 (1)**
AF13	**8 (1)**	0 (3)	0 (3)	0 (3)	0 (3)	4 (2)
AF14	**0 (1)**	**0 (1)**	**0 (1)**	**0 (1)**	**0 (1)**	**0 (1)**
Total	44	49	47	45	26	**17**

3.3.3 Conclusion

In this section, a modified species-based differential evolution with self-adaptive radius is introduced to overcome the difficulty of choosing niching parameters and improve the performance of the original SDE. The new radius adaptive strategy makes the algorithm adjusting the niching radius according to the population range while the modified selection strategy increases the diversity of the population. The proposed algorithm is compared with the original SDE as well as a number of commonly used multi-modal algorithms on a set of multi-modal benchmark test problems. As can be revealed from the experimental results, the proposed algorithm outperforms other niching algorithms.

3.4 Differential Evolution with Neighborhood Mutation for Multi-modal Optimization

In this section, a neighborhood mutation strategy is introduced and integrated with various niching differential evolution algorithms to solve multi-modal optimization problems. In the proposed neighborhood based differential evolution, the mutation is performed within each Euclidean neighborhood. The neighborhood mutation is able to maintain the multiple optima found during the evolution and evolve toward the respective global/local optimum. To test the performance of neighborhood mutation DE, a total of 29 problem instances are used. The proposed algorithms are compared with a number of state-of-the-art multi-modal optimization approaches and the experimental results suggest that niching DE with neighborhood mutation is able to provide better and more consistent performance over the state-of-the-art multi-modal algorithms [145], [146], [154], [155], [156], [157], [158], [162].

3.4.1 Neighborhood based differential evolution

In canonical DE, the mutation is performed between randomly picked individuals form the entire population. This will allow any two members to form the difference vector in DE/rand/1 (Equation 2-3-4), even if the two members are far apart from each other. This mutation is efficient when searching for single global peak optimization. It prevents premature local convergence and ensures global convergence in the final stage as all individuals in general evolve to one optimal point. However, when solving a multi-modal problem, multiple optima need to be located simultaneously. If the global version of the mutation is used, it will not be efficient for multiple localized convergences at the final search stage as required for multi-modal optimization. At final search stage, the whole population is distributed around different optimal regions. If the parameter space distances between different optima are large, efficient convergence to any of the optima will become impossible as the difference vectors can be generated using individuals from different optimal regions. Although some of the niching techniques can limit the mutation within each niche (SDE), they all depend on certain niching parameters such as the species radius. Choosing a proper niching parameter itself is almost impossible and the algorithm is greatly affected by the niching parameter.

Inspired by these observations, a neighborhood based mutation is proposed and integrated with three different DE niching algorithms namely CDE, SDE and sharing DE. Note that a modified version of fitness sharing DE is also introduced and used. The process of calculating the modified shared fitness and the steps of the modified fitness sharing DE are presented in Tables 3-4-1 and 3-4-2, respectively.

Table 3-4-1. The modified fitness sharing method

Input:	The population with original fitness value
Step 1	Sort the population in descending order of original fitness
	While the population is not empty

Step 2	Find the best unprocessed solution in the population and set it as the new niche centre.
Step 3	Indentify the solutions within the new niche using the specified niching radius.
Step 4	Penalize the solutions (except the niche centre) within the niche using the original fitness sharing formula (Section II B). Note that the best solution at the current niche centre uses the original fitness value.
Step 5	Remove the niche centre and the solutions within the niche from the population.
	Endwhile
Output:	The population with shared fitness computed using the modified fitness sharing method.

Table 3-4-2. Modified fitness sharing DE

Step 1	Randomly initialize the population and external archive. The archive size is twice of the Population size.
Step 2	Modify the fitness value of the solutions in archive using the modified fitness sharing method (Table III).
Step 3	Sort the solutions in archive in decreasing order of modified fitness.
Step 4	The first *NP* (population size) solutions in archive are used as parents to produce offspring using neighborhood mutation.
Step 5	Update the offspring with the nearest (Euclidean distance) member in the archive.
Step 6	Stop if a termination criterion is met. Otherwise go to Step 2.

In neighborhood mutation, difference vector generation is limited to a number (*parameter m*) of similar individuals as measured by Euclidean distance. The steps of generating offspring using neighborhood mutation are presented in Table 3-4-3. In this way, each individual is evolved towards its nearest optimal point and the possibility of between niche difference vector generation is reduced. The neighborhood based CDE (NCDE), neighborhood based SDE (NSDE) and neighborhood based sharing DE (NShDE) are presented in Tables 3-4-4, 3-4-5 and 3-4-6, repectively.

Table 3-4-3. The steps of generating offspring using neighborhood mutation

Input	A population of solutions of current generation (current parents)
Step 1	For i = 1 to NP (population size)
	1.1 Calculate the Euclidean distances between individual i and other members inside the population.
	1.2 Select m smallest Euclidean distance members to individual i and form a subpopulation (*subpop)* using these m members.
	1.3 Produce an offspring u_i using DE equations within $subpop_i$, i.e., pick r_1, r_2, r_3 from the subpopulation.
	2.3 Reset offspring u_i within the bounds if any of the dimensions the bounds.
	2.4 Evaluate offspring u_i using the fitness function.
	Endfor
Step 2	Selection NP fitter solutions for next generation using the according strategies of different algorithm.
Output	A population of solutions for next generation

Table 3-4-4. The NCDE algorithm

Step 1	Randomly generate NP number of initial trial solutions.
Step 2	For i = 1 to NP
	2.1 Find the parametrically most similar (the similarity is measured

	in terms of Euclidean distances in parameter space) *m* individuals of solution *i* to form a subpopulation $subpop_i$.
	2.2 Produce an offspring u_i using DE within $subpop_i$, i.e., pick r_1, r_2, r_3 from the i^{th} subpopulation.
	2.3 Calculate the Euclidean distance of u_i to the other individuals in the entire population.
	2.4 Compare the fitness of u_i with the most similar (w.r.t. Euclidean parameter space distance) individual and replace the most similar individual if the u_i has a better fitness.
	Endfor
Step 3	Stop if a termination criterion is satisfied. Otherwise go to Step 2.

Table 3-4-5. The NSDE algorithm

Step 1	Randomly generate *NP* number of initial trial solutions.
Step 2	Sort all individuals in descending order of their fitness values.
Step 3	While the sorted population is not empty
	Determine the species seed which is the best (fitness value) unprocessed individual. Find the parametrically most similar *m* individuals of the species seed and set them as one species. Remove the processed members for the current populations.
	Endwhile
Step 4	For each species, execute a global DE variant:
	If the child's fitness is the same as its species seed, replace this child with a randomly generated new individual.
Step 5	Keep only the *NP* fittest (objective value) individuals from the combined population.
Step 6	Stop if a termination criterion is satisfied. Otherwise go to Step 4.

Table 3-4-6. The NShDE algorithm

Step 1	Randomly initialize the population and external archive. The archive size is twice of the Population size.
Step 2	Modify the fitness value of the solutions in archive using the modified fitness sharing method (Table 3-4-1).
Step 3	Sort the solutions in archive in decreasing of the modified fitness order.
Step 4	The first *NP* (population size) solutions in archive are used as parents to produce offspring.
Step 5	For i=1:*NP* 5.1 Find the parametrically most similar *m* individuals of solution i to form a subpopulation $subpop_{i.}$ 5.2 Produce an offspring u_i using DE within $subpop_i$, i.e., pick r_1, r_2, r_3 from the i^{th} subpopulation. 5.3 Calculate the Euclidean distance of u_i to the other individuals in the entire external archive. 5.4 Compare the fitness of u_i with the most similar individual and replace the most similar individual if the u_i has a better fitness. Endfor
Step 6	Stop if a termination criterion is met. Otherwise go to Step 2.

In neighborhood mutation, there is only one parameter *m* which is the neighborhood size. This parameter controls how many individuals are selected in each subpopulation. Generally *m* should be chosen between 1/20 of the population size and 1/5 of the population size. It can also be dynamically set, from a relative large value to a small value. Different from other niching parameters, neighborhood size is easy to choose as it can be made proportional to the

population size. Moreover, the performance of the algorithm is not sensitive to the change of the neighborhood size as evidenced in Table 3-4-16 (NCDE).

3.4.2 Experimental setup

The DE parameters used in this work are as follow: $F = 0.9 \quad CR = 0.1$. Two experiments are conducted. In the first experiment, the proposed algorithm is compared with a number of commonly used niching algorithms; while the second experiment compares the performance of neighborhood based DE against one of the most recently reported multi-modal algorithms. In experiment two, we use the same test functions and settings as in [156] instead of reprogramming the algorithms in [156]. In total, eighteen different multi-modal algorithms are considered in our experiments:

1. NCDE: The neighborhood based crowding DE.
2. NSDE: The neighborhood based speciation DE.
3. NShDE: The neighborhood based sharing DE.
4. CDE [145][144]: The original crowding DE.
5. SDE [146]: Speciation-based DE.
6. ShDE: Modified Sharing DE.
7. FERPSO [154]: Fitness-Euclidean distance ratio PSO.
8. SPSO [154]: Speciation-based PSO.
9. *r*2pso [154]: An *lbest* PSO with a ring topology, each member interacts with only its immediate member to its right.
10. *r*3pso [154]: An *lbest* PSO with a ring topology, each member interacts with its immediate member on its left and right.
11. *r*2pso-*lhc* [154]: The same as *r*2pso, but with no overlapping neighborhoods.
12. *r*3pso-*lhc* [154]: The same as *r*3pso, but with no overlapping neighborhoods.
13. CMA-ES [155]: Niching covariance matrix adaptation evolution strategy
14. SCMA-ES[155]: Self-adaptive niching CMA-ES
15. TSC [157]; Topological species conservation

16. SCGA [158]: Species conserving genetic algorithm
17. DFS [162]: Dynamic fitness sharing
18. TSC2 [156]:Topological species conservation 2

- Test functions

In experiment one (E1), two sets of test functions are used in order to demonstrate the superior performance of neighborhood mutation based DE. These functions are widely used in multi-modal optimization and possess diverse characteristics. Set 1 contains 14 classical multi-modal problems (AF1-AF14) while set 2 has 15 novel composite multi-modal problems (Appendix B, BF1-BF15).

In experiment two (E2), the test functions and results in [156] are used. Table 3-4-7 shows the test functions. More details can be found in [156]. The experimental procedure is adopted from [156].

Table 3-4-7. Test Functions for Experiment Two

Experiment Two [156]			
Test Function name	Number of Global peaks	Test Function name	Number of Global peaks
E2-F1: Waves	10	E2-F8: Shubert	18
E2-F2: Six-hump camel back	2	E2-F9: Ackley	1
E2-F3: Sphere	1	E2-F10: Michalewicz	1
E2-F4: Shifted Rastrigin	1	E2-F11: Ursem F1	1
E2-F5: Rotated Hybrid Composition function	1	E2-F12: Ursem F3	1
E2-F6: Rescaled Six-hump camel back	2	E2-F13: Ursem F4	1
E2-F7: Branin RCOS	3		

- Population size and maximal number of evaluations

  In experiment one for test function set 1, the level of accuracy, niching radius, population size and maximal number of function evaluations are shown in Table 3-4-8. For test function set 2, a population size of 600 is used with the maximal number of function evaluations set at 300,000. The level of accuracy and niching radius are set as 0.5 and 1 respectively for all the problems in test function set 2. The detailed settings for experiments two can be found in [156].

  Table 3-4-8. Population size and function evaluation for E1-AF1 - E1-AF14

Function no.	ε	r	Population	No. of
E1-AF1	0.05	0.5	50	10000
E1-AF2	0.05	0.5	50	10000
E1-AF3	0.05	0.5	50	10000
E1-AF4	0.000001	0.01	50	10000
E1-AF5	0.000001	0.01	50	10000
E1-AF6	0.000001	0.01	50	10000
E1-AF7	0.000001	0.01	50	10000
E1-AF8	0.0005	0.5	50	10000
E1-AF9	0.000001	0.5	50	10000
E1-AF10	0.00001	0.5	50	10000
E1-AF11	0.05	0.5	250	100000
E1-AF12	0.0001	0.2	100	20000
E1-AF13	0.001	0.2	500	200000
E1-AF14	0.001	0.2	1000	400000

- Performance measure

  Experiment one uses success rate and average number of optima found as the criteria to access the performance. All performance indicators are calculated and averaged over 25 independent runs to eliminate the effect due to the random initialization, In experiment two; the criteria in [156] are used:

  1) Peak accuracy: For each desired peak to be located the closest individual x in the population is taken and absolute difference in

objective values is calculated. If the objective value of individual x is denoted by $f(x)$, the peak accuracy is calculated using the following equation:

$$peak\ accuracy = \sum_{i=1}^{\#\ peaks} \left| f(peak_i) - f(x) \right|$$

2) Distance accuracy: Peak accuracy may lead to erroneous results, if the peaks are close to each other or with identical height. The distance accuracy is used to avoid this error. It is calculated the same way as peak accuracy, with the only change that the fitness values are replaced by the Euclidean distance [156].

3.4.3 Experimental results

Experiment One

This section presents the experimental results and analyses of Experiment One. All algorithms were run until all known peaks were found or the maximum number of function evaluations was exhausted.

- Results on test function set 1

  The results of test function set 1 are shown in Tables 3-4-9 and 3-4-10. The second and third column of these tables indicate the level of accuracy and niche radius (for SDE and SPSO) used in the experiments. Success rates with ranks (inside the bracket) are listed in Table 3-4-9 while the average number of optima found is listed in Table 3-4-10. From these two tables, we can see that the proposed algorithms outperform the other algorithms in terms of both criteria. For these 14 problems, the proposed algorithms ranks top three among all the compared state-of-the-art algorithms. If we compare NCDE with the original CDE, we can see that NCDE improves 11

problems out of 14 while the remaining 3 cases are similar as both methods are able to locate all the optima in all runs, The superior performance is due to the neighborhood mutation, which makes it easier for the proposed algorithms to converge to different optima.

In order to determine the statistical significance of the advantage of neighborhood mutation based DE over other methods, *t*-test is applied on the average number of peaks found and the results are shown in Table 3-4-11. The numerical values 1, 0 represent that other methods are statistically inferior to, equivalent to the best algorithm with respect to each test function. From the results, we can observe that the neighborhood based DE methods always perform better or equal when compared with other niching methods on all test functions.

As stated by Mahfoud [106], a good niching algorithm must be able to locate global optima and maintain them for an exponential to infinite time period, with respect to population size. Maintaining found optima is crucial for multi-modal optimization algorithms. The neighborhood based DE algorithms are able to find and maintain the found optima to the end of the run. This is because of the replacement strategies. The replacement is either within the subpopulation or between most similar individuals. Therefore once a niche is formed around one global peak, it can maintain this peak until the end of the run.

Table 3-4-9. Success Rate for test function set 1

Function	NCDE	NShDE	NSDE	CDE	ShDE	SDE	FER-PSO	SPSO	r2pso	r3pso	r2pso-lhc	r3pso-lhc	SCMA-ES
E1-AF1	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	72 (10)	48 (13)	76 (9)	84 (8)	56 (12)	60 (11)	100 (1)
E1-AF2	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	44 (12.5)	88 (10)	96 (9)	44 (12.5)	56 (11)	100 (1)
E1-AF3	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	96 (7)	20 (8)	4 (12.5)	8 (10)	8 (10)	4 (12.5)	8 (10)	100 (1)
E1-AF4	100 (1)	100 (1)	100 (1)	28 (11)	4 (12)	72 (10)	84 (9)	88 (7.5)	92 (5.5)	88 (7.5)	100 (1)	92 (5.5)	0 (13)
E1-AF5	100 (1)	100 (1)	100 (1)	72 (12)	44 (13)	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)
E1-AF6	100 (1)	100 (1)	100 (1)	28 (11)	8 (12)	60 (10)	100 (1)	92 (6)	88 (8)	72 (9)	92 (6)	92 (6)	0 (13)
E1-AF7	100 (1)	100 (1)	100 (1)	60 (12)	40 (13)	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	96 (11)
E1-AF8	100 (1)	92 (3)	100 (1)	0 (12)	0 (12)	72 (4.5)	72 (4.5)	0 (12)	28 (7.5)	24 (9.5)	28 (7.5)	24 (9.5)	44 (6)
E1-AF9	100 (1)	100 (1)	100 (1)	0 (12)	0 (12)	100 (1)	96 (6)	0 (12)	56 (8.5)	60 (7)	56 (8.5)	52 (10)	100 (1)
E1-AF10	100 (1)	96 (4.5)	100 (1)	52 (11)	96 (4.5)	32 (12)	100 (1)	56 (10)	88 (6)	76 (7)	72 (8)	60 (9)	4 (13)
E1-AF11	100 (1)	100 (1)	100 (1)	72 (4)	28 (6)	0 (12)	52 (5)	0 (12)	4 (9)	4 (9)	4 (9)	20 (7)	0 (12)
E1-AF12	84 (2.5)	88 (1)	84 (2.5)	56 (8.5)	68 (5.5)	48 (11.5)	60 (7)	72 (4)	68 (5.5)	56 (8.5)	52 (10)	48 (11.5)	0 (13)
E1-AF13	88 (3)	96 (1)	24 (4)	8 (5)	92 (2)	0 (9.5)	0 (9.5)	0 (9.5)	0 (9.5)	0 (9.5)	0 (9.5)	0 (9.5)	0 (9.5)
E1-AF14	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)
Total Rank	**17.5**	**19.5**	**18.5**	105.5	83	82.5	65	114	91.5	97	99.5	103	96.5

Table 3-4-10. Average number of peaks found for test function set 1

Function	NCDE	NShDE	NSDE	CDE	ShDE	SDE	FER-PSO	SPSO	r2pso	r3pso	r2pso-lhc	r3pso-lhc	SCMA-ES
E1-AF1	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	0.72 (10)	0.48 (13)	0.76 (9)	0.84 (8)	0.56 (12)	0.6 (11)	1 (1)
E1-AF2	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	0.44 (12.5)	0.88 (10)	0.96 (9)	0.44 (12.5)	0.56 (11)	1 (1)

E1-AF3	2 (1)	2 (1)	2 (1)	2 (1)	2 (1)	1.96 (7)	0.8 (8)	0.24 (13)	0.48 (11.5)	0.6 (9.5)	0.48 (11.5)	0.6 (9.5)	2 (1)
E1-AF4	5 (1)	5 (1)	5 (1)	3.84 (11)	3.28 (12)	4.72 (10)	4.84 (9)	4.88 (7.5)	4.92 (5.5)	4.88 (7.5)	5 (1)	4.92 (5.5)	0.04 (13)
E1-AF5	1 (1)	1 (1)	1 (1)	0.72 (12)	0.44 (13)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)
E1-AF6	5 (1)	5 (1)	5 (1)	3.96 (11)	3.28 (12)	4.6 (10)	5 (1)	4.92 (5.5)	4.88 (7.5)	4.72 (9)	4.92 (5.5)	4.88 (7.5)	0 (13)
E1-AF7	1 (1)	1 (1)	1 (1)	0.6 (12)	0.4 (13)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	0.96 (11)
E1-AF8	4 (1)	3.92 (3)	4 (1)	0.32 (12)	0.16 (13)	3.72 (4)	3.68 (5)	0.84 (11)	2.92 (9)	2.76 (10)	3 (8)	3.12 (7)	3.44 (6)
E1-AF9	2 (1)	2 (1)	2 (1)	0.04 (12.5)	0.04 (12.5)	2 (1)	1.96 (6)	0.08 (11)	1.44 (10)	1.56 (7.5)	1.56 (7.5)	1.48 (9)	2 (1)
E1-AF10	1 (1)	0.96 (4.5)	1 (1)	0.52 (11)	0.96 (4.5)	0.32 (12)	1 (1)	0.56 (10)	0.88 (6)	0.76 (7)	0.72 (8)	0.6 (9)	0.04 (13)
E1-AF11	18 (1)	18 (1)	18 (1)	17.7 (4)	16.56 (6)	12.4 (11)	17.4 (5)	8.52 (12)	15.2 (9)	15.6 (8)	15.1 (10)	16.2 (7)	2.16 (13)
E1-AF12	5.8 (2)	5.88 (1)	5.84 (3)	5.56 (6)	5.6 (4.5)	4.88 (12)	5.36 (8.5)	5.6 (4.5)	5.52 (7)	5.16 (11)	5.36 (8.5)	5.28 (10)	1.52 (13)
E1-AF13	35.9 (3)	35.96 (1)	30.6 (5)	33.8 (4)	35.92 (2)	22.8 (9)	23.6 (7)	25.7 (6)	21.8 (12)	22.2 (11)	22.5 (10)	23.1 (8)	1.4 (13)
E1-AF14	179 (3)	198.96 (1)	84.28 (5)	152 (4)	197.88 (2)	50.6 (8)	68.6 (7)	70.1 (6)	40.6 (12)	45.4 (9)	42.2 (11)	43.3 (10)	0.04 (13)
Total Rank	**19**	**19.5**	**24**	102.5	97.5	88	70.5	114	110.5	108.5	107.5	106.5	113

Table 3-4-11 *t*-test on the average number of peaks found for test function set 1

Function	NCDE	NShDE	NSDE	CDE	ShDE	SDE	FER-PSO	SPSO	r2pso	r3pso	r2pso-lhc	r3pso-lhc	SCMA-ES
E1-AF1	N.A.	0	0	0	0	0	1	1	1	1	1	1	0
E1-AF2	N.A	0	0	0	0	0	0	1	1	1	1	1	0
E1-AF3	N.A.	0	0	0	0	0	1	1	1	1	1	1	0
E1-AF4	N.A.	0	0	1	1	1	1	0	0	0	0	0	1
E1-AF5	N.A.	0	0	0	1	1	0	0	0	0	0	0	0
E1-AF6	N.A.	0	0	1	1	1	0	0	0	1	0	0	1

E1-AF7	N.A	0	0	0	1	1	0	0	0	0	0	0	0
E1-AF8	N.A.	1	0	1	1	1	1	1	1	1	1	1	1
E1-AF9	N.A.	0	0	1	1	0	0	1	1	1	1	1	0
E1-AF10	N.A.	1	0	1	1	1	0	1	1	1	1	1	1
E1-AF11	N.A.	0	0	1	1	1	1	1	1	1	1	1	1
E1-AF12	0	N.A.	0	0	0	1	1	0	0	1	1	1	1
E1-AF13	0	N.A.	1	1	0	1	1	1	1	1	1	1	1
E1-AF14	1	N.A.	1	1	0	1	1	1	1	1	1	1	1

- Results on test function set 2

  All 15 functions in test function set 2 are composition functions. These composition functions are much more complex than the functions in set 1. Among these test problems, no algorithm is able to generate a nonzero success rate except the proposed algorithms (NCDE and NSDE). NCDE is able to generate the success rates of 0.8, 0.1 and 0.4 on *E1-BF3*, *E1-BF4* and *E1-BF5* respectively while NSDE is able to generate the success rates of 1, 0.5 and 0.9 on these test functions. Therefore, the average number of global optima found is used as the sole criterion. The results are shown in Table 3-4-12. NSDE ranks the best out of the thirteen algorithms on every problem. The statistical test results are shown in Table 3-4-13. By comparing the results of benchmark sets 1 and 2, we can conclude that as the complexity of the problems is increased, the proposed neighborhood mutation increases its advantage over all competing state-of-the-art

algorithms. Fig. 3-4-1 and Fig. 3-4-2 give a quick visual comparison on average number of peaks found by all algorithms.

Table 3-4-12 Average number of peaks found for test function set 2

Function	NCDE	NShDE	NSDE	CDE	ShDE	SDE	FER-PSO	SPSO	r2pso	r3pso	r2pso-lhc	r3pso-lhc	SCMA-ES
E1-BF1	5.18 (2)	3.7 (3)	6.7 (1)	0 (10)	0 (10)	1.79 (5)	1.08 (6)	0 (10)	0 (10)	0 (10)	0 (10)	0 (10)	2 (4)
E1-BF2	3.6 (2)	2.8 (3)	4 (1)	1.2 (6.5)	1.1 (8)	1.2 (6.5)	2 (4)	0 (11)	0 (11)	0 (11)	0 (11)	0 (11)	1.9 (5)
E1-BF3	5.8 (2)	4 (3)	6 (1)	0.7 (8)	1.11 (7)	1.5 (6)	2.5 (5)	0 (11)	0 (11)	0 (11)	0 (11)	0 (11)	2.7 (4)
E1-BF4	4.8 (2)	4.5 (3)	5.4 (1)	0 (9)	0 (9)	0 (9)	0 (9)	0 (9)	0 (9)	0 (9)	0 (9)	0 (9)	0.2 (4)
E1-BF5	5.2 (2)	3.6 (3)	5.9 (1)	1.1 (8)	1.3 (7)	1.3 (6)	2 (4)	0 (11)	0 (11)	0 (11)	0 (11)	0 (11)	1.9 (5)
E1-BF6	3 (1)	3 (1)	3 (1)	0 (10)	0 (10)	1.4 (5)	1.2 (6)	0 (10)	0 (10)	0 (10)	0 (10)	0 (10)	2.6 (4)
E1-BF7	1.8 (2)	1 (4)	1.9 (1)	0 (10)	0 (10)	1 (4)	0.5 (6)	0 (10)	0 (10)	0 (10)	0 (10)	0 (10)	1 (4)
E1-BF8	3 (1)	3 (1)	3 (1)	0 (10)	0 (10)	1.4 (6)	1.5 (5)	0 (10)	0 (10)	0 (10)	0 (10)	0 (10)	2.3 (4)
E1-BF9	3 (1)	3 (1)	3 (1)	0 (10)	0 (10)	1.8 (4)	1.5 (6)	0 (10)	0 (10)	0 (10)	0 (10)	0 (10)	1.7 (5)
E1-BF10	1.3 (2)	1 (6)	2 (1)	0 (10)	0 (10)	1.1 (4.5)	1.1 (4.5)	0 (10)	0 (10)	0 (10)	0 (10)	0 (10)	1.2 (3)
E1-BF11	2.8 (2)	2.2 (3)	4 (1)	0 (9.5)	0 (9.5)	1.3 (4)	0 (9.5)	0 (9.5)	0 (9.5)	0 (9.5)	0 (9.5)	0 (9.5)	0.7 (5)
E1-BF12	2.5 (2)	2 (3)	2.9 (1)	0 (10)	0 (10)	1.6 (5.5)	1.6 (5.5)	0 (10)	0 (10)	0 (10)	0 (10)	0 (10)	1.7 (4)
E1-BF13	2.3 (2)	1 (4)	3.8 (1)	0 (10)	0 (10)	0.9 (5)	0.3 (6)	0 (10)	0 (10)	0 (10)	0 (10)	0 (10)	1.4 (3)
E1-BF14	1 (1)	1 (1)	1 (1)	0 (10)	0 (10)	1 (1)	1 (1)	0 (10)	0 (10)	0 (10)	0 (10)	0 (10)	1 (1)
BF15	3.8 (2)	2.4 (3)	4 (1)	0 (10)	0 (10)	1.6 (5)	1.2 (6)	0 (10)	0 (10)	0 (10)	0 (10)	0 (10)	2 (4)
Total Rank	**26**	**42**	**15**	141	140.5	76.5	82	151.5	151.5	151.5	151.5	151.5	59

Table 3-4-13 *t*-test on the average number of peaks found for test function set 2

Function	NCDE	NSHDE	NSDE	CDE	SHDE	SDE	FER-PSO	SPSO	r2pso	r3pso	r2pso-lhc	r3pso-lhc	SCMA-ES
E1-BF1	1	1	N.A.	1	1	1	1	1	1	1	1	1	1
E1-BF2	0	1	N.A.	1	1	1	1	1	1	1	1	1	1
E1-BF3	0	1	N.A.	1	1	1	1	1	1	1	1	1	1
E1-BF4	1	1	N.A.	1	1	1	1	1	1	1	1	1	1
E1-BF5	0	1	N.A.	1	1	1	1	1	1	1	1	1	1
E1-BF6	0	0	N.A.	1	1	1	1	1	1	1	1	1	1
E1-BF7	0	1	N.A.	1	1	1	1	1	1	1	1	1	1
E1-BF8	0	0	N.A.	1	1	1	1	1	1	1	1	1	1
E1-BF9	1	1	N.A.	1	1	0	0	1	1	1	1	1	0
E1-BF10	1	1	N.A.	1	1	1	1	1	1	1	1	1	1
E1-BF11	0	1	N.A.	1	1	1	1	1	1	1	1	1	1
E1-BF12	1	1	N.A.	1	1	1	1	1	1	1	1	1	1
E1-BF13	0	0	N.A.	1	1	0	0	1	1	1	1	1	0
E1-BF14	0	1	N.A.	1	1	1	1	1	1	1	1	1	1
E1-BF15													

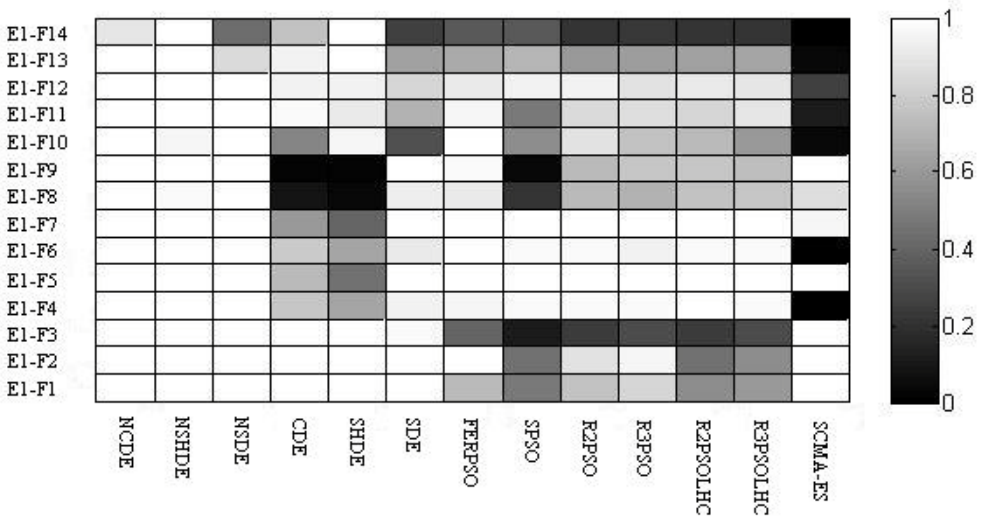


Fig. 3-4-1 Overview of average number of peaks found by each algorithm (Set 1). Results are normalized for each problem, so that 0 refers to the best and 1 to the worst algorithm.

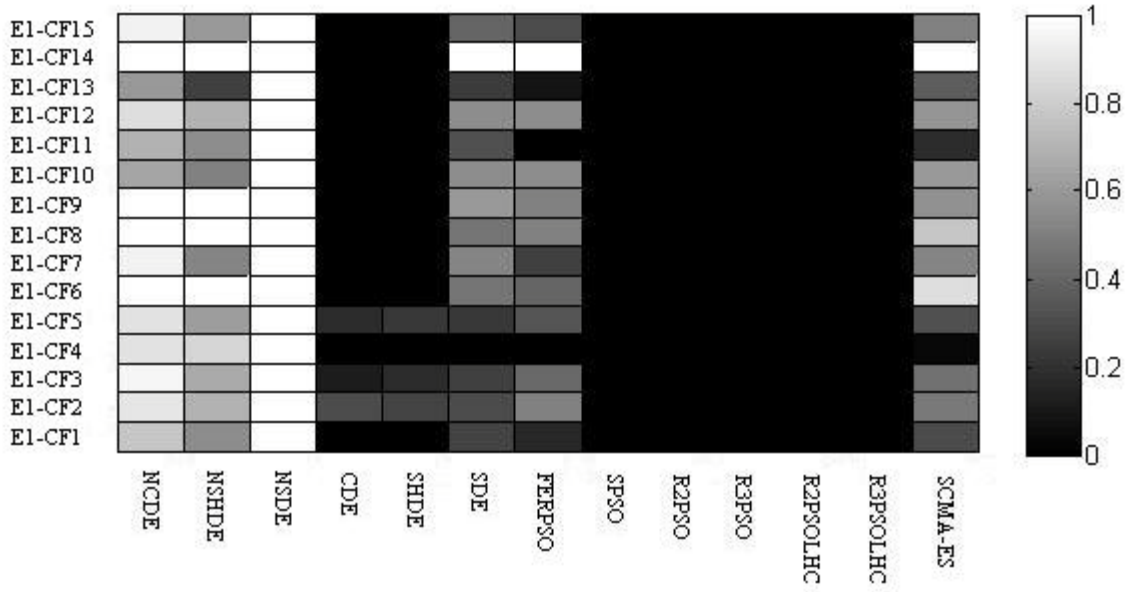


Fig. 3-4-2 Overview of average number of peaks found by each algorithm (Set 2). Results are normalized for each problem, so that 0 refers to the best and 1 to the worst algorithm.

- Locating local optima

In order to test the ability of locating local optima, five test functions (*E1-AF1*, *E1-AF2*, *E1-AF3*, *E1-AF5*, *E1-AF10*) which contain both global and local optima from Set 1 are used. The results are shown in Tables 3-4-14 and 3-4-15. It can be observed that with neighborhood mutation the ability of locating both global and local optima is greatly improved. To give a clear view, the final population of NCDE is also plotted (Fig. 3-4-3 to Fig. 3-4-5) for *E1-AF3*, *E1-AF5* and *E1-AF10*. Note that the population size and maximum number of function evaluations are set to 500 and 100,000 for *E1-AF10*. The settings of the remaining parameters are kept unchanged.

Table 3-4-14 Success rate in locating both global and local peaks

Function	NCDE	NSHDE	NSDE	CDE	SHDE	SDE	FER-PSO	SPSO	r2pso	r3pso	r2pso-lhc	r3pso-lhc	SCMA-ES
E1-AF1	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	84 (7)	64 (9)	44 (13)	72 (8)	56 (10)	48 (12)	52 (11)	100 (1)
E1-AF2	100 (1)	100 (1)	100 (1)	100 (1)	84 (7)	68 (10)	88 (6)	72 (9)	56 (11)	32 (13)	52 (12)	76 (8)	100 (1)
E1-AF3	100 (1)	100 (1)	40 (4)	44 (3)	12 (5)	4 (7)	0 (10.5)	0 (10.5)	0 (10.5)	0 (10.5)	8 (6)	0 (10.5)	0 (10.5)
E1-AF5	100 (1)	100 (1)	76 (4)	48 (6)	4 (7.5)	0 (11)	0 (11)	100 (1)	0 (11)	0 (11)	64 (5)	4 (7.5)	0 (11)
E1-AF10	100 (1)	100 (1)	100 (1)	0 (11.5)	96 (4)	0 (11.5)	0 (11.5)	92 (5)	60 (8)	52 (9)	84 (6)	76 (7)	0 (11.5)
Total Rank	5	5	11	22.5	24.5	46.5	48	38.5	48.5	53.5	41	44	35

Table 3-4-15 Average of optima found in locating both global and local peaks

Function	NCDE	NSHDE	NSDE	CDE	SHDE	SDE	FER-PSO	SPSO	r2pso	r3pso	r2pso-lhc	r3pso-lhc	SCMA-ES
E1-AF1	2 (1)	2 (1)	2 (1)	2 (1)	2 (1)	1.84 (7)	1.48 (11)	1.44 (13)	1.72 (8)	1.48 (11)	1.48 (11)	1.52 (9)	2 (1)
E1-AF2	2 (1)	2 (1)	2 (1)	2 (1)	1.84 (7)	1.68 (10)	1.88 (6)	1.72 (9)	1.36 (12)	1.24 (13)	1.52 (11)	1.76 (8)	2 (1)
E1-AF3	5 (1)	5 (1)	3.76 (4)	4.44 (3)	3.6 (5)	3.04 (7)	0.64 (12)	3.08 (6)	0.8 (11)	0.4 (13)	3 (8)	2.16 (9)	2 (10)
E1-AF5	5 (1)	5 (1)	4.76 (4)	4.28 (6)	3.12 (7)	1.52 (9)	1 (11)	5 (1)	1 (11)	1 (11)	4.52 (5)	2.8 (8)	0.48 (13)
E1-AF10	25 (1)	25 (1)	25 (1)	12.5 (10)	24.96 (4)	1.32 (12)	5.16 (11)	24.9 (5)	24.4 (8)	24.3 (9)	24.8 (6)	24.6 (7)	0.88 (13)
Total Rank	5	5	11	21	24	45	51	34	50	57	41	41	38

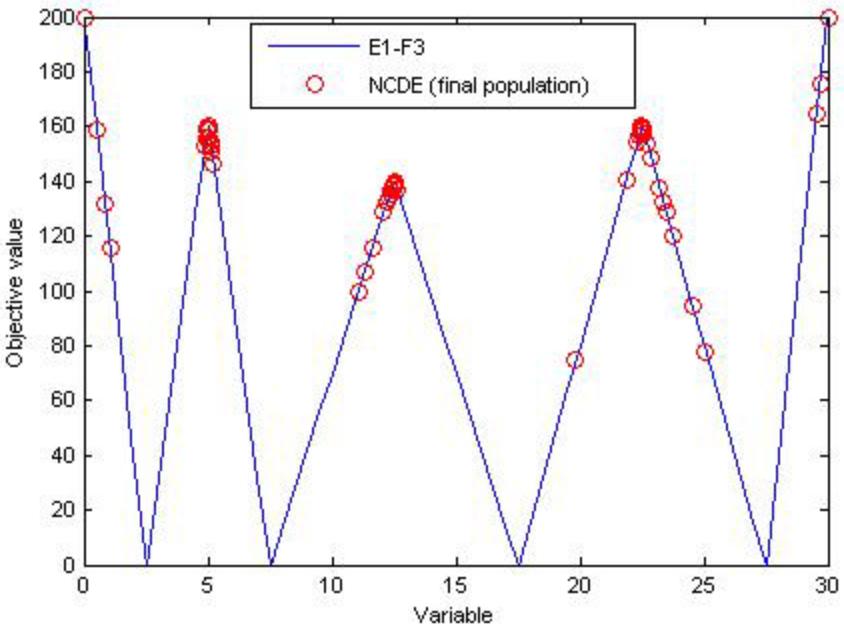


Fig. 3-4-3 The final population of NCDE for E1-AF3

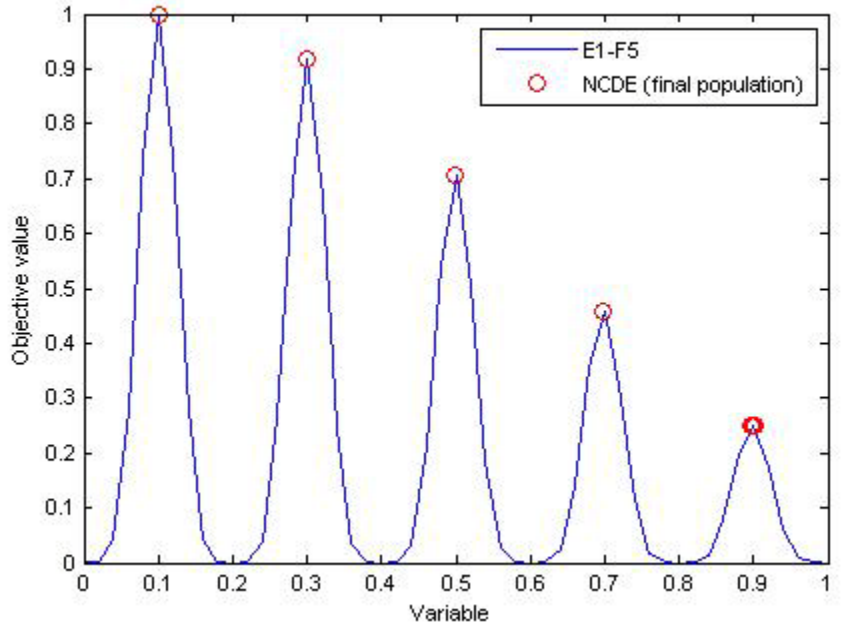


Fig. 3-4-4 The final population of NCDE for E1-F5

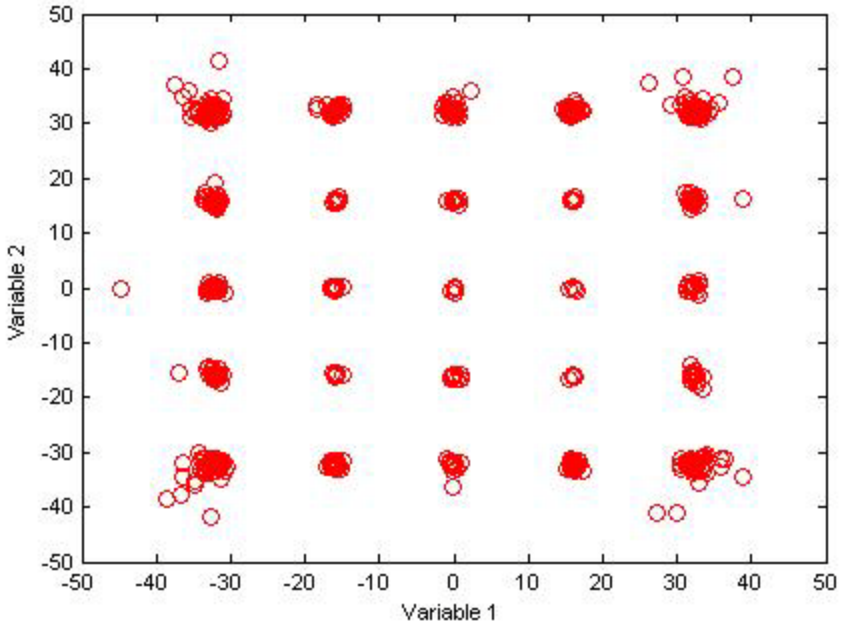


Fig. 3-4-5 The final population of NCDE for E1-F10

- **Summary of different algorithms on the test functions**

This section gives a brief summary on the performance of different algorithms on the benchmark functions.

