CONSTRAINED MULTI-OBJECTIVE PARTICLE SWARM OPTIMIZATION WITH APPLICATION IN POWER GENERATION

Lily Dujuan Li

MIT (University of Newcastle)

A thesis submitted

in fulfilment of the requirements for the degree of

Doctor of Philosophy

School of Computing Sciences

Faculty of Arts, Business, Informatics and Education

CQUniversity of Australia

April 2009

ABSTRACT

This thesis is devoted to the study of metaheuristic optimization algorithms and their application in power generation. The study focuses on constrained multi-objective optimization using Particle Swarm Optimization algorithm.

A multi-objective constraint-handling method incorporating a dynamic neighbourhood PSO algorithm is proposed for tackling single objective constrained optimization problems. The benchmark simulation results demonstrate the proposed approach is effective and efficient in finding the consistent quality solutions. Compared with the recent research results, the proposed approach is able to provide better or similar good results in most benchmark functions. The proposed performance-based dynamic neighbourhood topology has proved to be able to help make convergence faster than the static neighbourhood topology.

The thesis also presents a modified PSO algorithm for solving multi-objective constrained optimization problems. Based on the constraint dominance concept, the proposed approach defines two sets of rules for determining the cognitive and social components of the PSO algorithm. Simulation results for the four numerical optimization problems demonstrate the proposed approach is effective. The proposed approach has a number of advantages such as being applicable to any number of objective functions and computationally inexpensive.

As applications, three engineering design optimization problems and the power generation loading optimization problem are investigated. The simulation results for the engineering design optimization problems and the power generation loading optimization problem reveal the capability, effectiveness and efficiency of the proposed approaches. The methodology can be readily applicable to a broad range of applications.

TABLE OF CONTENTS

LIST OF TABLES

List of Figures

LIST OF ABBREVIATIONS

ACO	Ant Colony Optimization
API	Application Programming Interface
CNOP	Constrained Nonlinear Optimization Problem
DE	Differential Evolution
EA	Evolutionary Algorithm
ERP	External Repository
ETS	Emission Trading Scheme
GA	Genetic Algorithm
IBE	Iteration-based Experiment
JVM	Java Virtual Machine
MO	Multiobjective Optimization
MOOP	Multiobjective Optimization Problem
OO	Object-Oriented
PSO	Particle Swarm Optimization
SP	Spacing
TBE	Target-based Experiment
TSP	Travelling Salesman Problem
UML	Unified Modelling Language

GLOSSARY

a	parameter in ACO, coefficients of the polynomial to heat rate function
b	coefficients of the polynomial to emission curve function
C	criterion for TBE
C_r	cross factor in DE
c	PSO parameter
d	distance of arc in ACO
f	objective function
F	objective function, mutation factor in DE, feasible solutions
g	inequality constraint function
H_{saving}	heat consumption saving
h	equality constraint function, heat consumption per hour to a unit at a given load
i, j	index
k	index, order of polynomial function (superscript)
L, l	lower boundary
$lBest$	best particle in a local neighbourhood
M_{saving}	annual money saving
M_{total}	total power demand by market, total workload
$M_{\min}$	lowest power generated, lowest workload
$M_{\max}$	highest power generated, highest workload
m	number of constraints
N_p	population size in DE
n	dimension, total iterations (or generations)
P	particle, maximum NO_x emission license limit to each unit
$pBest$	best previous position for a particle
p	probability

q	NO_x emission level to a unit at a given load
R	penalty parameter
r	individual index in DE, random number in PSO, NO_x emission constraint function
S	decision space, search space
T	iteration, generation, target value
U	upper boundary
u	trail vector in DE , upper boundary
$V_{\max}$	maximum velocity in PSO
v	velocity in PSO, mutated individual in DE
w	inertia weight factor in PSO
x	decision variable
z	repair attempts in repairing algorithm
Φ, Ω	constraint violation
δ	minimum error criterion for equality constraint
ε	feasibility tolerance
χ	constriction coefficient in PSO
τ	amount of pheromone in ACO
ρ	rate of pheromone evaporation in ACO, relative size of feasible space
η	inverse distance of arc in ACO
β	parameter in ACO

ACKNOWLEDGMENTS

Nothing is impossible with God. – Luck 1:37-42 (Bible)

It is never an easy task to conduct a PhD research program. It is even harder when you have fulltime teaching commitments involved. When finishing this thesis, I am indeed filled with gratitude and joy. God has blessed me in all the ways during the past few years. In this section, I would like to thank and acknowledge the people who have carried God's grace and love for me throughout this journey.

Firstly, I would like to thank Prof. Xinghuo Yu, my principal supervisor, for his kindness to take me as his student. It cannot be over exaggerated the importance of Prof. Yu's support, encouragement and guidance in my research journey. I do learn a lot from his high level professional stands.

I am grateful to Dr Xiaodong Li and Dr Wei Li, my associate supervisors, for their time in providing guidance, advices and discussions. Dr Xiaodong Li's exceptional knowledge and prompt inputs has been such a valuable asset to me. Dr Wei Li has provided great advices at the early stage of this research which is much appreciated.

I cannot forget the support from my previous supervisor, Associate Professor Russel Stonier who has retired. Here I thank Russel for his support and patience.

I would like to thank my colleagues Dr Sylvia Ward, Dr Paul Waight, Dr Andrew Chiou, Dr Noel Patson, Dr William Guo and Dr Jo Luck who kindly offered to proofread my thesis and gave me invaluable feedback. Thanks to my friends Dr Fengling Han, Ms Jeni Richardson, Dr Shang Gao, Dr Michael Li, Ms Sandy Behrens, Mr Denial Pun, Ms Yi Hu, and Dr Jun Zhao for their continuous encouragement.

I want to thank the head of School of Computing Sciences, Mr Tim Roberts, for his support in adjusting my teaching commitments while I was finalising my thesis. Thanks also go to Prof. Kevin Tickle and Prof. Qinglong Han for their support and concern from the faculty management.

Thanks extended to my friends Mrs Amelia Evans and Mr Terry Parkin from Rockhampton City Band for reading some chapters of this thesis and for reminding me the importance of a balanced life.

I am in debt to my dear parents and my parent-in-laws. For many years, I did not have time to accompany them and was not able to look after them when they were sick. Deepest appreciations go to them for their love and understanding. I am grateful to my three beautiful sisters, Xiaojuan, Pei and Huafan and their husbands for their contributions in looking after our aged parents so that to allow me to complete this study.

Very special thanks to my husband, Jiping, for his endless support. He has been always there to share my joys and sadness, help me and tolerate my moodiness. This thesis would not have been completed without his unmeasurable support.

Lastly, thanks to my handsome and dearest son, Joe. He is so special for me. Because of him, I have got the courage and energy to deal with difficulties in my life. I do learn a lot from being a mum.

DECLARATION

I do hereby declare that the work contained in this thesis has not been previously submitted either in whole or in part for a degree or diploma at CQUniversity or any other higher education institution. Parts of this thesis have been published in the papers listed in the publication section. To the best of my knowledge and belief, the material presented in this thesis is original except where due reference is made in the thesis itself.

Lily Dujuan Li

Signature:

Date: 29 April 2009

Chapter 1

INTRODUCTION

1.1 PROLOGUE

Optimization is all about better choices. As human beings, we try to make optimal choices in our daily lives. When we are driving in busy city streets, we choose short and less crowded routes and avoid traffic lights to minimize our travelling time. When buying a car, we would choose the most comfortable model at the minimum cost. Better choices come from knowledge and experiences.

In most industrial activities, making an optimal choice is not that simple. The reasons may be that it is too complicated; there are too many choices; there is limited knowledge and experience; or environmental and human constraints exist. Such cases require complicated calculations and reasoning to help with our decision making. Fortunately, many of these applications can be formulated into mathematical models that can be represented and analysed systematically. This forms the base for the study of optimization.

Optimization is a branch of operations research. It involves topics in wide subfields such as nonlinear programming, stochastic programming, combinatorial optimization, and other computational intelligence technologies. It has a wide range of practical applications crossing industries such as production planning, transportation scheduling and mechanical design. Optimization plays a key role in improving the quality of decision making which leads to improved business performance for any organization.

This thesis is devoted to the study of optimization algorithms and their applications.

The rest of the chapter is organized as follows. Section 1.2 formally introduces optimization problems including basic concepts and terminologies. Section 1.3 presents an outline of the optimization algorithms. Section 1.4 describes the research motivation and scope. Section 1.5 states the goals and the contributions of the thesis. Section 1.6 previews the remaining chapters.

1.2 CONSTRAINED OPTIMIZATION PROBLEMS

Many real world optimization applications involve optimizing (minimizing or maximizing) one or more objectives subject to satisfying diverse equality and inequality constraints. Optimization problems are made up of three basic components: one (or more) objective function(s), a set of unknown decision variables and a set of constraints. The objective functions are numerical functions to determine how good a solution is. A set of unknown decision variables determine the value of the objective functions. The constraints are conditions that allow the decision variables to take on certain values but exclude others. The following sections introduce the optimization problems and some basic concepts.

1.2.1 Single Objective Optimization Problems

A single objective constrained optimization problem can be formalized as a pair (S, f), where $S \subseteq \mathbb{R}^n$ is a bounded set on $\mathbb{R}^n$ and $f : S \to \mathbb{R}$ is a *n*-dimensional real-valued function. The problem is to find a point $x^* \in S$ such that $f(x^*)$ is minimum or maximum on *S* (*S* is denoted as the decision space). Maximization of a real-value function $f_1(x)$ can be regarded as the minimization of the transformed function $f(x) = -f_1(x)$, the minimization problems are taken into consideration in general.