The Proposed three algorithms (NCDE, NSHDE and NSDE): As we can see from the above tables, all the three algorithms are able to generate consistent and satisfactory performance over a large extend of test functions. CDE, SHDE and SPSO: They can generate acceptable results over simple and low dimensional functions (test function set 1), but their fine tuning abilities are limited. Their performances over difficult and high dimensional problems (test function set 2) are poor.

SDE: Although SDE is always able to find certain number of global peaks and the fine tuning ability over the found optima is good, it fails to find most of the local optima. SDE algorithm also shows very poor performance if the number of global peaks is high. Therefore, if the user's target is finding a few global peaks with high accuracy, SDE could be a good choice. Otherwise, SDE may be unable to generate satisfactory result.

FERPSO: Compare to the proposed algorithm, FERPSO is able to generated relatively satisfactory result over a large extend of test functions. However, the ability to locate local optima is very poor. It cannot be used to locate both global and local optima.

R2PSO and R3PSO: These two methods perform well on simple problems and fail on difficult test functions. Their performances to locate local optima are also poor. Worthy of mentioning is that these algorithms do not require any of the niching parameters (same as the proposed algorithm).

This can be claimed as one of the advantages, as the performance of other algorithms may vary according to the niching parameter.

R2PSOLHC and R3PSOLHC: The performances are similar to R2PSO and R3PSO. The only difference is that these two algorithms showed slightly better abilities in locating local optima.

SCMA-ES: SCMA-ES is one of the most complicated multimodal algorithms. Their performances on complex and high dimensional problems are good. However, its poor fine tuning ability make it fails if the level of accuracy is high when solving even simple problems.

- Effect of varying the neighborhood size

Although the performance is not sensitive to the neighborhood size *m*, it can still affect the performance. In order to test the effect of varying the neighborhood size, three problems (*E1-AF1*, *E1-AF4*, and *E1-AF11*) from test function set 1 are used and examined on NCDE. The results are shown in Table 3-4-16. For *E1-AF1* and *E1-AF4*, *m* is set at 5 (1/10 of population size) and 10 (1/5 of the population size). As we can see, there is no difference in performance. For *E1-AF11*, the *m* is varied from 20 (2/25 of the population size) to 50 (1/5 of the population size). As we can see, only when *m=20*, the performance is reduced by a small margin.

Table 3-4-16 The effect of varying neighborhood size parameter *m*

fnc	ε	*m*	SR	Average no. of optima found
E1-AF1 (two-peak trap)	0.05	5	1	1
E1-AF1 (two-peak trap)	0.05	10	1	1
E1-AF4 (Equal Maxima)	0.000001	5	1	5
E1-AF4 (Equal Maxima)	0.000001	10	1	5
E1-AF11 (2D inverted Shubert function)	0.05	20	0.96	17.96
E1-AF11 (2D inverted Shubert function)	0.05	30	1	18
E1-AF11 (2D inverted Shubert function)	0.05	40	1	18
E1-AF11 (2D inverted Shubert function)	0.05	50	1	18

- Advantage of neighborhood mutation

To demonstrate the merit of neighborhood mutation, the best parents and their offspring of CDE and NCDE in different iterations (E1-AF8) are plotted in Fig. 3-4-6. As we can seen from the plot, the best parents of NCDE are always able to produce offspring around them, which means within the same niche and targeting on the same peak. However, the offspring produced are generally far away from their parents for CDE which may cause slow convergence and poor fine-tuning ability. The distributions of population of CDE and NCDE in different iterations (E1-AF8) are also presented in Fig. 3-4-7, which clearly shows NCDE converges faster than the original CDE.

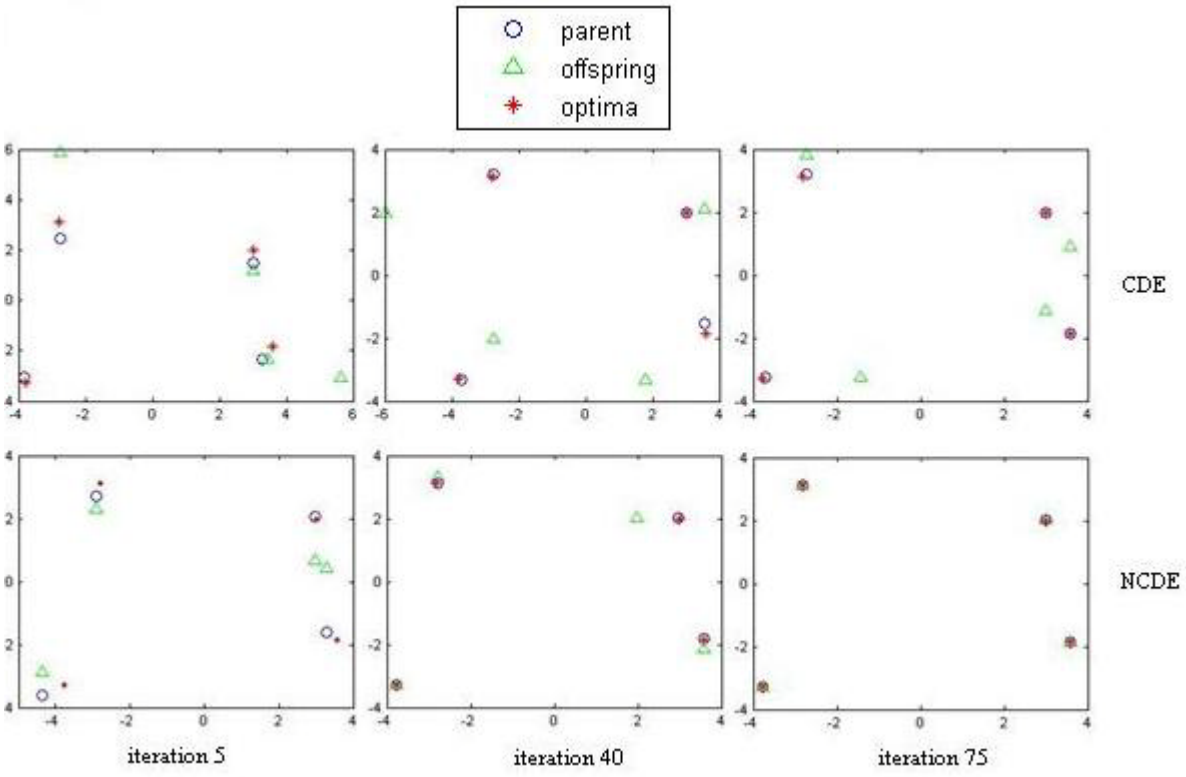


Fig. 3-4-6 The best parents and their offspring of CDE and NCDE in different iterations on E1-AF8

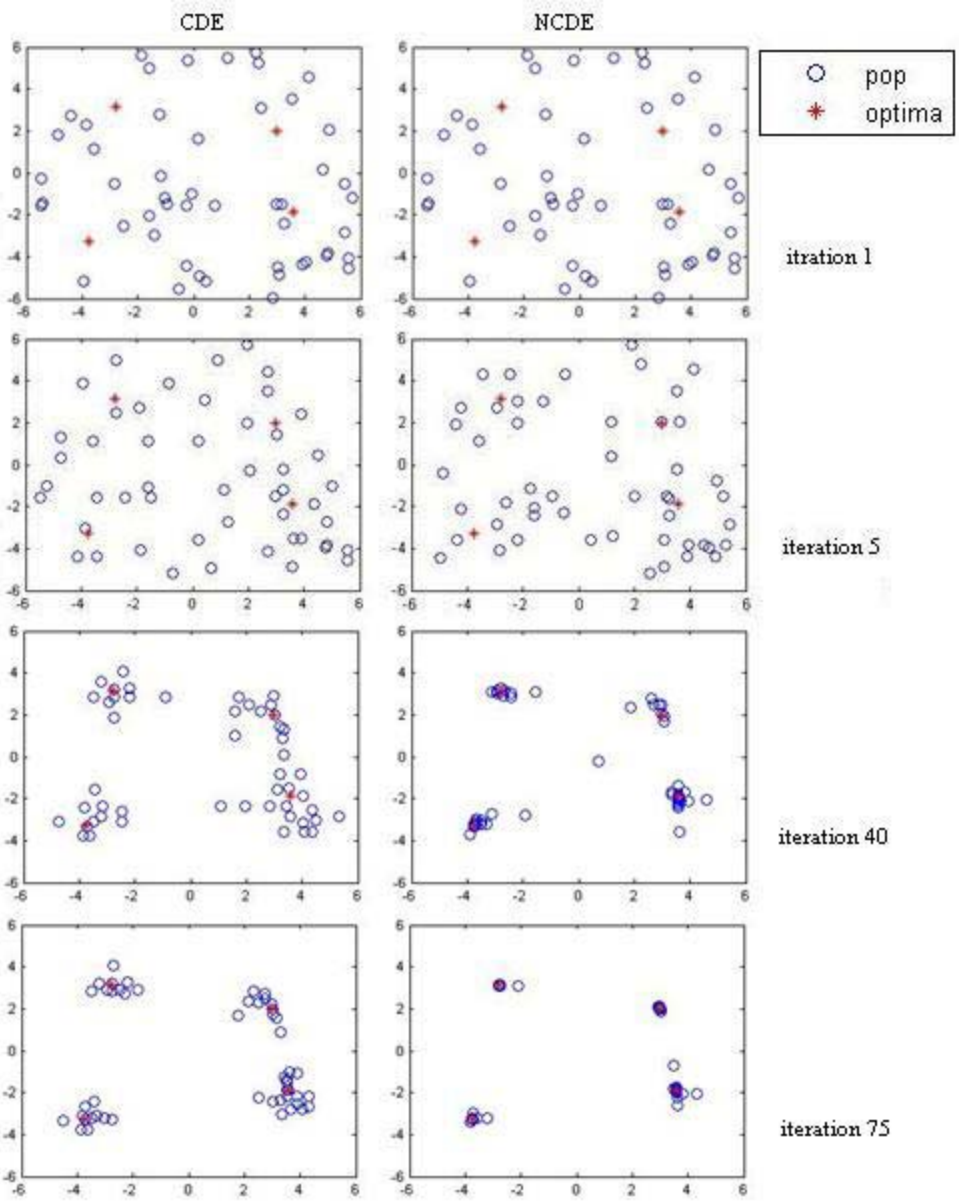


Fig. 3-4-7 Niching behavior of CDE and NCDE on E1-F8 (with identical initial solutions)

Experiment Two

In this section, the test functions/results reported in [156] are used and compared with the proposed neighborhood based DE algorithms. The results for the two criterions (peak accuracy/distance accuracy) are shown in Tables 3-4-17 and 3-4-18. From Table 3-4-17, we can see that the proposed algorithm performs the best except *E2-F8*. However, the superior performance of DFS on *E2-F8* is because of the peak accuracy error as explained in Section IV-C on performance measures.

The table of distance accuracy can truly reflect the performance of different algorithms where the neighborhood mutation based DEs rank on top for all test functions. Note that NSDE performs very well, if the target is to locate multiple global optima only. However, the performance decreases greatly if both local and global optima need to be found. This is because of the selection strategy used in SDE and NSDE. The selection method may discard some of the local peaks during the evolution process.

Table 3-4-17 The results of peak accuracy

	NCDE	NShDE	NSDE	TSC2	CDE	TSC	NCMA-ES	SCGA	DFS
E2-F1	0.10 (1)	0.14 (2)	6.72 (5)	1.84 (4)	1.59 (3)	7.7 (6)	8.89 (7)	18.59 (8)	20.93 (9)
E2-F2	2.75E-04 (1)	0.14 (2)	5.14 (6)	2.91 (3)	3.3 (4)	6.18 (7)	3.9 (5)	6.37 (8)	7.27 (9)
E2-F3	3.01E-33 (2)	6.85E-30 (3)	6.56E-47 (1)	1.81E-07 (4)	4.48E-04 (9)	4.90E-06 (8)	3.92E-06 (6)	2.86E-07 (5)	4.17E-06 (7)
E2-F4	0 (1)	0.0028 (3)	0 (1)	1.63 (8)	0.11 (6)	1.73 (9)	0.19 (7)	0.01 (4)	0.02 (5)
E2-F5	51.10 (2)	179.56 (5)	10.92 (1)	369.93 (7)	134.64 (3)	934.45 (8)	1840 (9)	317.24 (6)	164.85 (4)
E2-F6	1.95E-04 (1)	0.21 (2)	5.8 (7)	2.77 (4)	1.78 (3)	2.99 (5)	5.12 (6)	7.06 (9)	6.89 (8)
E2-F7	1.70E-04 (2)	3.51E-04 (3)	1.58E-05 (1)	0.02 (5)	0.1 (6)	1.79 (8)	1.96 (9)	0.73 (7)	3.42E-04 (4)
E2-F8	4.99 (3)	6.09 (4)	2.96 (2)	727.9 (6)	115.4 (7)	1628.46 (9)	52.6 (5)	1381.05 (8)	0.11 (1)
E2-F9	9.73E-04 (2)	1.10E-03 (3)	7.60E-14 (1)	0.85 (9)	0.24 (8)	0.23 (7)	0.01 (6)	0.003 (4.5)	0.003 (4.5)
E2-F10	4.39E-05 (1)	4.39E-05 (1)	0.02 (6)	0.009 (4)	0.006 (3)	0.01 (5)	0.07 (7)	0.48 (9)	0.37 (8)
E2-F11	3.18E-05 (1)	3.18E-05 (1)	3.18E-05 (1)	0.1 (6)	0.002 (4)	0.005 (5)	0.64 (7)	0.76 (8)	0.92 (9)
E2-F12	1.69E-04 (1)	4.26E-04 (2)	0.40 (5)	0.32 (4)	0.1 (3)	1.68 (6)	1.89 (7)	4.28 (8)	4.3 (9)
E2-F13	1.37E-06 (1)	9.20E-04 (3)	1.37E-06 (1)	0.28 (5)	0.12 (4)	0.89 (6)	1.02 (7)	2.53 (9)	2.11 (8)
Total Rank	**19**	**34**	**38**	69	63	89	88	93.5	85.5

Table 3-4-18 The results of distance accuracy

	NCDE	NShDE	NSDE	TSC2	CDE	TSC	NCMA-ES	SCGA	DFS
E2-F1	0.02 (1)	0.037 (2)	1.55 (5)	0.79 (4)	0.41 (3)	3.26 (6)	3.87 (7)	11.56 (9)	11.52 (8)
E2-F2	4.80E-03 (1)	0.10 (2)	2.23 (5)	2.09 (4)	1.99 (3)	6.18 (7)	3.19 (6)	7.02 (9)	6.22 (8)
E2-F3	2.63E-17 (2)	6.36E-016 (3)	1.87E-24 (1)	9.32E-05 (5)	5.25E-03 (9)	9.08E-05 (4)	5.84E-04 (7)	1.65E-04 (6)	8.12E-04 (8)
E2-F4	8.00E-05 (2)	7.72E-04 (3)	0 (1)	0.05 (7)	0.03 (6)	0.07 (8)	0.14 (9)	0.01 (4)	0.02 (5)
E2-F5	0.02 (2)	0.0304 (3)	6.30E-03 (1)	0.49 (5)	0.07 (4)	1.07 (7)	0.71 (6)	2.51 (8)	3.05 (9)
E2-F6	3.30E-03 (1)	0.03 (2)	0.28 (4)	0.54 (5)	0.22 (3)	3.94 (7)	1.48 (6)	5.31 (9)	4.55 (8)
E2-F7	0.01 (2)	0.02 (3)	1.80E-03 (1)	0.45 (5)	0.21 (4)	5.48 (8)	4.56 (7)	6.04 (9)	3.63 (6)
E2-F8	0.16 (2)	0.18 (3)	0.03 (1)	33.2 (7)	3.12 (4)	59.2 (8)	31.0 (6)	22.1 (5)	88.2 (9)
E2-F9	3.42E-04 (2)	3.76E-04 (3)	2.70E-14 (1)	0.21 (9)	0.05 (7)	0.06 (8)	2.96E-03 (6)	9.78E-04 (5)	8.18E-04 (4)
E2-F10	1.00E-05 (1)	3.20E-05 (2)	0.05 (6)	0.02 (4)	0.01 (3)	0.03 (5)	0.15 (7)	0.92 (9)	0.72 (8)
E2-F11	1.32E-06 (2)	6.80E-04 (3)	1.28E-09 (1)	0.21 (6)	9.00E-03 (4)	0.01 (5)	1.42 (7)	1.55 (8)	1.87 (9)
E2-F12	6.81E-04 (1)	1.00E-03 (2)	0.43 (5)	0.34 (4)	0.12 (3)	1.77 (6)	1.99 (7)	4.32 (9)	4.29 (8)
E2-F13	4.93E-10 (2)	6.93E-04 (3)	1.05E-16 (1)	0.95 (5)	036 (4)	7 (7)	5.42 (6)	7.68 (9)	7.36 (8)
Total Rank	**21**	**34**	**33**	65	57	86	87	99	98

3.4.4 Conclusion

DE has been widely used as an efficient optimization algorithm. Although different niching techniques have been integrated with DE, niching DE algorithms' performance is unsatisfactory on multi-modal optimization problems. This section proposed a neighborhood-based mutation and integrated it with various niching DE

algorithms to solve multi-modal optimization problems. Neighborhood mutation is able to restrict the production of offspring within a local area or the same niche as the parents. This method ensures that the algorithm converges faster with a high accuracy. We have demonstrated that the neighborhood mutation is able to induce stable niching behavior. The neighborhood based DE algorithm is able to locate multiple global optima and maintain them. The results of experimental studies suggest that the proposed algorithms can provide a better and more consistent performance than numerous state-of-the-art multi-modal optimization algorithms on a large number of test problems. Out of 29 problems, the proposed method improved almost all of them except a few test instances that both proposed and the original algorithms are able to solve perfectly.

3.5 Summary

In this chapter, DE-based niching algorithms are considered and three variants are proposed. Firstly, a dynamic grouping crowding DE (DECDE) is introduced which involves three sub-populations using different sets of control parameters. Secondly, a modified species-based DE in which a self-adaptive radius is presented to overcome the difficulty of selection of a radius value. Lastly, a neighborhood-based DE which limits mutation within a specific neighborhood is proposed. Experiments show that the proposed algorithms result in improvement of the performances.

Chapter 4

Particle Swarm Optimizer Based Niching Algorithms for Multi-modal Optimization

This chapter introduces some existing PSO based niching algorithms for multi-modal optimization. The proposed PSO based niching algorithms are also presented in this chapter. Note that section 4.2 proposed a new PSO based niching algorithm while section 4.3 introduced a local search technique which focused on improving the existing PSO based niching algorithms.

4.1 Existing PSO Based Niching Algorithms

For years, PSO has remained a favorite choice of the researchers working on multimodal optimization problems. To prevent convergence in local optima or locating a single optimal solution [72], [155], various techniques that commonly referred to as “niching” methods are incorporated into the classical PSO. These PSO variants such as Niche PSO [165] and SPSO [166] are able to promote and maintain formation of multiple stable subpopulations within a single population. In this section, some of the commonly used PSO based niching algorithms are introduced.

4.1.1 Niching PSO

In [165] Brits et al. introduced the niching technique that extends the unimodal particle swarm optimizer to efficiently locate multiple optimal solutions in multi-modal problems. Multiple subswarms are grown from an initial particle swarm by monitoring the fitness of individual particles. In niching PSO, the main swarm is trained using the *cognition only* model [55]. Equation 4-1-1 shows that in this model, only a conscience factor, in the form of a *personal best*, is considered when updating particle positions. Therefore no social information, in the form of a *global*

best solution, such as in the *gbest* and *lbest* algorithms, will influence position updates.

$$v_{i,j}(t+1) = wv_{i,j}(t) + c_1 r_{1,j}(t)(y_{i,j}(t) - x_{i,j}(t)) \qquad (4-1-1)$$

The steps of niching PSO are listed in Table 4-1-1.

Table 4-1-1 The steps for niching PSO

Step 1	Initialization of main particle swarm.
Step 2	Main swarm particles are trained by using the *cognition only* model.
Step 3	Update the fitness value of each main particle swarm.
Step 4	For every subswarm: 1. Subswarm particles are trained for one iteration using GCPSO algorithm [167]. 2. Update each particle's fitness value. 3. Update swarm radius.
Step 5	Merge subswarm if possible.
Step 6	Subswarms are allowed to combine any particle that has been moved into it from the main swarm.
Step 7	If any particle found in the main swarm that fulfil the partitioning criterion create a new subswarm with it and its nearest neighbour.
Step 8	Go to step 2 and continue till termination criterion meet.

4.1.2 Fitness Euclidean-distance ratio PSO (FERPSO)

FERPSO is proposed as an extension of FDR-PSO [73] [168] (Fitness-Distance-Ratio based PSO that was introduced in section 2.4) to solve multi-modal problems. In FERPSO, instead of global best the neighborhood best to each solution is used to lead the particles. The solutions move towards its personal best as well as its "fittest-and-closest" neighbors *(nbest)*, which are identified by the FER values. The *nbest* for *ith* particle is selected as the neighborhood personal best with the largest Fitness-Euclidean distance Ratio (FER):

$$FER_{(j,i)} = \alpha \cdot \frac{f(p_j) - f(p_i)}{\left\| p_j - p_i \right\|} \qquad (4-1-2)$$

where $\alpha = \frac{\|s\|}{f(p_g) - f(p_w)}$ is a scaling factor. p_w is the worst-fit particle in the current population. p_j and p_i are the *personal best* of the *jth* and *ith* particle respectively. $\|s\|$ is the size of the search space, which is estimated by its diagonal distance $\sqrt{\sum_{k=1}^{d} (x_k^u - x_k^l)^2}$ (where x_k^u and x_k^l are the upper and lower bounds of the *kth* dimension of the search space).

The velocity update equation in FERPSO is rewritten as:

$$V_i^d = \omega * V_i^d + c_1 * rand1_i^d * (pbest_i^d - X_i^d) + c_2 * rand2_i^d * (nbest^d - x_i^d) \qquad (4-1-3)$$

4.1.3 Speciation-based PSO (SPSO)

Speciation-based PSO (SPSO) was first proposed by Li [166] in 2004. In SPSO, different subpopulations are formed as species which is identified by the dominant particles known as the species seeds. To identify the species seeds and determine the size of species, a niching parameter known as radius must be specified by the user. The procedure for determining species and the species seeds was adopted from [158]. Each species and its corresponding species seed form a separate subpopulation that can be run as a separated PSO itself. Similar to FERPSO, SPSO replaces the global best by species best or species seed and all the particles in the same species at each iteration step share the same neighborhood best. Over successive iterations multiple global optima can be found in parallel.

4.1.4 Ring Topology PSO

Recently Li [154] proposed a *lbest* PSO with ring topology for niching. In ring topology PSO, each particle interacts only with its immediate neighbors. The implementation of ring topology PSO is shown in Table 4-1-2.

Table 4-1-2 The pseudocode of a *lbest* PSO using a ring topology

Step 1	Randomly generate an initial population
	Repeat
Step2	For $i=1$ to population_size do
	If $fit(\vec{x}_i) > fit(\vec{p}_i)$
	Then $\vec{p}_i \leftarrow \vec{x}_i$
	Endif
	Endfor
Step 3	For $i=1$ to population size do
	$\vec{p}_{n,i} \leftarrow neighborhoodBest(\vec{p}_{i-1}, \vec{p}_i, \vec{p}_{i+1})$
	Endfor
Step 4	For $i=1$ to population size do
	Apply a standard *lbest*
	Endfor
	Until termination criterion is met

4.1.5 Other algorithms

Besides the above mentioned PSO niching algorithms, various other approaches are also proposed in the literature. Susana et al. introduced a non-uniform mutation operator along with PSO [159] that has been proved to greatly improve PSO's performance for multimodal function optimization. Parsopoulos et al. [160] used the objective function "stretching" as a sequential PSO niching technique. Nickabadi et al. proposed a Dynamic Niching PSO (DNPSO) [161] technique where instead of determining the radius of the subswarm, the number of particles belonging to each niche is restricted.

4.2 Distance Based Locally Informed Particle Swarm Optimization for Multi-modal Optimization

Niching particle swarm optimization (PSO) has been widely used in the evolutionary computation as an important technique to solve multi-modal optimization problems. However, many PSO niching algorithms are difficult to be applied on practical problems because of their poor local search ability and requirement of prior knowledge to specify certain niching parameters. This work addresses these issues by proposing a distance based locally informed particle swarm optimization (LIPS), which eliminates the need to specify any niching parameter and enhance the fine search ability of PSO. Instead of using global best, LIPS uses several local bests to lead the particle. LIPS can operate as a stable niching algorithm by using the information provided by its neighborhoods. The neighborhoods are measured in Euclidean distance. The algorithms were tested on 29 multi-modal benchmark test functions. The experimental results suggested that the proposed technique was able to provide superior and more consistent performance over some existing niching algorithms on the test functions.

4.2.1 Locally Informed PSO (LIPS)

In 2004, Mendes et al. [76] introduced a fully informed particle swarm (FIPS) to solve single global optimization problems. The classical particle swarm algorithm works by continuously searching in a region that is defined by each particle's best previous success, the best success of one neighborhood particle, the particle's current position and its previous velocity [76]. The particle uses only one neighborhood best information to bias its search direction. However, there is no guarantee that the chosen neighborhood best will always lead to better solutions than other neighborhood's best. Important information that contained in neighborhood particles may be neglected through overemphasis on the single best neighbor, which may lead to poor local search or slower convergence.

Different from the canonical PSO, FIPS makes use of the information from all other particles around it, which is conceptually more concise and promises to perform better than the traditional particle swarm algorithm. In the fully informed particle swarm, all neighbors are a source of influence in leading the particles to fly. Therefore, how to select the neighbors determines how diverse the influence will be and how accurate the algorithm will be. In [76], FIPS was integrated with five different PSO social networks and showed the results were very promising in solving single global peak optimization problem. The five topologies were known as all, ring, four clusters, pyramid and square. The behavior of each particle was affected by its neighborhood identified by the topology used, which was able to make the whole population converge fast and more accurate.

Although FIPS is proven to be very effective in solving single objective global optimization problems, it is not suitable for multi-modal optimization due to the topology based neighborhood selection method. As can be easily revealed, all the particles and their neighborhoods are likely to converge to one point for single modal or single global peak optimization. However, for multi-modal case, multiple peaks need to be located simultaneously and these peaks can be far from each other. Thus the neighborhoods are likely from different areas or different niches in other words, if the topology neighborhood selection is used. Take ring topology as an example, the particle only interacts with its immediate members on its left and right. The possibility for these members from different niches is very high as the initial population is randomly generated. If the neighborhoods are not from the same niche or targeting on the same peak, it is almost impossible for the niching algorithm to converge and locate any of the peaks. Motivated by this observation, a distance based locally informed particle swarm optimization (LIPS) is presented in this work. Similar to FIPS, besides using its personal best, LIPS adopts the local information from its nearest neighborhood (measured in Euclidean distance) to lead the particles. In this way, LIPS can form different stable niches which can converge to different global peaks. The velocity update uses the equation listed below while the position update remains unchanged:

$$V_i^d \leftarrow \omega \times (V_i^d + \varphi(P_m^d - X_i^d)) \quad (4\text{-}2\text{-}1)$$

where φ is a constant equal to 4.1, $P_i = \frac{\sum_{j=1}^{nsize} (\varphi \cdot nbest_j) / nsize}{\varphi}$. $nbest_j$ is the *jth* nearest neighborhood to *ith* particle's *pbest*. *nsize* is the neighborhood size which will be discussed in the experimental result section. In this work *nsize* is dynamically increased from 2 to 5 over the function evaluations. There are two main advantages of LIPS to solve multi-modal optimization problems which are listed as below:

- Benefit of FIPS velocity update equation to ensure good usage of all neighborhood information especially at the late stage of the search process which leads to fast convergence and high accuracy.
- Euclidean distance based neighborhood selection to ensure the neighborhoods are from the same niches which increase the algorithm's local search and fine tuning ability.

With these two advantages, LIPS can easily find most of the peaks and maintain them until the end of function evaluations for multi-modal optimization. The details of LIPS algorithm are shown in Table 4-2-1.

Table 4-2-1. Steps of LIPS

Step 1	Randomly generate the initial solutions.
Step 2	Evaluate the initial solutions and initialize the *pbest*
	For *i=1* to *NP* (population size)
Step 2	Indentify the nearest (measured by parameter space Euclidean distance) *nsize* number neighborhood best to *ith* particle's *pbest*.
Step 3	Update the particles velocity using equations (4-2-1).
Step 4	Update the position of particle.
Step 5	Evaluate the newly generated particle.
Step 6	Update the *pbest* for the *ith* particle.

	Endfor
Step 7	Stop if a termination criterion is satisfied. Otherwise go to Step 2.

4.2.2 Experimental setup

In this work, ten different multi-modal algorithms (listed below) were examined in the experiments and the performance measures were success rate and average number of optima located. All the performances are measured over 25 runs.

1. LIPS: Locally Informed PSO
2. *r*2pso [154]: A *lbest* PSO with a ring topology, each member interacts with only its immediate member to its right.
3. *r*3pso [154]: A *lbest* PSO with a ring topology, each member interacts with its immediate member on its left and right.
4. *r*2pso-*lhc* [154]: The same as *r*2pso, but with no overlapping neighborhoods.
5. *r*3pso-*lhc* [154]: The same as *r*3pso, but with no overlapping neighborhoods.
6. SPSO [154]: Speciation-based PSO.
7. FERPSO [154]: Fitness-Euclidean distance ratio PSO.
8. SDE [146]: Speciation-based DE.
9. CDE [145][144]: The original crowding DE.
10. SCMA-ES[155]: Self-adaptive niching CMA-ES

- Test function

  In the experiment, 29 benchmark multi-modal test functions were used. These functions could be divided into two parts. The first 14 functions (AF1-AF15) were commonly used classical multi-modal test functions while the other 15 were composition multi-modal functions (Appendix B: BF1-BF15). All functions were considered for maximization.

- Population size and maximal number of evaluations

  The level of accuracy, niching radius, population size and maximal number of function evaluations are shown in Table 4-2-2.

Table 4-2-2. Test function settings

Function no.	ε	r	Population	No. of
AF1	0.05	0.5	50	10000
AF2	0.05	0.5	50	10000
AF3	0.05	0.5	50	10000
AF4	0.000001	0.01	50	10000
AF5	0.000001	0.01	50	10000
AF6	0.000001	0.01	50	10000
AF7	0.000001	0.01	50	10000
AF8	0.0005	0.5	50	10000
AF9	0.000001	0.5	50	10000
AF10	0.00001	0.5	50	10000
AF11	0.05	0.5	250	100000
AF12	0.0001	0.2	100	20000
AF13	0.001	0.2	500	200000
AF14	0.001	0.2	1000	400000

4.2.3 Experimental results

This section presents the experimental results and the analyses of experiments. All algorithms were run until all known peaks were found or the maximum number of function evaluations was exhausted. The effect of the parameter (neighborhood size) of LIPS is also discussed in this section.

- Success rate

  The success rates for all the algorithms on test function part one are recorded and presented in Table 4-2-3. The ranks of each algorithm are shown in the bracket while the total rank is listed in the last row of the table. As can been seen from this table, FIPS can get a much higher success rate than other niching algorithms on most of these 15 test functions. It also ranks the best if compare on the overall performance (total rank). The better

performance is due to the better fine search that generated by the local informed particles. For test function part two, the functions are much more complex than the functions in part 1. Among these test problems, no algorithm is able to generate a nonzero success rate. Therefore, the average number of optima found is used as the sole criterion and presented in the next section.

Table 4-2-3. Success rate (test function part one)

Test Func.	LIPS	r2pso	r3pso	r2psolhc	r3psolhc	SPOS	FERPSO	SDE	CDE	SACMA-ES
AF1	1.00 (1)	0.76 (6)	0.84 (5)	0.56 (9)	0.60 (8)	0.48 (10)	0.72 (7)	1.00 (1)	1.00 (1)	1.00 (1)
AF2	1.00 (1)	0.88 (7)	0.96 (6)	0.44 (9)	0.56 (8)	0.44 (9)	1.00 (1)	1.00 (1)	1.00 (1)	1.00 (1)
AF3	1.00 (1)	0.08 (6)	0.08 (6)	0.04 (9)	0.08 (6)	0.04 (9)	0.20 (5)	1.00 (1)	1.00 (1)	1.00 (1)
AF4	1.00 (1)	0.92 (3)	0.88 (5)	1.00 (1)	0.92 (3)	0.88 (5)	0.84 (7)	0.72 (8)	0.28 (9)	0.00 (10)
AF5	0.96 (2)	0.00 (8)	0.00 (8)	0.64 (3)	0.04 (5)	1.00 (1)	0.00 (8)	0.00 (8)	0.48 (4)	0.00 (8)
AF6	1.00 (1)	0.88 (6)	0.72 (7)	0.92 (3)	0.92 (3)	0.92 (3)	1.00 (1)	0.60 (8)	0.28 (9)	0.00 (10)
AF7	1.00 (1)	1.00 (1)	1.00 (1)	1.00 (1)	1.00 (1)	1.00 (1)	1.00 (1)	1.00 (1)	1.00 (1)	0.96 (10)
AF8	1.00 (1)	0.28 (5)	0.24 (7)	0.28 (5)	0.24 (7)	0.00 (9)	0.72 (2)	0.72 (2)	0.00 (9)	0.44 (4)
AF9	1.00 (1)	0.56 (6)	0.60 (5)	0.56 (6)	0.52 (8)	0.00 (9)	0.96 (4)	1.00 (1)	0.00 (9)	1.00 (1)
AF10	1.00 (1)	0.60 (5)	0.52 (6)	0.84 (3)	0.76 (4)	0.92 (2)	0.00 (7)	0.00 (7)	0.00 (7)	0.00 (7)
AF11	0.84 (1)	0.04 (6)	0.04 (6)	0.04 (6)	0.20 (4)	0.00 (9)	0.52 (3)	0.00 (9)	0.72 (2)	0.00 (9)
AF12	0.80 (1)	0.68 (3)	0.56 (5)	0.52 (7)	0.48 (8)	0.72 (2)	0.60 (4)	0.56 (5)	0.48 (8)	0.00 (10)
AF13	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.08 (1)	0.00 (2)
AF14	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)
Total rank	16	63	67	65	67	72	52	55	63	75

- Average number of optima found

  The average number of optima found is another important criterion for comparing different niching algorithms. The results for test functions part one and two are shown in Table 4-2-4 and 4-2-5, respectively. The mean value is highlighted in boldface. As can be revealed by the results, LIPS performs the best for both parts of the test functions. In order to determine the statistical significance of the advantage of LIPS, *t*-test is applied and shown in the last row of each test functions. The numerical values 1, 0 represent that other methods are statistically inferior to, equal to the proposed algorithm. The last three rows of both tables summarize how many cases that LIPS performs better, similar or worse than other algorithms. From the results, we can observe that LIPS always performs better or equal when compared with other niching methods on all test functions.

Table 4-2-4 Average number of optima found (test function part one)

Test Func.		LIPS	r2pso	r3pso	r2psolhc	r3psolhc	SPOS	FERPSO	SDE	CDE	SACMA-ES
F1	Worst	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00
	Best	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Mean	**1.00**	**0.76**	**0.84**	**0.56**	**0.60**	**0.48**	**0.72**	**1.00**	**1.00**	**1.00**
	Std	0.00	0.46	0.37	0.51	0.50	0.51	0.46	0.00	0.00	0.00
	t-test	-	1	1	1	1	1	1	0	0	0
F2	Worst	1.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00	1.00
	Best	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Mean	**1.00**	**0.88**	**0.96**	**0.44**	**0.56**	**0.44**	**1.00**	**1.00**	**1.00**	**1.00**
	Std	0.00	0.33	0.20	0.51	0.51	0.51	0.00	0.00	0.00	0.00
	t-test	-	1	1	1	1	1	0	0	0	0
F3	Worst	2.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	2.00	2.00
	Best	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00

	Mean	**2.00**	**0.48**	**0.60**	**0.48**	**0.60**	**0.24**	**0.80**	**1.96**	**2.00**	**2.00**
	Std	0.00	0.65	0.65	0.59	0.65	0.52	0.76	0.20	0.00	0.00
	t-test	-	1	1	1	1	1	1	1	0	0
	Worst	5.00	4.00	4.00	5.00	4.00	4.00	4.00	4.00	1.00	0.00
	Best	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	1.00
F4	Mean	**5.00**	**4.92**	**4.88**	**5.00**	**4.92**	**4.88**	**4.84**	**4.72**	**3.84**	**0.04**
	Std	0.00	0.28	0.33	0.00	0.28	0.33	0.37	0.46	1.03	0.20
	t-test	-	0	0	0	0	0	0	1	1	1
	Worst	4.00	1.00	1.00	3.00	1.00	5.00	1.00	1.00	2.00	1.00
	Best	5.00	1.00	1.00	5.00	5.00	5.00	1.00	3.00	5.00	1.00
F5	Mean	**4.96**	**1.00**	**1.00**	**4.52**	**2.80**	**5.00**	**1.00**	**1.52**	**4.28**	**1.00**
	Std	0.20	0.00	0.00	0.71	1.12	0.00	0.00	0.59	0.84	0.00
	t-test	-	1	1	1	1	0	1	1	1	1
	Worst	5.00	4.00	4.00	4.00	3.00	4.00	5.00	4.00	2.00	0.00
	Best	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	0.00
F6	Mean	**5.00**	**4.88**	**4.72**	**4.92**	**4.88**	**4.92**	**5.00**	**4.60**	**3.96**	**0.00**
	Std	0.00	0.33	0.46	0.28	0.44	0.28	0.00	0.50	0.89	0.00
	t-test	-	0	1	0	0	0	0	1	1	1
	Worst	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
	Best	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
F7	Mean	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**0.96**
	Std	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.20
	t-test	-	0	0	0	0	0	0	0	0	0
	Worst	4.00	1.00	1.00	0.00	1.00	0.00	2.00	3.00	0.00	3.00
	Best	4.00	4.00	4.00	4.00	4.00	2.00	4.00	4.00	1.00	4.00
F8	Mean	**4.00**	**2.92**	**2.76**	**3.00**	**3.12**	**0.84**	**3.68**	**3.72**	**0.32**	**3.40**
	Std	0.00	0.86	0.88	0.91	0.67	0.62	0.56	0.46	0.48	0.51
	t-test	-	1	1	1	1	1	1	1	1	1
	Worst	2.00	0.00	0.00	1.00	0.00	0.00	1.00	2.00	0.00	2.00
	Best	2.00	2.00	2.00	2.00	2.00	1.00	2.00	2.00	1.00	2.00
F9	Mean	**2.00**	**1.44**	**1.56**	**1.56**	**1.48**	**0.08**	**1.96**	**2.00**	**0.04**	**2.00**
	Std	0.00	0.71	0.58	0.51	0.59	0.28	0.20	0.00	0.20	0.00
	t-test	-	1	1	1	1	1	0	0	1	0
	Worst	25.00	23.00	22.00	24.00	23.00	24.00	3.00	1.00	7.00	0.00
	Best	25.00	25.00	25.00	25.00	25.00	25.00	7.00	2.00	20.00	4.00
F10	Mean	**25.00**	**24.44**	**24.32**	**24.84**	**24.60**	**24.92**	**5.16**	**1.32**	**12.52**	**0.88**
	Std	0.00	0.77	0.85	0.37	0.76	0.28	1.28	0.48	2.58	0.83
	t-test	-	1	1	0	1	0	1	1	1	1
	Worst	17.00	12.00	13.00	12.00	13.00	5.00	15.00	9.00	16.00	0.00
	Best	18.00	18.00	18.00	18.00	18.00	13.00	18.00	17.00	18.00	5.00
F11	Mean	**17.84**	**15.16**	**15.56**	**15.08**	**16.16**	**8.52**	**17.40**	**12.36**	**17.68**	**2.16**
	Std	0.37	1.57	1.23	1.32	1.34	2.38	0.76	1.91	0.56	1.21
	t-test	-	1	1	1	1	1	0	1	0	1
F12	Worst	4.00	4.00	3.00	4.00	4.00	4.00	4.00	3.00	5.00	0.00

	Best	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	4.00
	Mean	**5.60**	**5.52**	**5.16**	**5.36**	**5.28**	**5.60**	**5.36**	**4.88**	**5.56**	**1.52**
	Std	0.84	0.77	1.03	0.76	0.79	0.71	0.86	1.13	0.51	1.08
	t-test	-	0	1	0	0	0	0	1	0	1
	Worst	21.00	16.00	17.00	18.00	19.00	20.00	18.00	20.00	31.00	0.00
	Best	30.00	28.00	26.00	27.00	29.00	30.00	28.00	27.00	36.00	6.00
F13	Mean	**26.10**	**21.76**	**22.20**	**22.52**	**23.08**	**25.68**	**23.60**	**22.84**	**33.84**	**1.40**
	Std	2.65	2.95	2.25	2.12	2.86	2.72	2,63	1.89	1.34	1.84
	t-test	-	1	0	0	0	0	0	0	-1	0
	Worst	59.00	30.00	35.00	35.00	31.00	56.00	60.00	42.00	146	0.00
	Best	88.00	54.00	56.00	53.00	51.00	82.00	82.00	60.00	160	1.00
F14	Mean	**69.76**	**40.56**	**45.36**	**42.24**	**43.32**	**70.12**	**68.64**	**50.56**	**152.24**	**0.04**
	Std	6.53	5.33	5.46	4.27	5.89	6.27	6.12	4.07	4.18	0.20
	t-test	-	1	1	1	1	0	0	1	-1	1
t-test summary	Better	-	10	11	8	9	6	5	9	6	8
	Similar	-	4	3	6	5	8	9	5	6	6
	Worst	-	0	0	0	0	0	0	0	2	0