The problem can be stated more formally as to find x^* which

$$\left.\begin{array}{lll} \text{minimize} & f(x) & \\ \text{subject to} & g_i(x) \leq 0, & i = 1,2,...,m; \\ & h_j(x) = 0, & j = 1,2,...,p; \end{array}\right\} \qquad (1.1)$$

where *x* is the vector of solution $x = (x_1, x_2, ..., x_n)^T$, and each $x_i (i = 1,...n)$ is bounded by lower and upper limits $l_i \leq x_i \leq u_i$; m is the number of inequality constraints and p is the number of equality constraints. In both cases, constraints could be linear or non-linear.

The equality constraints $h_j(x) = 0$ can be translated into inequality constraints $|h_j(x)| - \delta \leq 0,$ where δ is a tolerance allowed. Equation (1.1) can be converted into:

$$\left.\begin{array}{lll} \text{minimize} & f(x) & \\ \text{subject to} & g_i(x) \leq 0, & i = 1,2,...,m; \end{array}\right\} \qquad (1.2)$$

where m is the total number of constraints.

More specifically, the optimization task is to find a $x^* \in S$ such that $\forall x = (x_1, x_2, ...x_n)^T \in S : f(x^*) \leq f(x)$ subject to satisfying all constraints.

Single objective optimization has one search space (decision space) and one global solution.

1.2.2 Multi-objective Optimization Problems

As in the single objective constrained optimization problem definition described earlier, a general multi-objective constrained optimization problem consists of a decision vector $x = (x_1, x_2, ..., x_n)^T$, an objective function vector $f(x) = (f_1(x), f_2(x), ..., f_k(x))$ and a constraint function vector $g(x) = (g_1(x), g_2(x), ..., g_m(x))$.

The problem can be stated as to find x^* which

$$\left.\begin{array}{lll} \text{minimize} & f_j(x), & j = 1, 2, ...k \\ \text{subject to} & g_i(x) \le 0, & i = 1, 2, ...m \end{array}\right\} \tag{1.3}$$

where k is the total number of objective functions and m is the total number of constraints.

In single-objective optimization, there exists a global optimum, while in the multi-objective case no optimal solution is clearly defined apart from a set of optima. The optimization is to find good compromises (or “trade-offs”) rather than a single solution as in the single objective optimization. All solutions to a multi-objective optimization problem are called Pareto-optimal solutions, the curve by joining these solutions is known as a Pareto-optimal front [1].

The optimization tasks for multi-objective optimization is to

- find a set of solutions as close as possible to the Pareto-optimal front and
- find a set of solutions as diverse as possible

Multi-objective optimization problems have two search spaces named decision space and objective space.

1.2.3 Basic Concept and Terminology

Since the scope (further described in Section 1.4) of this thesis is focused on integrating the multi-objective constraint-handling methods with the metaheuristic optimization algorithms, it is necessary to introduce some concepts and terminologies about multi-objective optimization. Figures 1.1 and 1.2 illustrate some basic concepts and terminologies based on a two-objective problem, which are then explained in detail.

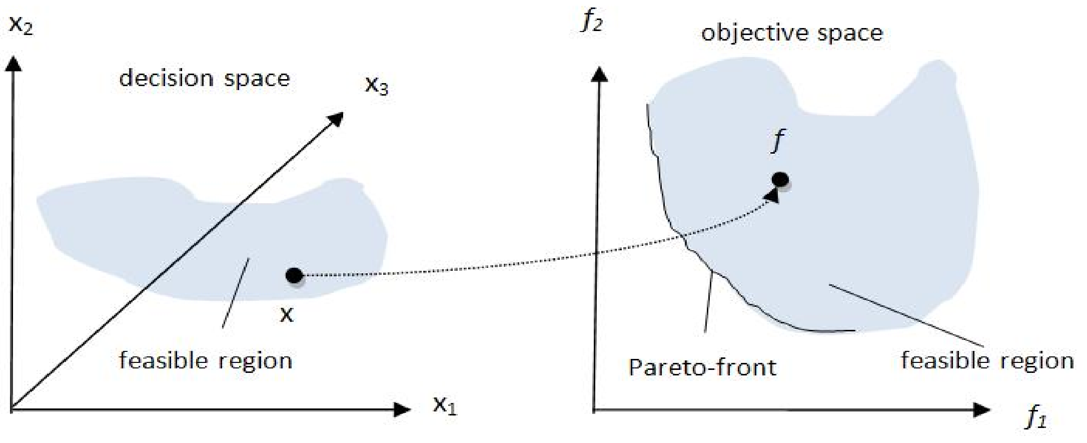


Figure 1.1 Illustration of the concepts of feasible region, Pareto-front, a decision point *x* and its corresponding objective value in decision and objective spaces

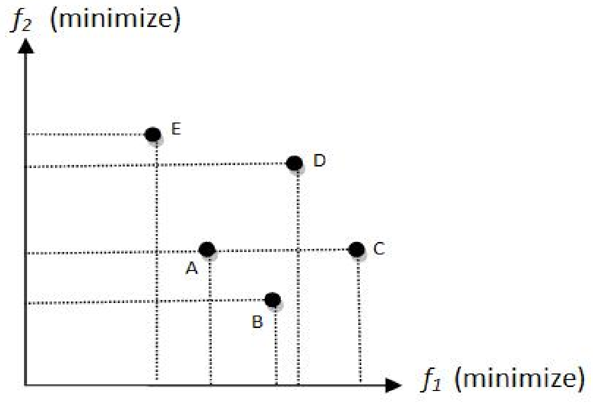


Figure 1.2 Illustration of concept of dominance

Definition 1 (Feasible Set):

A solution x is feasible if it satisfies all constraints that are imposed. That is, the feasible set x_f is a set of solution x that satisfy all the constraints $g(x)$:

$$x_f = \{x \in S \mid g(x) \leq 0\}$$

The feasible region is shadowed in light blue colour in Figure 1.1.

Definition 2 (Pareto Dominance):

A solution $x^{(1)}$ is said to dominate the other solution $x^{(2)}$, if both conditions a) and b) are true:

a) The solution $x^{(1)}$ is no worse than $x^{(2)}$ in all objectives j (j =1,2,…,k).

b) The solution $x^{(1)}$ is strictly better than $x^{(2)}$ in at least one objective j (j =1,2,…,k)

if $f(x^{(1)}) \leq f(x^{(2)})$, $x^{(1)}$ dominates $x^{(2)}$, denoted as $x^{(1)} \prec= x^{(2)}$

if $f(x^{(1)})\ !\leq f(x^{(2)})$ or $f(x^{(2)})\ !\leq f(x^{(1)})$, $x^{(1)}$ is indifferent to $x^{(2)}$, denoted as $x^{(1)} \sim x^{(2)}$.

Figure 1.2 illustrates a two-objective minimization optimization problem with five different solutions shown in the objective space. Solution B dominates solutions C and D because B has lower values in both objective functions f_1 and f_2 than those by C and D. Solution A dominates solution D and solution C because A has a lower value in f_1 than C and the same value f_2 as C. Solutions A, B and E are indifferent because they do not dominate each other.

Definition 3 (Non-dominated set):

Among a set of solutions P, the non-dominated set of solutions P' are those that are not dominated by any member of the set P. In Figure 1.2, for example, the non-dominated set consists of solution E, A and B.

Definition 4 (Pareto-optimal set):

The non-dominated set of the entire feasible search space S is the Pareto-optimal set. The curve by joining these solutions is known as a Pareto-optimal front as shown in Figure 1.1.

Definition 5 (Constraint dominance):

A solution $x^{(1)}$ is said to 'constraint-dominate' a solution $x^{(2)}$, denoted as $x^{(1)} \prec_c x^{(2)}$, if any of the following conditions are true:

- Solution $x^{(1)}$ is feasible and solution $x^{(2)}$ is not.
- Solution $x^{(1)}$ and solution $x^{(2)}$ are both infeasible, but solution $x^{(1)}$ has a smaller constraint violation.
- Solution $x^{(1)}$ and solution $x^{(2)}$ are feasible and solution $x^{(1)}$ dominates solution $x^{(2)}$ in the usual sense (see Definition 2).

1.3 OPTIMIZATION ALGORITHMS

1.3.1 Classification

Optimization algorithms can be classified into two categories. The first category is called deterministic methods. The second category is stochastic and heuristic/metaheuristic methods. The deterministic methods find the optimum up to certain accuracy while the stochastic and heuristic/metaheuristic methods find the optimum up to a certain probability. In other words, the deterministic methods find more accurate solutions if the problem representation meets certain requirements. The stochastic and heuristic/metaheuristic methods, however, can find solutions with a certain probability. A classification of the optimization techniques is presented in Figure 1.3. A description of these techniques follows.