Table 4-2-5 Average number of optima found (test function part two)

Test Func.		LIPS	R2PSO	R3PSO	R2PSOLHC	R3PSOLHC	SPSO	FERPSO	SDE	CDE	SACMA-ES
	Worst	3.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00	2.00
	Best	4.00	0.00	0.00	0.00	0.00	0.00	2.00	2.00	0.00	2.00
F16	Mean	**3.70**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**1.08**	**1.79**	**0.00**	**2.00**
	Std	0.48	0.00	0.00	0.00	0.00	0.00	0.28	0.43	0.00	0.00
	t-test	-	1	1	1	1	1	1	1	1	1
	Worst	2.00	0.00	0.00	0.00	0.00	0.00	2.00	1.00	1.00	1.00
	Best	3.00	0.00	0.00	0.00	0.00	0.00	2.00	2.00	2.00	2.00
F17	Mean	**2.10**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**2.00**	**1.20**	**1.20**	**1.90**
	Std	0.32	0.00	0.00	0.00	0.00	0.00	0.00	0.53	0.42	0.32
	t-test	-	1	1	1	1	1	0	1	1	0
	Worst	4.00	0.00	0.00	0.00	0.00	0.00	2.00	1.00	0.00	2.00
	Best	4.00	0.00	0.00	0.00	0.00	0.00	3.00	2.00	1.00	3.00
F18	Mean	**4.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**2.50**	**1.50**	**0.70**	**2.70**
	Std	0.00	0.00	0.00	0.00	0.00	0.00	0.53	0.53	0.48	0.48
	t-test	-	1	1	1	1	1	1	1	1	1
	Worst	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Best	3.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
F19	Mean	**1.90**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.20**
	Std	0.88	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.42
	t-test	-	1	1	1	1	1	1	1	1	1
F20	Worst	2.00	0.00	0.00	0.00	0.00	0.00	2.00	1.00	1.00	1.00

	Best	3.00	0.00	0.00	0.00	0.00	0.00	2.00	2.00	2.00	2.00
	Mean	**2.20**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**2.00**	**1.30**	**1.10**	**1.90**
	Std	0.42	0.00	0.00	0.00	0.00	0.00	0.00	0.48	0.32	0.32
	t-test	-	1	1	1	1	1	0	1	1	0
	Worst	3.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00	2.00
	Best	4.00	0.00	0.00	0.00	0.00	0.00	2.00	2.00	0.00	3.00
F21	Mean	**3.70**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**1.20**	**1.40**	**0.00**	**2.60**
	Std	0.48	0.00	0.00	0.00	0.00	0.00	0.42	0.52	0.00	0.52
	t-test	-	1	1	1	1	1	1	1	1	1
	Worst	4.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
	Best	4.00	0.00	0.00	0.00	0.00	0.00	1.00	2.00	0.00	1.00
F22	Mean	**4.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.50**	**1.00**	**0.00**	**1.00**
	Std	0.00	0.00	0.00	0.00	0.00	0.00	0.53	0.82	0.00	0.00
	t-test	-	1	1	1	1	1	1	1	1	1
	Worst	3.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00	2.00
	Best	4.00	0.00	0.00	0.00	0.00	0.00	2.00	2.00	0.00	3.00
F23	Mean	**3.20**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**1.50**	**1.40**	**0.00**	**2.30**
	Std	0.42	0.00	0.00	0.00	0.00	0.00	0.53	0.52	0.00	0.48
	t-test	-	1	1	1	1	1	1	1	1	1
	Worst	3.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00	1.00
	Best	4.00	0.00	0.00	0.00	0.00	0.00	2.00	3.00	0.00	2.00
F24	Mean	**3.50**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**1.50**	**1.80**	**0.00**	**1.70**
	Std	0.53	0.00	0.00	0.00	0.00	0.00	0.53	0.63	0.00	0.48
	t-test	-	1	1	1	1	1	1	1	1	1
	Worst	2.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00	1.00
	Best	3.00	0.00	0.00	0.00	0.00	0.00	2.00	2.00	0.00	2.00
F25	Mean	**2.10**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**1.10**	**1.10**	**0.00**	**1.40**
	Std	0.32	0.00	0.00	0.00	0.00	0.00	0.32	0.32	0.00	0.52
	t-test	-	1	1	1	1	1	1	1	1	1
	Worst	4.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
	Best	5.00	0.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00	1.00
F26	Mean	**4.10**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**1.30**	**0.00**	**0.70**
	Std	0.32	0.00	0.00	0.00	0.00	0.00	0.00	0.48	0.00	0.48
	t-test	-	1	1	1	1	1	1	1	1	1
	Worst	3.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00	1.00
	Best	4.00	0.00	0.00	0.00	0.00	0.00	2.00	3.00	0.00	2.00
F27	Mean	**3.50**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**1.60**	**1.60**	**0.00**	**1.70**
	Std	0.53	0.00	0.00	0.00	0.00	0.00	0.52	0.84	0.00	0.48
	t-test	-	1	1	1	1	1	1	1	1	1
	Worst	4.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Best	4.00	0.00	0.00	0.00	0.00	0.00	1.00	2.00	0.00	3.00
F28	Mean	**4.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.30**	**0.90**	**0.00**	**1.40**
	Std	0.00	0.00	0.00	0.00	0.00	0.00	0.48	0.57	0.00	1.17
	t-test	-	1	1	1	1	1	1	1	1	1

F29	Worst	1.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00	1.00
	Best	1.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00	1.00
	Mean	**1.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**1.00**	**1.00**	**0.00**	**1.00**
	Std	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	t-test	-	1	1	1	1	1	0	0	1	0
F30	Worst	4.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00	1.00
	Best	5.00	0.00	0.00	0.00	0.00	0.00	2.00	2.00	0.00	3.00
	Mean	**4.80**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**1.20**	**1.60**	**0.00**	**2.00**
	Std	0.42	0.00	0.00	0.00	0.00	0.00	0.42	0.52	0.00	0.67
	t-test	-	1	1	1	1	1	1	1	1	1
t-test summary	Better	-	15	15	15	15	15	12	14	15	12
	Similar	-	0	0	0	0	0	3	1	0	3
	Worst	-	0	0	0	0	0	0	0	0	0

- The effect of neighborhood size

  The performance of the proposed LIPS is dependent on the neighborhood size. A smaller neighborhood size will generate a better diversity for the population while a larger neighborhood size is good for fast convergence. Table 4-2-6 shows the performances on test function 11 and 16 with the neighborhood varied from 2 to 5. For test function 11, the average number of optima obtained drops as the neighborhood size increases. This is because the number of global optima for *F11* is high. Although a large neighborhood size increases the convergence, it decreases the diversity and misses some of the peaks. On the other hand, *F16* is composite function and finding one peak is already challenging. Therefore, the convergence is more important than diversity. How to choose the neighborhood size depends on the users' need. If the target is diversity, a smaller neighborhood size should be use otherwise a larger neighborhood size should be selected. In this paper, a dynamic neighborhood size is used. The neighborhood size is linearly increased from 2 to 5 over the function evaluations. In order to show the effect of neighborhood size more clearly. The convergence graph and distance to optima over function evaluations on *F16* are shown in Fig. 4-2-1 and 4-2-2. From the convergence graph we can see that a larger neighborhood size generates a higher average fitness value of the

population while a smaller neighborhood size generates small summation of distance to all peaks.

Table 4-2-6 Neighborhood size effect (*F11* and *F16*)

Test		Nsize=2	Nsize=3	Nsize=4	Nsize=5
F11	Worst	17	16	15	13
	Best	18	18	18	18
	Mean	**17.88**	**17.32**	**16.76**	**15.88**
	Std	0.3317	0.6272	1.0116	1.3329
F16	Worst	0	2	4	4
	Best	1	4	5	5
	Mean	**0.2**	**3**	**4.3**	**4.7**
	Std	0.42	0.67	0.57	0.48

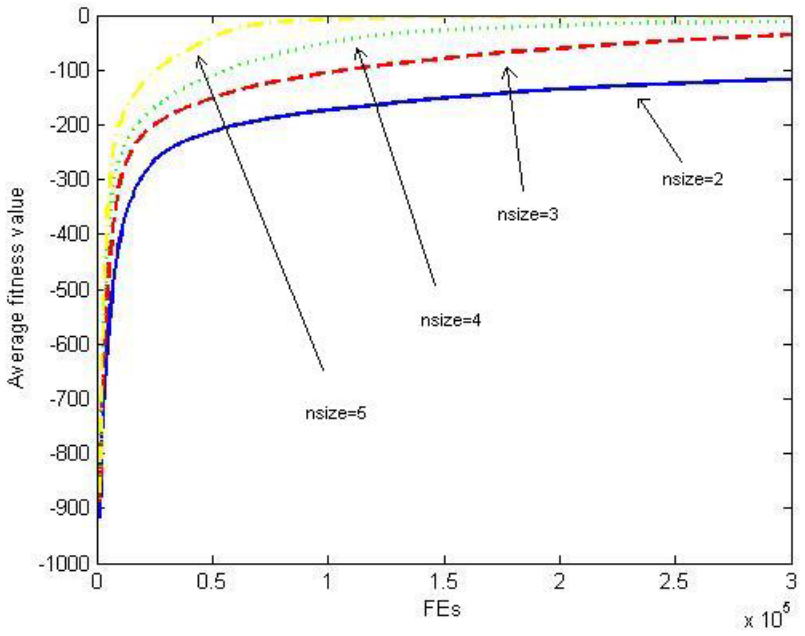


Fig. 4-2-1 The convergence graph for different neighborhood size (*F16*)

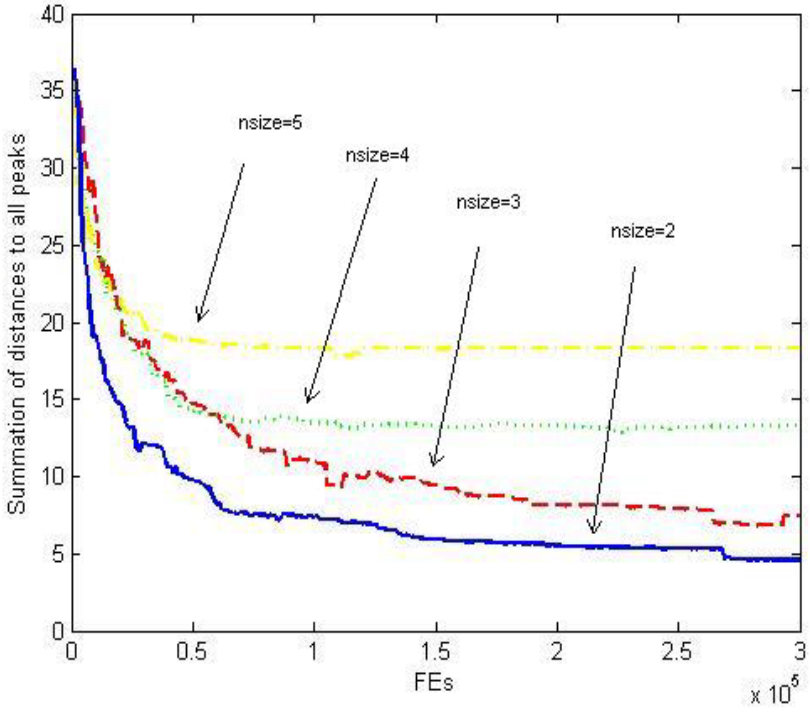


Fig. 4-2-2 The summation of distance to optima (*F16*)

- Maintaining found optima

Maintaining found optima is crucial for an effective and stable multi-modal optimization algorithm. LIPS is able to find and maintain the found optima until the end of the run. This is because the proposed algorithm uses neighborhood information to execute local search. Once the niche is formed around one global peak, the algorithm will continue to search better solutions within the niche. Only when a better solution is found within the niche, it will replace the current personal best which can maintain the found peak until the end of the run. Fig 4-2-3 shows the niching behavior of LIPS on *F1*. From the figure we can see that LIPS is able to develop stable niches around the global peaks.

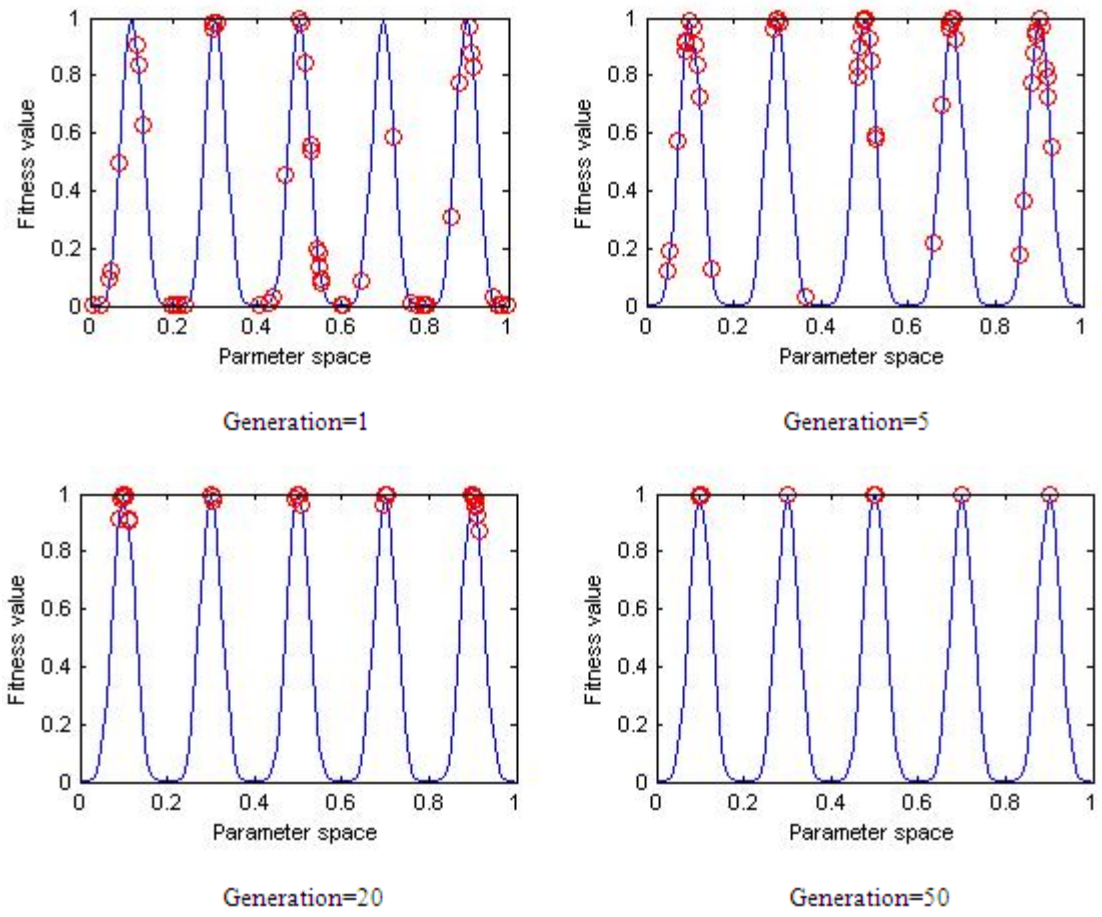


Fig. 4-2-3 Distribution of population over function evaluations (*F1*)

4.2.4 Conclusion

In this section, a locally informed PSO (LIPS) niching algorithm is proposed to solve multi-modal problems. LIPS makes use of the neighborhood information to realize niching behavior. In addition, LIPS eliminates the need to specify any niching parameters. With the neighborhood information the algorithm is enhanced with fine local search ability. We have demonstrated that the LIPS is able to effectively solve multi-modal problems. Experimental results also show that the proposed LIPS outperforms the classical niching algorithms compared in this work on the benchmark functions.

4.3 Niching Particle Swarm Optimization with Local Search Technique for Multi-modal Optimization

This work introduces a local search technique and is incorporated into some existing PSO based multi-modal optimization algorithms to enhance their fine search ability. The algorithms were tested on 14 commonly used multi-modal optimization problems. The experimental results suggested that the proposed technique not only increased the probability of finding both global and local optima but also reduced the average number of function evaluations.

4.3.1 Niching PSO with local search

In PSO, the current particles are responsible for searching of new points. However, the current position could be far from the personal best position. Due to this reason, the fine-searching ability of most niching PSO algorithms is not good. If the required accuracy is high, it will be difficult for the niching PSO to locate the required number of global/local optima. To overcome this problem, a local search method based on *pbest* member mutation is introduced.

In each iteration of PSO (after generating new solutions from the current position), the local search method makes use of *pbest* to produce another *NP* (population size) new solutions. The steps of generating new solutions by local search method are shown in Table 4-3-1 (assuming maximization problem).

Table 4-3-1 Steps of local search

Step 1	Update the current *pbest* by the original niching PSO (FERPSO or SPSO)
	For $i=1$ to NP (population size)
Step 2	Find $pbest_j$ (the nearest *pbest* member to $pbest_i$).
Step 3	If $pbest_j$ (objective value) > = $pbest_i$
	$nbest_1 = pbest_j$
	$nbest_2 = pbest_i$

	Else
	$nbest_1 = pbest_i$
	$nbest_2 = pbest_j$
	Endif
Step 4	Temp= $pbest_i + c_1 * rand * (nbest_1 - nbest_2)$
Step 5	Reset Temp within the bounds, if it exceeds the bounds and evaluate Temp.
Step 6	If Temp (objective value) > $pbest_i$
	$pbest_i$ = Temp
	Endif
	Endfor

With the local search technique, more solutions are produced around the personal best to enhance fine-searching ability of the original niching PSO algorithms. This local search method is incorporated into FERPSO, SPSO as well as ring topology PSO and tested on a set of benchmark functions. The results are presented in next section.

4.3.2 Experimental setup

14 multi-modal benchmark functions (AF1-AF14) were used to test the proposed algorithm. All the performances are measured over 25 runs. The population size, maximum number of function evaluations and level of accuracy are set to be the same for all tested algorithms and the details are listed in Table 4-3-2. Parameter settings for PSO are as below [76]:

$$C_1 = 2.05 \qquad C_2 = 2.05 \quad \omega = 0.729843788$$

Table 4-3-2 Population and no. of function evaluations

Function no.	Population size	No. of function evaluations	Level of accuracy
AF1-AF3	50	10000	0.05
AF4-AF7	50	10000	0.000001
AF8	50	10000	0.0005

AF9	50	10000	0.000001
AF10	500	100000	0.00001
AF11	250	100000	0.05
AF12	100	20000	0.0001
AF13	500	200000	0.001
AF14	1000	400000	0.001

Twelve different multi-modal algorithms are compared in the experiments:

1. FERPSO: Fitness-Euclidean distance ratio PSO.
2. FERPSOLS: Fitness-Euclidean distance ratio PSO with local search.
3. SPSO: Speciation-based PSO.
4. SPSOLS: Speciation-based PSO with local search.
5. *r*2pso: A *lbest* PSO with a ring topology, each member interacts with only its immediate member to its right.
6. *r*2psoLS: *r*2pso with local search
7. *r*2pso-*lhc*: The same as r2pso, but with no overlapping neighbourhoods, hence acting as multiple local hill climbers, more suitable for finding global as well as local optima.
8. *r*2pso-*lhc*LS: *r*2pso-*lhc* with local search
9. *r*3pso: A *lbest* PSO with a ring topology, each member interacts with its immediate member on its left and right.
10. *r*3psoLS: *r*3pso with local search
11. *r*3pso-*lhc*: The same as r3pso, but with no overlapping neighbourhoods. Basically multiple PSOs search in parallel, like local hill climbers. This variant is more appropriate if the goal of optimization is to find global optima as well as local optima.
12. *r*3pso-*lhc*LS: *r*3pso-*lhc* with local search

4.3.3 Experimental results

- Success rate

  The success rate for all the six PSO variants are recorded in pairs and presented in Table 4-3-3. The boldface indicates the better performance. As

can be seen from the results, the niching algorithms with the local search technique perform better in most of the cases. Especially on test function *AF1-AF4* and *AF12*, the success rate is improved by a big margin. For more challenging problems (*AF13* and *AF14*), although the success rate does not change, the average number of peaks found is increased (Table 4-3-4). The better performance is due to the better fine search yielded by the proposed local search technique.

Table 4-3-3 The success rate

Test Funct	FERPSO	FERPSO	SPSO	SPSOLS	R2pso	R2psoLS	R3pso	R3psoLS	R2psolhc	R2psolhc	R3psolhc	R3psolhc
AF1	0.64	**1**	0.44	**1**	0.72	**1**	0.56	**1**	0.48	**1**	0.52	**1**
AF2	0.88	**1**	0.72	**1**	0.56	**1**	0.32	**1**	0.52	**1**	0.76	**1**
AF3	0	**0.72**	0	**0.88**	0	**0.52**	0	**0.36**	0.08	**0.8**	0	**0.48**
AF4	0.84	**1**	0.88	**1**	0.92	**1**	0.88	**1**	**1**	**1**	0.92	**0.96**
AF5	0	**0.24**	**1**	**1**	0	**0.16**	0	**0.04**	0.48	**0.64**	0.04	**0.16**
AF6	**1**	**1**	0.92	**1**	0.88	**0.96**	0.72	**1**	0.92	**1**	0.92	**1**
AF7	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
AF8	0.72	**0.76**	0	**0.52**	0.28	**0.76**	0.24	**0.64**	0.28	**0.68**	0.24	**0.72**
AF9	0.96	**1**	0	**0.12**	0.56	**1**	0.6	**0.96**	0.56	**1**	0.52	**0.96**
AF10	0	**0.84**	0.92	**1**	0.6	**1**	0.52	**1**	0.84	**1**	0.76	**1**
AF11	**0.52**	0.48	0	**0.12**	0.04	**0.64**	0.04	**0.52**	0.04	**0.72**	0.2	**0.6**
AF12	0.6	**1**	0.72	**0.96**	0.68	**0.92**	0.56	**0.96**	0.52	**0.96**	0.48	**0.92**
AF13	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
AF14	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

- Average number of optima found

  The number of optima found for each test function is listed in Table 4-3-4. The mean value is highlight in boldface. As can be revealed by the results, the PSO niching variants with local search technique perform either better or comparable to their original versions.

Table 4-3-4 The number of optima found

Test Funct		FERPSO	FERPSOL	SPSO	SPSOLS	R2pso	R2psoLS	R3pso	R3psoLS	R2psolhc	R2psolhcL	R3psolhc	R3psolhcL
AF1	Min	0	2	1	2	1	2	0	2	1	2	1	2
	Max	2	2	2	2	2	2	2	2	2	2	2	2
	Mean	**1.48**	**2**	**1.44**	**2**	**1.72**	**2**	**1.48**	**2**	**1.48**	**2**	**1.52**	**2**
	Std	0.770	0	0.507	0	0.458	0	0.653	0	0.510	0	0.510	0
AF2	Min	1	2	1	2	0	2	0	2	1	2	1	2
	Max	2	2	2	2	2	2	2	2	2	2	2	2
	Mean	**1.88**	**2**	**1.72**	**2**	**1.36**	**2**	**1.24**	**2**	**1.52**	**2**	**1.76**	**2**
	Std	0.332	0	0.458	0	0.810	0	0.597	0	0.510	0	0.436	0
AF3	Min	0	3	2	4	0	2	0	2	1	3	0	2
	Max	2	5	4	5	2	5	1	5	5	5	3	5
	Mean	**0.64**	**4.48**	**3.08**	**4.88**	**0.8**	**3.96**	**0.4**	**3.24**	**3**	**4.72**	**2.16**	**3.92**
	Std	0.7	0.872	0.493	0.332	0.707	1.207	0.5	1.393	0.957	0.614	0.851	1.077
AF4	Min	4	5	4	5	4	5	4	5	5	5	4	4
	Max	5	5	5	5	5	5	5	5	5	5	5	5
	Mean	**4.84**	**5**	**4.88**	**5**	**4.92**	**5**	**4.88**	**5**	**5**	**5**	**4.92**	**4.96**
	Std	0.374	0	0.332	0	0.277	0	0.332	0	0	0	0.277	0.200
AF5	Min	1	1	5	5	1	1	1	1	3	3	1	1
	Max	1	5	5	5	1	5	1	5	5	5	5	5
	Mean	**1**	**2.16**	**5**	**5**	**1**	**1.98**	**1**	**1.24**	**4.32**	**4.52**	**2.8**	**2.8**
	Std	0	1.675	0	0	0	1.428	0	0.831	0.748	0.714	1.118	1.294
AF6	Min	5	5	4	5	4	4	4	5	4	5	3	5
	Max	5	5	5	5	5	5	5	5	5	5	5	5
	Mean	**5**	**5**	**4.92**	**5**	**4.88**	**4.96**	**4.72**	**5**	**4.92**	**5**	**4.88**	**5**
	Std	0	0	0.277	0	0.332	0.2	0.458	0	0.277	0	0.440	0
AF7	Min	1	1	1	1	1	1	1	1	1	1	1	1
	Max	1	1	1	1	1	1	1	1	1	1	1	1
	Mean	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

	Std	0	0	0	0	0	0	0	0	0	0	0	0
	Min	3	2	0	1	1	3	1	2	0	3	1	3
	Max	4	4	2	4	4	4	4	4	4	4	4	4
AF8	Mean	**3.68**	**3.68**	**0.84**	**3.32**	**2.92**	**3.76**	**2.76**	**3.6**	**3**	**3.68**	**3.12**	**3.72**
	Std	0.476	0.557	0.625	0.852	0.862	0.436	0.879	0.577	0.913	0.476	0.666	0.458
	Min	1	2	0	0	0	2	0	1	1	2	0	1
	Max	2	2	1	2	2	2	2	2	2	2	2	2
AF9	Mean	**1.96**	**2**	**0.08**	**0.76**	**1.44**	**2**	**1.56**	**1.96**	**1.56**	**2**	**1.48**	**1.96**
	Std	0.2	0	0.277	0.633	0.712	0	0.583	0.2	0.507	0	0.586	0.2
	Min	3	20	24	25	23	25	22	25	24	25	23	25
	Max	7	25	25	25	25	25	25	25	25	25	25	25
AF10	Mean	**5.16**	**24.44**	**24.92**	**25**	**24.44**	**25**	**24.32**	**25**	**24.84**	**25**	**24.6**	**25**
	Std	1.28	1.387	0.277	0	0.768	0	0.852	0	0.374	0	0.764	0
	Min	15	16	5	14	12	15	13	16	12	17	13	17
	Max	18	18	13	18	18	18	18	18	18	18	18	18
AF11	Mean	**17.4**	**17.24**	**8.52**	**16.56**	**15.16**	**17.56**	**15.56**	**17.36**	**15.08**	**17.72**	**16.16**	**17.6**
	Std	0.764	0.779	2.383	1.044	1.573	0.712	1.227	0.757	1.320	0.458	1.344	0.5
	Min	4	5	4	4	4	5	3	5	4	5	4	4
	Max	6	6	6	6	6	6	6	6	6	6	6	6
AF12	Mean	**5.36**	**5.96**	**5.6**	**5.88**	**5.52**	**5.92**	**5.16**	**5.96**	**5.36**	**5.96**	**5.28**	**5.88**
	Std	0.860	0.2	0.707	0.440	0.770	0.277	1.028	0.2	0.757	0.2	0.792	0.440
	Min	18	22	20	22	16	20	17	24	18	24	19	24
	Max	28	32	30	33	28	32	26	31	27	32	29	31
AF13	Mean	**23.6**	**27**	**25.68**	**27.24**	**21.76**	**26.64**	**22.2**	**26.92**	**22.52**	**27.28**	**23.08**	**27.44**
	Std	2.63	2.48	2.719	2.92	2.948	2.871	2.255	2.040	2.124	2.424	2.857	1.938
	Min	60	65	56	75	30	66	35	66	35	69	31	67
	Max	82	82	82	96	54	92	56	85	53	90	51	93
AF14	Mean	**68.64**	**73.16**	**70.12**	**84.36**	**40.56**	**77.20**	**45.36**	**76.28**	**42.24**	**78.40**	**43.32**	**77.99**
	Std	6.123	4.66	6.27	6.45	5.339	6.609	5.46	5.53	4.274	5.5	5.89	5.53

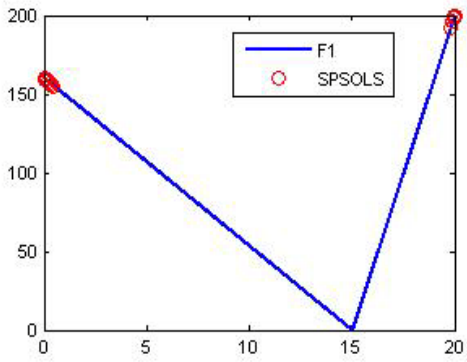


Fig. 4-3-1 The snapshot of AF1

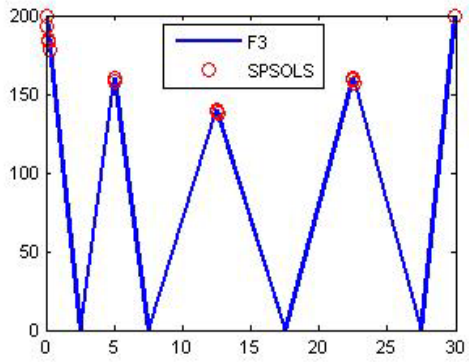


Fig. 4-3-2 The snapshot of AF3

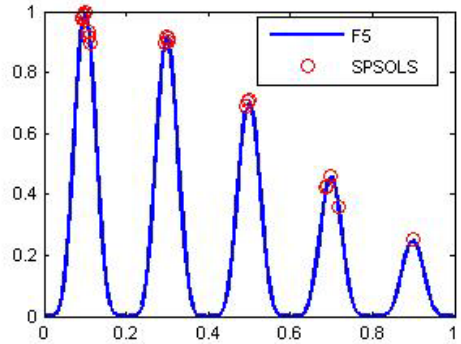


Fig. 4-3-3 The snapshot of AF5

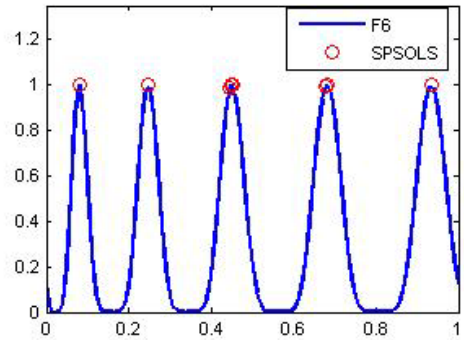


Fig. 4-3-4 The snapshot of AF6

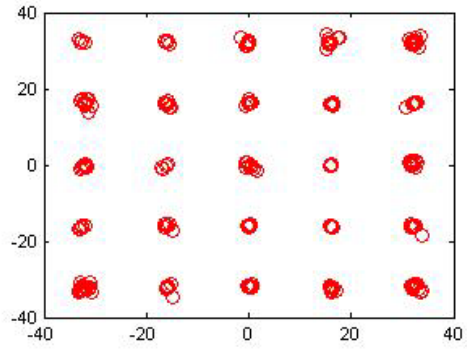


Fig. 4-3-5 The snapshot of AF10

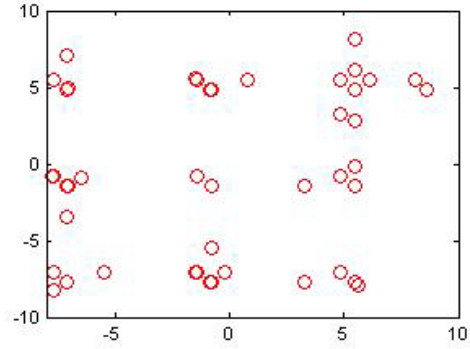


Fig. 4-3-6 The snapshot of AF11

4.3.4 Conclusion

In this section, a local search technique is proposed and combined with various PSO-based niching algorithms to solve multi-modal problems. The lack of fine searching ability of the original niching PSO curtails the ability to locate most local

or global optima. With the proposed local search method, niching PSO algorithms are enhanced with local search ability. Experimental results show that the niching algorithms integrated with the proposed local search procedure outperform the corresponding original niching PSO variants on the benchmark functions used.

4.4 Summary

In this chapter, PSO-based niching algorithms are considered and two variants are proposed. In the first algorithm, a neighborhood information usage technique is proposed which enhances the local search capability and eliminates the need to specify any niching parameters. For the second variant, a local search technique is incorporated into several existing PSO based niching algorithms to enhance their local search capability. Various experiments are undertaken to show the superior performances of the proposed algorithms.

Chapter 5

Evolutionary Algorithms for Constrained and Unconstrained Multi-objective Optimization

In this chapter, the proposed unconstrained and constrained multi-objective evolutionary algorithms are presented. For unconstrained MOEA, a summation of normalized objectives and diversified selection method is introduced to replace the commonly used non-domination sorting. While for constrained MOEA, an ensemble of constraints handling methods is proposed. Note that in this thesis, "*NP*" is population size, "*N*" is number of solution, and "*P*" is the percentage.

5.1 Multi-Objective Evolutionary Algorithms based on the Summation of Normalized Objectives and Diversified Selection

In literature, although some range-dependant ranking methods were proposed [141], [142], [143], [144], most multi-objective evolutionary algorithms (MOEAs) still use the concept of dominance in the search process to select the top solutions as parents in a purely elitist approach. However, as MOEAs are probabilistic search methods, some useful information may be wasted, if the dominated solutions are completely disregarded. In addition, the diversity may be lost during the early stages of the search process leading to a locally optimal or partial Pareto-front. Beside this, the non-domination sorting process is complex and time consuming. To overcome these problems, this work proposes multi-objective evolutionary algorithms based on Summation of Normalized Objective Values and Diversified **S**election (SNOV-DS). The performance of this algorithm was tested on a set of benchmark problems using multi-objective differential evolution (MODE) and multi-objective evolutionary programming (MOEP). With the proposed method, the performance metric has been improved significantly and the speed of the parent

selection process has also been increased when compared with the non-domination sorting.

5.1.1 Classical MOEP and MODE

During the last two decades, evolutionary computation techniques have been successfully used to solve a number of multi-objective optimization problems. Fig. 5-1-1 shows the flowchart of a general MOEA.

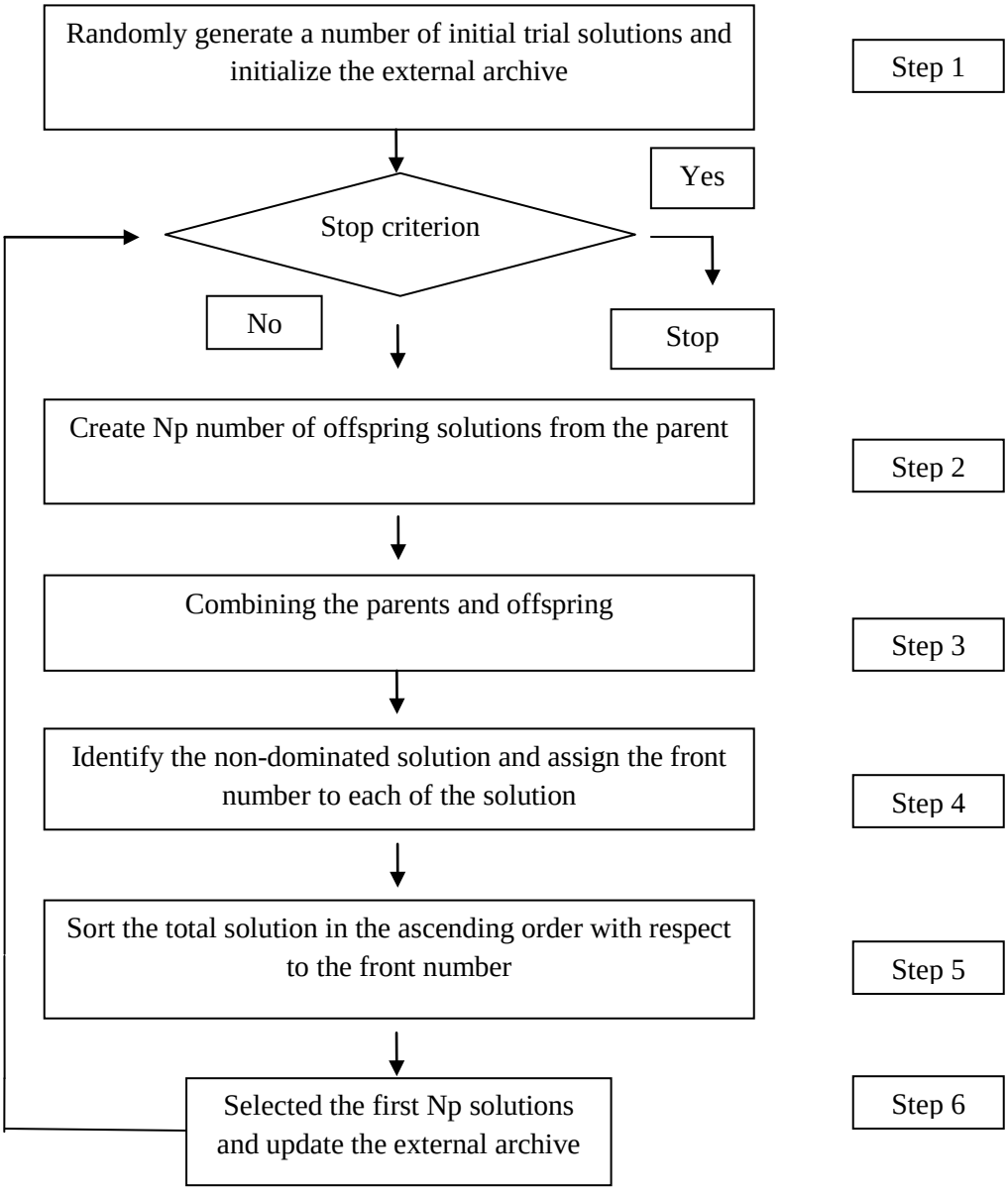


Fig. 5-1-1 Flowchart of a generic MOEA

- MOEP

  Evolutionary programming (EP) was first designed as an evolutionary approach to solve problems in artificial intelligence [5]. EP is a powerful global optimization technique. It starts with a randomly generated population and evolves towards better solutions over a number of iterations. There are two major stages in EP known as mutation and selection. Multi-objective Evolutionary programming (MOEP) is an extended work of single objective evolutionary programming to solve multi-objective optimization problems. The steps are the same as a general MOEA shown in Fig. 5-1-1.

- MODE

  Multi-objective differential evolution (MODE) is the implementation of DE to solve multi-objective optimization problems. The major difference is in the selection process. In single-objective optimization, the comparison and selection are based on objective function values. However, in MODE, the selection process is based on the non-domination concept whereby solutions with smaller front number will be selected as parents in the next iteration.

5.1.2 MOEA based on Summation of Normalized Objective Values and Diversified Selection (SNOV-DS)

In the proposed algorithm, steps 4 and 5 in Fig. 5-1-1 are replaced by Summation of Normalized Objective Values (SNOV) and Diversified Selection (DS). The details of SNOV and DS are presented as follows:

- Summation of normalized objective values (SNOV)

  Although many algorithms exist for non-dominated sorting of a population, the process is time consuming. The diversity may also be lost, if the selection is only based on the non-dominated elitist solutions. In the

literature, various mechanisms such as crowding density of solutions and gridding the objective space have been used to maintain the diversity of the solutions [169], [170]. However, the diversity maintenance mechanisms are not efficient and these mechanisms further increase the computational complexity. Motivated by these observations, a fast diversified selection procedure is proposed based on the normalized summation of objective values. Instead of using the non-dominated sorting, the SNOV-DS uses the summation of normalized objective values obtained as shown in Table 5-1-1.

Table 5-1-1 Obtaining the summation of normalized objective values

---

Step 1: For m = 1 to M (*number_of_objectives)*

Find the maximum f_{max} and minimum f_{min} objective values of the m^{th} objective and calculate the range of the m^{th} objective. Normalize the m^{th} objective values of all members using the Equation below:

$f_i'(x) = \frac{f_i(x) - f_{min}}{f_{max} - f_{min}}$ $\quad$ $f_m'(x) = \frac{f_m(x) - f_{min}}{f_{max} - f_{min}}$ $\quad$ where $f_m'(x)$ is the normalized m^{th} objective value.

End For

Step 2: For i = 1 to NP (*population_size)*

Sum all normalized objective values of the member to obtain a single value.

End For

- Diversified Selection

The challenge faced by the usage of summation of normalized objective values is the loss of diversity. To overcome this problem, the diversified selection method is used. During the selection process, the parents are chosen according to the summation of normalized objective values and diversity with respect to each objective. To maintain the diversity of the population, the solutions in the current population are divided into 2 sets,

preferential set and backup set. As the names indicate, the solutions inside the preferential set will be selected first for evolving. Solutions in the backup set will be selected based on the summation of normalized objective values to be evolved only if there are insufficient solutions in the preferential set. However, if the number of solutions inside the preferential set is greater than required (population) size, the required number of solutions will be randomly selected from the preferential set as parents for next generation. The algorithm for obtaining these sets is presented in Table 5-1-2.

Table 5-1-2 The algorithm for obtaining the diversified preferential and back-up solution sets

Step 1: For m = 1 to M (*number_of_objectives)*

(i) Evenly divide the range of the objective space into 100 bins

(ii) Scan/consider P percentage of the 100 bins (*i.e.* from bin 1 to P, P may be chosen as 80 or 90).

(iii) For each scanned/considered bin (if this bin is empty, just continue to next bin), the solution with the smallest summation of normalized objective values will be chosen to enter preferential set.

End For

Step 2: Accumulate the solutions excluded from the preferential set and store them in backup set.

In order to illustrate the diversified selection process clearly, an example is shown in Fig. 5-1-2 for a 2-objective minimization problem. There are 10 individuals (A-J) in the current population. For each objective, the range is divided into 10 bins. According to objective 1, we scan 80 percent of the bins (i.e. from bin 1 to 8). Note that the reason for not "scanning" the whole space is to remove the bad solutions. Since bins 1, 2 and 7 are empty, they will not be considered. For bin 3, the point with lowest summation of normalized objective values is G which will be put into the preferential set.