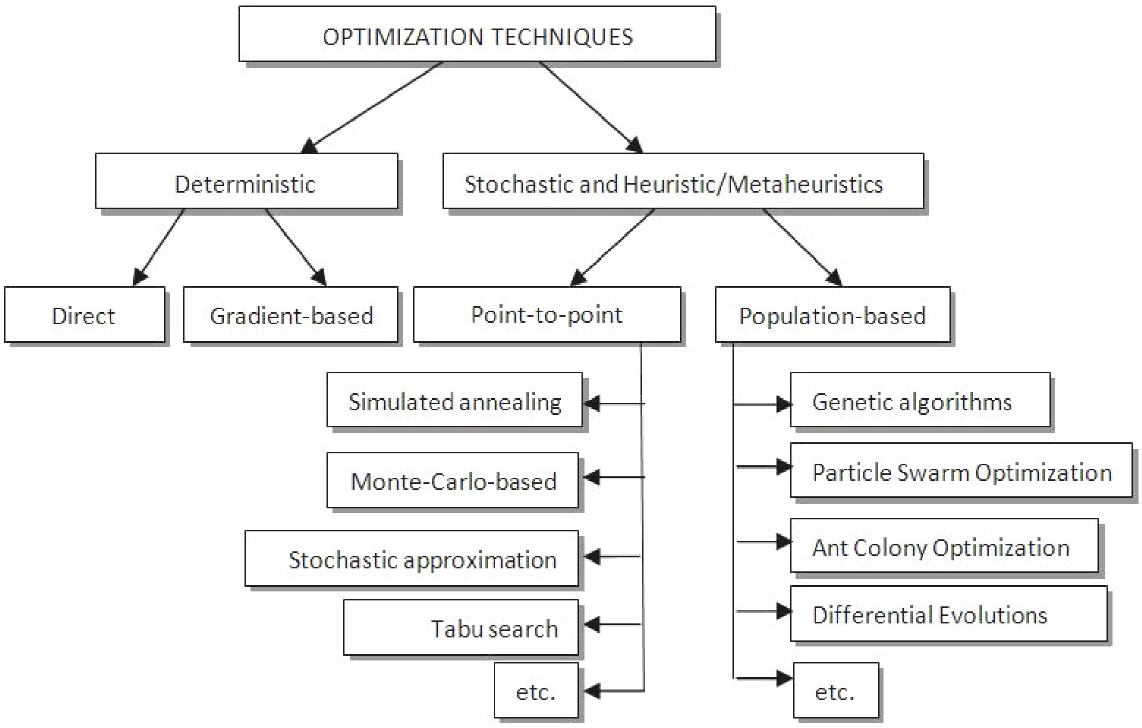


Figure 1.3 Classification of optimization algorithms

1.3.2 Deterministic Methods

The deterministic methods are also called classical methods. The classical optimization methods can be further classified into two groups: Direct methods and Gradient-based methods [1-3].

Direct search methods converge iteratively to the optimum. It starts from a random guess solution. Based on a pre-specified transition rule, the algorithm suggests a direction for successive search points [1, 3]. "*In direct search methods, only the objective function f(x) and the constraint values g(x) are used to guide the search strategy, which require many function evaluations for convergence, it is generally slow*" [1] .

Gradient-based methods compute the position of the minima by differentiating the objective function and setting the obtained gradient equation to zero:

$$\frac{\partial f}{\partial x_i} = 0, \qquad i \in \{1, 2, \ldots n\}, \tag{1.4}$$

while fulfilling the sufficient condition

$$\frac{\partial^2 f}{\partial x_i \partial x_j} > 0, \qquad i, j \in \{1, 2, \ldots n\}. \tag{1.5}$$

Obviously, gradient-based methods require the mathematical equations of the objective functions to be clearly defined and the functions must be continuous and differentiable. "*The gradient-based methods can quickly converge near an optimal solution. But they are not efficient in non-differentiable or discontinuous problems*"[1].

Deb [1] has summarized the difficulties that classical optimization methods have encountered, as follows:

- *The convergence to an optimal solution depends on the chosen initial solution.*
- *Most algorithms tend to get stuck to a suboptimal solution.*
- *An algorithm efficient in solving one optimization problem may not be efficient in solving a different optimization problem.*
- *Algorithms are not sufficient in handling problems having a discrete search space.*
- *Algorithms cannot be efficiently used on a parallel machine.*

With these difficulties, the applications of classical optimization methods have been restricted to the problems that have clear mathematical formulas for both objective functions and constraint functions, and the functions must be continuous and differentiable. They are not designed for processing inaccurate, noisy and complex data although they might excel at dealing with complicated data [4].

1.3.3 Stochastic and Heuristic/Metaheuristic Methods

The stochastic optimization methods are optimization algorithms which incorporate probabilistic (random) elements, either in the problem data (the objective function, the constraints), or in the algorithm itself (through random parameter values, random choices), or in both [5]. Heuristic/metaheuristic methods are those methods to search in the search space in a (more or less) intelligent way [6]. Most stochastic and heuristic/metaheuristic methods are inspired by nature since nature operates with random processes (e.g., for mutating genetic information, within the annealing process of metal, in molecular dynamics, or in swarm behaviours of birds) [3]. In this thesis, we refer metaheuristics as heuristic/metaheuristic stochastic methods.

The metaheuristic methods can be further classified into two groups: point-to-point methods and population-based methods [7, 8]. In point-to-point methods, the search invokes only one solution at the end of each iteration from which the search will start in the next iteration. Simulated annealing [9], Monte-Carlo-based Algorithms[10], and Stochastic Approximation[11] are typical examples of the point-to-point metaheuristics. Instead of one solution in each iteration, the population-based methods work with a population of solutions. By starting with a random set of solutions, a population-based algorithm modifies the current population to a different population in each iteration. Well-known representatives of the population-based metaheuristics are those Evolutionary Algorithms (EAs) like Genetic Algorithms (GA) [12-14], Particle Swarm Optimization (PSO) [15, 16], Ant Colony Optimization (ACO) [17] and Differential Evolution (DE) [18]. Working with a number of solutions provided algorithms with the ability to capture multiple optimal solutions in one single simulation run.

Compared with deterministic methods, the metaheuristic methods offer the following advantages:

- Wide applicability. Since the metaheuristic methods do not use any deterministic rules, and their representations are flexible (e.g., continuous, discrete, differentiable, non-differentiable), these properties make the algorithms flexible enough to be used in a wide variety of problem domains.
- Higher performance. The metaheuristic methods have ability to capture multiple solutions in one single simulation run. They often find optima in complicated optimization problems faster than classical optimization methods [1, 4, 19].

Chapter 2 will present a detailed review of the well-known population-based metaheuristic methods.

1.4 MOTIVATION AND SCOPE

1.4.1 Motivation

Several new emerging metaheuristics like PSO have proven to be effective and efficient for solving real-valued global **unconstrained** optimization problems [16, 20, 21]. However, their utilisations in solving **constrained** optimization problems remain problematic. Most real world applications have to cope with constraints. Often the constraints are many in numbers and are nonlinear. The traditional penalty functions approaches tend to have difficulties to deal with highly constrained search spaces [22, 23]. To seek other better constraint-handling methods incorporating the more superior algorithms is desirable. This research is motivated by the constrained real world applications and the fact of the unsatisfying performance of current constraint-handling methods in the metaheuristic algorithms.

1.4.2 Research Scope

This thesis is devoted to the study of metaheuristic algorithms in constrained optimization. New metaheuristic algorithms and the constraint handling methods are to be investigated. The study will focus on integrating multi-objective constraint-handling methods with the PSO algorithm for solving constrained optimization problems. The thesis also includes a study of the power generation loading optimization application.

1.5 THESIS GOALS AND CONTRIBUTIONS

1.5.1 Goals

The overall goal of this thesis is to investigate the PSO algorithm in constrained optimization and its application in power generation.

The objectives of the thesis are as following.

a) To integrate the relative superior multi-objective constraint-handling method with PSO algorithm for solving constrained optimization problems.

b) To conduct performance evaluation of the proposed approach including benchmark function experiments, search quality evaluation, computing consistency evaluation and comparison study.

c) To investigate how the different neighbourhood topologies affect the algorithm performance. To introduce and evaluate the dynamic neighbourhood topology in order to improve convergence rate.

d) To conduct performance evaluation of the proposed approach in engineering design optimization problems.

e) To investigate the power generation loading optimization application and to examine the utility of the proposed approaches in the application.

1.5.2 Contributions

In summary, the contributions of the thesis include:

- **A new approach to integrate the multi-objective constraint-handling method with PSO algorithm is proposed.** PSO algorithm offers some advantages over the genetic algorithms. There are few attempts made to integrate constraint-handling methods with PSO algorithm.

- **A novel performance-based dynamic neighbourhood topology is proposed.** Neighbourhood topology determines how particles communicate with each other. It is a key factor in PSO to determine how an individual makes use of its social experiences. The proposed performance-based dynamic neighbourhood topology groups those particles that have similar performance into groups to make communication more efficient.
- **An effective multi-objective PSO algorithm is proposed.** Most existing multi-objective PSO proposals do not address the constraints. Integrating constraint-handling mechanisms with multi-objective PSO is a quite challenging topic. This thesis is one of the few attempts to use PSO algorithm in constrained multi-objective optimization problems.
- **As a variant of the proposed approach, a goal-oriented multi-objective constraint-handling method via PSO algorithm is introduced.** Since the goals can be used as the exit criteria, each particle does not need to go through whole iterations. This makes the computation more cost-effective**.**
- **Power generation loading optimization problem is solved by using the proposed approaches.** The power generation loading optimization problem is of practical importance in the evolving carbon constrained power industry in terms of fuel saving and minimizing environmental impact. This thesis presents two optimization models of the power generation loading optimization, and applies the proposed approaches in the application. The simulation result demonstrates that the

loading optimization is significant and the proposed algorithm is effective and efficient. The methodology can be easily extended to other industrial applications.

1.6 STRUCTURE OF THE THESIS

The thesis contains seven chapters. A preview of the chapters is presented as follows.

Chapter 1 (the current chapter) introduces the constrained optimization problems, presents some basic concepts and terminologies, outlines the existing approaches, the research motivation and scope. The research goals and contributions of the thesis are also highlighted in this chapter.

Chapter 2 presents a literature review of the popular metaheuristic optimization methods including constraint-handling methods. A state-of-the-art PSO in constrained and multi-objective optimization are also included.

Chapter 3 presents a multi-objective constraint-handling method with a dynamic neighbourhood PSO algorithm. The chapter starts with problem formulation and transformation, followed by the description of PSO, a few selection rules and the idea of performance-based dynamic neighbourhood topology. The proposed algorithm is then presented. The algorithm is tested using some numerical bench mark functions. The numeric simulation results are to be presented. This chapter addresses the research goals a), b) and c) focusing on single objective constrained problems.

Chapter 4 proposes a modified PSO algorithm for tackling constrained multi-objective optimization problems. The proposed approach defines two sets of rules for determining the cognitive and social components of the PSO algorithm. The simulation results for the four constrained multi-objective optimization problems will be presented.