For bins 4, 5, 6 and 8, points B, D, A and I (solid circles) are respectively selected to be included in the preferential set. Similarly, points G, F, D, I, B, A and J are chosen to be in the preferential set according to objective 2 (dotted circles). Hence, the final preferential set contains solutions G, B, D, A, I, F and J, while the backup set contains E, C and H.

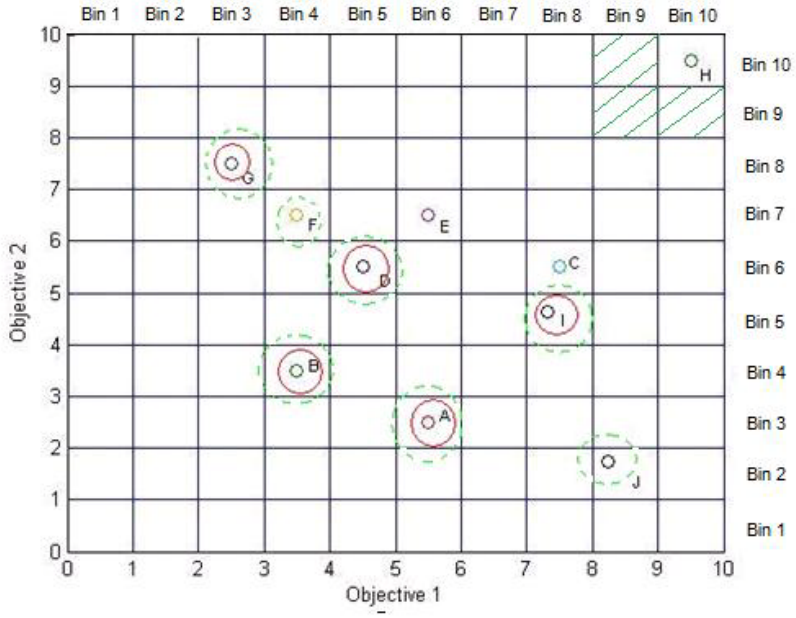


Fig. 5-1-2 Illustration of choosing preferential set

In Fig. 5-1-2, the shaded area shows the objective space included in the preferential set when *P=90* but excluded if *P=80*. When *P*=80, we scan from bin 1 to bin 8 of the objective space, while when *P=90*, we scan from bin 1 to bin 9. Any solution in the shaded area will not be included in the preferential set when *P=80*, but they may when *P=90*. Solution H will not be included in the preferential set for both *P*=80 and *P=90*. Note that the reason for not scanning the whole space is to remove the bad solutions (such as H).

- Discussion on Complexity

For the commonly used non-domination sorting, the complexity to obtain the overall non-dominated set is $O(MN^2)$ [1]. For the summation of normalized objective values method, it requires $O(N)$ comparisons to find

the maximum and minimum values and *O(N)* comparisons to identify the corresponding bins for each objectives. In total, the complexity of the method is *O(MN)*, where *M* is the number of objectives and *N* is the number of solutions. From the complexity calculations, we can observe that the complexity of the summation of normalized objective values method is in linear form while non-domination sorting is non-linear. This will reduce the CPU time of the SNOV-DS method compared to the non-domination sorting procedure.

5.1.3 Experiments Preparation

- Experimental setup

To test the SNOV-DS algorithm, 19 benchmark problems (Appendix C) are used which are introduced in [171]. Among these 19 problems, the first 7 are 2-objective problems and the next 6 are 3-objective while the last 6 are 5-objective problems. For the experiments, the maximal number of function evaluations is 500,000. The external archive sizes are 200, 300, 1600 for 2, 3, 5 objective problems respectively. The population sizes which are the same as the maximal number of solutions for computing the performance metric are set as 100, 150 800 for 2, 3, 5 objectives, respectively. The MODE parameters are $\mathrm{F} = 0.5, \mathrm{CR} = 0.1$.

- Performance measures

In order to compare the performance of the different algorithms quantitatively, a performance metric is needed. There are two goals in a multi-objective optimization: 1) convergence to the Pareto-optimal set and 2) diversity of solutions in the Pareto-optimal set. In this work, two different performance metrics are used (the indicators used in the congress on evolutionary computation CEC2007) [171]:

(1) R indicator (I_{R2}) [172]: $I_{R2} = \dfrac{\sum_{\lambda \in \Lambda} u^*(A, \lambda) - u^*(R, \lambda)}{|\Lambda|}$ where R is a reference set, u^* is the maximum value reached by the utility function u

with weight vector λ on an approximation set *A*, *i.e.*, $u^* = \max_{z \in A} u\ (z)$. We choose the augmented Tchebycheff function as the utility function [171] [172].

(2) Hypervolume difference to a reference set ($I_{\bar{H}}$) [172]: The hypervolume indicator I_H measures the hypervolume of the objective space that is weakly dominated by an approximation set *A*, and is to be maximized. Here we consider the hypervolume difference to a reference set *R*, and we will refer to this indicator as $I_{\bar{H}}$, which is defined as $I_{\bar{H}} = I_H(R) - I_H(A)$ where smaller values correspond to higher quality as opposed to the original hypervolume I_H .

Note that for both indicators, the value should be between -1 to 1 with the smaller being the better.

5.1.4 Experimental Results

- Performance Evaluation

Each test problem is run for 30 times. The performance metrics (*R* and *H* indicators) of MOEP with the SNOV-DS and MOEP with non-domination sorting (NDS) are presented in Table 5-1-3, while Table 5-1-4 presents the same for the MODE. The non-dominated sorting algorithms used in this simulation have the epsilon sorting and crowding distance techniques to increase the diversity. As evidenced from the Tables 5-1-3, 5-1-4, 5-1-5, 5-1-6 and Fig. 5-1-3 to Fig. 5-1-8, the performance of MOEAs with summation of normalized objective values method is either comparable or better than the performance of MOEAs with non-dominated sorting. Note that for indicators, the smaller the indicator value the better of the performance is.

In order to determine the statistical significance of the advantage of the summation of normalized objective values (SNOV-DS) method, *t*-test is applied on the *R and H* indicators. Statistically best results are highlighted

in boldface in Tables 5-1-3 and 5-1-4. Non-domination sorting method is compared with the proposed method. The numerical values -1, 0, 1 represent that the non-domination sorting method is statistically inferior to, equal to and superior to the SNOV-DS. From the results, we can observe that the SNOV-DS method always performs better or equal to non-domination sorting method with a faster speed. Superior performance of SNOV-DS is due to the stochastic nature of the diversified selection. Since the non-dominated sorting is a purely elitist method, it may lead to premature convergence due to loss of diversity. As we know, even in single objective optimization algorithm, pure elitist approaches lead to sub-optimal solutions when solving multi-modal optimization problems. Therefore, SNOV-DS is able to generate better quality solutions as well.

Table 5-1-3 *R and H* indicator values of the MOEP with the SNOV-DS and NDS

	R indicator					H indicator				
	MOEP(SNOV-DS)		MOEP(NDS)			MOEP(SNOV-DS)		MOEP(NDS)		
Problems	Mean	Std	Mean	Std	*h*-value	Mean	Std	Mean	Std	*h*-value
OKA2	-1.07E-03	0	-1.07E-03	0	**0**	-7.96E-04	9.13E-05	-7.51E-04	1.72E-04	**0**
SYMPART	8.25E-06	1.22E-05	1.05E-03	1.88E-06	**-1**	2.41E-05	3.58E-05	3.45E-04	6.46E-05	**-1**
S_ZDT1	-3.32E-04	3.04E-04	1.39E-02	3.35E-03	**-1**	5.54E-05	6.22E-04	3.86E-02	1.62E-02	**-1**
S_ZDT2	6.57E-04	1.93E-04	3.30E-02	8.80E-03	**-1**	2.09E-03	4.26E-04	5.18E-02	1.68E-02	**-1**
S_ZDT4	5.02E-03	7.85E-04	6.11E-02	1.31E-03	**-1**	1.55E-02	2.71E-03	1.84E-01	4.85E-03	**-1**
R_ZDT4	1.82E-03	1.40E-03	1.40E-02	3.08E-03	**-1**	5.88E-03	4.11E-03	4.37E-02	6.96E-03	**-1**
S_ZDT6	3.35E-02	4.40E-03	8.42E-02	3.00E-02	**-1**	3.35E-02	4.40E-03	2.04E-01	7.76E-02	**-1**
S_DTLZ2	6.57E-05	3.18E-05	1.03E-04	8.99E-06	**-1**	2.91E-03	4.04E-04	1.02E-02	2.70E-05	**-1**
R_DTLZ2	1.02E-04	2.73E-05	2.38E-04	6.89E-06	**-1**	2.37E-04	6.66E-05	3.25E-03	3.77E-04	**-1**

S_DTLZ3	8.47E-05	3.65E-05	2.27E-04	1.43E-05	**-1**	6.89E-04	9.45E-04	4.05E-03	1.93E-04	**-1**
WFG1	4.05E-02	7.37E-04	5.64E-02	3.07E-04	**-1**	2.27E-01	6.72E-03	2.89E-01	1.59E-03	**0**
WFG8	-2.89E-02	5.75E-05	2.09E-02	1.28E-02	**-1**	-1.74E-01	5.34E-04	1.16E-01	5.72E-02	**-1**
WFG9	-1.01E-02	1.02E-03	-6.16E-03	1.86E-03	**-1**	-3.97E-02	1.43E-02	-2.95E-02	1.56E-02	**-1**
S_DTLZ2	1.55E-05	2.35E-05	1.40E-05	2.61E-06	**0**	8.35E-05	9.92E-06	5.11E-05	7.36E-05	**0**
R_DTLZ2	3.20E-05	3.51E-06	3.07E-05	1.27E-05	**0**	-8.55E-05	1.18E-05	5.59E-05	1.30E-04	**-1**
S_DTLZ3	1.60E-05	3.38E-06	1.37E-05	2.27E-06	**0**	1.84E-05	4.14E-06	6.19E-05	2.72E-05	**-1**
WFG1	4.38E-02	3.96E-04	4.71E-02	9.55E-05	**-1**	5.03E-01	2.70E-03	5.30E-01	9.30E-04	**0**
WFG8	-1.16E-02	1.87E-04	1.23E-02	2.54E-03	**-1**	1.13E-01	1.86E-02	2.79E-01	1.10E-03	**-1**
WFG9	5.87E-04	1.80E-05	4.51E-03	8.92E-04	**-1**	-1.59E-02	1.25E-02	-1.56E-02	1.06E-02	**0**

Table 5-1-4 *R and H* indicator values of the MODE with the SNOV-DS and NDS

	R indicator					H indicator				
	MODE(SNOV-DS)		MODE(NDS)			MODE(SNOV-DS)		MODE(NDS)		
Problems	Mean	Std	Mean	Std	*h*-value	Mean	Std	Mean	Std	*h*-value
OKA2	-1.07E-03	0	-1.07E-03	0	**0**	-1.2E-03	2.77E-06	-1.2E-03	6.94E-06	**0**
SYMPART	1.15E-04	1.73E-05	9.00E-03	1.10E-02	**-1**	3.03E-06	1.87E-06	1.29E-02	2.48E-02	**-1**
S_ZDT1	-1.05E-03	8.66E-06	-1.05E-03	1.46E-07	**0**	-2.30E-03	1.05E-05	-2.00E-03	1.43E-04	**0**
S_ZDT2	-1.17E-04	7.34E-06	4.05E-04	5.59E-04	**-1**	4.05E-04	1.05E-04	1.20E-03	1.10E-03	**-1**
S_ZDT4	2.26E-06	3.86E-06	7.33E-03	6.71E-03	**-1**	2.93E-06	2.05E-06	1.74E-02	1.49E-02	**-1**
R_ZDT4	6.00E-04	3.03E-04	4.53E-03	1.23E-03	**-1**	2.10E-03	3.93E-04	1.38E-02	3.40E-03	**-1**

S_ZDT6	1.36E-04	2.86E-04	1.06E-01	1.61E-02	**-1**	2.59E-03	2.27E-04	2.59E-01	4.25E-002	**-1**
S_DTLZ2	2.01E-05	2.07E-05	2.27E-05	2.48E-05	**0**	9.33E-05	8.63E-05	1.89E-04	3.27E-05	**0**
R_DTLZ2	8.97E-05	4.06E-05	7.57E-05	2.00E-05	**0**	1.23E-05	4.38E-06	1.92E-05	1.69E-05	**-1**
S_DTLZ3	4.71E-07	6.36E-07	1.57E-06	1.89E-06	**-1**	3.54E-08	1.37E-08	2.61E-08	5.29E-08	**0**
WFG1	7.50E-03	3.90E-03	1.43E-02	3.08E-03	**-1**	3.11E-03	2.14E-02	4.71E-03	4.98E-03	**0**
WFG8	-2.92E-02	6.22E-05	-2.90E-02	9.30E-05	**-1**	-1.77E-01	3.65 E-04	-1.76E-01	5.48E-04	**0**
WFG9	-1.03E-02	9.78E-05	-1.01E-02	9.79E-04	**0**	-6.04E-02	4.15E-03	-5.63E-02	2.03E-04	**0**
S_DTLZ2	1.22E-05	5.28E-06	1.02E-05	2.30E-06	**0**	4.96E-09	2.87E-09	1.08E-06	1.03E-06	**-1**
R_DTLZ2	1.65E-05	1.08E-05	1.71E-05	2.24E-05	**-1**	-1.35E-04	1.67E-05	1.71E-05	2.23E-06	**-1**
S_DTLZ3	6.27E-07	4.12E-07	1.09E-06	3.59E-07	**-1**	2.23E-010	2.63E-01	7.19E-08	4.79E-08	**-1**
WFG1	3.40E-02	3.84E-04	3.89E-02	1.90E-04	**-1**	3.37E-02	4.44E-04	4.49E-01	1.96E-03	**-1**
WFG8	-1.24E-03	1.85E-05	-1.14E-02	1.43E-04	**-1**	-3.64E-01	1.8E-03	-3.44E-01	1.71E-03	**-1**
WFG9	5.31E-04	1.91E-05	5.85E-04	6.67E-05	**-1**	-1.38E-01	3.3E-03	-1.25E-01	2.81E-03	**-1**

The SNOV-DS is also compared with several other real parameter MOEAs. The results are shown in Tables 5-1-5 (a)-(c) and Tables 5-1-6 (a)-(c). As we can see in the right-most column in Table 5-1-5 (c) and 5-1-6 (c), the overall ranks of MODE and MOEP were improved drastically because of the SNOV-DS.

Table 5-1-5(a) The rank of mean *R* indicator on test problems 1-7. *M*=2

	1.OKA2	2.SYMPART	3.S_ZDT1	4.S_ZDT2	5.S_ZDT4	6.R_ZDT4	7.S_ZDT6	Rank
MODE(SNOV-DS)	-1.07E-032	1.15E-048	-1.05E-032	-1.17E-043	2.26E-063	6.00E-043	1.36E-044	25
MODE(NDS)	-1.07E-032	9.00E-0312	-1.05E-032	4.05E-044	7.33E-039	4.53E-038	1.06E-0112	49
MOEP(SNOV-DS)	-1.07E-032	8.25E-064	-3.32E-044	6.57E-045	5.02E-038	1.82E-036	3.35E-028	37
MOEP(ND	-1.07E-03	1.05E-03	1.39E-02	3.30E-02	6.11E-02	1.40E-02	8.42E-02	64

S)	2	10	11	10	11	11	9	
MOCLPSO [132]	4.71E-03 9	7.13E-03 11	4.88E-02 12	5.56E-02 11	7.42E-02 12	1.86E-02 12	9.79E-02 10	77
MO_PSO [133]	2.53E-03 11	1.64E-07 **1**	4.61E-03 10	6.50E-02 12	2.50E-02 10	6.56E-03 9	1.03E-01 11	64
NSGA2_SBX [138]	-1.06E-03 6	2.02E-05 7	2.09E-07 5	-2.06E-04 2	3.04E-08 2	2.13E-03 7	1.19E-02 5	34
NSGA2_PCX [139]	-1.64E-03 **1**	6.40E-05 6	7.98E-04 8	9.62E-04 6	5.35E-06 4	8.64E-04 5	1.86E-02 7	37
GDE3 [134]	-1.05E-03 7	1.56E-06 2	4.53E-06 7	3.21E-03 7	1.15E-05 5	6.72E-04 4	-1.08E-06 2	34
MOSaDE [135]	1.22E-01 12	2.53E-05 5	-5.19E-03 **1**	-8.27E-03 **1**	-1.58E-04 **1**	-7.14E-04 **1**	-8.96E-06 **1**	**22**
DEMOwSA [140]	-8.05E-04 8	4.19E-06 3	2.27E-06 6	1.44E-02 9	3.01E-03 7	3.66E-04 2	-1.02E-06 3	38
MTS [136]	6.53E-03 10	3.95E-04 9	1.00E-03 9	3.38E-03 8	1.21Ee04 6	6.65E-03 10	1.38E-02 6	58

Table 5-1-5(b) The rank of mean *R* indicator on test problems 8-13. *M*=3

	8. S_DTLZ2	9. R_DTLZ2	10. S_DTLZ3	11. WFG1	12. WFG8	13. WFG9	Rank
MODE (SNOV-DS)	2.01E-05 3	8.97E-05 5	4.71E-07 3	7.50E-03 2	-2.92E-02 **1**	-1.03E-02 2	**16**
MODE (NDS)	2.27E-05 4	7.57E-05 4	1.57E-06 6	1.43E-02 3	-2.90E-02 2	-1.01E-02 4	23
MOEP (SNOV-DS)	6.57E-05 6	1.02E-04 6	8.47E-05 9	4.05E-02 7	-2.89E-02 3	-1.01E-02 4	35
MOEP (NDS)	1.03E-04 11	2.38E-04 9	2.27E-04 10	5.64E-02 11	2.09E-02 12	-6.16E-03 10	63
MOCLPSO	9.98E-05 9	3.69E-04 11	2.74E-04 11	5.57E-02 10	-1.31E-02 8	-6.36E-03 9	58
MO_PSO	9.85E-05 8	3.07E-04 10	5.23E-04 12	7.41E-02 12	-1.68E-02 7	-1.04E-02 **1**	50
NSGA2_SBX	9.54E-05 7	2.14E-05 **1**	6.41E-08 **1**	2.25E-02 4	-1.24E-02 9	-1.02E-02 3	25
NSGA2_PCX	1.01E-04 10	3.69E-05 3	3.03E-05 8	5.21E-02 9	-2.70E-02 6	-9.42E-03 6	42
GDE3	6.05E-06 2	2.19E-05 2	3.50E-07 2	1.68E-03 **1**	-2.85E-02 4	-9.22E-03 7	18
MOSaDE	-8.47E-05 **1**	1.93E-04 8	4.83E-07 4	2.87E-02 5	-2.62E-04 11	3.62E-03 12	41
DEMOwSA	3.21E-05 5	6.17E-04 12	1.46E-05 5	3.78E-02 6	-2.75E-02 5	-7.82E-03 8	41
MTS	5.99E-04 12	1.83E-04 7	2.18E-05 7	4.75E-02 8	-8.70E-03 10	-3.51E-03 11	55

Table 5-1-5(c) The rank of mean *R* indicator on test problems 14-19. *M*=5

	14. S_DTLZ2	15. R_DTLZ2	16. S_DTLZ3	17. WFG1	18. WFG8	19. WFG9	Rank	**Overall Rank**
MODE (SNOV-DS)	1.22E-05 4	1.65E-05 2	6.27E-07 3	3.40E-02 4	-1.24E-02 **1**	5.31E-04 3	17	**58 (1)**
MODE (NDS)	1.02E-05 3	1.71E-05 3	1.09E-06 5	3.89E-02 6	-1.14E-02 4	5.85E-04 4	25	97 (4)
MOEP (SNOV-DS)	1.55E-05 7	3.20E-05 6	1.70E-05 9	4.38E-02 7	-1.16E-02 3	5.81E-04 5	37	109 (5)
MOEP (NDS)	1.40E-05 5	3.07E-05 5	1.77E-05 8	4.71E-02 11	1.23E-02 12	4.51E-03 11	52	179 (11)
MOCLPSO	7.74E-05 10	1.20E-04 11	9.95E-05 10	4.66E-02 10	1.89E-03 10	8.43E-04 6	57	193 (12)
MO_PSO	1.27E-04 12	1.03E-04 10	2.36E-04 11	5.31E-02 12	-3.40E-03 7	2.57E-03 9	61	175 (10)
NSGA2_SBX	2.81E-06 **1**	-1.18E-05 **1**	-1.47E-08 **1**	3.69E-02 5	-1.18E-02 2	-2.25E-03 **1**	**11**	70 (2)
NSGA2_PCX	1.25E-04 11	7.80E-05 9	3.91E-04 12	4.46E-02 8	-1.66E-03 8	-8.43E-04 2	50	129 (8)
GDE3	2.15E-05 8	3.31E-05 7	2.54E-07 2	4.61E-03 **1**	-1.14E-02 5	1.94E-03 7	30	82 (3)
MOSaDE	1.50E-05 6	7.74E-05 8	8.97E-06 7	1.97E-02 3	9.49E-03 11	7.24E-03 12	47	110 (6)
DEMOwSA	7.81E-06 2	1.34E-04 12	6.98E-06 6	4.59E-02 9	-3.96E-03 6	2.56E-03 8	43	122 (7)
MTS	7.51E-05 9	2.58E-05 4	6.59E-07 4	1.90E-02 2	-5.06E-06 9	3.58E-03 10	38	151 (9)

Table 5-1-6(a) The rank of mean *H* indicator on test problems 1-7. *M*=2

	1.OKA2	2.SYMPART	3.S_ZDT1	4.S_ZDT2	5.S_ZDT4	6.R_ZDT4	7.S_ZDT6	Rank
MODE(SNOV-DS)	-1.20E-03 2	3.03E-06 2	-2.30E-03 **1**	4.05E-04 **1**	2.93E-06 3	2.10E-03 3	2.59E-03 4	**16**
MODE(NDS)	-1.20E-03 2	1.29E-02 11	-2.00E-03 2	1.20E-03 2	1.74E-02 9	1.38E-02 8	2.59E-01 12	46
MOEP(SNOV-DS)	-7.96E-04 5	2.41E-05 5	5.54E-05 3	2.09E-03 4	1.55E-02 8	5.88E-03 6	3.35E-02 7	38
MOEP(NDS)	-7.51E-04 6	3.45E-04 9	3.86E-02 11	5.18E-02 10	1.84E-01 12	4.37E-02 11	2.04E-01 9	68
MOCLPSO	9.35E-03 8	1.77E-02 12	1.57E-01 12	9.51E-02 11	2.24E-01 11	5.76E-02 12	2.39E-01 10	76

MO_PSO	5.06E-02 11	4.02E-07 **1**	1.44E-02 9	1.29E-01 12	7.37E-02 10	1.97E-02 9	2.52E-01 11	63
NSGA2_SBX	-1.05E-03 4	6.30E-05 7	3.10E-04 6	1.21E-02 7	1.51E-06 2	6.52E-03 7	2.62E-02 5	38
NSGA2_PCX	8.56E-03 7	1.93E-04 8	1.69E-03 7	1.53E-03 3	3.47E-05 5	2.97E-03 5	4.13E-02 8	43
GDE3	-1.23E-03 **1**	4.65E-06 3	1.75E-04 5	4.03E-03 6	2.73E-05 4	2.27E-03 4	-2.31E-04 2	25
MOSaDE	4.01E-01 12	4.80E-05 6	1.85E-02 10	4.66E-02 9	-1.09E-03 **1**	-2.37E-03 **1**	-3.77E-03 **1**	40
DEMOwSA	1.79E-02 10	1.26E-05 4	1.45E-04 4	1.73E-02 8	9.31E-03 7	1.08E-03 2	-2.30E-04 3	38
MTS	1.53E-02 9	1.16E-03 10	2.12E-03 8	2.58E-03 5	3.66E-04 6	2.04E-02 10	3.06E-02 6	54

Table 5-1-6(b) The rank of mean *H* indicator on test problems 8-13. *M*=3

	8. S_DTLZ2	9. R_DTLZ2	10. S_DTLZ3	11. WFG1	12. WFG8	13. WFG9	Rank
MODE (SNOV-DS)	9.33E-05 3	1.23E-05 2	3.54E-08 5	3.11E-03 **1**	-1.77E-01 2	-6.04E-02 **1**	**14**
MODE (NDS)	1.89E-04 5	1.92E-05 3	2.61E-08 4	4.71E-03 2	-1.76E-01 3	-5.63E-02 4	21
MOEP (SNOV-DS)	2.91E-03 10	2.37E-04 5	6.89E-04 8	2.27E-01 6	-1.74E-01 5	-3.97E-02 9	43
MOEP (NDS)	1.02E-02 12	3.25E-03 9	4.05E-03 10	2.89E-01 10	1.16E-01 12	-2.95E-02 11	64
MOCLPSO	1.71E-03 8	9.07E-03 11	5.28E-03 11	2.86E-01 9	-8.67E-02 10	-3.37E-02 10	59
MO_PSO	6.48E-04 6	5.39E-03 10	1.67E-02 12	3.75E-01 11	-1.03E-01 8	-5.76E-02 3	50
NSGA2_SBX	1.29E-03 7	7.89E-04 6	1.48E-09 2	4.23E-01 12	-3.99E-01 **1**	-5.16E-02 6	34
NSGA2_PCX	1.79E-03 9	3.10E-05 4	1.43E-03 9	2.69E-01 8	-1.58E-01 7	-4.82E-02 8	45
GDE3	-6.83E-06 2	8.53E-06 **1**	2.21E-09 3	5.52E-03 3	-1.75E-01 4	-5.97E-02 2	15
MOSaDE	-2.10E-05 **1**	1.53E-03 8	-1.48E-07 **1**	1.68E-01 4	3.74E-02 11	5.07E-02 7	32
DEMOwSA	1.11E-04 4	1.35E-02 12	1.91E-06 6	1.96E-01 5	-1.71E-01 6	-5.40E-02 5	38
MTS	7.48E-03 11	9.61E-04 7	6.48E-05 7	2.60E-01 7	-5.14E-02 9	-1.35E-02 12	53

Table 5-1-6(c) The rank of mean *H* indicator on test problems 14-19. *M*=5

	14. S_DTLZ2	15. R_DTLZ2	16. S_DTLZ3	17. WFG1	18. WFG8	19. WFG9	Rank	**Overall Rank**
MODE (SNOV-DS)	4.96E-09 2	-1.35E-04 2	2.23E-10 2	3.37E-02 **1**	-3.64E-01 2	-1.38E-01 2	**11**	**41**
MODE (NDS)	1.08E-06 3	1.71E-05 5	7.19E-08 4	4.49E-01 6	-3.44E-01 4	-1.25E-01 3	25	92
MOEP (SNOV-DS)	1.35E-05 7	-8.55E-05 4	1.84E-05 8	5.03E-01 7	1.13E-01 9	-1.59E-02 9	44	125
MOEP (NDS)	5.11E-05 8	5.59E-05 6	6.19E-05 9	5.30E-01 11	2.79E-01 12	-1.56E-02 10	56	188
MOCLPSO	4.67E-04 9	2.33E-03 11	1.33E-03 10	5.26E-01 10	-1.41E-01 8	-8.71E-02 6	54	189
MO_PSO	6.69E-04 10	1.66E-03 10	9.41E-03 11	5.98E-01 12	-1.89E-01 7	-4.62E-02 7	57	170
NSGA2_SBX	5.54E-06 5	-1.83E-04 **1**	2.04E-16 **1**	4.38E-01 4	-4.17E-01 **1**	-2.15E-01 **1**	13	85
NSGA2_PCX	9.66E-04 11	8.26E-04 9	4.35E-02 12	5.07E-01 8	-2.68E-01 5	-1.92E-02 8	53	141
GDE3	1.00E-05 6	-1.34E-04 3	4.95E-10 3	5.30E-02 2	-3.47E-01 3	-1.13E-01 5	22	62
MOSaDE	-2.60E-04 **1**	4.19E-04 8	1.62E-06 6	2.72E-01 3	1.58E-01 10	1.06E-01 12	40	112
DEMOwSA	2.01E-06 4	9.88E-03 12	8.05E-07 5	5.21E-01 9	-2.58E-01 6	-1.20E-01 4	40	116
MTS	2.52E-03 12	2.55E-04 7	2.17E-06 7	4.47E-01 5	1.67E-01 11	5.87E-02 11	53	160

In order to compare the convergence speed of the proposed algorithm, the convergence graphs of MODE and MOEP with both NDS and SNOV-DS on some test problems are shown in Figs. 5-1-3 to 5-1-8. These figures demonstrate competitive convergence performance of the SNOV-DS procedure. Fig. 5-1-9 to Fig. 5-1-12 show the final front obtained by 4 methods on test problems OKA2, S_ZDT1, S_ZDT2 and R_ZDT2. As can be revealed by these figures, MOEAs with SNOV-DS converge better to the optimal front than MOEAs with NDS most of the times especially for complicated problems such as R_ZDT2. Fig. 5-1-9 shows similar performance between two methods. This is because OKA2 is a relatively simple test function and most of the MOEAs are able to get the true Pareto front.

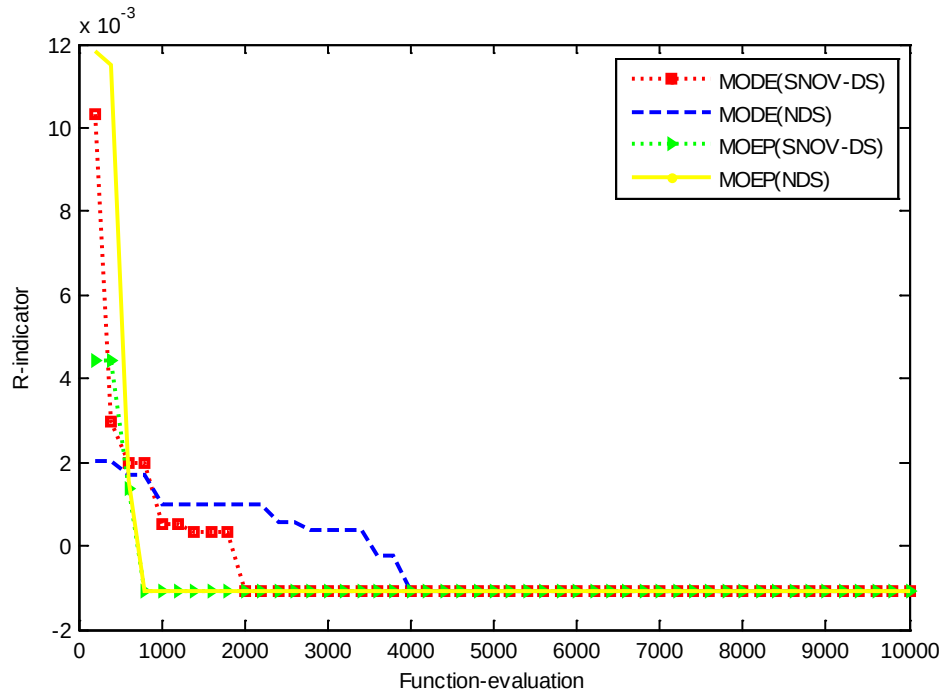


Fig. 5-1-3 The convergence graph of OKA2 (M=2)

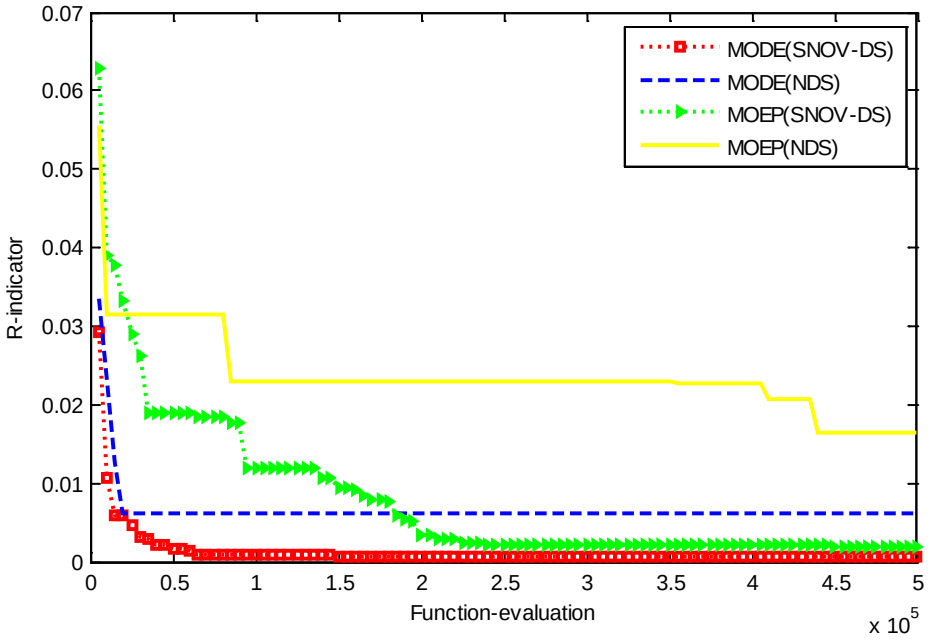


Fig. 5-1-4 The convergence graph of R_ZDT4 (M=2)

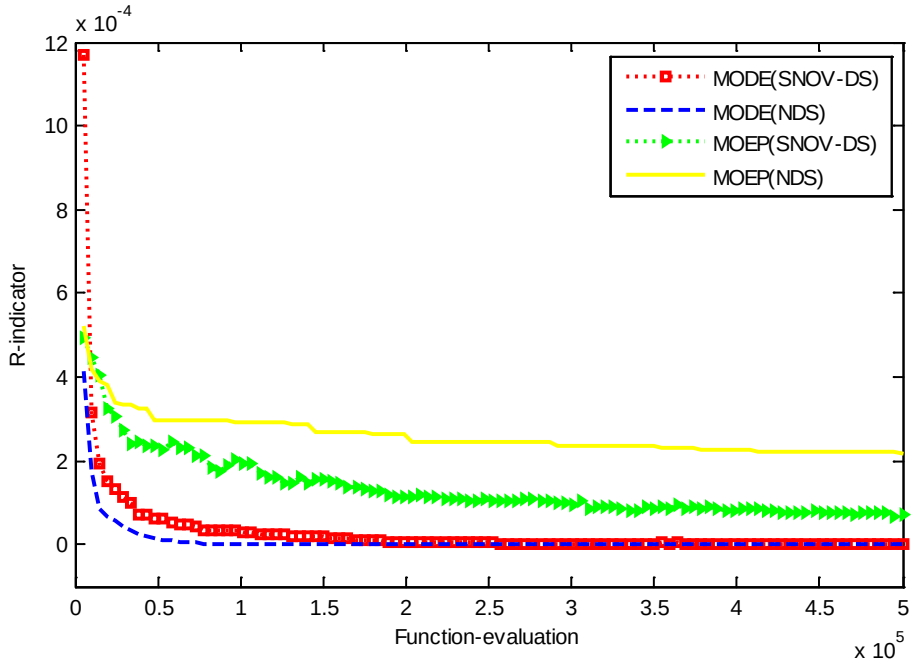


Fig. 5-1-5 The convergence graph of S_DTLZ3 (M=3)

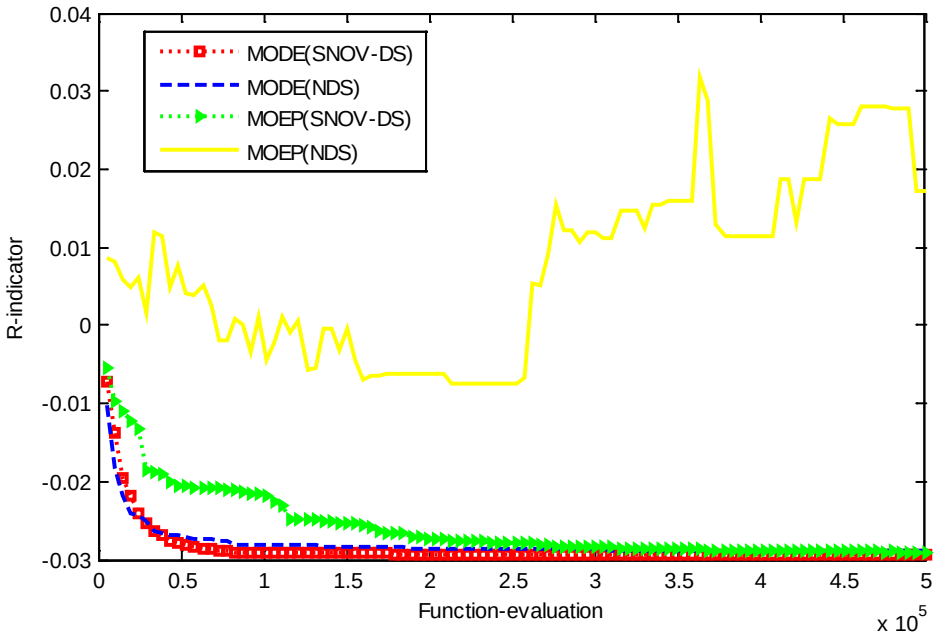


Fig. 5-1-6 The convergence graph of WFG8 (M=3)

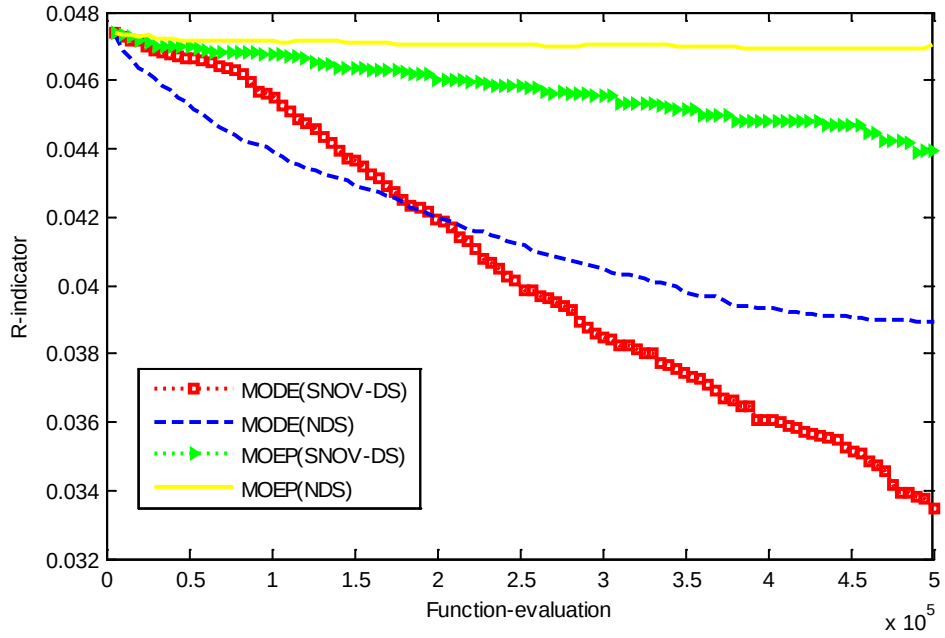


Fig. 5-1-7 The convergence graph of WFG1 (M=5)

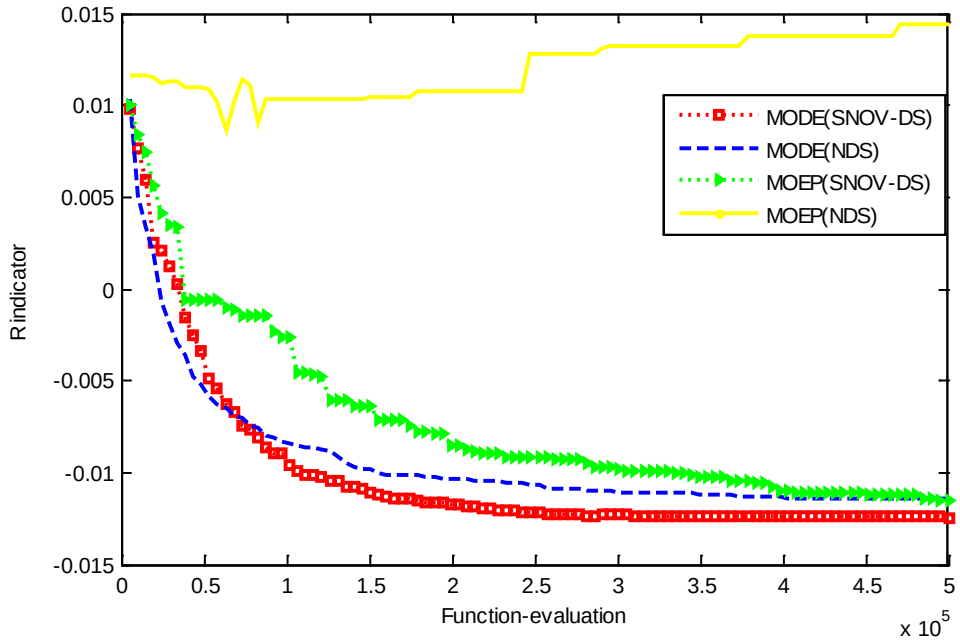


Fig. 5-1-8 The convergence graph of WFG8 (M=5)