This chapter continues to address research goals a), and b) focusing on multi-objective constrained problems.

Chapter 5 consists of two parts. The first part describes the design and implementation issues for the proposed approaches. The second part presents the simulation results for three engineering design optimization problems. For special case, where the objective function has a predefined goal, the optimization task is to find solutions that achieve the predefined goal, is addressed. This chapter is to address the research goal d).

In Chapter 6, a challenging real-world application, the power generation loading optimization problem is presented. Based on the problem description and specification, the power generation loading optimization models are presented. Then the proposed two approaches with PSO are applied to the application. Optimization results are presented and discussed. The research goal e) is addressed in this chapter.

Chapter 7 concludes the thesis by discussing the contributions and suggesting possible directions for future research.

In addition to the main text, two appendices are included in the thesis. Appendix I contains the thirteen constrained numerical optimization testing functions. And Appendix II contains the three constrained engineering design optimization functions.

Chapter 2

METAHEURISTIC OPTIMIZATION METHODS – A LITERATURE REVIEW

2.1 INTRODUCTION

The term “metaheuristics” was first proposed by Glover in 1986 [24]. Metaheuristics contain all heuristics methods that show evidence of achieving good quality solutions for the problems of interest within an acceptable time [25, 26]. In the operations research discipline, a metaheuristic is a general-purpose algorithmic framework that can be applied to different optimization problems with relatively few modifications [27]. However, the metaheuristics offer no guarantee of obtaining the global solutions.

The evolutionary algorithms (EAs) are typical metaheuristics. By detecting the optimal solution through cooperation and competition among the individuals of the population, evolutionary optimization often finds optima in complicated optimization problems faster than classical optimization methods [1]. The concept of a genetic algorithm (GA) was first conceived by John Holland in 1975. Since then, a number of evolutionary algorithms such as ACO [28], PSO[15] and DE[29] have emerged. These emerging metaheuristics have been increasingly attracting attention both in academia and industry. Over the last decade or so, evolutionary algorithms have been extensively studied as search and optimization tools in various problem domains. The primary reasons for their success are their broad applicability, ease of use and global perspective [13]. However, the original versions of these metaheuristics have no mechanisms to deal with constraints. Most real world optimization applications have to deal with constraints. Integrating constraint-handling methods with these metaheuristics becomes imperative.

There are a number of constraint-handling methods available in evolutionary optimization [1, 22, 30]. The research on integrating constraint-handling methods with metaheuristics has been concentrated on genetic algorithms[30-32]. Integrating these constraint-handling methods with other algorithms has not been properly studied.

This chapter presents a literature review of the four most salient metaheuristics, that is, GA, DE, ACO and PSO. Two questions are particularly focused in the review: How do the algorithms work? What are the advantages and disadvantages of the algorithms? The constraint-handling methods and their pros and cons will also be included in the review. Furthermore, a review of the state-of-the-art of the PSO algorithms in constrained optimization will be presented as this is the primary focus of this research.

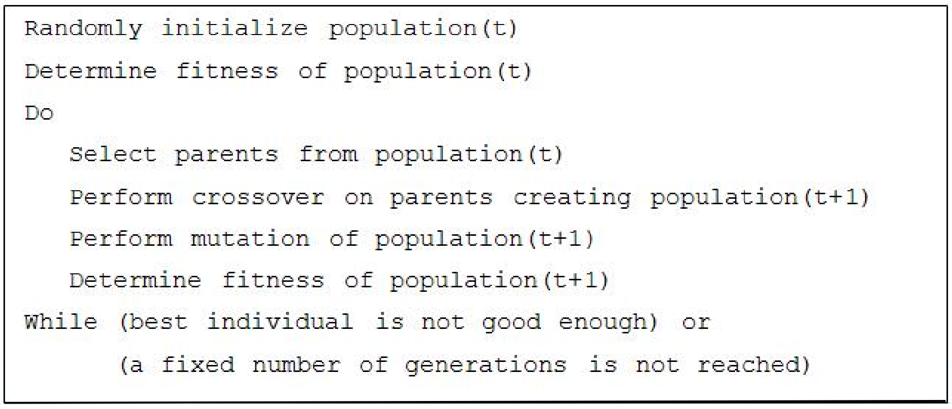
The rest of the chapter is organized as follows. Section 2.2 presents an overview of the four metaheuristic optimization algorithms. Section 2.3 overviews the existing constraint-handling methods. Section 2.4 reviews the state-of-the-art PSO in constrained optimization. Section 2.5 reviews the PSO in multi-objective optimization and Section 2.6 summarizes the chapter.

2.2 METAHEURISTIC OPTIMIZATION ALGORITHM OVERVIEW

2.2.1 Genetic Algorithms

Genetic Algorithms are heuristic search algorithms inspired by evolutionary biology such as inheritance, mutation, selection, and crossover (also called recombination). The basic techniques of the GAs are designed to simulate processes in natural systems for evolution especially those that follow Darwin’s principles of "survival of the fittest". The genetic algorithms are attributed to Holland [33, 34] and Goldberg [13].

Genetic algorithms are implemented based on natural selection. After an initial population is randomly generated, the algorithm evolves through three operators: selection, crossover and mutation. A typical GA is illustrated in Figure 2.1. An explanation to the three operators follows.



```
Randomly initialize population(t)
Determine fitness of population(t)
Do
   Select parents from population(t)
   Perform crossover on parents creating population(t+1)
   Perform mutation of population(t+1)
   Determine fitness of population(t+1)
While (best individual is not good enough) or
      (a fixed number of generations is not reached)
```

Figure 2.1 Pseudo-code for a typical genetic algorithm

Selection: Selection operator gives preference to better individuals, allows them to pass on their genes to the next generation. The goodness of each individual depends on its fitness which may be determined by an objective function or by a subjective judgment.

Goldberg and Deb [1, 35] evaluated four well-known selection schemes: proportionate reproduction (or "roulette wheel selection"), ranking selection, tournament selection and Genitor (or "steady state") selection. The proportionate selection is found to have a large computational complexity as well as a scaling problem [1]. Compared to linear ranking selection, the binary tournament selection is recommended because of its better time complexity [35]. The Genitor selection, tournament selection with larger tournament sizes, or nonlinear ranking can have higher growth ratio in time complexity [35].

Crossover: Crossover operator is the prime distinguished factor of GA from other optimization techniques. In this operation, two individuals are picked from the population using the selection operator. A crossover site along the bit strings is randomly chosen. The values of the two strings are exchanged up to this point. The two new offspring created from this mating are put into the next generation of the population. By recombining portions of good individuals, this process is likely to create even better individuals.

Mutation: With some low probability, a portion of the new individuals will have some of their bits exchanged. The purpose of mutation is to maintain diversity within the population and inhibit premature convergence. Mutation alone induces a random walk through the search space.

Since initiated, GAs have been well studied by many researchers for decades. The major advantages and disadvantages for GAs are summarized as follows.

Advantages: As the population-based evolutionary algorithm, GAs can quickly scan a vast solution set and locate good solutions rapidly. Bad individuals do not affect the end solution negatively as they are simply discarded. The inductive nature of the GA means that it doesn't have to know any rules of the problem [36]. This is very useful for complex or loosely defined problems.

Disadvantages: In many problems, GAs may have a tendency to converge towards local optima or even arbitrary points rather than the global optimum of the problem [4, 37]. The likelihood of this occurring depends on the shape of the fitness landscape: certain problems may provide an easy ascent towards a global optimum; others may make it easier for the function to find the local optima. This problem can be alleviated by increasing the rate of mutation or by using selection techniques that maintain a diverse population of solutions [33]. However, tuning the parameters such as mutation probability, recombination probability and population size remains controversial. Another difficulty for GAs is that generally the problems need to be encoded into binary format.

2.2.2 Differential Evolution

Differential evolution is a stochastic, population-based optimization algorithm introduced by Storn and Price in 1995 [29]. It is one of the most promising novel EAs [29, 38-40]. The procedure for a DE algorithm is the same as GAs, after an initial population is randomly generated; the algorithm evolves through three operators: mutation, crossover and selection.

Mutation: "*A mutation process begins by randomly selecting three individuals from the population to form a triplet. In the triplet, one member is randomly taken as the donor and the other two members are taken to make up perturbation to the donor*" [41]. For a given solution vector $x_i = (x_{i,1}, ..., x_{i,n})$, where $i \in [0, N_P - 1]$ indexes the population;

N_p is the population size; *n* is dimension; the perturbed individual is therefore generated based on the three chosen individuals as in the form:

$$v_i = x_{r1} + F \bullet (x_{r2} - x_{r3}) \tag{2.1}$$

where $r_1, r_2, r_3 \in \{0, \ldots, N_p - 1\}$ are randomly selected and satisfy: $r_1 \neq r_2 \neq r_3 \neq i$; the mutation factor $F \in [0, 2]$ introduced by Storn and Price [29] in Equation (2.1) is a control parameter of DE.

Crossover: After the mutation operation, the perturbed individual $v_i = (v_{i,1}, \ldots, v_{i,n})$ and the current individual x_i (that is, individual being mutated) are then subject to the crossover operation. Then a "trial" vector $u_i = (u_{i,1}, \ldots, u_{i,n})$ is generated by the follow equation:

$$u_{i,j} = \begin{cases} v_{i,j} & \text{if } \ rand_j \leq C_r \ \text{ or } \ j = k \\ x_{i,j} & \text{otherwise} \end{cases} \tag{2.2}$$

where $j = 1, \ldots, n$, $k \in \{1, \ldots, n\}$ is a random parameters index, chosen once for each i , and the crossover factor, $C_r \in [0,1]$, is set by the user.