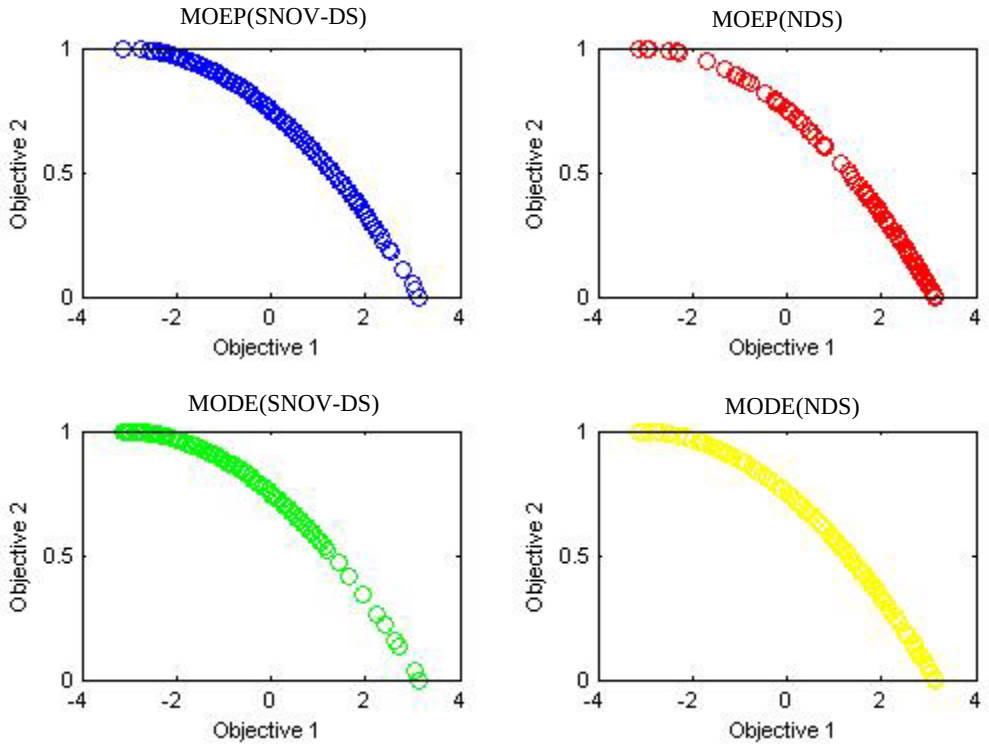


Fig. 5-1-9 Results of test function OKA2

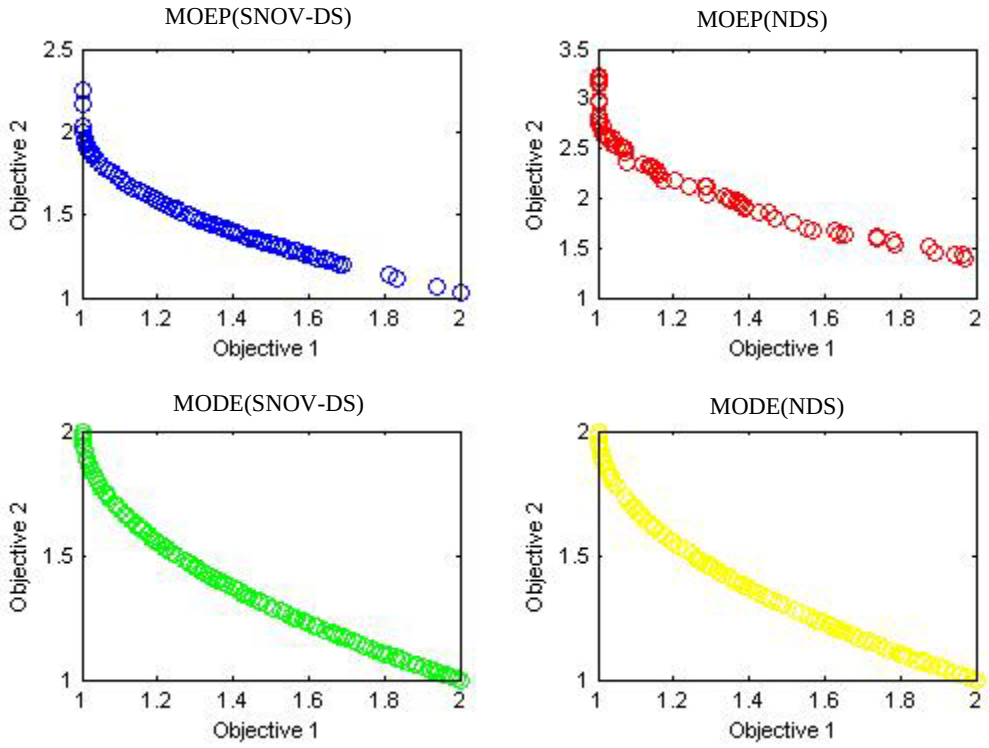


Fig. 5-1-10 Results of test function S_ZDT1

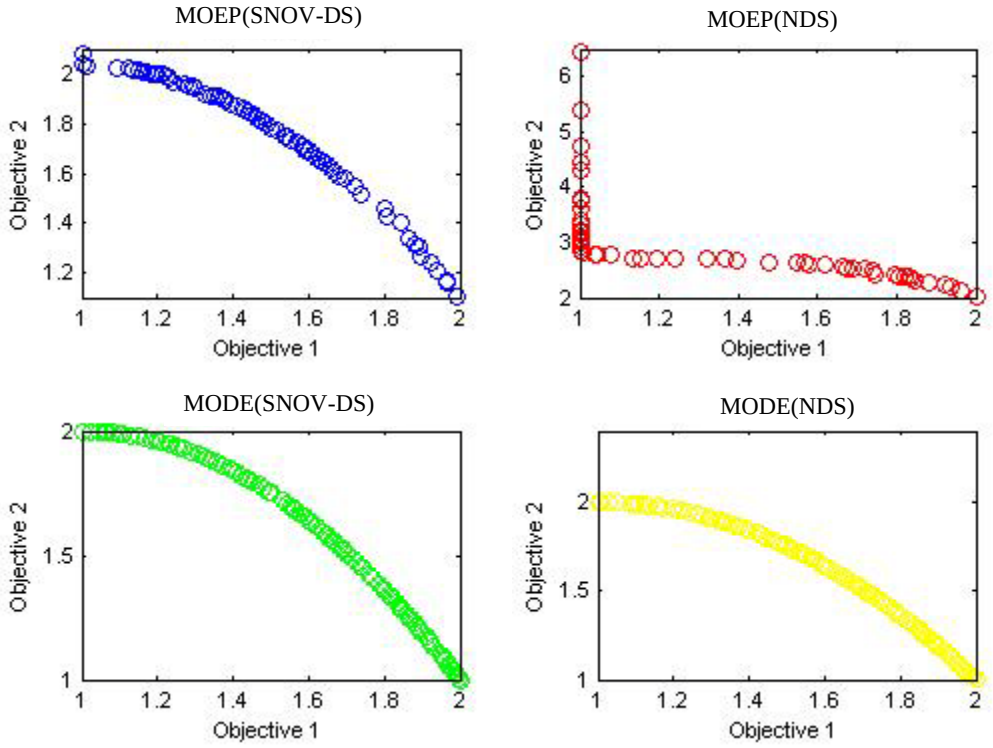


Fig. 5-1-11 Results of test function S_ZDT2

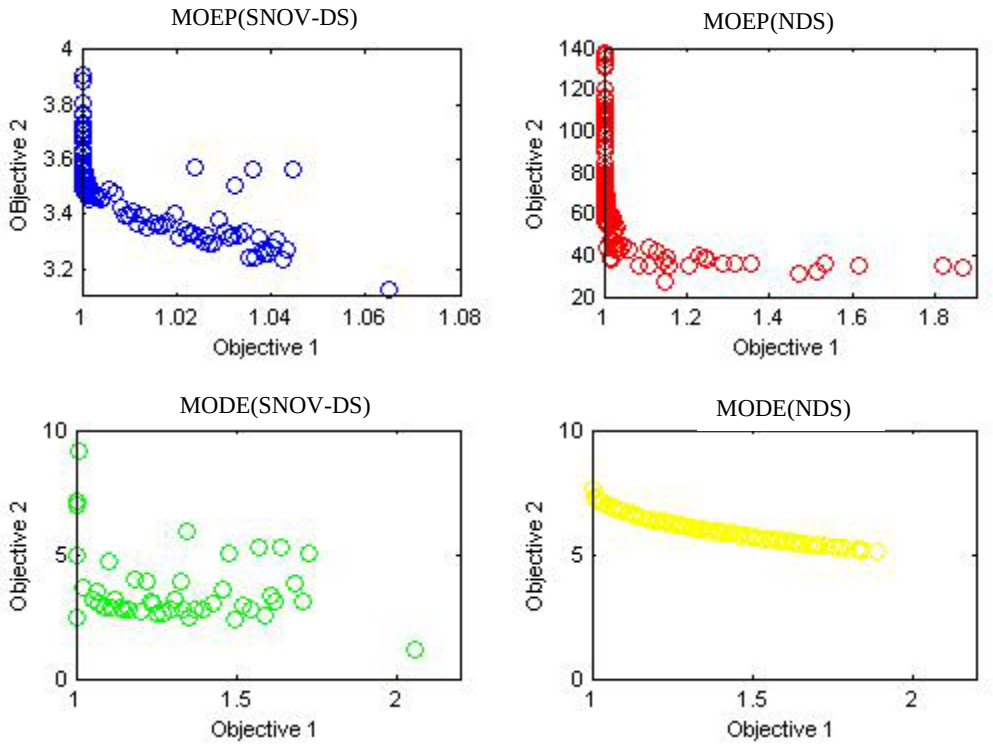


Fig. 5-1-12 Results of test function R_ZDT2

- Parameter sensitivity study

The parameter settings will be influential on the performance of the algorithm. For the parameter *P (scanning percentage)*, generally it should be chosen from 0.7-0.9 and it should not be smaller than 0.6, as small *P* value will decrease the diversity. This parameter may also be dynamically set, *i.e.* starting form 0.9 and gradually decreasing to 0.7 with increasing function evaluations. Table 5-1-7 shows how the performance varies when varying *P* from 0.7-0.9 for some problems with the number of bins set to 100. For the other parameter, namely the number of bins, generally it should not be too small, as it will also affect the diversity. Table 5-1-8 shows how the performance varies when varying the number of bins with *P* set to 0.9.

Table 5-1-7 Performance evaluation with varying *P* values

P	S_ZDT2(M=2)	R_ZDT4(M=2)	S_DTLZ2(M=3)	WFG9(M=3)	S_DTLZ3(M5)
0.7	4.12E-02	4.51E-04	4.20E-05	-1.27E-02	1.14E-06
0.8	3.99E-02	4.28E-04	3.32E-05	-1.26E-02	1.51E-06
0.9	-1.16E-04	2.61E-04	2.79E-07	-9.46E-03	6.25E-07

Table 5-1-8 Performance evaluation by varying the number of bins

Bin	S_ZDT2(M=2)	R_ZDT4(M=2)	S_DTLZ2(M=3)	WFG9(M=3)	S_DTLZ3(M=5)
50	7.90E-03	1.50E-03	6.30E-05	-9.40E-03	4.18E-07
75	-1.14E-04	6.74E-04	3.37E-05	-1.05E-02	8.32E-07
100	-1.15E-04	3.68E-04	1.12E-05	-1.06E-02	6.28E-07
125	-1.14E-04	4.18E-04	2.51E-06	-1.14E-02	6.94E-07
150	-1.13E-04	4.82E-04	1.75E-06	-9.40E-03	6.42E-07
200	-1.12E-04	4.10E-04	5.32E-07	-9.97E-03	6.56E-07

- Comparison based on simulation speed

In order to compare the running speed of SNOV-DS and non-dominated sorting method, a simple experiment is conducted. Fifty sets of random solutions are generated to simulate the parent selection on 2, 3, 5 objectives by both methods. The results are shown in Table 5-1-9.

Table 5-1-9 Computation time comparison between the SNOV-DS and NDS

No. of objectives	Population size	SNOV-DS running time (50 runs/Average)	Non-domination running time (50 runs/Average)	Ratio (NDS / SNOV-DS)
2	200	0.2973 secs / 0.0059 secs	6.0996 secs / 0.1220 secs	20.5
3	300	0.4656 secs / 0.0093 secs	9.7554 secs / 0.1951 secs	20.9
5	1600	2.6838 secs / 0.0537 secs	171.3166 secs / 3.4263 secs	63.8

5.1.5 Conclusion

This section presents a method based on summation of normalized objective values with diversified selection to select parents in multi-objective evolutionary algorithms in order to reduce the complexity of non-domination sorting and also to improve the performance. In order to maintain diversity, the method also introduces a diversified selection method. When compared to the commonly used non-domination sorting method, SNOV-DS performs better with significantly reduced computation time on the two implementations of multi-objective evolutionary algorithms using the differential evolution and the evolutionary programming as the search methods. Moreover, the proposed MODE with SNOV-DS performs the best among 10 different algorithms reported in the literature on the benchmark test problems used in this work.

5.2 Constrained Multi-Objective Optimization Algorithm with Ensemble of Constraint Handling Methods

Different constraint handling techniques have been used together with multi-objective evolutionary algorithms (MOEA) to solve constrained multi-objective optimization problems. It is impossible for a single constraint handling technique to outperform all other constraint handling techniques always on every problem irrespective of the exhaustiveness of the parameter tuning. To overcome this selection problem, we use an ensemble of constraint handling methods (ECHM) which is inspired by the work on single-objective constrained optimization [204] to tackle constrained multi-objective optimization problems. The ECHM is integrated with multi-objective differential evolution (MODE) algorithm which is one of the most effective multi-objective optimization algorithms. Note that SNOV-DS is not included in the constrained multi-objective optimization as the current version is not suitable for constrained problems. The performance is compared between the ECHM and the same single constraint handling methods using the same MODE. The results show that ECHM overall outperforms the single constraint handling methods.

5.2.1 Constraint Handling Methods

Many different constraint handling techniques have been proposed in the literature. Michalewicz and Schoenauer [173] grouped the methods for handling constraints within EAs into four categories: preserving feasibility of solutions, penalty functions, make a separation between feasible and infeasible solutions, and hybrid methods.

Based on the number of feasible solutions present in the current solutions, the search process of a constrained problem can be divided into three phases [174] considering the combined parent-offspring population: (1) No feasible solution, (2) At least one feasible solution, and (3) Combined offspring-parent population has

more feasible solutions than the size of next generation parent population. Various constraint-handling techniques have been proposed over the years to handle diverse constraints in EAs. The differences between these techniques are how to deal with the infeasible individuals throughout the three search phases.

Although solving single objective constrained optimization problems has been studied for several decades, very few works have been done in solving constrained multi-objective optimization problems. Deb et al. [175] proposed a constrained multi-objective algorithm based on the concept of constrained-domination, which is also known as superiority of the feasible solution. Woldesenbet *et al.* [176] introduced a constraint handling technique based on adaptive penalty functions and distance measures by extending the corresponding version for the single objective constrained optimization [177].

In this work, three different constraint handling methods namely self adaptive penalty [176], superiority of feasible solution [181], and ε-constraint [182] are extended from single objective constrained optimization to multi-objective optimization. For all three CHM, the overall constraint violation $v(\mathbf{x})$ is calculated using the following steps:

Step (1): Transform equality constraints into inequality form, and then we can combine all the constraints as:

$$G_j(X) = \begin{cases} \max\{g_j(X), 0\} & j = 1, \ldots J. \\ \max\{|h_j(X)| - \delta, 0\} & j = J+1, \ldots, J+K \end{cases} \quad (5\text{-}2\text{-}1)$$

where δ is a tolerance parameter for the equality constraints.

Step (2): Get the constraint violation for $v_j(\mathbf{x})$, j=1 to $J+K$ (*number_of_constraints*)

Step (3): Normalize the constraint violation using equation:

$$v_{j\_normalized}\ (\boldsymbol{x}) = v_j(\boldsymbol{x}) / v_{max} \quad (5\text{-}2\text{-}2)$$

Where $\boldsymbol{v}_{max}$ is the current maximum violation of constraint j

Step (4): Calculate the overall constraint violation using equation:

$$v(\boldsymbol{x}) = \sum_{j=1}^{J+K} v_{j\,normalized} \quad (\boldsymbol{x}) \tag{5-2-3}$$

- Self adaptive penalty (SP) [176]

  The most common and the earliest approach in the EA community to handle constraints is to use penalty functions. The idea of penalty functions were first introduced by Courant [178]. This method is to transform a constrained optimization problem into an unconstrained problem by adding a penalty factor to the fitness value of each infeasible individual so that it is penalized for violating the constraints. If the penalties added depend only on the degree of violation, then the penalty function is called static penalty. On the other hand, if the penalty function depends on the current generation count also, the penalty function is called dynamic penalty [179]. In adaptive penalty functions [180], information gathered from the search process will be used to control the amount of penalty added to infeasible individuals. Penalty-based constraint handling techniques for multi-objective is similar to single objective except that the penalty factor is added to all the objectives instead of only one objective.

  A self adaptive penalty function is proposed by Woldesenbet and Tessema [176] to solve constrained multi-objective optimization problems using evolutionary algorithm. The method keeps track of the number of feasible individuals in the population to determine the amount of penalty added to infeasible individuals. If there are a few feasible individuals in the whole population, a larger penalty factor is added to infeasible ones. Otherwise, a small penalty factor is used. Self adaptive penalty function uses modified objective function values instead of using the original objective values. The modified objective value has two components: distance measure and adaptive penalty.

Distance values are found for each objective function by incorporating the effect of an individual's constraint violation into its objective values. The value of individual $\boldsymbol{x}$ in each objective function $\boldsymbol{m}$ can be formulated as follows:

$$d_i^{(x)} = \begin{cases} v(x) if r_f = 0 \\ \sqrt{f_i''(x) + v(x)^2}, otherwise \end{cases}$$

where $\boldsymbol{v(x)}$ is the overall constraint violation, $r_f = \frac{Number\ \ of\ feasible\ \ individuals}{population\ \ \ size}$, $f_i''(x)$ is the normalized objective value for m^{th} objective and defined as $f_m''(\boldsymbol{x}) = \frac{f_m(\boldsymbol{x}) - f_{min}}{f_{max} - f_{min}}$. Here, f_{max} and f_{min} are the current maximum and minimum values of the corresponding objective function.

According to equation (5-2-4), if there is no feasible solution in the current population, an infeasible solution with a smaller constraint violation will dominate another infeasible individual with a higher constraint violation. On the other hand, if the number of feasible solution is not zero. The distance value will include both objective values and constraint violations. Besides the distance values, two other penalty functions are also used to include additional penalty for infeasible individuals:

$$p_m(\boldsymbol{x}) = (1 - r_f) X_m(\boldsymbol{x}) + r_f Y_m(\boldsymbol{x}) \qquad (5\text{-}2\text{-}5)$$

where $$X_m(\boldsymbol{x}) = \begin{cases} 0, & if\ r_f = 0 \\ v(\boldsymbol{x}) & otherwise \end{cases} \qquad (5\text{-}2\text{-}6)$$

$$Y_m(\boldsymbol{x}) = \begin{cases} 0, & if\ x\ is\ a\ feasible\ individual \\ f_m'', & if\ x\ is\ an\ infeasible\ individual \end{cases} \qquad (5\text{-}2\text{-}7)$$

The final modified objective value for the m^{th} objective of individual $\boldsymbol{x}$ is formulated as the sum of the distance values and penalty functions:

$$F_m(\boldsymbol{x}) = d_m(\boldsymbol{x}) + p_m(\boldsymbol{x}) \qquad (5\text{-}2\text{-}8)$$

This modified objective value formulation is flexible and will allow the search to utilize infeasible solutions. Table 5-2-1 shows how the fitness values are modified by the self-adaptive penalty method.

Table 5-2-1. Fitness modification by using the self-adaptive penalty method

Step 1: Find feasible solution(s) from the combined population and calculate

the feasible percentage of the current population r_f

Step 2: If $r_f = 0$

For *m* = 1 to *M* (*number_of_objectives)*

$\text{distance}_i(x) = v(x)$

$X_m(\boldsymbol{x}) = 0$

End For

Else

For *m* = 1 to *M (number _of_objectives)*

$d_m(\boldsymbol{x}) = \sqrt{f_m''(\boldsymbol{x}) + v(\boldsymbol{x})^2}$

$X_m(\boldsymbol{x}) = v(\boldsymbol{x})$

End For

End If

Step 3: If $v(\boldsymbol{x}) = 0$

For *m* = 1 to *M (number _of_objectives)*

$Y_m(\boldsymbol{x}) = 0$

End For

Else

For *m* = 1 to *M (number_of_objectives)*

$Y_m(\boldsymbol{x}) = f_m''$

End For

End If

Step 4: Apply equation (5-2-5) to obtain $p_m(\boldsymbol{x})$.

Step 5: Form the new fitness values for all the objectives of all solutions using equation (5-2-8).

- Superiority of feasible solution (SF) [181]

This constraint handling method was first introduced by Powell and Skolnick [172]. The SF as used in multi-objective optimization is expressed as:

$$fitness_m(\boldsymbol{x}) = \begin{cases} f_m(\boldsymbol{x}) & if\ x\ is\ feasible \\ f_{worst}^m + v(\boldsymbol{x}) & otherwise \end{cases} \quad (5\text{-}2\text{-}9)$$

where f_{worst}^{i} is the m^{th} objective value of the worst feasible solution with respect to objective m in the population and $v(\boldsymbol{x})$ is the overall constraint violation. If there is no feasible solution in the current population, f_{worst}^m is zero.

In this approach, there is no penalty factor involved. Feasible solutions are always better than the infeasible ones. In this way, the infeasible solutions are expected to evolve towards the feasible region and feasible solutions are expected to evolve towards the global optimal front. Table 5-2-2 shows how the fitness values are modified by the superiority of feasible solution method.

Table 5-2-2 Fitness modification by using the SF method

Step 1:	Extract feasible solution(s) from the combined population. If there is no feasible solution, the new fitness value is overall constraint violation value. If all solutions are feasible, the fitness values are unchanged.
Step 2:	For m = 1 to *M (number_of_objectives)* Find the maximum objective values $f_{i\_max}^{''}$ for each objective among the feasible solutions. End For
Step 3:	Form the new fitness values for infeasible solutions using equation (5-2-9) and keep the fitness values of feasible solutions unchanged.

- ε-constraint (EC) [182]

The ε-constraint handling technique was first proposed by Takahama [182]. The basic idea is similar to the superiority of feasible solutions and the difference is the relaxation of the constraint violation during the early stages of the search process using the ε parameter. As some useful information may be contained in the infeasible solutions with relatively small overall constraint violation, the relaxation of the constraint violations may include more infeasible solutions in the population during the early stages of evolution. The ε value is updated until the generation counter k reaches the control generations T_c. After T_c generations, the ε value is set to zero and the EC becomes the same as the superiority of feasible method. The ε value is updated according to the following equations:

$$\varepsilon(0) = v(x_\theta) \tag{5-2-10}$$

$$\varepsilon(k) = \begin{cases} \varepsilon(0)(1 - \frac{k}{T_c})^{cp}, & 0 < k < T_c \\ 0, & k \geq T_c \end{cases} \tag{5-2-11}$$

where x_θ is the top θ-th individual at initialization. cp parameter is recommended to be [2,10]. Table 5-2-3III shows how the fitness values are modified by the ε-constraint method.

Table 5-2-3 Fitness modification by using the EC method

Step 1:	Find infeasible solution(s) from the combined population (if all solutions are feasible, ε is set to be zero throughout the evolution)

Step 2:	If the generation count k=1
	Use equation (5-2-10) to obtain $\varepsilon(0)\varepsilon(0) = \mathrm{v}(\mathrm{x}_\theta)$
	Else if the generation count $k < T_c$
	$\varepsilon(k) = \varepsilon(0)(1 - \frac{k}{T_c})^{cp}$,
	Else
	$\varepsilon(k) = 0$
	End If

Step 3: Find the solutions that have the constraint violation less than ε and from these solutions find the maximum objective values f''_{i_max} for each objective.

Step 4: Form the new fitness values for the solutions that have constraint violations greater than ε using the following equation while others are kept unchanged:

$$fitness(\boldsymbol{x}) = f''_{m\_max} + v(\boldsymbol{x})$$

- Comparative discussion on constraint handling methods

When solving constrained problems, one of the following three scenarios can be encountered: (1) no feasible solution, (2) at least one feasible solution, and (3) combined offspring-parent population has more feasible solutions than the size of next generation parent population. When an ensemble of constraint handling method is constructed, it is beneficial to choose competitive CHMs which also perform differently during these three scenarios.

The SF method ranks feasible solutions better than the infeasible ones. Two infeasible solutions are compared based on their overall constraint violations only, while two feasible solutions are compared based on their objective function values only. Therefore, in scenario (1), infeasible solutions with low overall constraint violation are selected from the combined parent and offspring population. In scenario (2), first all feasible ones are selected and then infeasible ones with the lowest overall constraint violations are selected. In scenario (3), only feasible ones with best objective values are selected. The SP method always selects individuals based on a value determined by the overall constraint violation and objective values. Thus, an individual with lower overall constraint violation and higher fitness can be selected over a feasible individual with lower fitness in scenario (3) too. The ε-constraint (EC) method selects similar to

the SF, but in the EC, a solution is regarded as feasible if its overall constraint violation is smaller than $\varepsilon(k)$. Therefore, it is obvious that these competitive methods rank solutions differently.

5.2.2 MODE with Ensemble of Constraint Handling Methods

According to the no free lunch (NFL) theorem [153], it is impossible for a single constraint handling technique to outperform all other constraint handling methods on every problem. Each problem has its own characteristics such as the ratio between feasible search space and the whole search space, linearity/non-linearity of constraint functions and so on. Each constraint handling method may also work differently during different search phases. Hence, different constraint handling methods may be suitable for solving a problem during different stages of the search process.

The flowchart of the ensemble algorithm is shown in Fig. 5-2-1. In our implementation, an ensemble of three constraint handling methods is used. Each constraint handling method has a population associated with it. Each population generates its offspring population. For every offspring in the three offspring populations, objective / constraint function values and overall constraint violation are computed. The three offspring populations are combined and passed to the three fitness modification algorithms in Tables 5-2-1 to 5-2-3 simultaneously to undergo different fitness modification processes according to the three CHMs. After the completion of three fitness modification processes, each offspring population is combined with the respective parent population. Therefore, we perform the objective and constraint function evaluation only once per offspring and perform fitness modification three times according to the CHMs.

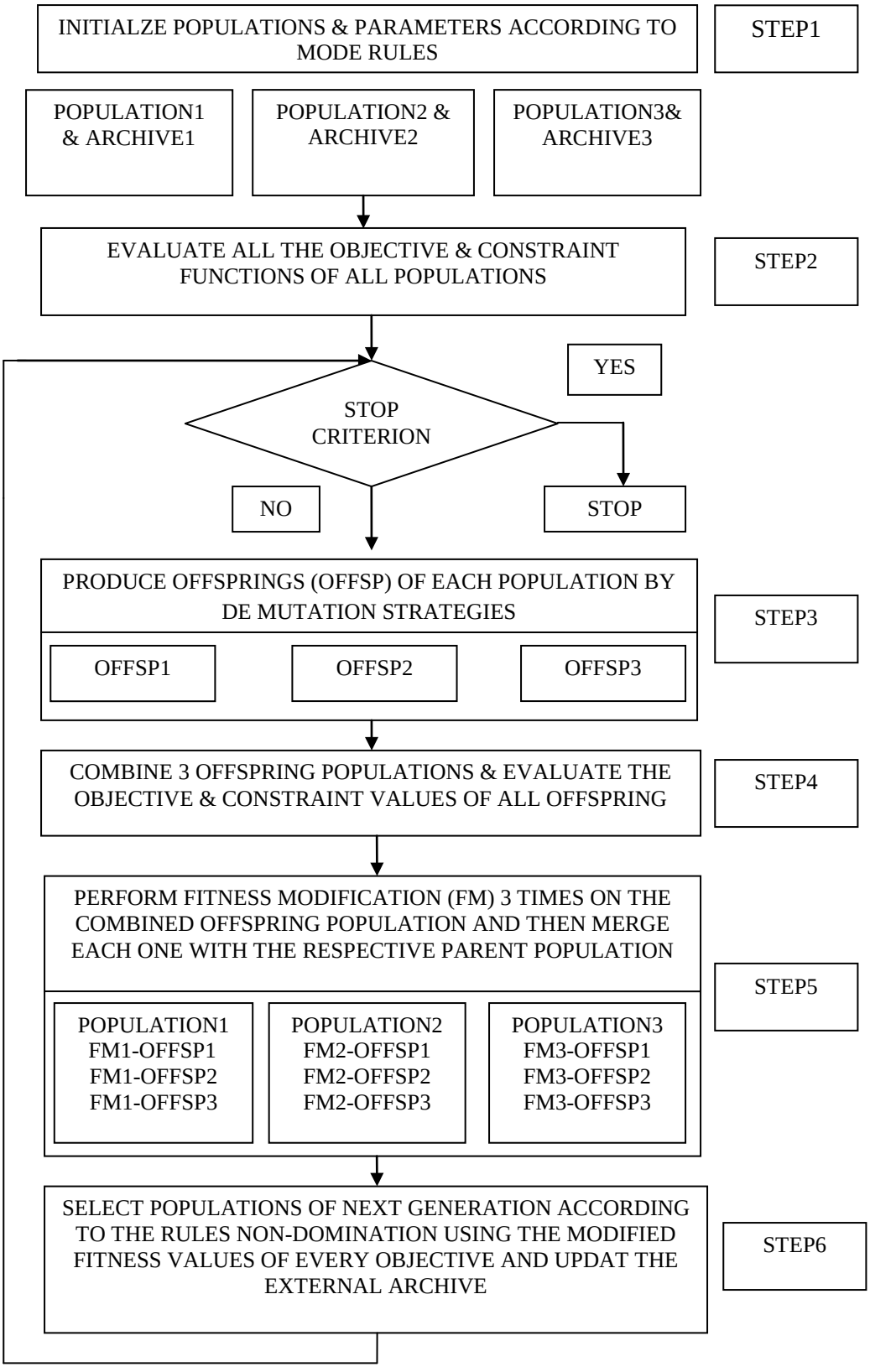


Fig. 5-2-1 Flowchart of ensemble of three constraint handling methods

When we employ single constraint handling methods to solve a particular test problem, we may observe that a particular CHM generates better quality offspring than the other CHMs. Generating better quality solutions implies that these solutions have lower overall constraint violation and/or better fitness values. Such solutions can be detected by any CHM. Therefore, even if one particular CHM may not be the best choice for generating good quality offspring, this CHM can definitely identify good quality offspring generated by another CHM. Therefore, if a particular constraint handling method is best suited for the search method and the problem during a point in the search process, the offspring population produced by the particular constraint handling technique will dominate the offspring generated by other populations and enter populations corresponding to other constraint handling methods too. In the subsequent generations, these superior offspring will become parents in other populations too. For difficult problems, more constraint handling methods can be included in the ensemble in order to benefit more from each function call.

5.2.3 Experiments and Results

- Experimental setup

The population size is set at 50 for ensemble method, 50 and 150 for single constraint handling methods. The maximum external archive size is set at 100. The maximum number of function evaluations (FES) is set as 2e+5. All the performances are measured over 30 runs. The parameters used in the algorithms are listed as below:

$$\text{DE: } F = 0.3, CR = 0.3$$

$$\text{EC: } cp = 5, T_c = 60000, \theta = 20$$

Seven different methods have been compared experimentally. They are:

(1) Method A: MODE with an ensemble of three constraint handling methods;

(2) Method B: MODE with superiority of feasible solution constraint handling method (population size=50);

(3) Method C: MODE with ε-constraint handling method (population size=50);

(4) Method D: MODE with self adaptive penalty constraint handling method (population size=50);

(5) Method E: MODE with superiority of feasible solution constraint handling method (population size=150);

(6) Method F: MODE with ε-constraint handling method (population size=150);

(7) Method G: MODE with self adaptive penalty constraint handling method (population size=150);

To test the performances of these algorithms, ten constrained multi-objective benchmark problems available from the literature are used. These problems are presented in Appendix D.

- Performance measures

In order to compare the performances of different constraint handling methods quantitatively, some performance metrics are needed. There are two goals in a multi-objective optimization: 1) convergence to the Pareto-optimal set and 2) diversity of solutions in the Pareto-optimal set. The following two indicators are used:

(1) R indicator (I_{R2}) [172]: $I_{R2} = \frac{\sum_{\lambda \in \Lambda} u^*_\lambda(A) - u^*_\lambda(R)}{|\Lambda|}$ where R is a reference set, u^* is the maximum value reached by the utility function u with weight vector λ on an approximation set A, *i.e.*, $u^* = \max_{z \in A} u(z)$. We choose the augmented Tchebycheff function as the utility function.

(2) Hypervolume difference to a reference set ($I_{\bar{H}}$) [172]: The hypervolume indicator I_H measures the hypervolume of the objective

space that is weakly dominated by an approximation set *A*, and is to be maximized. Here we consider the hypervolume difference to a reference set *R*, and we will refer to this indicator as $I_{\bar{H}}$, which is defined as $I_{\bar{H}} = I_H(R) - I_H(A)$ where smaller values correspond to higher quality as opposed to the original hypervolume I_H .

- Comparison of ECHM with single constraint handling methods

From the results in Tables 5-2-4 to 5-2-7, it can be observed that the ensemble of constraint handling methods perform either similar to or better than the single constraint handling methods. From the summation of the rankings shown in the last row, it can be observed that ensemble constraint handling method outperforms all the single constraint handling methods with respect to both the *R*-indicator and *H*-indicator. Further, as each single constraint handling method has similar summation of the rankings, it also suggests that different methods perform differently on each problem. Moreover, as the population size of single constraint handling methods varies, the performance of different problems also varies. It suggests that different population sizes are suitable for different test problems.

The superior performance of ensemble over the single constraint handling methods in the given number of function evaluations is due to the efficient use of every function call by all three populations and the search process in the ensemble benefiting from the best performance of different constraint handling methods during different stages of the search process. In every generation only the best offspring from the three populations will survive and have the opportunity to reproduce, which means the offspring not only compete with their parents but also compete with the offspring produced by the populations of other constraint handling methods.

In order to demonstrate the advantage of the ECHM more clearly, *t*-test is applied to the indicators and the *h* values are shown in Tables 5-2-4 to 5-2-7 below the function names in the first column. All the three single constraint handling methods are compared with the proposed ECHM. The

numerical values -1, 0, 1 represent that the proposed ECHM is statistically superior to, equal to and inferior to the single constraint handling method. From the result, we can observe that the proposed ECHM performs either better or similar to all three single constraint handling methods.

Table 5-2-4. The *R*-indicators for 30 runs and *h*-values on the test problems. *h*-values are shown below the function names in the first column. (Population size=50)

Function		Ensemble	rank	SF	rank	EC	rank	SP	rank
TNK	Mean	2.25E-04	2	2.60E-04	4	2.29E-04	3	2.16E-04	1
	Worst	3.88E-04	2	3.67E-04	1	4.82E-04	4	4.44E-04	3
[0,0,0]	Best	5.45E-05	1	7.18E-04	3	6.31E-05	2	8.97E-05	4
	Std	8.66E-05		9.37E-05		1.46E-04		1.03E-04	
SRN	Mean	8.21E-05	1	3.53E-04	3	3.92E-04	4	2.86E-04	2
	Worst	3.06E-04	1	9.77E-04	4	8.76E-04	3	6.66E-04	2
[-1,-1,-1]	Best	2.75E-06	1	2.78E-05	2	8.82E-05	4	5.45E-05	3
	std	8.04E-05		2.79E-04		2.29E-04		1.80E-04	
CONSTR	Mean	1.56E-05	2	1.94E-05	4	1.55E-05	1	1.66E-05	3
	Worst	2.44E-05	1	3.74E-05	4	3.28E-05	2	3.38E-05	3
[0,0,0]	Best	7.46E-07	1	3.50E-06	3	1.77E-06	2	5.83E-06	4
	std	6.44E-06		1.09E-05		8.42E-06		7.46E-06	
OSY	Mean	7.00E-03	1	1.03E-02	3	3.49E-02	4	7.10E-03	2
	Worst	2.02E-02	1	3.92E-02	3	3.94E-02	4	3.89E-02	2
[-1,-1,0]	Best	1.95E-06	1	3.66E-05	2	1.95E-02	4	3.76E-05	3
	std	9.50E-03		1.16E-02		7.70E-03		1.13E-02	
CTP1	Mean	1.19E-06	1	2.30E-03	4	1.90E-03	3	8.93E-04	2
	Worst	4.46E-06	1	8.00E-03	2	8.50E-03	3	8.50E-03	3
[-1,-1,-1]	Best	3.45E-09	1	3.71E-07	4	2.41E-08	2	4.81E-08	3
	std	1.23E-06		3.10E-03		3.20E-03		2.10E-03	
CTP2	Mean	5.27E-06	1	7.91E-04	4	4.01E-04	3	3.80E-05	2
	Worst	1.36E-05	1	1.74E-02	4	5.50E-03	3	7.14E-05	2
[-1,-1,-1]	Best	3.60E-07	1	3.60E-06	3	3.08E-06	2	9.11E-06	4
	std	4.21E-06		3.30E-03		1.40E-03		2.24E-05	
CTP3	Mean	1.04E-04	1	5.91E-04	4	1.53E-04	2	1.61E-04	3
	Worst	1.87E-04	1	8.50E-04	4	2.71E-04	2	3.39E-04	3
[-1,-1,-1]	Best	4.88E-06	2	5.21E-05	4	1.81E-07	1	4.26E-06	3
	std	5.30E-05		1.80E-03		7.33E-05		7.52E-05	
CTP4	Mean	7.49E-04	1	1.00E-03	3	1.60E-03	4	8.28E-04	2
	Worst	2.00E-03	1	3.90e-03	3	1.22E-02	4	2.10E-03	2
[-1,-1,0]	Best	7.34E-05	1	4.37E-04	4	1.10E-04	2	2.86E-04	3
	std	4.00E-04		8.02E-04		3.10E-03		5.47E-04	

CTP5	Mean	1.00E-03	1	2.90E-03	2	3.90E-03	4	2.90E-03	2
	Worst	1.50E-03	1	1.37E-02	2	1.37E-02	2	1.48E-02	4
[-1,-1,-1]	Best	7.60E-05	2	9.32E-04	4	7.70E-05	2	2.88E-05	1
	std	4.20E-04		3.10E-03		3.30E-03		4.70E-03	
CTP6	Mean	5.87E-08	1	2.80E-03	3	2.80E-03	3	2.50E-03	2
	Worst	2.70E-07	1	1.18E-02	4	6.40E-03	2	9.10E-03	3
[-1,-1,-1]	Best	7.59E-10	1	1.30E-09	2	1.50E-03	4	1.90E-09	3
	std	6.96E-08		2.90E-03		1.60E-03		3.10E-03	
Summation of Ranks:			35		92		85		79

Table 5-2-5. The *H*-indicators for 30 runs and *h*-values on the test problems. *h*-values are shown below the function names in the first column. (Population size=50)

Function		Ensemble	rank	SF	rank	EC	rank	SP	rank
TNK	Mean	6.20E-04	2	7.15E-04	4	6.34E-04	3	5.97E-04	1
	Worst	1.20E-03	2	1.00E-03	1	1.30E-03	4	1.20E-03	3
[0,0,0]	Best	1.71E-04	1	2.17E-04	3	1.91E-04	2	2.64E-04	4
	std	2.28E-04		2.47E-04		3.84E-04		2.71E-04	
SRN	Mean	1.60E-03	1	3.80E-03	4	3.70E-03	2	3.70E-03	2
	Worst	1.90E-03	1	4.20E-03	3	4.40E-03	4	4.00E-03	2
[-1,-1,-1]	Best	1.40E-03	1	3.50E-03	3	3.50E-03	3	3.40E-03	2
	std	1.11E-04		1.96E-04		2.41E-04		1.64E-04	
CONSTR	Mean	2.44E-04	2	2.43E-04	1	2.44E-04	2	2.45E-04	4
	Worst	2.60E-04	1	2.69E-04	4	2.66E-04	2	2.67E-04	3
[0,0,0]	Best	1.46E-04	1	2.19E-04	2	2.22E-04	3	2.23E-04	4
	std	3.51E-05		1.53E-05		1.44E-05		1.147-05	
OSY	Mean	1.10E-02	1	1.89E-02	3	7.23E-02	4	1.35E-02	2
	Worst	2.70E-02	1	8.97E-02	4	8.89E-02	3	7.65E-02	2
[0,-1,0]	Best	1.50E-03	1	2.00E-03	2	2.60E-02	4	2.10E-03	3
	std	1.93E-02		2.13E-02		2.43E-02		1.83E-02	
CTP1	Mean	2.34E-05	1	7.90E-03	4	6.40E-03	3	3.40E-03	2
	Worst	3.25E-05	1	2.56E-02	2	2.68E-02	3	2.68E-02	3
[-1,-1,-1]	Best	1.98E-05	1	2.51E-05	4	2.24E-05	2	2.39E-05	3
	std	3.54E-06		9.10E-03		9.80E-03		6.60E-03	
CTP2	Mean	1.14E-05	1	2.40E-03	4	1.30E-03	3	1.21E-04	2
	Worst	2.51E-05	1	5.22E-02	4	1.66E-02	3	5.11E-04	2
[-1,-1,-1]	Best	3.55E-06	2	1.43E-05	1	1.80E-05	3	1.97E-05	4
	std	6.29E-06		9.90E-03		4.30E-02		1.45E-04	
CTP3	Mean	1.80E-04	1	9.59E-04	4	2.37E-04	2	2.53E-04	3
	Worst	3.06E-04	1	1.39E-02	4	4.11E-04	2	5.80E-04	3
[-1,-1,-1]	Best	2.47E-05	2	8.74E-05	4	3.54E-07	1	2.90E-05	3
	std	7.82E-05		2.90E-03		1.10E-04		1.18E-04	
CTP4	Mean	2.20E-03	1	3.30E-03	3	4.80E-03	4	2.50E-03	2

[-1,-1,0]	Worst	6.10E-03	1	9.60E-02	4	3.59E-02	3	6.20E-03	2
	Best	3.84E-04	1	1.30E-03	4	4.28E-04	2	9.31E-04	3
	std	1.10E-03		2.30E-03		9.00E-03		1.50E-03	
CTP5	Mean	1.60E-03	1	4.50E-03	2	6.00E-03	4	4.70E-03	3
	Worst	2.20E-03	1	2.13E-02	2	2.13E-02	2	2.32E-02	4
[-1,-1,-1]	Best	1.48E-04	2	1.50E-03	4	1.83E-04	3	1.58E-04	1
	std	6.21E-04		5.40E-03		6.50E-03		7.40E-03	
CTP6	Mean	9.41E-07	1	4.40E-03	3	4.41E-03	4	3.90E-03	2
	Worst	1.26E-06	1	1.81E-02	4	1.02E-02	2	1.45E-02	3
[-1,-1,-1]	Best	7.91E-07	1	1.02E-06	3	2.20E-03	4	9.89E-07	2
	std	1.14E-07		4.60E-03		2.60E-03		4.90E-03	
Summation of Ranks:			36		94		86		79

Table 5-2-6 The *R*-indicators for 30 runs and *h*-values on the test problems. *h*-values are shown below the function names in the first column. (Population size=150)

Function		Ensemble	rank	SF	rank	EC	rank	SP	rank
TNK	Mean	2.25E-04	1	3.44E-04	4	3.42E-04	3	2.49E-04	2
	Worst	3.88E-04	2	7.09E-04	3	7.75E-04	4	3.87E-04	1
[-1,-1,0]	Best	5.45E-05	1	8.21E-05	3	7.47E-05	2	1.52E-04	4
	Std	8.66E-05		1.91E-04		1.73E-04		9.29E-05	
SRN	Mean	8.21E-05	1	6.05E-04	3	1.40E-03	4	4.72E-04	2
	Worst	3.06E-04	1	1.10E-03	2	5.60E-03	4	1.80E-03	3
[-1,-1,-1]	Best	2.75E-06	1	2.10E-05	3	2.80E-05	4	1.57E-05	2
	std	8.04E-05		3.75E-04		1.30E-03		4.69E-04	
CONSTR	Mean	1.56E-05	3	1.55E-05	2	1.83E-05	4	1.08E-05	1
	Worst	2.44E-05	1	2.56E-05	2	5.23E-05	4	3.38E-05	3
[0,0,0]	Best	7.46E-07	1	1.97E-06	4	8.55E-07	2	1.87E-06	3
	std	6.44E-06		7.52E-06		1.12E-05		8.08E-06	
OSY	Mean	7.00E-03	1	1.68E-02	3	3.17E-02	4	1.55E-02	2
	Worst	2.02E-02	1	3.89E-02	3	3.92E-02	4	3.88E-02	2
[-1,-1,-1]	Best	1.95E-06	1	3.76E-05	3	1.94E-02	4	1.34E-05	2
	std	9.50E-03		1.48E-02		9.40E-03		1.14E-02	
CTP1	Mean	1.19E-06	1	7.78E-05	3	1.56E-04	4	5.13E-05	2
	Worst	4.46E-06	1	3.77E-04	2	4.20E-03	4	3.77E-04	2
[-1,-1,-1]	Best	3.45E-09	1	4.26E-08	4	2.41E-08	3	5.23E-09	2
	std	1.23E-06		1.52E-04		7.76E-04		1.30E-04	
CTP2	Mean	5.27E-06	2	6.96E-06	3	7.57E-06	4	4.92E-06	1
	Worst	1.36E-05	1	2.01E-	3	2.61E-05	4	1.84E-05	2
[0,-1,0]	Best	3.60E-07	1	6.34E-07	3	1.24E-06	4	5.00E-07	2
	std	4.21E-06		5.66E-06		6.29E-06		4.29E-	
CTP3	Mean	1.04E-04	1	1.59E-04	4	1.41E-04	2	1.49E-04	3
	Worst	1.87E-04	1	2.83E-04	3	2.50E-04	2	2.86E-04	4

[-1,-1,-1]	Best	4.88E-06	1	3.22E-05	3	4.63E-05	4	2.56E-05	2
	std	5.30E-05		6.18E-05		6.14E-05		5.69E-05	
CTP4	Mean	7.49E-04	2	6.24E-04	1	1.06E-03	4	7.56E-04	3
	Worst	2.00E-03	2	1.40E-03	1	2.50E-03	4	2.10E-03	3
[0,-1,0]	Best	7.34E-05	1	2.92E-04	3	2.28E-04	2	3.10E-04	4
	std	4.00E-04		2.02E-04		6.08E-04		4.58E-04	
CTP5	Mean	1.00E-03	1	1.20E-03	2	1.20E-03	2	1.40E-03	4
	Worst	1.50E-03	1	2.10E-03	2	2.80E-03	3	5.50E-03	4
[0,0,-1]	Best	7.60E-05	2	7.68E-05	3	6.82E-05	1	7.69E-05	4
	std	4.20E-04		4.51E-04		6.36E-04		1.10E-03	
CTP6	Mean	5.87E-08	1	5.36E-04	2	2.90E-03	4	1.10E-03	3
	Worst	2.70E-07	1	7.80E-03	3	3.00E-03	2	7.80E-03	3
[-1,-1,-1]	Best	7.59E-10	1	3.42E-09	3	3.04E-09	2	7.97E-09	4
	std	6.96E-08		2.00E-03		5.49E-04		2.70E-03	
Summation of Ranks:			37		83		98		79

Table 5-2-7 The *H*-indicators for 30 runs and *h*-values on the test problems. *h*-values are shown below the function names in the first column. (Population size=150)

Function		Ensemble	rank	SF	rank	EC	rank	SP	rank
TNK	Mean	6.20E-04	1	9.27E-04	4	9.21E-04	3	6.75E-04	2
	Worst	1.20E-03	1	1.90E-03	3	2.10E-03	4	1.20E-03	1
[-1,-1,0]	Best	1.71E-04	1	2.35E-04	3	2.16E-04	2	4.18E-04	4
	std	2.28E-04		5.02E-04		4.56E-04		2.44E-04	
SRN	Mean	1.60E-03	1	3.60E-03	3	3.50E-03	2	3.70E-03	4
	Worst	1.90E-03	1	4.60E-03	4	4.10E-03	2	4.10E-03	2
[-1,-1,-1]	Best	1.40E-03	1	3.50E-03	2	3.60E-03	4	3.50E-03	2
	std	1.11E-04		2.02E-04		1.20E-04		1.45E-04	
CONSTR	Mean	2.44E-04	2	2.44E-04	2	2.49E-04	4	2.42E-04	1
	Worst	2.60E-04	1	2.76E-04	4	2.68E-04	3	2.66E-04	2
[0,0,0]	Best	1.46E-04	1	2.18E-04	3	2.11E-04	2	2.21E-04	4
	std	3.51E-05		1.49E-05		1.13E-05		1.35-05	
OSY	Mean	1.10E-02	1	2.75E-02	3	5.33E-02	4	2.27E-02	2
	Worst	2.70E-02	1	7.29E-02	3	8.64E-02	4	6.33E-02	2
[-1,-1,-1]	Best	1.50E-03	1	2.10E-03	3	2.43E-02	4	2.03E-03	2
	std	1.93E-02		2.48E-02		2.15E-02		1.73E-02	
CTP1	Mean	2.34E-05	1	1.97E-04	3	2.79E-04	4	1.31E-04	2
	Worst	3.25E-05	1	9.57E-04	2	7.30E-03	4	9.57E-04	2
[-1,-1,-1]	Best	1.98E-05	1	2.03E-05	3	2.05E-05	4	2.01E-05	2
	std	3.54E-06		3.87E-04		1.30E-03		3.29E-04	
CTP2	Mean	1.14E-05	2	1.22E-03	3	1.35E-05	4	1.01E-05	1
	Worst	2.51E-05	1	2.72E-02	3	2.88E-05	4	2.57E-05	2
[0,0,0]	Best	3.55E-06	1	4.33E-06	2	4.91E-06	4	4.38E-06	3
	std	6.29E-06		5.95E-05		4.30E-02		4.13E-06	

CTP3	Mean	1.80E-04	1	2.66E-04	4	2.37E-04	2	2.41E-04	3
	Worst	3.06E-04	1	4.57E-04	4	3.93E-04	2	4.56E-04	3
[-1,-1,-1]	Best	2.47E-05	1	7.00E-05	3	1.02E-04	4	6.61E-05	2
	std	7.82E-05		9.34E-05		9.00E-05		8.41E-05	
CTP4	Mean	2.20E-03	2	1.90E-03	1	2.70E-03	4	2.30E-03	3
	Worst	6.10E-03	2	4.20E-03	1	7.20E-03	4	6.10E-03	2
[0,0,0]	Best	3.84E-04	1	9.66E-04	3	7.88E-04	2	1.00E-03	4
	std	1.10E-03		2.30E-03		1.70E-03		1.30E-03	
CTP5	Mean	1.60E-03	1	1.80E-03	2	1.90E-03	3	2.20E-03	4
	Worst	2.20E-03	1	3.20E-03	2	4.20E-03	3	8.70E-03	4
[0,0,-1]	Best	1.48E-04	2	1.54E-04	3	1.43E-04	1	1.86E-04	4
	std	6.21E-04		6.71E-04		9.50E-04		1.70E-03	
CTP6	Mean	9.41E-07	1	1.50E-03	2	7.60E-03	4	3.10E-03	3
	Worst	1.26E-06	1	2.22E-02	3	7.90E-03	2	2.22E-02	3
[-1,-1,-1]	Best	7.91E-07	1	1.19E-06	2	1.20E-06	3	1.22E-06	4
	std	1.14E-07		5.70E-03		1.50E-03		7.80E-03	
Summation of Ranks:			35		83		96		79

5.2.4 Conclusion

In this work, an ensemble of three different constraint handling techniques is employed with multi-objective differential evolution algorithm, since there is no single constraint handling technique can outperform all other constraint handling techniques on every problem. The important property of the ECHM is that every function call is used by every population associated with each constraint handling technique. As evolutionary algorithms are stochastic in nature, the search process passes through different phases at different points during the search process. Different constraint handling methods can be effective during different stages of the search process. The ensemble of constraint handling method will bring the best solutions generated by the most suitable technique to all the populations. In this way, the search process will continue using the best technique to generate offspring. Hence, ECHM has the potential to perform well over diverse problems when compared to a single CHM. We tested the performances of the proposed ECHM as well as the three individual constraint handling methods. Experimental results showed that the ECHM overall outperforms all three single constraint handling methods.

5.3 Summary

This Chapter considers evolutionary algorithms for constrained and unconstrained multi-objective optimization. For unconstrained multi-objective optimization, a new sorting and selection method is proposed which has lower complexity and better performance than the non-dominated sorting selection, while for constrained case, an ensemble of different constraint handling methods is introduced. Experimental results suggest that both proposed techniques result in improvements of the performances.

Chapter 6

Application

In this chapter, the proposed multi-objective differential evolution (SNOV-DS) algorithm is employed to solve the nonlinear constrained multi-objective environmental/economic dispatch (EED) problem. This algorithm is integrated with superiority of feasible solution constraint handling technique.

6.1 Multi Objective Differential Evolution Algorithms to Solve Environmental Economic Dispatch Problem

In recent decades, environmental concern has been considered as one of the most important factors because of the fossil fuel fired electric generators exacerbate global warming. Multi-objective optimization algorithm is used to find the best tradeoff solution for this environmental/economic dispatch problem. In this work, the standard IEEE 30-bus six-generator test system is studied with fuel cost and emission as two conflicting objectives to be optimized simultaneously. A multi-objective differential evolution algorithm with SNOV-DS is compared with the classical NSGAII algorithm as well as some results in the literature by using other multi-objective optimization algorithms. The results confirm the effectiveness of the proposed approach with respect to computation speed and solution quality in relation to the state-of-the-art literature.

6.1.1 Environmental/Economic Dispatch Problem

The operations of electrical power systems are designed to meet the continuous variation of power demand. In essence, to ensure economic operation, power generation scheduling is performed based on two important tasks namely, unit commitment and economic dispatch, of which, latter is the topic of present

research. The purpose of traditional economic dispatch is to allocate generation levels to various generators in the system in order to meet the load demand in the most economic way. However, the optimum schedule obtained may not be the best, in case environmental criteria are also considered. The passage of the clean air act amendments in 1990 has forced the utilities to reduce their SO_2 and NO_X emissions by 40 percent from 1980 levels [163]. Therefore, apart from cost, emission objective must also be taken into account.

Environmental/Economic dispatch (EED) is a multi-objective problem having conflicting objectives, as the minimization of cost maximizes the pollution, leads to the necessity of trade-off analysis to define admissible dispatch policies for any demand level [183]. There has been much research pertaining to EED problem. The influence of power pools on power dispatch with environmental consideration is analyzed by taking emission as the constraints of the model and power is dispatched by minimizing cost as single objective [184]. The constrained emission approach is considered in [185], and solved for dynamic power dispatch using Han-Powell algorithm. The multi-objective problem considering cost, emission and line over load index as objectives is solved using ε-constrained method in [186]. The method involves the optimization of most preferred objective while considering other objectives as constraints, which are bounded by some allowable level ε. However, the approach provided only weakly non-dominated solution and that too by taking considerably large time. The economic emission dispatch is solved by weighted min-max approach along with risk in expected power deviations as third objective [187]. The EED problem with line flow security constraint is solved by weighted sum method in [188] to convert the multi-objective EED problem in single objective optimization problem. These methods generate the non-dominated solution by varying the weights, thus requiring multiple runs to generate the desired Pareto set of solutions. Moreover, these methods are not efficient in solving problems having non-convex Pareto optimal fronts. A linear programming based technique has been proposed in [189] which consider one objective at a time. But, the approach failed to give any information regarding the trade-off front. The multi-objective EED problem with security

constraints is solved for longitudinal power system by using weighted sum method and the combined objective is optimized by simulated annealing approach [190] and genetic algorithm [191]. Recently, the research focus has shifted towards handling both the objectives simultaneously. Over the past decade, this option has received much interest due to the development of a number of multi-objective evolutionary search strategies [1]. The combination of real coded genetic algorithm and simulated annealing techniques to solve the EED problem is given in [192] as Multi-objective Stochastic Search Technique (MOSST). The novel Nondominated Sorting Genetic Algorithm (NSGA) [193], Niched Pareto Genetic Algorithm (NPGA) [194] and Strength Pareto Evolutionary Algorithm (SPEA) [195] have been successfully applied to EED problem by Abido. Robert et al. [196], [197] developed elitist multi-objective evolutionary algorithm called NSGA-II and applied to EED problem. The EED problem with three unit and six unit systems was solved by Multi-objective Evolutionary programming approach (MOEP) in [198]. Multiobjective Particle Swarm Optimization (MOPSO) [199], Fuzzified multi-objective particle swarm optimization (FMOPSO) [200], Fuzzy clustering based Particle swarm optimization (FCPSO) [201], Multi-objective chaotic particle swarm optimization (MOCPSO) [202], and an improved quantum-behaved particle swarm optimization method (QPSO) [203], etc., constitutes the pioneering multiobjective approaches based on particle swarm optimization that have earlier been applied to solve the multi-objective environmental/economic dispatch problem. These algorithms were tested on standard IEEE 30 bus 6-generator system to obtain a trade-off between the cost and emission. In this work, we use a multi-objective differential evolution (MODE) based on SNOV-DS and combined with superiority of feasible solution constraint handling technique to solve the EED problem.

6.1.2 EED Problem Formulation

- Objectives

1) Fuel Cost Objective: The objective of classical economic dispatch is the minimization of total generation cost while satisfying several constraints. Mathematically,

$$Minimize \quad F(P_G) = \sum_{i=1}^{N_G} F_i(P_{Gi}) \tag{6-1-1}$$

In Eqn. (6-1-1), $F_i(P_{Gi})$ is the i^{th} generator cost function and is approximated as a quadratic function of the power output from the generating units, *i.e.*

$$F_i(P_{Gi}) = a_i P_{Gi}^2 + b_i P_{Gi} + c_i \qquad i = 1,2,3, ...,N_G \tag{6-1-2}$$

where, a_i, b_i, c_i are the cost coefficients, N_G represents the number of generating units in the system and P_{Gi} is the power output of the i^{th} generator. The generator cost function with non-smooth cost characteristics due to the valve point effect is given by

$$F_i(P_{Gi}) = a_i P_{Gi}^2 + b_i P_{Gi} + c_i + \left| e_i \sin\left(f_i \left(P_{Gi}^{\min} - P_{Gi} \right) \right) \right|, \ i = 1,2,3, ...,N_G \tag{6-1-3}$$

where e_i and f_1 are the cost of coefficients related to non-smooth part adding rectified sinusoidal effect with the quadratic part.

2) Emission Objective (NO_X): The minimum emission dispatch is the simultaneous minimization of classical economic dispatch including NO_X emission objective which is modeled as:

$$E\left(P_G\right) = \sum_{i=1}^{N} 10^{-2} (\alpha_i + \beta_i P_{Gi} + \gamma_i \left(P_{Gi}\right)^2) + \zeta_i \exp\left(\lambda_i P_{Gi}\right) \tag{6-1-4}$$

Here, $\alpha_i, \beta_i, \gamma_i, \zeta_i \text{ and } \lambda_i$ are the emission coefficients of the i^{th} generator. Although a number of other substances such as SO_2, dust particles, etc. are also emitted during the operation of electrical generators, the NO_X objectives are considered in this study as they are representative of the emissions generated by the fossil fuel fired generators.

- Constraints

1) Power Balance Constraint or Demand Constraint*:* The total power generated must cover total demand P_D and total transmission losses P_{LOSS}. Therefore,

$$\sum_{i=1}^{N_G} P_{Gi} - P_D - P_{Loss} = 0 \tag{6-1-5}$$

Transmission losses are calculated by solving the load flow problem given as:

$$P_{Gi} - P_{Di} - V_i \sum_{j=1}^{N_B} V_j \left[G_{ij} \cos\left(\delta_i - \delta_j\right) + B_{ij} \sin\left(\delta_i - \delta_j\right) \right] = 0 \tag{6-1-6}$$

$$Q_{Gi} - Q_{Di} - V_i \sum_{j=1}^{N_B} V_j \left[G_{ij} \sin\left(\delta_i - \delta_j\right) - B_{ij} \cos\left(\delta_i - \delta_j\right) \right] = 0 \tag{6-1-7}$$

Where, N_B : Total number of buses

Q_{Gi} : Reactive power generated at the i^{th} bus

P_{Di} : Real power at the i^{th} bus

Q_{Di} : Reactive Power at the i^{th} bus

G_{ij} : Transfer conductance between buses i and j

B_{ij} : Transfer susceptance between buses i and j

V_i & V_j : Voltage magnitude at bus i and j

δ_i & δ_j : Voltage angle at bus i and j

Equations (6-1-6) and (6-1-7) are nonlinear constraint equations, which are solved using Newton-Raphson method, and the load flow solution thus obtained gives all bus magnitudes and angles. Then the real power loss is calculated as

$$P_{Loss} = \sum_{k=1}^{N_L} g_k \left[V_i^2 + V_j^2 - 2 V_i V_j \cos\left(\delta_i - \delta_j\right) \right] \tag{6-1-8}$$

where N_L is the total number of transmission lines and g_k is the conductance of the k^{th} line which connects buses i and j.

2) Generation Capacity Constraints*:* The real output power of each generator is constrained by lower and upper limits, *i.e.*

$$P_{Gi}^{min} \leq P_{Gi} \leq P_{Gi}^{max} \quad (6\text{-}1\text{-}9)$$

where, P_{Gi}^{min} and P_{Gi}^{max} are the minimum and maximum operating outputs of unit *i* respectively.

3) Line flow constraints: The line flow constraint is used to avoid undesired line loadings due to power distribution. Thus, transmission line loading S_l is restricted by its upper limit as:

$$|S_{l_i}| \leq S_{l_i}^{max}, \qquad i = 1, 2, ..., N_L \quad (6\text{-}1\text{-}10)$$

6.1.3 Experimental Setup

The experimental analysis of all algorithms is done on the standard IEEE 30-bus 6-generator system [193]. This test system has load demand of 2.834 p.u. and all buses are connected through 41 transmission lines. The base MVA of the test system is considered as 100MVA. The detailed data of fuel cost and NO_X emission coefficients could be obtained from [195]. The constraint handling method used for both algorithms is superiority of feasible solution [181]. In this approach, feasible solutions are always better than the infeasible ones. If two infeasible solutions are compared, the solution with smaller constraint violation is better. On the other hand, two feasible solutions are compared using their fitness values. To demonstrate the effectiveness of the proposed MOEA, three different cases have been considered as follows:

Case 1: Lossless 6 units system considering only generation capacity constraint and the power balance constraint.

Case 2: Lossless system with larger number of units system considering only generation capacity constraint and the power balance constraint.

Case 2A: 60 units system & Case 2B: 120 units system.

Case 3: 6 units system including the losses and all constraints.

Two MOEA algorithms namely MODE and NSGA-II were implemented in Matlab 7.1 using Intel® Core 2 Duo CPU, 2.66 GHz, 3 GB RAM computer. The operating system of the computer is Microsoft Windows XP. The distribution indices for crossover and mutation operators are all set as 20 for NSGA_II while the parameter F and CR are chosen 0.5, 0.1 respectively for MODE. The population sizes are tuned according to the problems for both algorithms which are shown in Table 6-1-1.

Table 6-1-1 Tuned population size for both algorithms

	Population size	Max. achieve	Max. FEs
Case 1	100	100	10000
Case 2A	500	500	250000
Case 2B	500	1000	500000
Case 3	100	100	10000

To compare the performance of MODE and NSGA-II, three criteria are used: 1) The minimum cost of generation and corresponding emission; 2) The minimum emission and the corresponding cost; 3) The best compromise solution

6.1.4 Determining best compromise solution

After obtaining the final non-dominated solutions, we extracted the best compromise solution by using a fuzzy-based mechanism [195]. The best compromise solution means the best tradeoff solution between the two considered objectives. This approach is commonly used in EED optimization [193], [194], [195]. Membership value of each individual lying in Pareto-optimal set F_i is computed using the membership function defined in the following way:

$$\mu_i = \begin{cases} 1 & \text{if} \quad F_i \le F_i^{min} \\ \dfrac{F_i^{max} - F_i}{F_i^{max} - F_i^{min}} & \text{if} \quad F_i^{max} < F_i < F_i^{min} \\ 0 & \text{if} \quad F_i \ge F_i^{max} \end{cases} \tag{6-1-11}$$

where μ_i stands for the membership value of the i^{th} function (F_i). For each nondominated solution k, the normalized membership value ($\mu[k]$) is calculated using

$$\mu[k] = \frac{\sum_{i=1}^{M} \mu_i[k]}{\sum_{j=1}^{N_{pareto}} \sum_{i=1}^{M} \mu_i[j]} \tag{6-1-12}$$

M is the number of objectives and N_{pareto} is the number of solutions in pareto-optimal front. The best compromise solution is that for which $\mu[k]$ is the largest.

Each of the algorithms is run for ten times and the best run is used to test the three criteria above. In order to test the diversity and convergence as well as variation of different runs, two performance metrics [172] are used, namely R indicator (I_{R2}) and hypervolume difference to a reference set ($I_{\bar{H}}$).

6.1.5 Experimental results

The results obtained by MODE is compared against NSGA-II as well as those reported by other peer approaches, except for Case 2 as there is no previously reported results for this case. In all Tables, the best results are highlighted using boldface. The results are shown in Tables 6-1-2 to 6-1-4. From these tables, we can observe that the MODE algorithm based on SNOV-DS outperforms NSGA-II in all the cases with respect to simulation time and solution quality. The performance metrics shown in Table 6-1-9 demonstrate the superior performance of the MODE. We also observe in Table 6-1-9 that NSGA-II found feasible solutions only in one run out of ten runs for case 2A and no feasible solution was found for case 2B in all ten runs. The non-dominated solutions (first front) of the median run for both algorithms are shown in Fig. 6-1-1 to Fig. 6-1-4. As can been seen from these plots, MODE is able to generate a much better non-dominated front than NSGA-II. Note that Fig. 6-1-3 only plotted the first front of MODE as NSGA-II was not able to obtain any feasible solution. In order to demonstrate the advantage of MODE more

clearly, MODE was also compared with results in the literature in Tables 6-1-5 to 6-1-8. As revealed by these comparisons, the MODE algorithm performs the best among all the algorithms on the EED problem.

Table 6-1-2 Results for case 1

Case 1		MODE			NSGA-II	
	Best emission	Best cost	Best compromise	Best emission	Best cost	Best compromise
pg1	0.4097	0.1127	0.2493	0.4148	0.114	0.2677
pg2	0.4602	0.2954	0.3674	0.4271	0.2969	0.3426
pg3	0.5375	0.5082	0.5377	0.4806	0.4653	0.4704
pg4	0.3897	1.0379	0.7083	0.439	1.0084	0.7157
pg5	0.5381	0.521	0.5385	0.5235	0.5859	0.5818
pg6	0.4988	0.3589	0.4328	0.549	0.3635	0.4559
Cost ($/hr)	637.6974	**600.1538**	**608.8429**	635.0121	600.4101	609.4166
Emission	**0.1942**	0.2236	**0.2015**	0.1947	0.2218	0.2018
Simulation time		**1.753 seconds**			16.184 seconds	

Table 6-1-3 Results for case 2

Case 2A		MODE			NSGA-II	
	Best emission	Best cost	Best compromise	Best emission	Best cost	Best compromise
Cost ($/hr)	6.28E+03	**6.04E+03**	**6.13E+03**	6.86E+03	6.85E+03	6.85E+03
Emission	**1.9548**	2.0686	**1.9871**	2.4358	2.4468	2.4442
Simulation time		**51.96 seconds**			2.34 hours	
Case 2B						
Cost ($/hr)	1.24E+04	**1.21E+04**	**1.22E+04**	**No feasible solution was found**		
Emission	**3.9301**	4.1173	**3.997**			
Simulation time		**136.5 seconds**			4.92 hours	

Table 6-1-4 Results for case 3

Case 3		MODE			NSGA-II	
	Best emission	Best cost	Best compromise	Best emission	Best cost	Best compromise
pg1	0.4009	0.1768	0.3127	0.4068	0.1014	0.2283
pg2	0.468	0.3619	0.4112	0.4499	0.4751	0.3984
pg3	0.5442	0.7334	0.556	0.5578	0.8945	0.7509
pg4	0.3962	0.5945	0.5818	0.4152	0.6245	0.5277
pg5	0.544	0.5832	0.5411	0.5511	0.4864	0.5042
pg6	0.5106	0.4084	0.4599	0.4825	0.2748	0.4487
Cost ($/hr)	643.9757	**619.2015**	**624.4553**	641.3736	624.2947	625.2153
Emission	**0.1942**	0.2029	**0.1968**	**0**.1943	0.213	0.2
Simulation time		**78.42 seconds**			83.95 seconds	

Table 6-1-5 Best cost of MODE compared to methods in the literature (case 1)

Case 1	LP [189]	MOSST [192]	NSGA [193]	NPGA [194]	SPEA [195]	FCPSO [201]	MODE
P_G^1	0.1500	0.1125	0.1567	0.1080	0.1062	0.1070	0.1127
P_G^2	0.3000	0.3020	0.2870	0.3284	0.2897	0.2897	0.2954
P_G^3	0.5500	0.5311	0.4671	0.5386	0.5289	0.5250	0.5082
P_G^4	1.0500	1.0208	1.0467	1.0067	1.0025	1.0150	1.0379
P_G^5	0.4600	0.5311	0.5037	0.4949	0.5402	0.5300	0.521
P_G^6	0.3500	0.3625	0.3729	0.3574	0.3664	0.3673	0.3589
Cost ($/hr)	606.31	605.89	600.57	600.26	600.15	**600.13**	600.1538
Emission (tons/h)	0.2233	0.2222	0.2228	**0.2212**	0.2215	0.2223	0.2236

Table 6-1-6 Best emission of MODE compared to literature methods (case 1)

Case 1	LP [189]	MOSST [192]	NSGA [193]	NPGA [194]	SPEA [195]	FCPSO [201]	MODE
P_G^1	0.4000	0.4095	0.4394	0.4002	0.4116	0.4074	0.4097
P_G^2	0.4500	0.4626	0.4511	0.4474	0.4532	0.4577	0.4550
P_G^3	0.5500	0.5426	0.5105	0.5166	0.5329	0.5389	0.5363
P_G^4	0.4000	0.3884	0.3871	0.3688	0.3832	0.3837	0.3842
P_G^5	0.5500	0.5427	0.5553	0.5751	0.5383	0.5352	0.5348
P_G^6	0.5000	0.5152	0.4905	0.5259	0.5148	0.5110	0.5140
Cost ($/hr)	639.60	644.112	639.23	639.18	638.51	638.27	638.36
Emission (tons/h)	**0.1942**	**0.1942**	0.1944	0.1943	**0.1942**	**0.1942**	**0.1942**

Table 6-1-7 Best cost of MODE compared to methods in the literature (case 3)

Case 3	NSGA [193]	NPGA [194]	SPEA [195]	FCPSO [201]	MODE
P_G^1	0.1358	0.1127	0.1319	0.1596	0.1768
P_G^2	0.3151	0.3747	0.3654	0.3535	0.3619
P_G^3	0.8418	0.8057	0.7791	0.7974	0.7334
P_G^4	1.0431	0.9031	0.9282	0.9719	0.5945
P_G^5	0.0631	0.1347	0.1308	0.0862	0.5832
P_G^6	0.4664	0.5331	0.5292	0.4961	0.4084
Cost ($/h)	620.87	620.46	619.60	620.18	**619.2015**
Emission (tons/h)	0.2368	0.2243	0.2244	0.2283	**0.2029**

Table 6-1-8 Best emission of MODE compared to methods in the literature (case 3)

Case 3	NSGA [193]	NPGA [194]	SPEA [195]	FCPSO [201]	MODE
P_G^1	0.4403	0.4753	0.4419	0.4797	0.4009
P_G^2	0.4940	0.5162	0.4598	0.5287	0.468
P_G^3	0.7509	0.6513	0.6944	0.6711	0.5442
P_G^4	0.5060	0.4363	0.4616	0.5318	0.3962
P_G^5	0.1375	0.1896	0.1952	0.1257	0.544
P_G^6	0.5364	0.5988	0.6131	0.5299	0.5106
Cost ($/h)	649.24	657.59	651.71	651.62	**643.9757**
Emission (tons/h)	0.2048	0.2017	0.2019	0.2047	**0.1942**

Table 6-1-9 The performance metrics for MODE and NSGA-II

		MODE		NSGA-II	
		R_indicator	H_indicator	R_indicator	H_indicator
Case 1	Best	2.85E-05	9.15E-05	3.96E-04	1.16E-03
	Worst	2.90E-04	8.88E-04	2.03E-02	6.30E-02
	Mean	**1.16E-04**	**3.58E-04**	7.07E-03	2.18E-02
	Std	1.06E-04	3.22E-04	7.92E-03	2.46E-02
Case 2A	Best	5.10E-04	1.19E-03	**Only 1 run was able to locate feasible solutions**	
	Worst	3.15E-03	7.51E-03		
	Mean	**1.64E-03**	**3.90E-03**		
	Std	7.15E-04	1.71E-03		
Case 2B	Best	1.82E-05	5.21E-05	**No feasible solution was located**	
	Worst	3.90E-04	1.10E-03		
	Mean	**2.45E-04**	**6.90E-04**		
	Std	1.23E-04	3.47E-04		
Case 3	Best	2.92E-06	5.58E-06	3.79E-03	1.15E-02
	Worst	1.48E-04	4.57E-04	2.71E-02	8.60E-02
	Mean	**7.21E-05**	**2.25E-04**	1.06E-02	3.33E-02
	Std	6.10E-05	1.89E-04	8.41E-03	2.68E-02

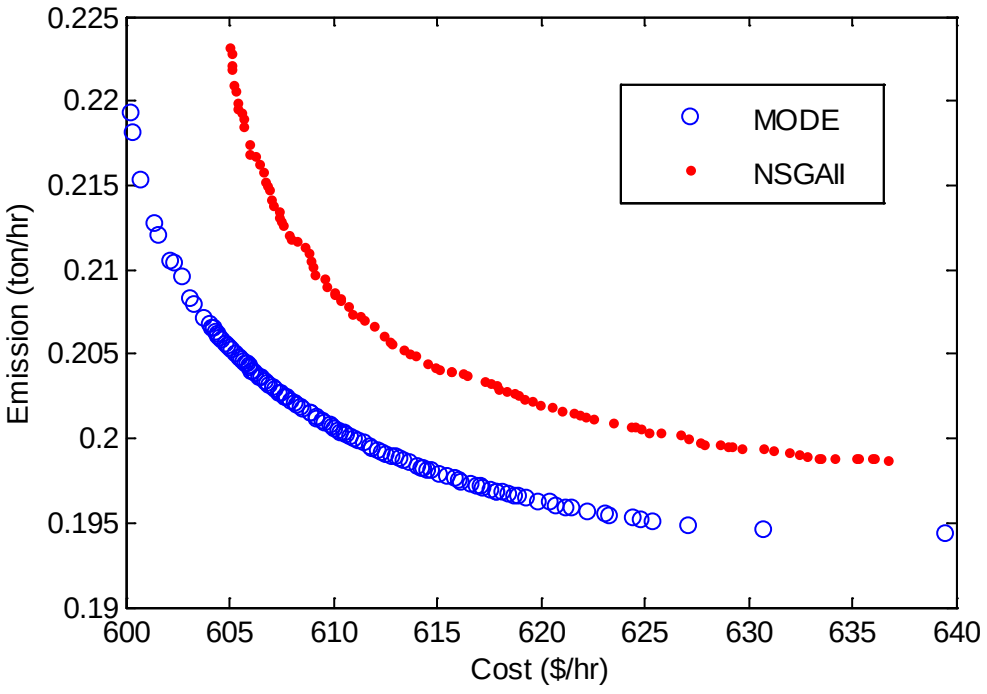


Fig. 6-1-1 Non-dominated solutions for the median run, Case 1

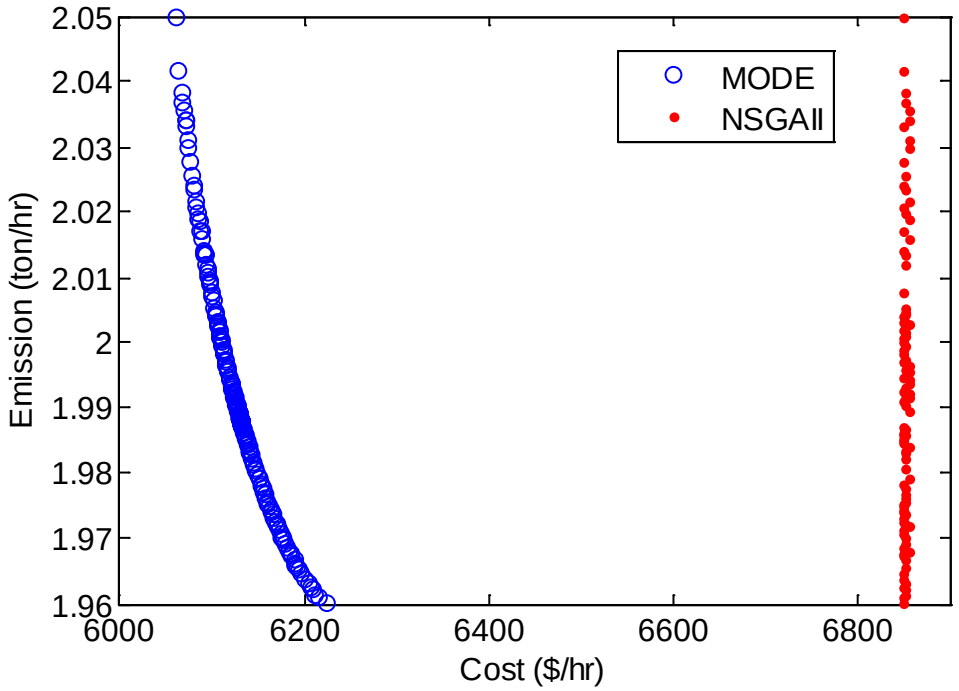


Fig. 6-1-2 Non-dominated solutions for the median run, Case 2A

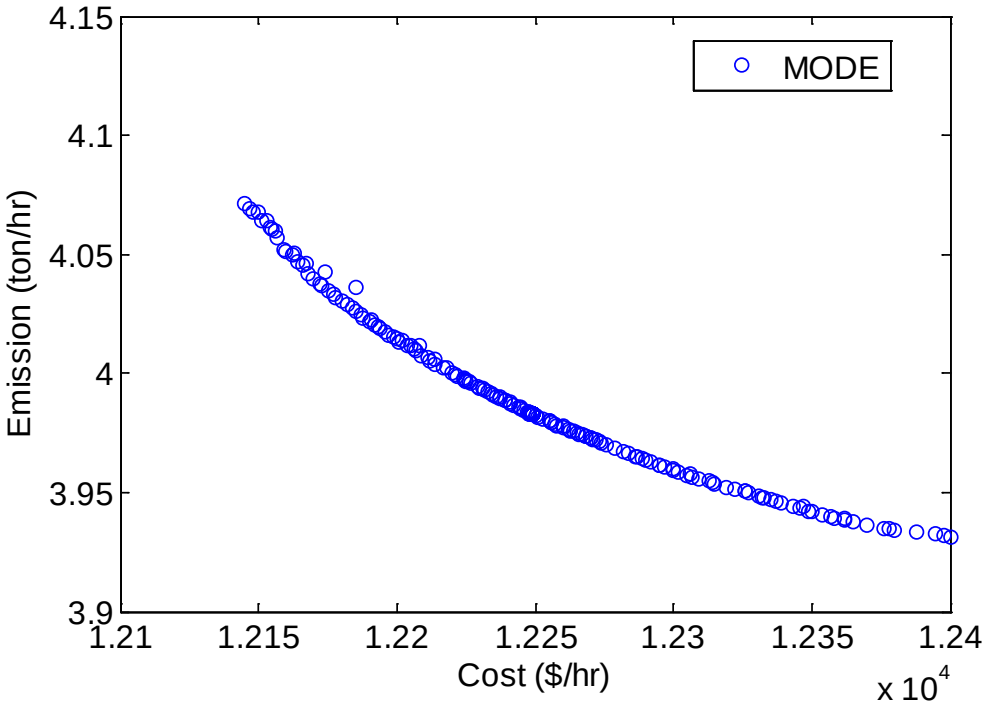


Fig. 6-1-3 Non-dominated solutions for the median run, Case 2B

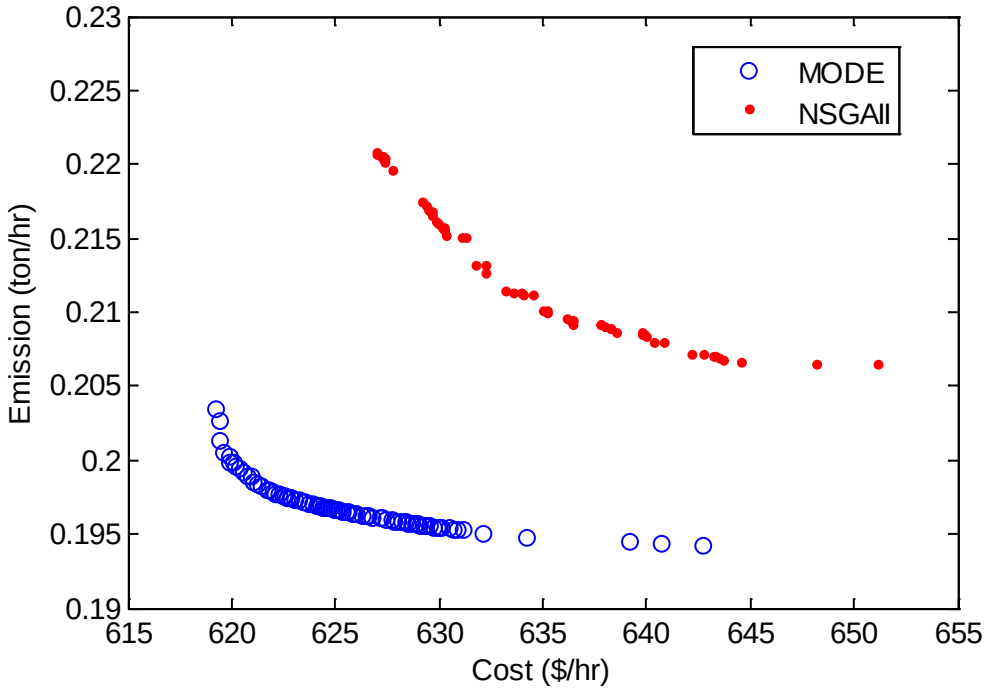


Fig. 6-1-4 Non-dominated solutions for the median run, Case 3

6.1.6 Conclusion

A multi-objective differential evolution based on SNOV-DS has been employed to solve the environmental / economic dispatch problem. The fuzzy based approach to find the best compromise solution is also employed. IEEE 30 bus 6-generator system has been chosen to test the proposed method in this paper. The proposed algorithm is compared with the classical NSGA-II and a number of algorithms in the literature. From the experimental results, we can observe that the proposed algorithm performs the best among the state-of-the-art algorithms for solving the EED problem.

Chapter 7

Conclusions and Future Work

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

7.1 Conclusions

This thesis proposed improved multi-modal optimization algorithms based on Differential Evolution (DE) and Particle Swarm Optimizer (PSO). The thesis also investigated the improvement of constrained and unconstrained multi-objective optimization. Conclusions can be summarized as follows.