Selection: The selection scheme is also different from those in GAs. The population for the next generation is selected from an individual in the current population and its corresponding trial vector, that is, $u_{i,j}$. The selection rule can be expressed as follows:

$$x_i^{t+1} = \begin{cases} u_i^{t+1} & \text{if } f(u_i^{t+1}) \le f(x_i^t) \\ x_i^t & \text{otherwise} \end{cases} \tag{2.3}$$

where t is the generation number.

In summary, at each generation, new vectors v_i are generated by the combination of vectors (x_{r1}, x_{r2}, x_{r3}) randomly chosen from the current population (mutation). The new vectors are then mixed with the predetermined target vectors x_i and produce the trial vectors u_i (crossover). Finally, the trial vector is accepted for the next generation if and only if it yields a reduction in the value of the objective function (selection).

Advantages: The strength of DE lies on the replacement of the current population by a new population that is surely a better fit. This ensures the "fittest to survive". DE is simple to implement and has empirically demonstrated excellent performance that is superior to some traditional EAs [41]. DE can be easily extended for handling continuous, discrete and integer variables as well as multiple non-linear and non-trivial constraints[40]. In addition, the algorithm doesn't require the binary encoding or scaling like those in GAs, which make the algorithm easy to implement. It has been noticed that DE has exceptional performance compared to other search heuristics methods in numerical optimization [42]. Sickel et al [43] claimed that DE achieves the same level of performance as PSO based on a power plant control application.

Disadvantage: Krink et al [42] found that DE performs poorly for noisy optimization problems (where the fitness function cannot be clearly formulated) compared to conventional EA and PSO. This is because that DE uses a rather greedy and less stochastic approach to problem solving [42]. Since the algorithm is relatively new, there are not many drawbacks reported in the literature.

2.2.3 Ant Colony Optimization

Ant Colony Optimization, introduced by Marco Dorigo in 1992 [28], is a probabilistic technique for solving optimization problems inspired by the behaviour of ants in finding paths from the colony to food. At the beginning, ants move at random. During moving, pheromone is deposited on its path. Ants detect the lead ant's path and are inclined to follow. The more pheromone deposited on the path, the more probability of the path being followed. Over time, the pheromone trail starts to evaporate to reduce its attractive strength. ACO algorithms are often applied to the problems that can be represented in the form of sets of components (nodes) and transitions (arcs), or by a set of weighted graphs such as Travelling Salesman Problems (TSPs), Vehicle Routing Problems, Network Routing Problems and Scheduling Problems.

There are some variants of ACO algorithms available. Figure 2.2 illustrates a generic colony optimization algorithm.

```
Initialize pheromone trails
Do While (termination condition not met)
   Do Until (Each ant complete a tour)
       Local trail update
   End Do
   Analyze Tours
   Global Trail Update
End Do
```

Figure 2.2 Pseudo-code for a generic ACO algorithm

At the beginning of the search process, a constant amount of pheromone is assigned to all arcs (initialization). Then ants start moving. An ant k will move from node i to node j with probability

$$p_{i,j}{}^{k}(t) = \frac{[\tau_{i,j}(t)]^{\alpha} \cdot [\eta_{i,j}]^{\beta}}{\sum_{l \in J_i^k} [\tau_{i,l}(t)]^{\alpha} \cdot [\eta_{i,j}]^{\beta}} \tag{2.4}$$

where

- J_i^k is the set of nodes that ant k still has to visit when it is on node i .
- $\tau_{i,j}(t)$ is the amount of pheromone on arc i,j at time *t*.
- α is a parameter to control the influence of $\tau_{i,j}$.
- $\eta_{i,j} = 1/d_{i,j}$ is the inverse distance of arc i,j.
- β is a parameter to control the influence of $\eta_{i,j}$.

where the arc i, j is traversed, the pheromone amount

$$\Delta\tau_{i,j}{}^{k}(t) = \begin{cases} 1/d_{i,j} & \text{if ant } k \text{ travels on arc } i, j \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

is deposited along the path. Then, the local pheromone trail is updated by

$$\tau_{i,j}(t) = \rho\tau_{i,j}(t) + \Delta\tau_{i,j}(t) \tag{2.6}$$

where ρ is the rate of pheromone evaporation.

Advantages: ACO algorithms suit better for those problems that can be graphically represented in the forms of nodes and arcs. Convergence is guaranteed but time to converge is uncertain. For TSPs with a small number of nodes, the ACO performs better against other optimization techniques. The ACO algorithms have an advantage over simulated annealing and GA approaches when the graph may change dynamically; the ant colony algorithm can be run continuously and adapt to changes in real time. This is of interest in network routing and urban transportation systems [17, 27].

Disadvantages: Theoretical analysis is difficult due to the sequences of random decisions and the dynamic probability distribution by iteration. Coding is somewhat complicated because the formulas are not straightforward. It is found for problems that have a large number of nodes, an ACO algorithm takes an exponential time to converge [17, 27].

2.2.4 Particle Swarm Optimization

Particle Swarm Optimization is a stochastic metaheuristic method for optimizing numerical functions on the metaphor of social behaviours of flocks of birds and schools of fish [15]. A PSO algorithm consists of individuals, called particles that form a swarm. Each particle represents a candidate solution to the problem. Particles change their positions by flying in a multi-dimensional search space looking for the optimal position. During flight, each particle adjusts its position according to its own experience and the experience from its neighbouring particles, making use of the best position encountered by itself and the best position in the entire population (global PSO) or its local neighbourhood (local PSO). The performance of each particle is measured by a predefined fitness function (objective function) which is problem-dependent.

The original local PSO algorithm is expressed as follows:

$$v_{id}^{(t+1)} = v_{id}^{(t)} + c_1 r_{1id}^{(t)} (pBest_{id}^{(t)} - x_{id}^{(t)}) + c_2 r_{2id}^{(t)} (lBest_{id}^{(t)} - x_{id}^{(t)}) \tag{2.7}$$

$$x_{id}^{(t+1)} = x_{id}^{(t)} + v_{id}^{(t+1)} \tag{2.8}$$

where $x_i = (x_{i1}, x_{i2}, ...x_{iD})$ denotes the *i-th* particle in a D-dimensional search space; $pBest_i = (p_{i1}, p_{i2}, ...p_{iD})$ denotes the best previous position of the *i-th* particle in the flight history; $lBest_i = (p_{g1}, p_{g2}, ...p_{gD})$ denotes the best particle of the neighbourhood; $v_i = (v_{i1}, v_{i2}, ...v_{iD})$ denotes the velocity for particle i; c_1 and c_2 are two positive constants, called the cognitive and social parameters respectively (or acceleration constants); r_{1id} and r_{2id} are two random numbers uniformly distributed in the range [0, 1], which are used to maintain the diversity of the population; t denotes the iteration.

So far, several variants of the PSO algorithm have been developed. Among them, two variants are prominent. One is the *Inertia weight PSO* and the other one is the *Constriction PSO* (as cited in [44]). Equation (2.9) and Equation (2.10) below give a generic PSO form containing these two parameters.

$$v_{id}^{(t+1)} = \chi[w v_{id}^{(t)} + c_1 r_{1id}^{(t)} (pBest_{id}^{(t)} - x_{id}^{(t)}) + c_2 r_{2id}^{(t)} (lBest_{id}^{(t)} - x_{id}^{(t)})] \tag{2.9}$$

$$x_{id}^{(t+1)} = x_{id}^{(t)} + v_{id}^{(t+1)} \tag{2.10}$$

where w is the inertia weight which is considered critical for the PSO's convergence behaviour; χ is a constriction coefficient. These two new parameters are used for controlling particles' velocities.

Three variants of PSO can be observed from the generic PSO as in Equation (2.9) and Equation (2.10): when $\chi = 1$, the generic PSO is an inertia weighted PSO; when $w = 1$, the generic PSO is a constriction PSO; when $w = 0$, $\chi = 1$, the generic PSO is called a bare bones PSO [45].

According to the PSO formulas, the computation is straightforward once the parameters are settled. A suitable value for the inertia weight w usually provides balance between global and local exploration abilities and consequently results in a reduction of the number of iterations required to locate the optimum solutions. The experimental results indicated that it is better to initially set the inertia to a large value, in order to promote global exploration of the search space, and gradually decrease it to get more refined solutions [20]. Thus, an initial value around 1.2 and a gradual decline towards 0 can be considered as a good choice for w [19, 46]. The c_1 and c_2 are not critical for PSO's convergence. However, proper fine-tuning may result in faster convergence and alleviation of local minima. Kennedy proposed to use $c_1 = c_2 = 2$ (as cited in [19]). But experimental results indicated that $c_1 = c_2 = 0.5$ might provide even better results [19]. It is reported by Carlisle and Dozier (as cited in [19]) that it might be even better to choose a larger cognitive parameter c_1 than a social parameter c_2 but with $c_1 + c_2 \leq 4$. However, most parameters recommended are based on the experiments for unconstrained optimization problems. The parameter setting for constrained problems varies.

Among the EAs, PSO is the only one that does not follow the “survival of the fittest” concept. It does not utilize a direct selection function. The particles with lower fitness can survive during the optimization and potentially visit any point of the search space [47].

Advantages: Compared with other EAs, PSO can be easily implemented and it is computationally inexpensive because it has lower requirements for memory and CPU speed [48]. Moreover, it does not require gradient information of the objective function under consideration, but only its values, and it used only primitive mathematical operators [19]. PSO has been proved to be an efficient method for many global optimization problems (single objective) and in some cases it does not suffer the difficulties encountered by other EA techniques [15]. Research has also shown that PSO usually results in faster convergence rate than the GAs [16, 19, 46].

Disadvantages: PSO algorithms have more parameters than other algorithms. Some parameters like w and χ are sensitive. A set of parameters may work well for some functions but work badly for other functions. The parameters suggested so far are all based on trial-and-error methods. Thus, how to select appropriate parameters for different applications is an issue.

2.2.5 Discussion

Like other EAs, the four main stream algorithms reviewed in section 2.2 work with a population of solutions instead of one solution in each iteration. Working with a number of solutions provides an EA with the ability to capture multiple optimal solutions in one single simulation run [1].