A thorough literature survey on DE, covering its origin, fundamental rationale, improved DE variants and applications had been conducted. To improve the niching behavior of DE for multi-modal optimization, a Dynamic Grouping Crowding Differential Evolution (DGCDE) with ensemble of parameter had been presented. The algorithm started with a single population. Throughout the evolution process, the population was dynamically regrouped into 3 equal subpopulations every *m* generations, and each of the subpopulations was assigned with a separated set of parameters. The information was exchanged and shared between different subpopulations. The fittest offspring from the most suitable parameter set/population would survive and kept for the consequent generation. The novel structure increased the diversity of the particles and discouraged the premature convergence. Experiments were conducted on 12 benchmark functions and the comparison results with the original Crowding Differential Evolution (CDE) had been presented to show the superior performance of the DGCDE.

The author pointed out the drawback of the Species-based Differential Evolution (SDE) and introduced a modified SDE with a self-adaptive radius to overcome the difficulty of selecting the proper radius. The algorithm was able to adjust the niching radius according to the population. The author also proposed a novel

neighborhood mutation strategy and integrated it with three different niching DE algorithms, namely the Crowding Differential Evolution (CDE), Species-based Differential Evolution (SDE) and Sharing DE (ShDE). With the neighborhood mutation strategy, stable niches were able to be generated to find and maintain multiple optima during the evolution. The novel strategy reduced the size of the difference vector which showed great advantage in fine-tuning ability for multi-modal optimization. The proposed method was tested on a total of 29 problem instances and compared with a number of state-of-the-art multi-modal optimization approaches. The result showed that the neighborhood mutation was able to provide better and more consistent performance over the state-of-the-art multi-modal algorithms.

In Chapter 4, two PSO based niching algorithms had been proposed. First the author presented a distance based Locally Informed Particle Swarm optimization (LIPS). LIPS not only removed the need to specify a niching parameter, but also made use of Fully Informed Particle Swarm (FIPS) velocity update equation to ensure good usage of all neighborhood information. In addition, the Euclidean distance based neighborhood selection was able to increase the algorithm's local search and fine tuning ability. In the second work, the author enhanced the fine search ability of some existing PSO based niching algorithms by introducing a local search technique. The local search technique generated solutions around the *pbest* in current generation to increase the fine-tuning ability. The experimental results showed that the local search technique not only increased the probability of finding global/local optima but also speeded up the searching process or reduced the average number of function evaluations.

The improvement of constrained and unconstrained multi-objective optimization algorithms had been presented in Chapter 5. To replace the complex and time consuming non-domination sorting process, the author proposed a Summation of Normalized Objective Values and Diversified **S**election (SNOV-DS) method. This method was integrated with both multi-objective evolutionary programming (MOEP) and multi-objective differential evolution (MODE). With proposed

method, the selection speed was increased up to 60 times. The performances of both algorithms were also improved significantly. For constrained multi-objective optimization, instead of attempting to choose one method by extensive trial-and-error procedure, the author developed an ensemble of constraint handling method (ECHM) incorporating three different constraint handling techniques. ECHM made use of three different populations and each of them associated with one constraint handling method. ECHM exchanged information between different populations and kept the best offspring that was generated by the most suitable constraint handling method. The experimental results showed that ECHM overall outperformed the single constraint handling methods.

In Chapter 6, Environmental/Economic dispatch (EED) problem was introduced and solved by the proposed multi-objective differential evolution based on SNOV-DS. EED is a typical optimization problem with two conflicting objectives, as the minimization of cost and the minimization of pollution. The proposed algorithm was compared with NSGAII and a number of state-of-the-art algorithms on the standard IEEE 30-bus six-generator test system. The experimental results showed that the proposed algorithm performed the best among the state-of-the-art algorithms for solving the EED problem.

7.2 Future Work

- Extending the ensemble idea to other evolutionary algorithms

  Ensemble technique is a general idea. Other than ensemble of constraint handling methods, it can be extended and applied to many algorithms and places.

  - Ensemble of niching parameter: since the performance of niching algorithm depends on the parameter greatly. Different parameter sets work for different optimization problems. This provides a good perspective for the idea of ensemble. With ensemble of different parameter sets, we can always generate the good offspring using the most suitable parameter and maintain stable niches.

- Ensemble of different niching algorithms: as different niching algorithms are suitable for different problems and it is impossible to find one algorithm that can outperform the rest on all the test functions. With this observation, an ensemble of different niching algorithms can be useful to replace the procedure of choosing suitable niching method by extensive trial-and-error.
- Ensemble of different MOEAs: Different MOEAs also work differently on various kinds of problems. Several MOEAs can be ensembled together to make it works for most of the problems.

- Extending the ensemble idea in Chapter 3

  In Chapter 3, the ensemble idea makes use of three different DE parameter sets to solve multimodal problems. However, more parameter sets should be included if different problems are to be solved. In future, a pool of parameters can be used and incorporated with the probability selection strategy in [23] to further improve the performance of Crowding DE.

- Developing local search MODE

  The commonly used MODE is an extended version from single objective DE. For single objective DE, as the search goes on, the parameter space will be close to each other, which will be near to the optimal point. This ensures a very good local search at the end of the searching process. However, for MODE, the optimal solutions is a Pareto-front not a single point, the parameter space could be very far from each other even during the end of the search. This will lead to a poor local search. Therefore, a good local search MODE need to be developed to overcome this problem.

- Extending the proposed SNOV-DS method on other MOEAs

  The proposed SNOV-DS method can be extended to many multi-objective evolutionary algorithms which use the non-domination sorting as the

ranking and selection method. The SNOV-DS is expected to generat a faster simulation speed and a better performance.

- Applying the developed MOEAs to solve financial optimization problems

  In modern financial markets, portfolio allocation is one of the major problems faced by investors and fund managers. Investors need to choose several assets from thousands of available assets and form a single portfolio from an enormous set of possibilities in order to simultaneously maximize the return and minimize the risk. The problem of developing this return-risk efficient frontier can be formulated as a large-scale optimization problem with two objectives. As the large scale portfolio involves large number of variable to be optimized, it is almost impossible to use conventional optimization techniques, such as quadratic programming. Therefore, the proposed MOEAs can be a good choice to solve these kinds of problems in future.

- Extending the MOEA to solve the constrained asset allocations problems using the proposed constraint handling method

  The portfolio optimization/asset allocation problems mentioned above are unconstrained problems. However, in real work application, the problem has to be subjected to various constraints such as minimum/maximum transaction lot. In future, the proposed constraint handling method can be used to solve constrained portfolio optimization problems.

- Applying the developed MOEAs to solve linear antenna array design problem

  Linear antenna array design is one of the most important electromagnetic optimization problems of current interest. There are two design objectives involved: the minimum average SLL and null control in specific directions that are to be minimized simultaneously in order to achieve the optimal

spacing between the array elements. In future, the proposed MOEAs can be used to solve such problems.

- Applying the developed niching algorithms to solve real-world problems

  Many real-world optimization problems can be classified as multi-modal optimization problems. In future, the developed niching algorithms can be used to solve these practical problems such as clustering problems.

Author's Publications

Journal Papers

1. **B. Y. Qu**, P. N. Suganthan, "Neighborhood based Crowding Differential Evolution for Multi-modal Optimization", *accepted by IEEE Transaction on Evolutionary Computations*, Doi: 10.1109/TEVC.2011.2161873.
2. **B. Y. Qu**, P. N. Suganthan and J. J. Liang, "Niching Particle Swarm Optimization with Local Search Technique for Multi-modal Optimization", *accepted by Information Sciences*, Doi: 10.1016/j.ins.2012.02.011.
3. D. L. Jia, G. X. Zheng, **B. Y. Qu** and M. K. Khan, "A hybrid particle swarm optimization algorithm for high-dimensional problems," *Computer and Industrial Engineering*, July 2011.
4. S. Das, S. Maity, **B. Y. Qu**, P.N. Suganthan, "Real-parameter evolutionary multimodal optimization — A survey of the state-of-the-art", Vol. 1, No. 2, pp. 71-88, *Swarm and Evolutionary Computation*, June 2011.
5. A. Zhou, **B. Y. Qu**, H. Li, S. Z. Zhao, P. N. Suganthan and Q. Zhang, "Multiobjective Evolutionary Algorithms: A Survey of the State-of-the art," *Swarm and Evolutionary Computation*, Mar 2011.
6. **B. Y. Qu**, P. N. Suganthan, "Multi-Objective Evolutionary Algorithms based on the Summation of Normalized Objectives and Diversified Selection", *Information Sciences*, Vol. 180, No. 17, pp. 3170-3181, 1 Sept. 2010.
7. **B. Y. Qu**, P. N. Suganthan, "Constrained Multi-Objective Optimization Algorithm with Ensemble of Constraint Handling Methods", *Engineering Optimization*, DOI: 10.1080/0305215X.2010.493937.
8. **B. Y. Qu**, P. N. Suganthan, "Multi-Objective Differential Evolution with Diversity Enhancement", *Journal of Zhejiang University-SCIENCE C*, Vol. 11, No. 7, pp. 538-543, 2010.
9. S. Pal, **B. Y. Qu**, S. Das, and P. N. Suganthan, "Linear Antenna Arrays Synthesis with Constrained Multi-objective Differential Evolution", *Progress*

in Electromagnetics Research, PIER B, Vol. 21, pp. 87-111, 2010.

10. **B. Y. Qu**, P. N. Suganthan and S. Das, "Distance based locally informed particle swarm optimization for multi-modal optimization," accepted *IEEE Transaction on Evolutionary Computation*.

Conferences

1. **B. Y. Qu**, P. N. Suganthan, "Multi-objective Evolutionary Programming without Non-domination Sorting is up to Twenty Times Faster", *IEEE Congress on Evolutionary Computation,* pp. 2934-2939, Norway, May 2009.
2. **B. Y. Qu** and P. N. Suganthan, "Constrained Multi-Objective Optimization Algorithm with Diversity Enhanced Differential Evolution", *IEEE Congress on Evolutionary Computation,* Barcelona, Spain, pp. 1675-1679, July 2010.
3. **B. Y. Qu** and P. N. Suganthan, "Novel Multi-modal Problems and Differential Evolution with Ensemble of Restricted Tournament Selection", *IEEE Congress on Evolutionary Computation*, Barcelona, Spain, pp. 3480-3486, July 2010.
4. R. Mallipeddi, P. N. Suganthan and **B. Y. Qu**, "Diversity Enhanced Adaptive Evolutionary Programming for Solving Single Objective Constrained Optimization Problems", *IEEE Congress on Evolutionary Computation,* pp. 2106 - 2113**,** Norway, May 2009.
5. **B. Y. Qu**, V. R. Pandi, B. K. Panigrahi and P. N. Suganthan, "Multi Objective evolutionary programming to Solve Environmental Economic Dispatch Problem"*, 11th International Conference on Control, Automation, Robotics and Vision*, Singapore, 2010.
6. S. Z. Zhao, **B. Y. Qu**, M. Willjuice Iruthayarajan, S. Baskar and P. N. Suganthan, "Multi-objective Robust PID controller tuning using multi-objective differential evolution", *11th International Conference on Control, Automation, Robotics and Vision, Singapore,* 2010.
7. **B. Y. Qu**, V. R. Pandi, B. K. Panigrahi, P. N. Suganthan and S. Z. Zhao, "Two Local Best Based Multi Objective Particle Swarm Optimization to Solve Environmental Economic Dispatch Problem", India.
8. **B. Y. Qu**, P. Gouthanan, and P. N. Suganthan, "Dynamic Grouping Crowding

Differential Evolution with Ensemble of Parameters for Multi-modal Optimization", *SEMCCO: Int. Conf. on Swarm, Evolutionary and Memetic Computing*, Chennai, India, 2010.

9. **B. Y. Qu**, P. N. Suganthan and S. Z. Zhao, "Current Based Fitness Euclidean-distance Ratio Particle Swarm Optimizer for Multi-modal Optimization", *Second World Congress on Nature and Biologically Inspired Computing*, Japan, 2010.
10. S. Z. Zhao, P. N. Suganthan and **B. Y. Qu**, "Multi-objective particle swarm optimizer with dynamic e-dominance sorting," *Second World Congress on Nature and Biologically Inspired Computing*, Japan, 2010.
11. **B. Y. Qu** and P. N. Suganthan, "Modified species-based differential evolution with self –adaptive radius for multi-modal optimization," *Inernational Conference on Computational Problem-solving*, China, 2010.
12. **B. Y. Qu** and P. N. Suganthan, "Multi-objective differential evolution based on the summation of normalized objectives and improved selection method," *SDE-2011 IEEE Symposium on Differential Evolution*, Paris, France, 2011.
13. J. J. Liang, **B. Y. Qu**, S. T. Ma and P. N. Suganthan, "Memetic Fitness Euclidean-Distance Particle Swarm Optimization for Multi-modal Optimization," *ICIC*, Zhengzhou, China, 2011.
14. J. J. Liang, **B. Y. Qu** and P. N. Suganthan, "Dynamic Multi-Swarm Particle Swarm Optimization for Multi-Objective Optimization Problems," has been accepted by 2012 IEEE Congress on Evolutionary Computation.
15. J. J. Liang, S. T. Ma and **B. Y. Qu**, "Strategy Adaptative Memetic Crowding Differential Evolution for Multimodal Optimization," has been accepted by 2012 IEEE Congress on Evolutionary Computation.
16. J. J. Liang, X. B, Mao, **B. Y. Qu**, B. Niu and T. J. Chen, "Elite Multi-Group Differential Evolution," has been accepted by 2012 IEEE Congress on Evolutionary Computation.
17. **B. Y. Qu**, J. J. Liang, P. N. Suganthan & T. J. Chen, "Ensemble of Clearing Differential Evolution for Multi-modal Optimization," has been accepted by ICSI2012.

Bibliography

[1] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. Chichester, England: John Wiley & Sons, 2002.

[2] A. M. Anile, V. Cutello, G. Nicosia, R. Rascuna, and S. Spinella, "Comparison among Evolutionary Algorithms and Classical Optimization Methods for Circuit Design Problems," in *IEEE Conference on Evolutionary Computation*. vol. 1 Vancouver, Canada, pp. 765-772, 2005.

[3] J. H. Holland, *Adaptation in natural and artificial systems*. The University of Michigan Press, Ann Arbor, MI, 1975.

[4] L. J. Fogel. "Autonomous automata," *Industrial Research*, 4: pp. 14-19, 1962.

[5] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. Wiley, New York, 1966.

[6] I. Rechenberg , "Evolution strategy: optimization of technical systems according to principles of biological evolution" *Ph.D. dissertation*, TU Berlin, 1971.

[7] R. Storn and K. V. Price, "Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces," ICSI, *Tech. Rep*. TR-95-012, 1995 [Online]. Available: http://icsi.berkeley.edu/storn/litera.html.

[8] K. V. Price, "Differential evolution vs. the functions of the 2nd ICEO," in *Proc. IEEE Int. Conf. Evol. Comput.*, pp. 153–157, 1997.

[9] K. V. Price and R. Storn, "Differential evolution: A simple evolution strategy for fast optimization," *Dr. Dobb's J.*, vol. 22, no. 4, pp. 18–24, 1997.

[10] R. Storn, "On the usage of differential evolution for function optimization," in *Proc. North Am. Fuzzy Inform. Process. Soc.*, pp. 519–523, 1996.

[11] 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.

[12] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks*, Piscataway, NJ, pp. 1942-1948, 1995.

[13] Barricelli, Nils Aall, "Esempi numerici di processi di evoluzione," *Methodos*: 45-68, 1954.

[14] Fraser, Alex, "Simulation of genetic systems by automatic digital computers. I. Introduction". *Aust. J. Biol. Sci.* vol. 10, pp. 484-491, 1957.

[15] J. H. Holland, "Outline for a Logical Theory of Adaptive Systems," *Journal of the Association for Computing Machinery (ACM)*, vol. 9, pp. 297 - 314 1962.

[16] D. E. Goldberg and J. Richardson, “Genetic algorithms with sharing for multi-modal function optimization,” *in Proceedings of the second international conference on genetic algorithms*, pp.41-49, 1987.
[17] I. Rechenberg, "Cybernetic solution path of an experimental problem," in *Library Translation No 1122* Royal Aircraft Establishment, Farnborough, UK, 1965.
[18] R. Mallipeddi, “Ensemble strategies with evolutionary programming and differential evolution for solving single objective optimization problems,” Doctoral dissertation, Nanyang Technological University, 2010.
[19] A. Pétrowski, “A clearing procedure as a niching method for genetic algorithms,” Proc. of the IEEE Int. Conf. on Evolutionary Computation, New York, USA, pp. 798–803, 1996.
[20] T. Back, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford University Press, 1996.
[21] S. Das and P. N. Suganthan, "Differential Evolution: A Survey of the State-of-the-art", *IEEE Trans. on Evolutionary Computation*, DOI: 10.1109/TEVC.2010.2059031.
[22] K. V. Price, R. M. Storn, and J. A. Lampinen, "Differential Evolution : A Practical Approach to Global Optimization," in *Natural Computing Series* Berlin: Springer, 2005.
[23] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential Evolution Algorithm with Strategy Adaptation for Global Numerical Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, pp. 398-417, April 2009.
[24] V. L. Huang, “Self adaptive differential evoltion,” Doctoral dissertation, Nanyang Technological University, 2009.
[25] S. Das, A. Abraham, and U. K. Chakraborthy, "Differential Evolution Using a Neighborhood-Based Mutation Operator," *IEEE Transactions on Evolutionary Computation*, vol. 13, pp. 526-553, 2009.
[26] R. Gämperle, S. D. Müller, and P. Koumoutsakos, "A Parameter Study for Differential Evolution," in *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation* Interlaken, Switzerland: WSEAS Press, pp. 293-298, 2002.
[27] S. Das, A. Konar, and U. K. Chakraborty, "Two Improved Differential Evolution Schemes for Faster Global Search," *Proceedings of the 2005 conference on Genetic and Evolutionary Computation*, pp. 991-998, 2005.
[28] D. Zaharie, "Control of Population Diversity and Adaptation in Differential Evolution Algorithms," *In: Proceedings of the 9th International Conference on Soft Computing, Brno 41-46*, 2003.
[29] R. Storn and K. Price, "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, pp. 341-359, 1997.
[30] J. Rönkkönen, S. Kukkonen, and K. V. Price, "Real-parameter optimization with differential evolution," in *IEEE Congress on Evolutionary Computation*, pp. 506-513, 2005.

[31] M. M. Ali and A. Törn, “Population set based global optimization algorithms: Some modifications and numerical studies,” *Comput. Oper. Res.*, vol. 31, no. 10, pp. 1703–1725, 2004.
[32] J. Brest, S. Greiner, B. Boˇskovi´c, M. Mernik, and V. ˇZ 2306 umer, “Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems,” *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 646–657, 2006.
[33] J. Liu and J. Lampinen, “A fuzzy adaptive differential evolution algorithm,” *Soft Comput. A Fusion Founda. Methodol. Applicat.*, vol. 9, no. 6, pp. 448–462, 2005.
[34] H. Yousefi, H. Handroos, and A. Soleymani, "Application of Differential Evolution in System Identification of a Servo-hydraulic System with a Flexible Load," *Mechatronics*, vol. 18, pp. 513-528, 2008.
[35] H. Tang, S. Xue, and C. Fan, "Differential Evolution Strategy for Structural System Identification," *Computers and Structures*, vol. 86, pp. 2004-2012, 2008.
[36] A. Chatterjee, "Differential Evolution Tuned Fuzzy Supervisor Adapted Extended Kalman Filtering for Slam Problems in Mobile Robots," *Robotica*, vol. 27, pp. 411-423, 2009.
[37] J. Chakraborty, A. Konar, L. C. Jain, and U. K. Chakraborty, "Cooperative Multi-Robot Path Planning using Differential Evolution," *Journal of Intelligent and Fuzzy Systems*, vol. 20, pp. 13-27, 2009.
[38] L. S. Coelho and V. C. Mariani, "Combining of Chaotic Differential Evolution and Quadratic Programming for Economic Dispatch Optimization with valve-point effect " *IEEE Transactions on Power Systems*, vol. 21, pp. 989–996, 2006.
[39] X. Yuan, L. Wang, Y. Zhang, and Y. Yuan, "A Hybrid Differential Evolution Method for Dynamic Economic Dispatch with Valve-point Effects," *Expert Systems with Applications*, vol. 36, pp. 4042-4048, 2009.
[40] N. Noman and H. Iba, "Differential Evolution for Economic Load Dispatch Problems," *Electric Power Systems Research*, vol. 78, pp. 1322-1331, 2008.
[41] Y. P. Chang and C. Low, "An Ant Direction Hybrid Differential Evolution Heuristic for the Large-scale Passive Harmonic Filters Planning Problem," *Expert Systems with Applications*, vol. 35, pp. 894-904, 2008.
[42] K. A. Michalski, "Electromagnetic Imaging of Circular-cylindrical Conductors and Tunnels using a Differential Evolution Algorithm," *Microwave and Optical Technology Letters*, vol. 27, pp. 330-334, 2000.
[43] D. G. Kurup, M. Himdi, and A. Rydberg, "Synthesis of Uniform Amplitude Unequally Spaced Antenna Arrays using the Differential Evolution Algorithm," *IEEE Transactions on Antennas and Propagation*, vol. 51, pp. 2210-2217, 2003.
[44] S. Yang and Z. Nie, "Mutual Coupling Compensation in Time Modulated Linear Antenna Arrays," *IEEE Transactions on. Antennas and Propagation*, vol. 53, pp. 372-376, 2005.

[45] R. Xu, G. K. Venayagamoorthy, and D. C. Wunsch, "Modeling of Gene Regulatory Networks with Hybrid Differential Evolution and Particle Swarm Optimization," *Neural Networks,* vol. 20, pp. 917-927, 2007.
[46] N. Noman and H. Iba, "Inferring Gene Regulatory Networks using Differential Evolution witl Local Search Heuristics," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 4, pp. 634-647, 2007.
[47] H. Silverio and L. R. Bitello, "A Differential Evolution Approach for Protein Folding using a Lattice Model," *Journal of Computer Science and technology*, vol. 22, 2007.
[48] B. V. Babu and R. Angira, "Modified Differential Evolution for Optimization of Non-linear Chemical Processes," *Computers & Chemical Engineering,* vol. 30, pp. 989-1002, 2006.
[49] B. V. Babu and K. K. N. Sastry, "Estimation of Heat Transfer Parameters in a Trickle-bed Reactor using Differential Evolution and Orthogonal Collocation," *Computers & Chemical Engineering*, vol. 23, pp. 327-339, 1999.
[50] S. Das and A. Konar, "Automatic Image Pixel Clustering with an Improved Differential Evolution " *Applied Soft Computing,* vol. 9, pp. 226-236, 2009.
[51] V. Aslantas, "An Optimal Robust Digital Image Watermarking based on SVD using Differential Evolution Algorithm," *Optics Communications,* vol. 282, pp. 769-777, 2009.
[52] J. Ilonen, J. K. Kamarainen, and J. Lampinen, "Differential Evolution Training Algorithm for Feed-forward Neural networks," *Neural Processing Letters,* vol. 17, pp. 93-105, 2003.
[53] L. S. Coelho and F. A. Guerra, "B-spline Neural Network Design using Improved Differential Evolution for Identification of an Experimental Nonlinear Process," *Applied Soft Computing,* vol. 8, pp. 1513-1522, 2008.
[54] R. Storn, "Differential Evolution Design of an IIR-filter," in *IEEE International Conference on Evolutionary Computation* IEEE, pp. 268-273, 1996.
[55] S. Das and A. Konar, "Two-dimensional IIR Filter Design with Modern Search Heuristics: A comparative study," *International Journal of Computational Intelligence and Applications,* vol. 6, pp. 329-355, 2006.
[56] W. D. Chang, "Two-dimensional Fractional-order Digital Differentiator Design by using Differential Evolution Algorithm," *Digital Signal Processing,* vol. 19, pp. 660-667, 2009.
[57] Z. Fan, J. Liu, T. Sørensen, and P. Wang, "Improved Differential Evolution based on Stochastic Ranking for Robust Layout Synthesis of MEMS Components," *IEEE Transactions on Industrial Electronics,* vol. 56, pp. 937-948, 2009.
[58] R. Storn, "System Design by Constraint Adaptation and Differential Evolution," *IEEE Transactions on Evolutionary Computation,* vol. 3, pp. 22-34, 1999.

[59] G. C. Onwubolu, "Design of Hybrid Differential Evolution and Group Method of Data Handling Networks for Modeling and Prediction," *Information Sciences,* vol. 178, pp. 3616-3634, 2008.
[60] M. K. Venu, R. Mallipeddi, and P. N. Suganthan, "Fiber Bragg Grating Sensor Array Interrogation using Differential Evolution," *Optoelectronics and Advanced Materials - Rapid Communications*, vol. 2, pp. 682-685, 2008.
[61] R. Poli, J. Kennedy and T. Blackwell, "Particle swarm optimization an overview," *Swarm Intelligence*, pp. 33-57, DOI: 10.1007/s11721-007-0002-0.
[62] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proc. of the IEEE Congress on Evolutionary Computation (CEC 1998)*, Piscataway, NJ, pp. 69-73, 1998.
[63] J. J. Liang, "Self adaptive differential evoltion," Doctoral dissertation, Nanyang Technological University, 2010.
[64] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proc. of the 1999 Congress on Evolutionary Computation,* Piscataway, NJ, pp. 1945-1950, 1999.
[65] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," *Evolutionary Programming VII: Proceedings of the Seventh Annual Conference on Evolutionary Programming,* New York, pp. 591-600, 1998.
[66] R. C. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources," in *Proc. of the IEEE Congress on Evolutionary Computation (CEC 2001),* pp. 81-86, Seoul, Korea, 2001.
[67] J. Kennedy, "The behavior of particles." In V.W. Porto, N. Saravanan, D.Waagen & A. E. Eiben (Eds.), *Lecture notes in computer science*. *Evolutionary programming VII: proceedings of the 7-th annual conference on evolutionary programming,* pp. 581–589, San Diego, CA. Berlin: Springer, 1998.
[68] M. Clerc, "The swarm and the queen: towards a deterministic and adaptive particle swarm optimization," in *Proc of the IEEE Congress on Evolutionary Computation (CEC 1999),* pp. 1951-1957, 1999.
[69] 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.
[70] J. Kennedy, R. C. Eberhart and Y. Shi, *Swarm Intelligence*, Morgan Kaydmann, 2001.
[71] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proc. of the IEEE Congress on Evolutionary Computation (CEC 2002),* Honolulu, Hawaii USA, 2002.
[72] P. N. Suganthan, "Particle swarm optimizer with neighborhood operator," in *Proc. of the IEEE Congress on Evolutionary Computation (CEC 1999),* Piscataway, NJ, pp. 1958-1962, 1999.

[73] T. Peram, K. Veeramachaneni, and C. K. Mohan, "Fitness-distance-ratio based particle swarm optimization," in *Proc. IEEE Swarm Intelligence System,* Indianapolis, Indiana, USA, pp. 174-181, 2003.
[74] J. J. Liang, and P. N. Suganthan, "Dynamic Multi-Swarm Particle Swarm Optimizer," *IEEE Swarm Intelligence Symposium*, pp. 124-129, Pasadena, CA, USA, June 2005.
[75] M. Clerc, "Stagnation analysis in particle swarm optimization or what happens when nothing happens," *Technical Report CSM-460*, Department of Computer Science, University of Essex, August 2006.
[76] 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.
[77] R. Poli, "An analysis of publications on particle swarm optimization applications," *Technical Report CSM-469*, Department of Computer Science, University of Essex.
[78] M. Donelli, R. Azaro, F. D. Natale, and A. Massa, "An innovative computational approach based on a particle swarm strategy for adaptive phased-arrays control," *IEEE Transactions on Antennas and Propagation*, vol. 54, no. 3, pp. 888–898, 2006.
[79] D. Boeringer and D. Werner, "Particle swarm optimization versus genetic algorithms for phased array synthesis," *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 3, pp. 771–779, 2004.
[80] M. Khodier and C. Christodoulou, "Linear array geometry synthesis with minimum sidelobe level and null control using particle swarm optimization," *IEEE Transactions on Antennas and Propagation*, vol. 53, no. 8, pp. 2674–2679, 2005.
[81] N. Jin and Y. Rahmat-Samii, "Parallel particle swarm optimization and finite-difference time -domain (PSO/FDTD) algorithm for multiband and wide-band patch antenna designs," *IEEE Transactions on Antennas and Propagation*, vol. 53, no. 11, pp. 3459–3468, 2005.
[82] Y. Kim, S. Keely, J. Ghosh, and H. Ling, "Application of artificial neural networks to broadband antenna design based on a parametric frequency model," *IEEE Transactions on Antennas and Propagation*, vol. 55, no. 3 I, pp. 669–674, 2007.
[83] S. Selvan, C. Xavier, N. Karssemeijer, J. Sequeira, R. Cherian, and B. Dhala, "Parameter estimation in stochastic mammogram model by heuristic optimization techniques," *IEEE Transactions on Information Technology in Biomedicine*, vol. 10, no. 4, pp. 685–695, 2006.
[84] K. Veeramachaneni, L. Osadciw, and P. Varshney, "An adaptive multi-modal biometric management algorithm," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 35, no. 3, pp. 344–356, 2005.
[85] Z.-L. Gaing, "A particle swarm optimization approach for optimum design of PID controller in AVR system," *IEEE Transactions on Energy Conversion*, vol. 19, no. 2, pp. 384–391, 2004.

[86] J. Heo, K. Lee, and R. Garduno-Ramirez, "Multiobjective control of power plants using particle swarm optimization techniques," *IEEE Transactions on Energy Conversion*, vol. 21, no. 2, pp. 552–561, 2006.
[87] H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, and Y. Nakanishi, "A particle swarm optimization for reactive power and voltage control considering voltage security assessment," *IEEE Transactions on Power Systems*, vol. 15, no. 4, pp. 1232–1239, 2000.
[88] W. Wang, Y. Lu, J. Fu, and Y. Z. Xiong, "Particle swarm optimization and finite-element based approach for microwave filter design," *IEEE Transactions on Magnetics*, vol. 41, no. 5, pp. 1800–1803, 2005.
[89] W.-C. Liu, "Design of a multiband CPW-fed monopole antenna using a particle swarm optimization approach," *IEEE Transactions on Antennas and Propagation*, vol. 53, no. 10, pp. 3273–3279, 2005.
[90] S. Kannan, S. Slochanal, and N. Padhy, "Application and comparison of metaheuristic techniques to generation expansion planning problem," *IEEE Transactions on Power Systems*, vol. 20, no. 1, pp. 466–475, 2005.
[91] J. Vlachogiannis and K. Lee, "Determining generator contributions to transmission system using parallel vector evaluated particle swarm optimization," *IEEE Transactions on Power Systems,* vol. 20, no. 4, pp. 1765–1774, 2005.
[92] C.-F. Juang and C.-H. Hsu, "Temperature control by chip implemented adaptive recurrent fuzzy controller designed by evolutionary algorithm," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, no. 11, pp. 2376–2384, 2005.
[93] A. Adly and S. Abd-El-Hafiz, "Using the particle swarm evolutionary approach in shape optimization and field analysis of devices involving nonlinear magnetic media," *IEEE Transactions on Magnetics*, vol. 42, no. 10, pp. 3150–3152, 2006.
[94] W. Slade, H. Ressom, M. Musavi, and R. Miller, "Inversion of ocean color observations using particle swarm optimization," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 42, no. 9, pp. 1915–1923, 2004.
[95] M. Wachowiak, R. Smolikova, Y. Zheng, J. Zurada, and A. Elmaghraby, "An approach to multi-modal biomedical image registration utilizing particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 289–301, 2004.
[96] Y. Song, Z. Chen, and Z. Yuan, "New chaotic PSO-based neural network predictive control for nonlinear process," *IEEE Transactions on Neural Networks*, vol. 18, no. 2, pp. 595–601, 2007.
[97] C.-M. Huang and F.-L.Wang, "An RBF network with OLS and EPSO algorithms for real-time power dispatch," *IEEE Transactions on Power Systems*, vol. 22, no. 1, pp. 96–104, 2007.
[98] J. Vlachogiannis and K. Lee, "A comparative study on particle swarm optimization for optimal steady-state performance of power systems," *IEEE Transactions on Power Systems*, vol. 21, no. 4, pp. 1718–1728, 2006.

[99] A. Esmin, G. Lambert-Torres, and A. Z. De Souza, "A hybrid particle swarm optimization applied to loss power minimization," *IEEE Transactions on Power Systems*, vol. 20, no. 2, pp. 859–866, 2005.
[100] B. Liu, L. Wang, and Y.-H. Jin, "An effective PSO-based memetic algorithm for flow shop scheduling," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 1, pp. 18–27, 2007.
[101] T.-Y. Lee, "Operating schedule of battery energy storage system in a time-of-use rate industrial user with wind turbine generators: a multipass iteration particle swarm optimization approach," *IEEE Transactions on Energy Conversion*, vol. 22, no. 3, pp. 774–782, 2007.
[102] Engelbrecht.A.P, *Computational Intelligence An Introduction*, John Wiley & Sons, Chichester, 2nd ed. 2007.
[103] K.A. De Jong, "An Analysis of the Behavior of a Class of Genetic Adaptive Systems", Doctoral Dissertation, University of Michigan, 1975.
[104] S. W. Mahfoud, "A comparison of parallel and Sequential Niching Method" *International Conference on Genetic algorithms*, pp136-143, 1995.
[105] K. Deb and D. E. Goldberg, "An investigation of niche and species formation in genetic function optimization," in *Proc. 3rd Int. Conf. Genetic Algorithms*, J. D. Schaffer, Ed. San Mateo, CA: Morgan Kaufmann, pp. 42–50, 1989.
[106] S. Mahfoud, "Niching method for genetic algorithms". Doctoral dissertation. Technical report, Department of Computer Science, University of Lllinoisat Urbana-Champaign, Urbana, IL, USA, Illinois Genetic Algorithms Laboratory (IlliGAL) report No.95001, 1995.
[107] S. Mahfoud. "Crowding and preselection revisited." In Parallel Problem Solving from Nature2, pages27– 37, 1992.
[108] S. Mahfoud, "Simple analytical models of genetic algorithms for multi-modal function optimization." Technical report, Department of Computer Science, University of Lllinoisat Urbana-Champaign, Urbana, IL, USA, Illnois Genetic Algorithm Laboratory Report No.94005, 1993.
[109] O. Mengsheol and D. Goldberg, "Probabilistic crowding: Deterministic crowding with probabilistic replacement." *In Proceedings of the Genetic and Evolutionary Computation Conference*-1999(GECCO -99), pages409–416, 1999.
[110] G. Harick, "Finding multi-modal solutions in problems of bounded difficulty." Technical report, Illinois Genetic Algorithms Laboratory, report No.94002, 1994.
[111] G. Harick. "Finding multi-modal solutions using restricted tournament selection." *In Proceedings of the Sixth International Conference on Genetic Algorithms* (ICGA-95), pp. 24–31, 1997.
[112] A. Petrowski. "An efficient hierarchical clustering Technique for speciation evolution", Technical report, Institute National des Telecommunications, Evry, France, Technique Report, 1997.
[113] X. Yin and N. Germay. "Investigations on solving load flow problems by genetic algorithms.", In Electric Power System Research, pages151–163, 1991.

[114] X. Yin and N.Germay, “A fast genetic algorithm with sharing scheme using cluster analysis methods in multi-modal function optimization.” *In Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms*, pages450–457, 1993.
[115] A. Ghosh and S. Dehuri, “Evolutionary algorithms for multi-criterion optimization: a survey,” *International Journal of Computing & Information Sciences*, Vol. 2, pp. 38-57, 2004.
[116] J. D. Schaffer, “Multiobjective optimization with vector evaluated genetic algorithms,” *Doctoral Dissertation*, Vanderbilt University, Nashville, Tannessee.
[117] F. Kursawe, “A variant of evolution strategies for vector optimization,” in *Parellel problem solving from nature I (PPSN-I)*, pp. 193-197, 1990.
[118] C. M. Fonseca, and P. J. Fleming, “Genetic algorithms for multi-objective optimization: formulation, discussion and generalization,” in *Proceedings of the fifth international conference on genetic algorithms*, pp 415-423, 1993.
[119] C. M. Fonseca and P. J. Fleming, “An overview of evolutionary algorithms in multi-objective optimization,” *Evolutionary Computation*, vol. 3, pp. 1-16, 1995.
[120] J. Horn, N. Nafploitis and D. E. Goldberg, “A niched Pareto genetic algorithm for multi-objective optimization,” in *Proceedings of the first IEEE conference on evolutionary computation*, pp. 82-87, 1994.
[121] E. Zizler and L. Thiele, “Multi-objective optimization using evolutionary algorithms-a comparative case study,” *Parallel problem solving from nature*, pp. 292-301, Springer, Berlin, Germany, 1998.
[122] E. Zizler and L. Thiele, “Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach,” *IEEE Transactions on Evolutionary Computation*, pp. 257-271, 1999.
[123] K. Deb, “Multi-objective genetic algorithms: problem difficulties and construction of test problems,” *Evolutionary Computing Journal*, pp. 205-230, 1999.
[124] N. Srinivas and K. Deb, “Multi-objective optimization using nondominated sorting in genetic algorithms,” *Evolutionary Computation*, pp. 221-248, 1994.
[125] P. Hajela and C. Y. Lin, “Genetic search strategies in multicriterion optimal design,” *Structural optimization*, pp. 99-107, 1992.
[126] P. Hajela and J. Lee, “Constrained genetic search via scheme adaptation: an immune network solution,” *Structural optimization*, pp. 11-15, 1996.
[127] M. Laumanns, G. Rudolph and H. P. Schwefel, “A spatial predator-prey approach to multi-objective optimization: a preliminary study,” in *Proceedings of the parallel problem solving from nature*, pp. 241-249, 1998.
[128] K. Deb, S. Agrawal, A. Pratap and T. Meyarivan, “A fast and elitist multi-objective genetic algorithm: NSGAII,” Technical Report 200001, India Institute of Technology, Kanpur: Kanpur Genetic Algorithms Laboratory, 2000.

[129] K. Deb, S. Agrawal, A. Pratap and T. Meyarivan, “A fast and elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II.” in *Proceedings of the parallel problem solving from nature VI*, pp.849-858, 2000.
[130] H. Kita, Y. Yabumoto, N. Mori and Y. Nishikawa, “ Multi-objective optimization by means of the thermodynamical genetic algorithm,” in *Proceedings of the parallel problem solving from nature IV*, pp. 504-512, 1996.
[131] A. Osyczka and S. Kundu, “A new method to solve generalized multi-criteria optimization problems using the simple genetic algorithm,” *Structural optimization,* pp. 94-99, 1995.
[132] V. L. Huang, P. N. Suganthan and J. J. Liang, “Comprehensive Learning Particle Swarm Optimizer for Solving Multiobjective Optimization Problems”, *Int. Journal of Intelligent Systems*, vol. 21, pp. 1-18, 2006.
[133] K. Zielinski, R. Laur, “Adaptive Parameter Setting for a Multi-Objective Particle Swarm Optimization Algorithm”, *Proc. of IEEE Congress of Evolutionary Computation*, Singapore. pp. 3585 – 3592, 2007,
[134] S. Kukkonen and J. Lampinen, “Performance assessment of Generalized Differential Evolution 3 (GDE3) with a given set of problems”, *Proc. of the IEEE Congress on Evolutionary Computation*, Singapore. pp. 3593 – 3600, 2007.
[135] V. L. Huang, A. K. Qin, P. N. Suganthan and M. F. Tasgetiren, “Multi-objective optimization based on self-adaptive differential evolution algorithm”, *Proc. of the IEEE Congress on Evolutionary Computation*, Singapore. pp. 3601 – 3608, 2007.
[136] L-Y. Tseng and C. Chen, “Multiple trajectory search for multiobjective optimization”, *Proc. of the IEEE Congress on Evolutionary Computation*, Singapore. pp. 3609 – 3616, 2007.
[137] K. Deb, A. Pratap, S. Agrawal and T. Meyarivan, “A fast and elitist multi-objective genetic algorithm: NSGAII,” IEEE Transactions on Evolutionary Computation, Vol. 6, pp. 182-197, 2002.
[138] D. Sharma, A. Kumar, K. Deb and K. Sindhya, " Hybridization of SBX based NSGA-II and sequential quadratic programming for solving multi-objective optimization problems," . of the IEEE Congress on Evolutionary Computation, Singapore, pp. 3003-3010, 2007.
[139] A. Kumar, D. Sharma and K. Deb, " A hybrid multi-objective optimization procedure using PCX based NSGA-II and sequential quadratic programming," Proc. of the IEEE Congress on Evolutionary Computation, Singapore, pp. 3011-3018, 2007.
[140] A. Zamuda, J. Brest, B. Boskovic and V. Zumer, "Differential evolution for multiobjective optimization with self adaptation," Proc. of the IEEE Congress on Evolutionary Computation, Singapore, pp. 3617-3624, 2007.
[141] Richard Balling and Scott Wilson. “The Maximin Fitness Function for Multi-objective Evolutionary Computation: Application to City Planning,” In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '2001), pp. 1079-1084. San Francisco, California, 2001.

[142] P. J. Bentley and J. P. Wakefield. "Finding Acceptable Solutions in the Pareto-Qptimal Range using Multiobjective Genetic Algorithms," Soft Computing in Engineering Design and Manufacturing, pp. 231-240, 1997.
[143] Carlos A. Coello Coello, Gary B. Lamont, and David A. Van Veldhuizen. Evolutionary Algorithmsfor Solving Multi-Objective Problems. Springer, New York, second edition, 2007.
[144] Li, H. and Q. Zhang, Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II," *IEEE Trans. on Evolutionary Computation*, Vol. 12, No. 2, 284-302, 2009.
[145] R. Thomsen, "Multi-modal optimization using Crowding-based differential evolution," *in Proceedings of the IEEE 2004 congress on evolutionary computation*, pp. 1382-1389, 2004.
[146] X. Li, "Efficient differential evolution using speciation for multi-modal function optimization," *in Proceedings of the conference on genetic and evolutionary computation.* Washington DC, USA, pp. 873-880, 2005.
[147] D. Zaharie, "A multipopulation differential evolution algorithm for multi-modal optimization", *In Proceedings of the Tenth MENDEL International Conference on Soft Computing*, Brno, Czech Republic, pp. 17-24, 2004.
[148] D. Zaharie, " Extensions of differential evolution algorithms for multi-modal optimization," in *Proceedings of 6th International Symposium of Symbolic and numeric Algorithms for Scientific Computing*, Timisoara, Romania, pp. 523-534, 2004.
[149] Y. Gao and Y. Wang, "A memetic differential evolutionary algorithm for high dimensional functions' optimization," in *Proceedings of the third international conference on natural computation*, Haikou, Hainan, China, pp.188-192, 2007.
[150] J. Chiou and F. Wang, "Hybrid method of evolutionary algorithms for static and dynamic optimization problems with application to a fed-batch fermentation process," *Computers and Chemical Engineering*, vol. 23, pp. 1277-1291, 1999.
[151] V. Tirronen, F. Neri, T. Karkkainen, K. Majava and T. Rossi, "A memetic differential evolution in filter design for defect detection in paper production," Applications of Evolutionary Computing, Lectures notes in Computer Science, pp. 320-329, 2007.
[152] V. Tirronen, "Global optimization using memetic differential evolution with applications to low level machine vision," *Doctoral dissertation*, University of Jyvaskyla, 2008.
[153] D H. Wolpert, W. G. Macready, No free lunch theorems for optimization, *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 67–82, 1997.
[154] X. Li, "Niching without niching parameters: particle swarm optimization using a ring topology," *IEEE Transactions on Evolutionary Computation*, vol. 14, 2010.
[155] O. M. Shir, M. Emmerich and T. Back, "Adaptive niche radii and niche shapes approaches for niching with the CMA-ES", *Evolutionary Computation*, vol. 18, pp. 97-126, 2010.

[156] C. Stoean, M. Preuss, R. Stoean, and D. Dumitrescu, “Multi-modal optimization by means of a topological species conservation algorithm”, *IEEE Trans. Evol. Comput*., 2010.
[157] C. Stoean, M. Preuss, R. Stoean, and D. Dumitrescu, “Disburdening the species conservation evolutionary algorithm of arguing with radii,” in *Proc. Genetic Evol. Comput. Conf. (GECCO)*, pp.1420-1427, 2007.
[158] J.-P. Li, M. E. Balazs, G. T. Parks, and P. J. Clarkson, “A species conserving genetic algorithm for multi-modal function optimization,” *Evol. Comput.*, vol. 10, no. 3, pp. 207-234, 2002.
[159] S. C. Esquivel and C. A. C. Coello, "On the use of particle swarm optimization with multimodal functions", *in Proceedings of 2003 IEEE Congress on Evolutionary Computation (CEC'2003)*, Vol. 2, pp. 1130--1136, IEEE Press, Canberra, Australia, December, 2003.
[160] K. E. Parsopoulos, V. P. Plagianakos, G. D. Magoulas, M. N. Vrahitis, “Stretching technique for obtaining global minimizers through particle swarm optimization”, *In Proceedings of the Particle Swarm Optimization Workshop*, pp. 22-29, Indianapolis, USA, 2001.
[161] A. Nickabadi, M.M. Ebadzadhe and R. Safabakhsh, “A dynamic niching particle swarm optimizer for multi-modal optimization”, *In Proceedings of the Congress on Evolutionary Computation (CEC 2008)*, pp. 26 – 32, Hong Kong, 2008.
[162] A. D. Cioppa, C. D. Stefano, and A. Marcelli, “Where are the niches? Dynamic fitness sharing,” *IEEE Trans. Evol. Comput*., vol. 11, no. 4, pp. 453-465, 2007.
[163] IEEE Current Operating Problems Working Group, *Potential impacts of clean air regulations on system operations*. pp. 647-656, 1995.
[164] J. Kennedy, “Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance”, *In Proceedings of the Congress on Evolutionary Computation*, pp. 1931-1938, Washington DC, USA, IEEE Service Center, Piscataway, NJ, 1999.
[165] R. Brits, A. 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, 2002.
[166] X. Li. “Adaptively choosing neighborhood bests using species in a particle swarm optimizer for multi-modal function optimization.”, In K. Deb, editor, *Proc. Of Genetic and Evolutionary Computation Conference 2004(LNCS 3102)*, pp. 105–116, 2004.
[167] F van den Bergh, “An Analysis of Particle Swarm Optimizers” , PhD Thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.
[168] K. Veeramachaneni, T. Peram, C. Mohan, and L. Osadciw. “Optimization using particle swarm with near neighbor interactions.” In Proc. of Genetic and Evolutionary Computation Conference, pp. 110-121, Chicago, 2003.
[169] J. Knowles and D. Corne, "Approximating the nondominated front using the Pareto archived evolution strategy," *Evolutionary Computation*, pp. 149-172, 2000.

[170] D. J. White, "Epsilon dominance and constraint partitioning in multiple objective problems," *Journal of Global Optimization*, pp. 435-448, 1998.
[171] 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 on Multi-objective Optimization Algorithms," *Technical Report*, Nanyang Technological University, Singapore, 2007.
[172] J. Knowles, L. Thiele, and E. Zitzler, "A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers," Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland, 2006.
[173] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *Evolutionary Computation*, vol. 4, pp. 1-32, 1996.
[174] Y. Wang, Z. Cai, Y. Zhou and W. Zeng, "An Adaptive Tradeoff Model for Constrained Evolutionary Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, pp. 80-92, 2008.
[175] K. Deb, "*An efficient constraint handling method for genetic algorithms*, Computer Methods" in *Applied Mechanics and Engineering*, vol. 186, pp. 311-338, 2002.
[176] Y. G. Woldesenbet, B. G. Tessema and G. G. Yen, "Constraint handling in multi-objective evolutionary optimization," *in 2007 IEEE Congress on Evolutionary Computation*, Piscataway, NJ, USA, 2007.
[177] B. Tessema and G. G. Yen, "A self adaptive penalty function based algorithm for constrained optimization," in *IEEE Congress on Evolutionary Computation*, Piscataway, NJ, USA, 2007.
[178] R. Courant, "Variational methods for the solution of problems of equilibrium and vibrations," *Bulletin of the American Mathematical Society*, vol. 49, pp. 1-23, 1943.
[179] J. Joines and C. Houck, "On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with Gas," in *First IEEE Congress on Evolutionary Computation*, Orlando, 1994.
[180] R. Farmani and J. A. Wright, "*Self-Adaptive Fitness Formulation for Constrained Optimization*," IEEE Transactions on Evolutionary Computation, vol. 7, pp. 445-455, 2003.
[181] D. Powell D. and M. M. Skolnick, "Using genetic algorithms in engineering design optimization with non-linear constraints," in *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, USA, 1993.
[182] T. Takahama and S. Sakai, "Constrained Optimization by Constrained Particle Swarm Optimizer with level Control," in *Proc. of 4th IEEE International Workshop on Soft Computing as Transdisciplinary Science and Technology*, Muroran, Japan, 2005.
[183] J. Zahavi and L. Eisenberg, "An application of the Economic-environmental power dispatch," *IEEE Trans Syst, Man*, Cybernet, pp. 523-530, 1977.

[184] S. F. Brodesky and R. W. Hahn, "Assessing the influence of power pools on emission constrained economic dispatch," *IEEE Trans Power Systems*, pp. 57-62, 1986.
[185] G. P. Granelli, M. Montagna and G. L. Pasini, "Emission constrained dynamic dispatch," *Electric Power Systems Research*, pp. 55-64, 1992.
[186] R. Yokoyama, S. H. Bae, T. Morita and H. Sasaki, "Multiobjective optimal generation dispatch based on probability security criteria," *IEEE Trans Power Systems*, pp. 317-324, 1988.
[187] J. S. Dhillon, S. C. Parti and D. P. Kothari, "Stochastic economic emission load dispatch," *Electric Power Systems Research*, pp. 186-197, 1993.
[188] J. Nanda, L. Hari and M. L. Kothari, "Economic Emission load dispatch with line flow constraints using a classical technique," *IEE Proc Generation Transmission and Distribution*, pp.1-10, 1994.
[189] A. Farag, S. Baiyat and T. C. Cheng, "Economic load dispatch multiobjective optimization procedures using linear programming techniques," *IEEE Trans Power Systems*, pp. 731-738, 1995.
[190] C. S. Chang, K. P. Wong and B. Fan, "Security-constrained multiobjective generation dispatch using bicriterion global optimization," *IEE Proc Generation Transmission and Distribution*, pp. 406-414, 1995.
[191] J. X. Xu, C. S. Chang and X. W. Wang, "Constrained multiobjective global optimization of longitudinal interconnected power system by genetic algorithm," *IEE Proc Generation Transmission and Distribution*, pp. 435-446, 1996.
[192] D. B. Das and C. Patvardhan, "New Multi-Objective Stochastic Search Technique for Economic Load Dispatch," *IEE Proc Generation Transmission and Distribution*, pp. 747-752, 1998.
[193] M. A. Abido, "A Novel Multiobjective Evolutionary Algorithm for Environmental / Economic Power Dispatch," *Electric Power Systems Research*, pp. 71-81, 2003.
[194] M. A. Abido, "A Niched Pareto Genetic Algorithm for Environmental/ Economic Power Dispatch," *Electric Power Systems Research*, pp. 97-105, 2003.
[195] M. A. Abido, "Environmental / Economic Power Dispatch using Multiobjective Evolutionary Algorithms," *IEEE Trans Power Systems*, pp. 1529-1537, 2003.
[196] T. F. Robert, A. H. King, C. S. Harry, Rughooputh and K. Deb, "Evolutionary Multi-Objective Environmental/Economic Dispatch: Stochastic vs. Deterministic Approaches," *KanGAL Report*, pp. 1-15, 2004.
[197] T. F. Robert, A. H. King, C. S. Harry, Rughooputh and K. Deb, "Stochastic Evolutionary Multiobjective Environmental/Economic Dispatch," *In Proceedings of IEEE Congress on Evolutionary Computation*, Vancouver, BC, Canada, pp. 946-953, 2006.

[198] D. N. Jeyakumar, P. Venkatesh and L. Y. Lee. “Application of Multi objective evolutionary programming to combined economic emission dispatch problem,” *In Proceedins of Int. Conf. on Neural Networks*, Florida, USA, pp. 12-17, 2007.

[199] B. Zhao and Y. J. Cao, “Multiple objective particle swarm optimization technique for economic load dispatch,” *Journal of Zhejiang University Science* pp. 420-427, 2005.

[200] L. Wang and C. Singh, “Environmental/Economic power dispatch using a fuzzified multi-objective particle swarm optimization,” *Electric Power Systems Research*, pp. 1654-1664, 2007.

[201] A. Shubham, B. K. Panigrahi and M. K. Tiwari, “Multiobjective Particle Swarm Algorithm with Fuzzy Clustering for Electrical Power Dispatch,” *IEEE Trans Evolutionary Computation*, pp. 529-541, 2008.

[202] J. Cai, X. Ma, Q. Li, L. Li and H. Peng, “A multi-objective chaotic particle swarm optimization for environmental/economic dispatch,” *Energy Conversion and Management*, pp. 1318-1325, 2009.

[203] S. Lu, C. Sun and Z. Lu, “An improved quantum-behaved particle swarm optimization method for short-term combined economic emission hydrothermal scheduling,” *Energy Conversion and Management*, pp. 561-571, 2010.

[204] R. Mallipeddi, P. N. Suganthan, “Ensemble of Constraint Handling Techniques”, *IEEE Trans. on Evolutionary Computation*, pp. 561 - 579, Aug. 2010.

Appendix A

AF1: Two-Peak Trap

$$f_1(x) = \begin{cases} \frac{160}{15}(15 - x), & \text{for } 0 \le x \le 15 \\ \frac{200}{5}(x - 15), & \text{for } 15 \le x \le 20 \end{cases}$$

Range: $0 \le x \le 20$

AF2: Central Two-Peak Trap

$$f_2(x) = \begin{cases} \frac{160}{10}x, & \text{for } 0 \le x \le 10 \\ \frac{160}{5}(15 - x) & \text{for } 10 \le x \le 15 \\ \frac{200}{5}(x - 15), & \text{for } 15 \le x \le 20 \end{cases}$$

Range: $0 \le x \le 20$

AF3: Five-Uneven-Peak Trap

$$f_3(x) = \begin{cases} 80(2.5 - x) & \text{for } 0 \le x < 2.5 \\ 64(x - 2.5) & \text{for } 2.5 \le x < 5 \\ 64(7.5 - x) & \text{for } 5 \le x < 7.5 \\ 28(x - 7.5) & \text{for } 7.5 \le x < 12.5 \\ 28(17.5 - x) & \text{for } 12.5 \le x < 17.5 \\ 32(x - 17.5) & \text{for } 17.5 \le x < 22.5 \\ 32(27.5 - x) & \text{for } 22.5 \le x < 27.5 \\ 80(x - 27.5) & \text{for } 27.5 \le x \le 30 \end{cases}$$

Range: $0 \le x \le 20$

AF4: Equal Maxima

$$f_4(x) = \sin^6(5\pi x)$$

Range: $0 \le x \le 1$

AF5: Decreasing Maxima

$$f_5(x) = \exp[-2\log(2)\cdot(\frac{x-0.1}{0.8})^2]\cdot\sin^6(5\pi x)$$

Range: $0 \le x \le 1$

AF6: Uneven Maxima

$$f_6(x) = \sin^6(5\pi(x^{3/4} - 0.05))$$

Range: $0 \le x \le 1$

AF7: Uneven Decreasing Maxima

$$f_7(x) = \exp[-2\log(2)\cdot(\frac{x-0.08}{0.854})^2]\cdot\sin^6(5\pi(x^{3/4} - 0.05))$$

Range: $0 \le x \le 1$

AF8: Himmelblau's function

$$f_8(x, y) = 200 - (x^2 + y - 11)^2 - (x + y^2 - 7)^2$$

Range: $-6 \le x, y \le 6$

AF9: Six-Hump Camel Back

$$f_9(x, y) = -4[(4 - 2.1x^2 + \frac{x^4}{3})x^2 + xy + (-4 + 4y^2)y^2]$$

Range: $-1.9 \le x \le 1.9;$
$-1.1 \le y \le 1.1$

AF10: Shekel's foxholes

$$f_{10}(x, y) = 500 - \frac{1}{0.002 + \sum_{i=0}^{24} \frac{1}{1 + i + (x - a(i))^6 + (y - b(i))^6}}$$

where $a(i) = 16(i \bmod 5) - 2)$, and $b(i) = 16(\lfloor (i/5) \rfloor - 2)$

Range: $-65.536 \le x, y \le 65.535$

AF11: 2D Inverted Shubert function

$$f_{11}(\vec{x}) = -\prod_{i=1}^{2}\sum_{j=1}^{5} j\cos[(j+1)x_i + j]$$

Range: $-10 \le x_1, x_2 \le 10$

AF12-14: Inverted Vincent function

$$f(\vec{x}) = \frac{1}{n}\sum_{i=1}^{n}\sin(10.\log(x_i))$$

where n is the dimesnion of the problem

Range: $0.25 \le x_i \le 10$

The differences between AF12-AF14 are the numbers of the dimensions. (n is 1 for AF12, 2 for AF13 and 3 for AF14).

Appendix B

The composition function are defined as follow:

$F(x)$: new composition function

$f_i(x)$: i[th] basic function used to construct the composition function.

n: number of basic functions (number of optima)

x: decision variable

D: dimensions (can be chosen from 1-100)

M_i : linear transformation matrix for each $f_i(x)$

o_i : new shifted optima position for each $f_i(x)$

$$F(x) = \sum_{i=1}^{n} \left\{ w_i * [f_i'((x - o_i) / \lambda_i * M_i)] \right\}$$

w_i : weight value for each $f_i(x)$, calculated as follow:

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

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

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

σ_i : used to control each $f_i(x)$'s coverage range.

λ_i : used to stretch compress the function.

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

$\left| f_{\max i} \right|$ is estimated using: $\left| f_{\max i} \right| = f_i((x' / \lambda_i) * M_i), x' = [5, 5, ..., 5]$

BF1: Composition Function 1 (*n*=8)

$f_{1-2}(x)$: Rastrigin's Function

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

$f_{3-4}(x)$: Weierstrass Function

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

$a = 0.5, b = 3, k_{\max} = 20$

$f_{5-6}(x)$: Griewank's Function

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

$f_{7-8}(x)$: Sphere Function

$$f_i(x) = \sum_{i=1}^{D} x_i^2$$

$\sigma_i = 1$ for all i

λ = [1, 1, 10, 10, 5/60, 5/60, 5/32, 5/32]

M_i : are all identity matrices

These formulas are basic functions; shift and rotation should be added to these functions. Take f_1 as an example, the following function should be evaluated:

$$f_i(z) = \sum_{i=1}^{D} (z_i^2 - 10\cos(2\pi z_i) + 10)$$

where $z = ((x - o_i) / \lambda_1) * M_1$.

BF2: Composition Function 2 (*n*=6)

$f_{1-2}(x)$: Griewank's Function

$f_{3-4}(x)$: Weierstrass Function

$f_{5-6}(x)$: Sphere Function

$\sigma_i = 1$ for all i

λ = [1, 1, 10, 10, 5/60, 5/60,]

M_i : are all identity matrices

BF3: Composition Function 3 (*n*=6)

$f_{1-2}(x)$: Rastrigin's Function

$f_{3-4}(x)$: Griewank's Function

$f_{5-6}(x)$: Sphere Function

$\sigma_i = 1$ for all i

λ = [1, 1, 10, 10, 5/60, 5/60,]

M_i : are all identity matrices

BF4: Composition Function 4 (*n*=6)

$f_{1-2}(x)$: Rastrigin's Function

$f_{3-4}(x)$: Weierstrass Function

$f_{5-6}(x)$: Griewank's Function

$\sigma_i = 1$ for all i

λ = [1, 1, 10, 10, 5/60, 5/60,]

M_i : are all identity matrices

BF5: Composition Function 5 (*n*=6)

$f_{1-2}(x)$: Rastrigin's Function

$f_{3-4}(x)$: Weierstrass Function

$f_{5-6}(x)$: Sphere Function

$\sigma_i = 1$ for all i

λ = [1, 1, 10, 10, 5/60, 5/60,]

M_i : are all identity matrices

BF6: Composition Function 6 (*n*=6)

$f_{1-2}(x)$: F8F2 Function

$$F8(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$$

$$F2(x) = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$$

$$f_i(x) = F8(F2(x_1, x_2)) + F8(F2(x_2, x_3)) + ... + F8(F2(x_{D-1}, x_D) + F8(F2(x_{D,} x_1))$$

$f_{3-4}(x)$: Weierstrass Function

$f_{5-6}(x)$: Griewank's Function

$\sigma = [1,1,1,1,1,2]$,

$\lambda = [5*5/100; 5/100; 5*1; 1; 5*1; 1]$

M_i : are all orthogonal matrix

BF7: Composition Function 7 (*n*=6)

$f_{1-2}(x)$: Rotated Expanded Scaffer's F6 Function $F(x, y) = 0.5 + \frac{(\sin^2(\sqrt{x^2 + y^2}) - 0.5)}{(1 + 0.001(x^2 + y^2))^2}$

$$f_i(x) = F(x_1, x_2) + F(x_2, x_3) + ... + F(x_{D-1}, x_D) + F(x_D, x_1)$$

$f_{3-4}(x)$: F8F2 Function

$f_{5-6}(x)$: Weierstrass Function

$\sigma = [1,1,1,1,1,2]$,

$\lambda = [5;10;5;1;5*5/100;5/100]$

M_i : are all orthogonal matrix

BF8: Composition Function 8 (*n*=6)

$f_{1-2}(x)$: Rotated Expanded Scaffer's F6 Function

$f_{3-4}(x)$: F8F2 Function

$f_{5-6}(x)$: Griewank's Function

$\sigma = [1,1,1,1,1,2]$,

$\lambda = [5*5/100;5/100;5*1;1;5*1;1]$

M_i : are all orthogonal matrix

BF9: Composition Function 9 (*n*=6)

$f_{1-2}(x)$: Rotated Expanded Scaffer's F6 Function

$f_{3-4}(x)$: Weierstrass Function

$f_{5-6}(x)$: Griewank's Function

$\sigma = [1,1,1,1,1,2]$,

$\lambda = [5;10;5*5/100;5/100;5;1]$

M_i : are all orthogonal matrix

BF10: Composition Function 10 (*n*=6)

$f_{1-2}(x)$: Rastrigin's Function

$f_{3-4}(x)$: F8F2 Function

$f_{5-6}(x)$: Weierstrass Function

$\sigma = [1,1,1,1,1,2]$,

$\lambda = [5;10;5*5/100;5/100;5;1]$

M_i : are all orthogonal matrix

BF11: Composition Function 11 (*n*=8)

$f_{1-2}(x)$: Rastrigin's Function

$f_{3-4}(x)$: F8F2 Function

$f_{5-6}(x)$: Weierstrass Function

$f_{7-8}(x)$: Griewank's Function

$\sigma = [1,1,1,1,1,2,2,2]$,

$\lambda = [5;1;5;1;50;10;5*5/200;5/200]$

M_i : are all orthogonal matrix

BF12: Composition Function 12 (F*n*=8)

$f_{1-2}(x)$: Rotated Expanded Scaffer's F6 Function

$f_{3-4}(x)$: F8F2 Function

$f_{5-6}(x)$: Weierstrass Function

$f_{7-8}(x)$: Griewank's Function

$\sigma = [1,1,1,1,1,2,2,2]$,

$\lambda = [5*5/100;5/100;5;1;5;1;50;10]$

M_i : are all orthogonal matrix

BF13: Composition Function 13 (*n*=10)

$f_{1-2}(x)$: Rotated Expanded Scaffer's F6 Function

$f_{3-4}(x)$: Rastrigin's Function

$f_{5-6}(x)$: F8F2 Function

$f_{7-8}(x)$: Weierstrass Function

$f_{9-10}(x)$: Griewank's Function

$\sigma = [1,1,1,1,1,2,2,2,2,2]$,

$\lambda = [5*5/100; 5/100; 5; 1; 5; 1; 50; 10; 5*5/200; 5/200]$

M_i : are all orthogonal matrix

BF14: Composition Function 14 (F28 *n*=10)

All settings are the same as F13, except M_i's condition numbers are [10 20 50 100 200 1000 2000 3000 4000 5000]

BF15: Composition Function 15 (F29 *n*=10)

$f_1(x)$: Weierstrass Function

$f_2(x)$: Rotated Expanded Scaffer's F6 Function

$f_3(x)$: F8F2 Function

$f_4(x)$: Ackley's Function

$f_i(x) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + e$ $f_5(x)$: Rastrigin's Function

$f_6(x)$: Griewank's Function

$f_7(x)$: Non-Continuous Expanded Scaffer's F6 Function

$$F(x,y) = 0.5 + \frac{(\sin^2(\sqrt{x^2+y^2}) - 0.5)}{(1+0.001(x^2+y^2))^2}$$

$$f_i(x) = F(y_1, y_2) + F(y_2, y_3) + \ldots + F(y_{D-1}, y_D) + F(y_D, y_1)$$

$$y_i = \begin{cases} x_j & |x_j| < 1/2 \\ round(2x_j)/2 & |x_j| > 1/2 \end{cases} \text{ for } j = 1, 2, ..., D$$

$$round(x) = \begin{cases} a-1 \text{ if } x \le 0 \,\&\, b \ge 0.5 \\ a \quad\quad\quad \text{if } b < 0.5 \\ a+1 \text{ if } x > 0 \,\&\, b \ge 0.5 \end{cases}$$

$f_8(x)$: Non-Continuous Rastrigin's Function

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

$$y_i = \begin{cases} x_j & |x_j| < 1/2 \\ round(2x_j)/2 & |x_j| > 1/2 \end{cases} \text{ for } j = 1, 2, ..., D$$

$f_9(x)$: High Conditioned Elliptic Function

$$f(x) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} x_i^2$$

$f_{10}(x)$: Sphere Function with Noise in Fitness

$$f_i(x) = (\sum_{i=1}^{D} x_i^2)(1 + 0.1|N(0,1)|)$$

n=10

$\sigma_i = 2$ for all i

$\lambda = [10; 5/20; 1; 5/32; 1; 5/100; 5/50; 1; 5/100; 5/100]$

M_i are all rotation matrices, condition number are [100 50 30 10 5 5 4 3 2 2];

Appendix C

1. OKA2

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

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

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

Pareto Set $(x_1, x_2, x_3) = (x_1, 5\cos(x_1), 5\sin(x_1)), x_1 \in [-\pi, \pi]$

2. SYM-PART

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

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

where

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

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

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

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

3. Extended shifted ZDT1 (S_ZDT1)

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

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

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

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

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

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

D : dimension

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

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

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

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

$x_i \in [x\min_i, x\max_i]$, $\mathbf{xmin}=[x\min_1, x\min_2,...x\min_D]$ and $\mathbf{xmax}=[x\max_1, x\max_2,...x\max_D]$

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

4. Extended Shifted ZDT2 (S_ ZDT2)

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

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

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

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

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

D : dimension

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

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

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

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

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

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

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

5. Extended Shifted ZDT4 (S_ZDT4)

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

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

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

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

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

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

D : dimension

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

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

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

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

$x_i \in [x\min_i, x\max_i]$, $\mathbf{xmin} = [x\min_1, x\min_2, ...x\min_D]$ and $\mathbf{xmax} = [x\max_1, x\max_2, ...x\max_D]$ are included in the data file.

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

6. Extended rotated ZDT4 (R_ZDT4)

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

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

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

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

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

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

D : dimension

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

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

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

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

7. Extended shifted ZDT6 (S_ZDT6)

$$f_1(x) = \begin{cases} 1 - \exp(-4z_1')\sin^6(6\ z_1') + 1, & z\ \ _1 \ge 0 \\ S(p_1)\left(1 - \exp(-4z_1')\sin^6(6\ z_1') + 1\right), & z_1 < 0 \end{cases}$$

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

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

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

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

D : dimension

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

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

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

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

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

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

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

8. Extended shifted DTLZ2 (S_DTLZ2)

$$f_1(\mathbf{x}) = \begin{cases} (1+g(\mathbf{x}_M))\cos(z_1'\pi/2)\cos(z_2'\pi/2)\ldots\cos(z_{M-2}'\pi/2)\cos(z_{M-1}'\pi/2)+1, & z_i \geq 0 \\ S(psum_1)\left((1+g(\mathbf{x}_M))\cos(z_1'\pi/2)\cos(z_2'\pi/2)\ldots\cos(z_{M-2}'\pi/2)\cos(z_{M-1}'\pi/2)+1\right), & \text{otherwise} \end{cases}$$

$$f_2(\mathbf{x}) = \begin{cases} (1+g(\mathbf{x}_M))\cos(z_1'\pi/2)\cos(z_2'\pi/2)\ldots\cos(z_{M-2}'\pi/2)\sin(z_{M-1}'\pi/2)+1, & z_i \geq 0 \\ S(psum_2)\left((1+g(\mathbf{x}_M))\cos(z_1'\pi/2)\cos(z_2'\pi/2)\ldots\cos(z_{M-2}'\pi/2)\sin(z_{M-1}'\pi/2)+1\right), & \text{otherwise} \end{cases}$$

$$f_3(\mathbf{x}) = \begin{cases} (1+g(\mathbf{x}_M))\cos(z_1'\pi/2)\cos(z_2'\pi/2)\ldots\sin(z_{M-2}'\pi/2)+1, & z_i \geq 0 \\ S(psum_3)\left((1+g(\mathbf{x}_M))\cos(z_1'\pi/2)\cos(z_2'\pi/2)\ldots\sin(z_{M-2}'\pi/2)+1\right), & \text{otherwise} \end{cases}$$

$\vdots$

$$f_{M-1}(\mathbf{x}) = \begin{cases} (1+g(\mathbf{x}_M))\cos(z_1'\pi/2)\sin(z_2'\pi/2)+1, & z_i \geq 0 \\ S(psum_{M-1})\left((1+g(\mathbf{x}_M))\cos(z_1'\pi/2)\sin(z_2'\pi/2)+1\right), & \text{otherwise} \end{cases}$$

$$f_M(\mathbf{x}) = \begin{cases} (1+g(\mathbf{x}_M))\sin(z_1'\pi/2)+1, & z_i \geq 0 \\ S(psum_M)\left((1+g(\mathbf{x}_M))\sin(z_1'\pi/2)+1\right), & \text{otherwise} \end{cases}$$

$$g(\mathbf{x}_M) = \sum_{x_i \in \mathbf{X}_M} (z_i' - 0.5)^2$$

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

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

D : dimension

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

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

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

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

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

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

The Pareto-optimal solutions $x_i^* = 0.5 + o_i \left(x_i^* \in \mathbf{x}_M\right)$ and the objective function values must satisfy: $\sum_{m=1}^{M} \left(f_m^*\right)^2 = 0.5$

9. Extended Rotated DTLZ2 (R_DTLZ2)

$$f_1(\mathbf{x}) = \begin{cases} (1+g(\mathbf{x}_M))\cos\left(z_1'\pi/2\right)\cos\left(z_2'\pi/2\right)\ldots\cos\left(z_{M-2}'\pi/2\right)\cos\left(z_{M-1}'\pi/2\right)+1, & z_i \geq 0 \\ S(psum_1)\left((1+g(\mathbf{x}_M))\cos\left(z_1'\pi/2\right)\cos\left(z_2'\pi/2\right)\ldots\cos\left(z_{M-2}'\pi/2\right)\cos\left(z_{M-1}'\pi/2\right)+1\right), & \text{otherwise} \end{cases}$$

$$f_2(\mathbf{x}) = \begin{cases} (1+g(\mathbf{x}_M))\cos\left(z_1'\pi/2\right)\cos\left(z_2'\pi/2\right)\ldots\cos\left(z_{M-2}'\pi/2\right)\sin\left(z_{M-1}'\pi/2\right)+1, & z_i \geq 0 \\ S(psum_2)\left((1+g(\mathbf{x}_M))\cos\left(z_1'\pi/2\right)\cos\left(z_2'\pi/2\right)\ldots\cos\left(z_{M-2}'\pi/2\right)\sin\left(z_{M-1}'\pi/2\right)+1\right), & \text{otherwise} \end{cases}$$

$$f_3(\mathbf{x}) = \begin{cases} (1+g(\mathbf{x}_M))\cos\left(z_1'\pi/2\right)\cos\left(z_2'\pi/2\right)\ldots\sin\left(z_{M-2}'\pi/2\right)+1, & z_i \geq 0 \\ S(psum_3)\left((1+g(\mathbf{x}_M))\cos\left(z_1'\pi/2\right)\cos\left(z_2'\pi/2\right)\ldots\sin\left(z_{M-2}'\pi/2\right)+1\right), & \text{otherwise} \end{cases}$$

$\vdots$

$$f_{M-1}(\mathbf{x}) = \begin{cases} (1+g(\mathbf{x}_M))\cos\left(z_1'\pi/2\right)\sin\left(z_2'\pi/2\right)+1, & z_i \geq 0 \\ S(psum_{M-1})\left((1+g(\mathbf{x}_M))\cos\left(z_1'\pi/2\right)\sin\left(z_2'\pi/2\right)+1\right), & \text{otherwise} \end{cases}$$

$$f_M(\mathbf{x}) = \begin{cases} (1+g(\mathbf{x}_M))\sin\left(z_1'\pi/2\right)+1, & z_i \geq 0 \\ S(psum_M)\left((1+g(\mathbf{x}_M))\sin\left(z_1'\pi/2\right)+1\right), & \text{otherwise} \end{cases}$$

$$g\left(\mathbf{x}_M\right) = \sum_{x_i \in \mathbf{X}_M} \left(z_i' - 0.5\right)^2$$

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

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

D : dimension

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

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

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

$x_i \in [x\min_i, x\max_i]$, $\mathbf{xmin} = [x\min_1, x\min_2, ...x\min_D]$ and $\mathbf{x}\max = [x\max_1, x\max_2, ...x\max_D]$

10. Extended shifted DTLZ3 (S_DTLZ3)

$$f_1(\mathbf{x}) = \begin{cases} (1+g(\mathbf{x}_M))\cos\left(z_1'\,\pi/2\right)\cos\left(z_2'\,\pi/2\right)\ldots\cos\left(z_{M-2}'\,\pi/2\right)\cos\left(z_{M-1}'\,\pi/2\right), & z_i \geq 0 \\ S(psum_1)\left((1+g(\mathbf{x}_M))\cos\left(z_1'\,\pi/2\right)\cos\left(z_2'\,\pi/2\right)\ldots\cos\left(z_{M-2}'\,\pi/2\right)\cos\left(z_{M-1}'\,\pi/2\right)+1\right), & \text{otherwise} \end{cases}$$

$$f_2(\mathbf{x}) = \begin{cases} (1+g(\mathbf{x}_M))\cos\left(z_1'\,\pi/2\right)\cos\left(z_2'\,\pi/2\right)\ldots\cos\left(z_{M-2}'\,\pi/2\right)\sin\left(z_{M-1}'\,\pi/2\right), & z_i \geq 0 \\ S(psum_2)\left((1+g(\mathbf{x}_M))\cos\left(z_1'\,\pi/2\right)\cos\left(z_2'\,\pi/2\right)\ldots\cos\left(z_{M-2}'\,\pi/2\right)\sin\left(z_{M-1}'\,\pi/2\right)+1\right), & \text{otherwise} \end{cases}$$

$$f_3(\mathbf{x}) = \begin{cases} (1+g(\mathbf{x}_M))\cos\left(z_1'\,\pi/2\right)\cos\left(z_2'\,\pi/2\right)\ldots\sin\left(z_{M-2}'\,\pi/2\right), & z_i \geq 0 \\ S(psum_3)\left((1+g(\mathbf{x}_M))\cos\left(z_1'\,\pi/2\right)\cos\left(z_2'\,\pi/2\right)\ldots\sin\left(z_{M-2}'\,\pi/2\right)+1\right), & \text{otherwise} \end{cases}$$

$\vdots$

$$f_{M-1}(\mathbf{x}) = \begin{cases} (1+g(\mathbf{x}_M))\text{co}\ \left(z_1'\,\pi/2\right)\sin\left(z_2'\,\pi/2\right), & z_i \geq 0 \\ S(psum_{M-1})\left((1+g(\mathbf{x}_M))\cos\left(z_1'\,\pi/2\right)\sin\left(z_2'\,\pi/2\right)+1\right), & \text{otherwise} \end{cases}$$

$$f_M(\mathbf{x}) = \begin{cases} (1+g(\mathbf{x}_M))\sin\left(z_1'\,\pi/2\right), & z_i \geq 0 \\ S(psum_M)\left((1+g(\mathbf{x}_M))\sin\left(z_1'\,\pi/2\right)+1\right), & \text{otherwise} \end{cases}$$

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

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

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

D : dimension

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

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

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

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

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

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

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

11. WFG1

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

h_M = mixed$_M$(with α=1 and A=5)

t^1 $t^1_{i=1:k} = y_i$

$t^1_{i=k+1:n}$ = s_linear(y_i,0.35)

t^2 $t^2_{i=1:k} = y_i$

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

t^3 $t^3_{i=1:n}$ = b_poly(y_i,0.02)

t^4 $t^4_{i=1:M-1}$ = r_sum ($\{y_{(i-1)k/(M-1)+1},\ldots, y_{ik/(M-1)}\}$,

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

t^4_M = r_sum ($\{y_{k+1},\ldots, y_n\}$,$\{2(k+1),\ldots,2n\}$

Optimal solutions:

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

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

12. WFG8

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

t^1 $t^1_{i=1:k} = y_i$

$t^1_{i=k+1:n}$ = b_param(y_i, r_sum($\{y_1,\ldots, y_{i-1}\}$,$\{1,\ldots,1\}$), 0.98/49.98, 0.02, 50)

t^2 $t^2_{i=1:k} = y_i$

$t^2_{i=k+1:n}$ = s_linear(y_i,0.35)

t^3 $t^3_{i=1:M-1}$ = r_sum ($\{y_{(i-1)k/(M-1)+1},\ldots, y_{ik/(M-1)}\}$,$\{1,\ldots,1\}$)

t^3_M = r_sum ($\{y_{k+1},\ldots, y_n\}$,$\{1,\ldots,1\}$)

Optimal solutions:

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

$$z_{i=k+1:n} = 2i \times 0.35^{(0.02+49.98(\frac{0.98}{49.98}-(1-2u)\left\lfloor 0.5-u \right\rfloor+\frac{0.98}{49.98}))^{-1}}$$
$$u = r\_sum(\{z_1,...,z_{i-1}\},\{1,...,1\}).$$

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

13. WFG9

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

t^1 $t^1_{i=1:n-1}$ = b_param(y_i, r_sum({y_{i+1},…, y_n},{1,…,1}), 0.98/49.98, 0.02, 50)

$t^1_n = y_n$

t^2 $t^2_{i=1:k}$ =s_decept(y_i, 0.35, 0.001, 0.05)

$t^2_{i=k+1:n}$ =s_multi(y_i, 30, 95, 0.35)

t^3 $t^3_{i=1:M-1}$ = r_nonsep ({$y_{(i-1)k/(M-1)+1}$,…, $y_{ik/(M-1)}$}, k/(M–1))

t^3_M = r_nonsep ({y_{k+1},…, y_n}, l)

Optimal solutions:

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

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

Appendix D

Test problem 1: TNK

The first problem is proposed by Tanaka:

Minimize $f_{1(x)} = x_1,$

Minimize $f_{2(x)} = x_2,$

Subject to $C_{1(x)} \equiv x_1^2 + x_2^2 - 1 - 0.1\cos\left(16arctan\frac{x_1}{x_2}\right) \geq 0,$

$C_{2(x)} \equiv (x_1 - 0.5)^2 + (x_2 - 0.5)^2 \leq 0.5,$

$0 \leq x_1 \leq \pi,$

$0 \leq x_2 \leq \pi.$

Test problem 2: SRN

The second problem is proposed by Srinivas :

Minimize $f_1(x) = 2 + (x_1 - 2)^2 + (x_2 - 1)^2,$

Minimize $f_2(x) = 9x_1 - (x_2 - 1)^2,$

Subject to $C_1(x) \equiv x_1^2 + x_2^2 \leq 225,$

$C_2(x) \equiv x_1 - 3x_2 + 10 \leq 0,$

$-20 \leq x_1 \leq 20,$

$-20 \leq x_2 \leq 20.$

Test problem 3: CONSTR

Minimize $f_1(x) = x_1$

Minimize $f_2(x) = (1 + x_2)/x_1$

Subject to $C_1(x) = 9x_1 + x_2 - 6 \geq 0$

$C_2(x) = 9x_2 - x_2 - 1 \geq 0$

$0.1 \leq x_1 \leq 1,$

$0 \leq x \leq 51.$

Test problem 4: OSY

Osyczka and Kundu used the following six-variable test problem:

Minimize $f_1(x) = -[25(x_1 - 2)^2 + (x_2 - 2)^2 + (x_3 - 1)^2 + (x_4 - 4)^2 + (x_5 - 1)^2]$

$f_1(x) = -[25(x_1 - 2)^2 + (x_2 - 2)^2 + (x_3 - 1)^2 + (x_4 - 4)^2 + (x_5 - 1)^2],$

Minimize $f_2(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2,$

Subject to $C_1(x) \equiv x_1 + x_2 - 2 \geq 0,$

$C_2(x) \equiv 6 - x_1 - x_2 \geq 0,$

$C_3(x) \equiv 2 - x_2 + x_1 \geq 0,$

$C_4(x) \equiv 4 - (x_3 - 3)^2 - x_4 \geq 0,$

$$C_5(x) \equiv 2 - x_1 + 3x_2 \geq 0,$$

$$C_6(x) \equiv (x_5 - 3)^2 + x_6 - 4 \geq 0,$$

$$x_1 \geq 0,\ x_2, x_6 \leq 10,$$

$$1 \leq x_3, x_5 \leq 5, \qquad 0 \leq x_4 \leq 6.$$

Test problem 4: CTP1

Minimize $f_1(x) = x_1$

Minimize $f_2(x) = g(x)\exp(-\frac{f_1(x)}{g(x)})$

Subject to $f_2(x) - a_j \exp(-b_j f_1(x)) \geq 0,$

$j = 1, \ldots, J$

Where $0 \leq x_1 \leq 1, -5 \leq x_2, x_3, x_4 \leq 5 \ \ and$

$g(x) = 31 + \sum_{i=2}^{4}(x_i^2 - 10\cos(4\pi x_i))$

$J = 2,\ a_1 = 0.858, a_2 = 0.541, a_2 = 0.728\ and\ b_2 = 0.295$

Test problems 5-10: CTP2-CTP6

Minimize $f_1(x) = x_1$

Minimize $f_2(x) = g(x)(1 - \sqrt{\frac{f_1}{g(x)}})$

Subject to $\cos(\theta)\,(f_2(x) - e) - \sin(\theta)\, f_1(x) \geq$

$$a|\sin(b\pi(\sin(\theta)\,(f_2(x)-e)+\cos(\theta)\,f_1(x))^c)|^d$$

Where $0 \le x_1 \le 1, -5 \le x_2, x_3, x_4 \le 5 \quad and$

$$g(x) = 31 + \sum_{i=2}^{4}(x_i^2 - 10\cos(4\pi x_i))$$

The parameters chosen for the different CTP2 to CTP6 problems are listed in the following table:

The parameters for CTP2 to CTP6

	θ	a	b	c	d	e
CTP2	-0.2π	0.2	10	1	6	1
CTP3	-0.2π	0.1	10	1	0.5	1
CTP4	-0.2π	0.75	10	1	0.5	1
CTP5	-0.2π	0.75	10	2	0.5	1
CTP6	-0.05π	40	5	1	6	0