The population-based metaheuristic algorithms do not assume any particular structure of a problem to be solved. The flexible representation makes the algorithms suitable to be used in a wide variety of problem domains.

GA and DE have two operations, namely *selection* and *search (crossover* and *mutation)*. In the selection operation, better solutions in the current population are emphasized. In the search operation, new solutions are created by exchanging partial information among solutions of the mating pool and by perturbing them in their neighbourhood.

ACO and PSO belong to a discipline called Swarm Intelligence[19, 48]. One of the basic principles of swarm intelligence is "adaptability". That is, the ants or swarm are able to alter their behaviours toward the better solutions.

Regarding implementation difficulty, it seems that PSO and DE are easier because the formulas are straightforward. ACO is somehow difficult because of the random decisions and the dynamic probability distribution. GA has the difficulties in encoding problems into binary format and scaling.

The common issues for these algorithms include integration of constraint-handling strategies, diversity maintaining, and speed-diversity trade-off. These issues remain open.

2.3 CONSTRAINT-HANDLING METHODS OVERVIEW

The real world optimization cannot escape from handling constraints. Unfortunately, the original EAs like those reviewed in the previous sections do not have constraint-handling mechanisms integrated. How to integrate the constraint-handling methods with the EAs becomes very necessary.

Constraint-handling methods for EAs have been grouped into four categories [22, 49], as follows:

- methods based on preserving feasibility of solutions,
- methods based on penalty functions,
- methods which make a clear distinction between feasible and infeasible solutions,
- other hybrid methods.

We review each of these methods in turn.

2.3.1 Preservation of Feasibility

Preservation of feasibility methods assume that all individuals start at the feasible region. It does not allow any solutions that violate any of the assigned constraints to evolve to the next generation. To implement this, firstly, all individuals have to be initialized in the feasible region. During evolution, any infeasible solutions should be disregarded (rejected) or fixed (repaired) to be feasible in some way. One example is GENOCOP systems [14] which use so called *specialized operators*. By using these operators, the feasible individuals are transformed into other feasible individuals in order to ensure the offspring solution vectors are always feasible in any case [37]. The GENOCOP assumed that all constraints are linear.

In order to keep population size, if any infeasible solutions are disregarded during evolution, new feasible individuals should be regenerated.

Obviously, the drawback of this constraint-handling method is that the initialization process may be impractically long or almost impossible for those constrained nonlinear optimization problems (CNOPs) that have extremely small feasible spaces[50].

Disregarding the infeasible solutions is questionable in EAs. Recent research shows that maintaining infeasible solutions during evolution can improve the computing performance [51]. An EA-hard problem can be transformed to an EA-easy problem by exploiting infeasible solution [52]. Thus, handling constraints by using the preservation of feasibility approach may not be a good idea unless the problems are simple. "Simple" means the problem has a large feasible space which makes the initialization easier and the constraints are linear only.

2.3.2 Penalty Function Approach

Penalty function approach is the most common approach for solving constrained optimization problems. The penalty functions are used to degrade the quality of an infeasible solution [37] . In this manner, the constrained problem is transformed to an unconstrained one by using the modified evaluation function

$$F(x) = f(x) + R \bullet \Omega\ (x) \tag{2.11}$$

where *R* is the penalty parameter; $\Omega(x)$ is the constraint violation which can be calculated by

$$\Omega(x) = \sum_{j=1}^{m} \begin{cases} |g_j(x)|\ , & \text{if } g_j(x) > 0; \\ 0\ , & \text{otherwise.} \end{cases} \tag{2.12}$$

where $g_j(x),\ j = 1, 2, ...m,$ are the constraint functions. If no constraint violation occurs, $\Omega(x)$ is zero, the penalty is zero, $F(x) = f(x)$; otherwise, it is positive (minimization problems are assumed). The penalty parameter *R* is used to make both of the terms on the right side of the Equation (2.11) to have the same order of magnitude [1]. *R* varies from case to case.

Penalty functions can be static (or stationary) or dynamic (or non-stationary). Static penalty functions use fixed penalty values *R* throughout the minimization, while in dynamic penalty functions, the penalty values *R* are dynamically modified [1]. Literature shows, for single objective optimization problems, results obtained using dynamic penalty functions are almost always superior to those obtained through static function [46]. However, most studies in multi-objective optimization problems (MOOPs) use carefully chosen static values of *R* [1].

There are some difficulties in choosing penalty parameters, as indicated by Deb [1]:

- *If a smaller than adequate penalty parameter value is chosen, the penalty effect is less and the resulting optimal solution may be infeasible.*
- *If a larger than adequate penalty parameter value is chosen, the constraints will be over-emphasized, and the minimization algorithms usually get trapped in local minima.*

Unfortunately, there is no rule for choosing an adequate penalty parameter but trial-and-error. Thus, the penalty function based approach for constraint-handling is problem dependent. This is a major drawback of the approach.

2.3.3 Methods Based on Searching for Feasibility

The constraint-handling methods based on searching for feasibility emphasize a distinction between feasible and infeasible solutions[22, 37]. One example is the so called "Behavioural memory method" proposed by Schoenauer and Xanthakis (as cited in [37]) which considers the problem constraints in sequence. That is, it only considers one constraint at a time. Once a sufficient number of feasible solutions are found in the presence of one constraint, the next constraint is considered. Eventually, all constraints will be satisfied.

Another method called "superiority of feasible points" is based on a classical penalty approach with one notable exception [53]. Each individual is evaluated by the formula:

$$F(x) = f(x) + r\sum_{j=1}^{m} f_j(x) + \theta(t,x) \tag{2.13}$$

where r is a constant; the original component $\theta(t,x)$ is an additional iteration-dependent function that influences the evaluations of infeasible solutions [37]. This method distinguishes feasible and infeasible individuals by adopting an additional heuristic rule suggested earlier in Rechardson et al [54]: for any feasible individual x and any infeasible individual y: $f(x) \prec f(y)$, that is, any feasible solution is better than any infeasible one. This can be achieved in many ways. The point is to penalize the infeasible individuals such that they cannot be better than the worst feasible individual.

The third example in this category is the repairing of infeasible individuals, introduced in GENOCOP III [37] . The method needs to find some feasible individuals as reference points and then apply an algorithm to repair the infeasible individuals to be feasible. For example, if an individual s is not feasible, the system selects one of the reference points, say r from P_r (P_r is a population that contain all feasible references), and creates a sequence of random points z from a segment between s and r by generating random numbers a from the range (0,1): $z = a \times s + (1 - a) \times r$. Once a fully feasible z is found, replace s by z.

However, the *behavioural memory method* requires that there is a linear order of all constraints, and the order in which the constraints are processed influences the results provided by the algorithm in terms of total running time and precision [30]. The *superiority of feasible points* approach will fail in cases where the ratio between the feasible region and the whole search space is too small (for example, when there are constraints very difficult to satisfy) unless a feasible point is introduced in the initial population [30]. The *repairing infeasible individuals* approach may be a good choice when an infeasible solution can be easily transformed into a feasible solution. However this approach is problem-dependent too since a specific repair algorithm has to be designed for each particular problem [30, 37].

2.3.4 Other Hybrids

Some earlier approaches combine evolutionary computation techniques with deterministic procedures for numerical optimization problems. One is to combine EA and the direction set method (as cited in [37]). However, these methods have some limitations as deterministic methods do.

Another interesting method is to use evolutionary multi-objective optimization techniques to handle constraints. These multi-objective constraint-handling approaches can be categorized into two groups:

- approaches transform a constrained optimization problem into an unconstrained **bi-objective** optimization problem;
- approaches transform a constrained optimization problem into an unconstrained **multi-objective** optimization problem.

However, although a problem is now a multi-objective unconstrained optimization problem (MOOP), there is no need for good trade-offs between multi-objectives. We want to find the best possible solutions that do not violate any constraints.

In the following paragraphs, we present a review to the typical approaches for the two categories of the multi-objective constraint-handling methods. The first group (that is, bi-objective model) is specifically focused since the model will be adopted in our research.

Approaches for bi-objective optimization model:

The idea for the first group of the multi-objective constraint-handling methods is to restate a single objective optimization problem in such a way that two objectives would be considered: the first would be to optimize the original objective function f and the second would be to minimize

$$\Phi(x) = \sum_{i=1}^{m} \max(0, g_i(x)) \qquad (2.14)$$

where $\Phi(x)$ is the total amount of constraint violations; $g_i(x)$ for $1 \le i \le m$ are the constraint functions. The optimization problem is transformed to find x that minimize $F(x) = (f(x), \Phi(x))$. An ideal solution x would have $\Phi(x) = 0$ (that is, satisfy all constraints) and $f(x) \le f(y)$ for all feasible y (minimization problem is assumed).

The examples which fall in the bi-objective optimization model can be found in [12, 31, 55-58].

Camponogara and Talukdar [12] proposed a scheme that calculates improvement directions from Pareto sets defined by the objective function $f(x)$ and constraint violations $\Phi(x)$. The direction search $d = (x_i - x_j)/|x_i - x_j|$, where $x_i \in S_i, x_j \in S_j$, and S_i and S_j are

Pareto sets, is intended to simultaneously minimize all the objectives. The linear search is performed during the crossover stage of GA. The scheme outperformed GAs based on penalty methods [12]. However, the scheme has problems to maintain diversity [59]. Using line search within a GA adds some extra computational cost [30, 59]. It is not clear what the impact is on the segment chosen to search in the overall performance of the algorithm[30, 59].

Surry et al [55] proposed COMOGA (Constrained Optimization by Multi-objective Optimization Genetic Algorithms) where the population was ranked based on constraint violations. One portion of the population was selected based on constraint ranking, and the rest based on real cost (fitness) of the individuals. "*The aim of the proposed approach to solve this bi-objective problem is based on reproducing solutions which are good in one of the two objectives with other competitive solutions in the other objective (e.g., constraint violation)"* [59] . COMOGA was tested on a gas network design problem providing slightly better results than those provided by a penalty function approach. Its main drawbacks are that it requires several extra parameters and that it has not been tested extensively [30, 59].

Coello [31] proposed a ranking procedure where "*each individual is assigned a rank based on its degree of dominance over the rest of the population. Feasible individuals are always ranked higher than infeasible ones, and the degree of constraint violation determines the rank among infeasible individuals".* This approach was tested on a set of engineering design problems providing competitive results. Its main drawback is the computational cost of the technique and its difficulty to handle equality constraints [60].

Zhou et al [56] proposed a ranking procedure based on Pareto Strength [61] for the bi-objective problem. The simplex crossover operator is used to generate a set of offspring where the individual with the highest Pareto strength and the solution with the lowest sum of constraint violation are both selected to take part in the population for the next generation. The approach was tested on a subset of the well-known benchmark functions. "*The results were competitive but using different set of parameters for different functions, which made evident the sensitivity of the approach to the values of its parameters*"[59].

Similar to [56], Wang and Cai [57] also employed a simplex crossover operator with a set of parents to generate a set of offspring. Additionally, they used an external archive to store infeasible solutions with a low sum of constraint violation in order to replace some random solutions in the current population. The approach provided good results in 13 well-known test problems. However, a different set of values for the parameter were used, depending of the dimensionality of the problem [59].

Venkatraman and Yen [58] proposed a generic, two-phase framework for constrained optimization problems using GAs. In the first phase, the objective function is completely disregarded and the constrained optimization problem is treated as a constraint satisfaction problem. The second phase starts when the first feasible solution was found. Now both objectives are taken into account and nondominated sorting [62] is used to rank the populations. The approach provided good quality results in 11 well-known benchmark problems but lacked consistency.

Approaches for multi-objective optimization model:

The procedure for the second group of the multi-objective constraint-handing methods is to redefine the single objective optimization of $f(x)$ as a multi-objective optimization problem in which we will have $m+1$ objectives, where m is the number of constraints. Then any multi-objective optimization techniques to the new multi-objective vector $F(x) = (f(x), f_1(x), ..., f_m(x))$, where $f_1(x), ..., f_m(x)$ are the original constraints of the problem, can be applied. An ideal solution x would thus have $f_i(x) \le 0$ for $1 \le i \le m$ (that is, satisfy all constraints) and $f(x) \le f(y)$ for all feasible y .

A number of approaches have been developed in solving a multi-objective problem with objective function and constraints as separate objectives. Some examples are: Coello [63], Liang et al [64], Ray et al [65-67] and Coello et al [68, 69]. Briefly, these approaches make use of the multi-objective optimization techniques plus some special considerations to the objectives of constraint satisfaction. However, no approach dominates. All approaches have certain drawbacks. For details, please refer to the multi-objective optimization references such as the one by Deb [1].

It is observed that most approaches that appoint multi-objective constraint-handling methods are via GAs.

2.4 PSO IN SINGLE OBJECTIVE CONSTRAINED OPTIMIZATION

Although EAs have been successful in many applications, their utilisation in solving constrained optimization problems remains problematic because their original versions lack a mechanism to incorporate constraints with the fitness function [30, 59, 60, 70]. As a member of the evolutionary optimization techniques, PSO offers some advantages over other algorithms and has proven to be effective and efficient for solving real-valued global unconstrained optimization problems [16, 20]. For constrained optimization problems, there have been only a few attempts [70]. These attempts mainly focus on adopting those constraint-handling methods that have been used in genetic algorithms.

Hu et al [71, 72] presented a modified PSO algorithm for constrained nonlinear optimization problems. In their research, the *preserving feasibility* strategy is employed to deal with constraints. That is, all particles start with feasible individuals. During flying, only those feasible particles are counted and those infeasible particles are ignored. Eleven well-known constrained benchmark numerical functions are tested. Results reveal that PSO can find the optimum for most cases, although some cases need a large population size and more iterations to converge. However, the computation cost is questionable since it assumes that all particles start with feasible individuals, which requires a longer initialization process. This approach also has diversity problems. The computing performance depends on the initial random population.

Parsopoulos and Vrahatis [46, 73] reported an investigation to appoint the *penalty function approach* to deal with constrained optimization problems through PSO. The simulation results to the six constrained optimization problems and four engineering optimization problems demonstrated the capability of the PSO method in solving constrained optimization problems with promising results. The main drawback for this approach is that many parameters are problem-dependent. The approach has not addressed the diversity issue.

Pulido and Coello [70] proposed a criterion-based selection scheme which is based on the constraint dominance concept [1] for handling constraints. The idea is: when two feasible particles are compared, the particle that has the highest fitness value wins; if one of the particles is infeasible and the other one is feasible, the feasible particle wins; if both particles compared are infeasible, the particle that has the lowest value in its total violation of constraints wins. They also used a turbulence operator to perturb the swarm as to avoid local convergence. The approach was tested on thirteen benchmark functions with the competitive results. However, the approach did not perform consistently in problems G5, G10 and G13 which are considered complicated problems. Also, the constraint violation measurement seems not easy to implement in this approach.

He et al [74] also adopted the preserving feasibility and searching for feasibility constraint-handling method in the engineering optimization problems. The drawback for this approach is the same as [71, 72], that is, a higher computation cost was caused by the longer initialization process. No diversity control was mentioned in the approach.

Zavala et al [75] proposed PESO (Particle Evolutionary Swarm Optimization Algorithm) in 2005 for tackling constrained optimization problems. Similar to Pulido and Coello [70], constraints were handled by the feasibility rules. The approach proposed two new perturbation operators: "c-perturbation" and "m-perturbation". The goal of these operators is to fight premature convergence and poor diversity issues [75] . The simulation results for the benchmark functions were very competitive. However, the approach did not achieve a satisfactory result in G13 which has a large search space with a small feasible region.

Wei and Wang [76] adopted the multi-objective constraint-handling method into PSO algorithm. The approach converts a single objective constrained optimization problem into a bi-objective unconstrained problem. Then a particle can find its personal best and the global best by applying the feasibility rules like those applied in [70, 75]. They also proposed a three-parent crossover operator to modify the new particle generated from the previous iteration by the PSO algorithm. The authors claimed that the new crossover operator would make the offspring *having greater probability to locate near the feasible region*. However, the approach has only tested on four simple problems (lower constrained) with moderately good results. The crossover operator was also questionable since it disturbs the PSO formulation.

Zielinkski and Laur [77] proposed an approach similar to [70]. The total sum of constraint violation was used for measurement in the selection rules. The simulation results showed that the approach was successful in many test functions. It also demonstrates the approach has some difficulties for the problems that have high dimension and high number of equality constraints.

He and Wang [78] proposed a co-evolutionary particle swarm optimization for constrained engineering design problems, where PSO is applied with two kinds of swarms for evolutionary exploration and exploitation in spaces of both solutions and penalty factors. The drawback is the penalty factors are problem dependent. Some results reported from this reference are not feasible.

To summarize, the "feasibility rules" (or selection rules) approach has been attracting more attention than other constraint-handling methods in PSO. The reason is: PSO algorithm has straightforward formulas and a fundamental decision is to determine the best neighbourhood particle *lBest* and the best personal experience *pBest.* The feasibility rules serve this purpose better. The results reported from PESO [75] seems the best-so-far results. It introduced two perturbation operators which make PSO more stochastic. The adoption of multi-objective constraint-handling methods in PSO, especially to transfer the original problem into multi-objective optimization model has not been properly investigated.

2.5 PSO IN MULTI-OBJECTIVE OPTIMIZATION

There are a number of proposals that discussed the multi-objective optimization using PSO algorithm.

Hu and Eberhart [79] introduced a dynamic neighbourhood strategy to select the global best. In their method, one objective called *fixed objective* must be selected firstly. Then the distance of the current particle from other particles (in the objective space) is calculated in terms of the fixed objective function. Then find the nearest *m* (neighbourhood size) particles as the neighbours of the current particle based on the distances. Lastly, find the local optima among the neighbours in terms of the fitness value of the second objective function. The personal best is determined by the Pareto-dominance concept. This approach is tested by seven bi-objectives functions with effective results. No comparison was made with any other models, or the true Pareto fronts for the problems. How to select the fixed objective and how it affects the results are unknown. The formulation suits bi-objective problems only.

Coello and Lechuga [80] proposed a grid method. The objective space is divided into many small hypercubes, and a fitness value is assigned to each hypercube depending on the number of elite particles that lie in it. The more elite particles the hypercube have, the less fitness value of it. Then one of the hypercubes is selected by roulette-wheel method. Global best is a random particle selected from the selected hypercube. The personal best is updated by the Pareto-dominance concept. This approach is tested by three bi-objective test functions with effective results. Diversity is well maintained by using a grid method. However, the implementation for selecting global best is complicated when a large number of objective functions are involved.

Parsopoulos and Vrahatis [81] adopted the *Weighted Aggregation* technique in applying PSO in multi-objective problems. According to this approach, all objectives are summed to a weighted combination. Then a multi-objective problem is converted into a single objective problem. The weighted aggregate algorithm needs to be run many times to produce estimated Pareto optimal points. Therefore, it is computationally expensive.

Parsopoulos et al [82] also introduced the vector evaluated PSO which uses multi swarms. Each swarm is evaluated using only one of the objective functions. And the best particle of each swarm is selected to act as the global best particle to another swarm. The implementation for this approach is complicated since there is information exchange between multi swarms.

Fieldsend and Singh [83] used a dominated tree for storing the particles, which consists of a list of composite points ordered by weakly dominated relations. In this method the selection of the best global in the population is based on its closeness to a particle in the archive. It has been shown to be significantly better than the methods used in a recent alternative multi-objective PSO. However, the authors mentioned that the approach may experience problems if there is little or no relationship between “closeness” in objective space and “closeness” in decision space[83] .

Salazar-Lechuga and Rowe [82] adopted a fitness sharing concept [84] in PSO algorithm. The idea of fitness sharing is to distribute a population of individuals along a set of resources. When an individual is sharing resources with other individuals, its fitness is disregarded in proportion to the number and closeness to individuals that surround it [85]. The fitness sharing helps to maintain diversity between solutions in the non-dominated repository. The global best is selected by using Roulette Wheel method from the repository. The approach was tested and compared with other methods and the effectiveness of the approach was shown.

Huo et al [86] presented an approach which select a global best in such a way: firstly, best particles for each objective function are selected; secondly, the mean value of those particles is calculated. Then the mean value is set as the *gBest*. This *gBest* is not the true position but a virtue position. The personal best is selected by two steps: the first step calculates the distance between the current particles to the non-dominated archive solutions; then the archive member with minimal distance to one particle will be selected as *pBest*. A *distance valve* strategy is used for diversity control. The approach was tested by three functions with good results. However, the parameters were differently set.

There are more than 25 proposals [87] that have addressed multi-objective optimization using PSO algorithm. However, the majority of them are constraint free. Two papers addressed the constrained multi-objective optimization using PSO algorithm. Ji [88] presented a divided range multi-objective PSO for distributed computing. For constraint-handling, the author adopted the symbiosis mechanism where the feasible particles evolve towards Pareto-front; and infeasible particles evolve toward feasibility guided by an unfeasibility function. A gradually decreased threshold is used for the proportion of infeasible particles. The approach was tested by three constrained MO functions. The author claimed that the Pareto fronts were achieved but no detailed results were given. The unfeasibility function used to guide the infeasible particles is unclear. Reddy and Kumar [89] presented EM-MOPSO which combines PSO technique with Pareto dominance criteria to evolve non-dominated solutions. The constraint-handling is based on the constraint dominance concept [1]. The global best particle (*gBest* of *lBest*) is randomly selected from the ERP (an external repository) where all non-dominated solutions are stored. The personal best particle is determined by the Pareto-dominance concept. The approach was tested by four constrained MO functions with the promising results. However, the implementation looks very complicated and the use of ERP makes the computation expensive.

As we can see, constrained multi-objective optimization using PSO algorithm has not been well studied yet.

2.6 SUMMARY

This chapter has reviewed the popular metaheuristic optimization algorithms and the constraint-handling methods in evolutionary optimization. A review of PSO in constrained optimization and in multi-objective optimization has also been presented.

The population-based algorithms have the ability to capture multiple optimal solutions in one single simulation run which leads to a high computing performance. The flexible representations make the algorithms appropriate to be used in a wide variety of problem domains. The four population-based algorithms (that is, GA, DE, ACO and PSO) and their pros and cons have been specially focused in the review. GA and DE use operators to produce better population for the next generation. ACO and PSO are able to alter their behaviours toward the better solutions by their "adaptability" feature. In regard to implementation difficulty, PSO and DE are easier because the formulas are straightforward. The common open issues for these algorithms are integrating constraint-handling strategies, diversity maintaining, and speed-diversity trade-off.

A review of the four types of constraint-handling mechanisms has been conducted. The preserving feasibility constraint-handling methods are easy to implement; but they are not efficient since a long initialization process is needed. Since the penalty factors are problem-dependent, the applicability is restricted by penalty-based constraint-handling approaches. The methods based on searching for feasibility also have some drawbacks like problem-dependent. Comparatively, the multi-objective constraint-handling methods offer some advantages over other approaches. We have specifically reviewed the literature in multi-objective constraint-handling methods.

The state-of-the-art PSO in single objective constrained optimization has been reviewed. Most of the papers focus on adopting those constraint-handling methods that have been used in GAs in PSO algorithm. The "selection rules" based constraint-handling approach has been popular than the others in PSO. Integrating the multi-objective constrained-handling method with the PSO algorithm is in need of further study.

Lastly, the PSO in multi-objective optimization has been reviewed. Most multi-objective PSO proposals are constraint free. Constraint-handling in multi-objective optimization problems via PSO has not been well investigated.

Chapter 3
A MULTI-OBJECTIVE CONSTRAINT-HANDLING METHOD WITH THE PSO ALGORITHM

3.1 INTRODUCTION

In Chapter 2, we reviewed the most popular metaheuristic optimization algorithms and the constraint-handling methods. It is apparent that constraint-handling in evolutionary optimization remains problematic. The drawback for the preserving feasibility method is that the initialization process may be impractically long or almost impossible for those CNOPs (Constrained Nonlinear Optimization Problems) that have extremely small feasible spaces [50]. The computation is very costly. The penalty function approaches have drawbacks in finding appropriate penalty factors which need to be carefully fine-tuned [60]. Therefore, they are problem-dependent [1, 32]. The criteria-based selection approaches are becoming popular because the PSO algorithm has straightforward formulas and the selection rules make the comparison (between particles) possible, which helps in determining the best neighbourhood particle *lBest* and the best personal particle *pBest*. The adoption of the multi-objective constraint-handling method in PSO, especially to transfer the original problem into multi-objective optimization models, needs further investigation.

Multi-objective constraint-handling was firstly proposed by Fonseca and Fleming back in 1995 (as cited in [59]). The main idea is to treat the constraints as extra objectives. By doing so, an original single objective constrained optimization problem can be transformed into a multi-objective unconstrained optimization problem. Then the techniques for multi-objective optimization can be employed. Since 1995, a number of models have been developed using this idea. Several representative examples include COMOGA [55], Camponogara and Talukdar [12], Mezura-Montes and Coello [31, 90, 91] and Jimenez et al.[92]. Unfortunately, these models have some shortcomings. For example, some of them add extra computational cost [12, 31]; others require extra parameters [55, 68, 92]. A detailed review of these models has been presented in Chapter 2. It is noticed that most of these models are built on GAs.

In this chapter, we propose an approach to integrate the multi-objective constraint-handling mechanism with a dynamic neighbourhood PSO algorithm. By converting a single objective constrained optimization problem into a bi-objective unconstrained optimization problem, the proposed approach aims to minimize the original objective function and the total amount of constraint violations (the second objective). The concept of Pareto domination from multi-objective optimization is adopted in determining a particle's best past experience and the best social experience in the group. The second objective is used as a benchmark to select particles (defined in selection rules). An adaptive inertia weight factor and a minor perturbation are introduced to improve the convergence and the diversity. The dynamic neighbourhood topology is proposed for improving the algorithm performance. The simulation results to the thirteen numerical benchmark functions will be presented.

The rest of the chapter is organized as follows: Section 3.2 presents the problem formulation and transformation; Section 3.3 describes the proposed multi-objective constraint handling incorporating with a dynamic neighbourhood PSO algorithm. Section 3.4 presents the simulation results to the numerical benchmark functions. Section 3.5 presents the results of experiments of two comparative studies. Section 3.6 summarizes the chapter.

3.2 PROBLEM FORMULATION AND TRANSFORMATION

As mentioned in Chapter 1, a general single objective constrained optimization problem can be stated as:

$$\left.\begin{array}{lll} \text{minimize} & f(x) & \\ \text{subject to} & g_i(x) \le 0, & i = 1, 2, ..., m; \end{array}\right\}$$

where m is the total number of constraints. An equality constraint h is regarded as an inequality constraint with a tolerance δ, that is, $|h_j(x)| - \delta \le 0$.

The multi-objective constraint-handling method (as addressed in Chapter 2) transforms a global optimization problem into a bi-objective problem where the first objective is to optimize the original objective function $f(x)$ and the second is to minimize

$$\Phi\ (x) = \sum_{i=1}^{m} \max\ (0, g_i(x))$$

where $\Phi(x)$ is a total amount of constraint violations. From the above equation, if a solution vector $x = (x_1, x_2, ..., x_n)$ satisfies all constraints, that is, $g_i(x) \le 0$ for $i = 1, 2, ..., m$, $\Phi(x)$ returns a zero. Otherwise, it returns a positive number indicating the total amount of constraint violations. Thus, the optimum value for $\Phi(x)$ is zero. Therefore, the single objective constrained optimization problem as in Equation (1.2) can be transformed into:

$$\left.\begin{array}{ll} \text{minimize} & F(x) = (f(x)\ ,\ \Phi(x)\ ) \\ \text{where} & \Phi(x) = \sum_{i=1}^{m} \max\ (0,\ g_i(x)) \end{array}\right\} \quad (3.1)$$

Equation (3.1) is a bi-objective unconstrained optimization problem.

For a general multi-objective optimization problem, the ideal procedure is to find a set of Pareto-optimal solutions first and then choose one solution from the set by using some other higher-level information for consideration [1]. For global constrained optimization as in model (3.1), constraint satisfaction is a must and it is more important than real objective function minimization. That is, if a solution is not feasible, no matter how fit its objective function is, it is useless. In other words, if a solution is feasible, even if it is not fit enough, it can be still considered as a candidate solution. Therefore, the second objective $\Phi = 0$ (totally constraint satisfied) or $\Phi \le \varepsilon$ (total constraint nearly satisfied), can be used as higher-level information to guide decision making during the search. The ε is a small positive number which indicates the feasibility tolerance. This is the so called "decision making during the search" approach in multi-objective optimization [93]. Figure 3.1 below is an example to illustrate the Pareto-front, feasible solutions and the desired solution to the established bi-objective optimization problem as described by Equation (3.1), and the final solution will fall in A to B depending on how ε is selected.

Most multi-objective optimization methods use a Pareto dominance concept to search for non-dominated solutions, since this concept allows a way to compare solutions with multiple objectives. The definition for Pareto dominance can be found from Chapter 1.

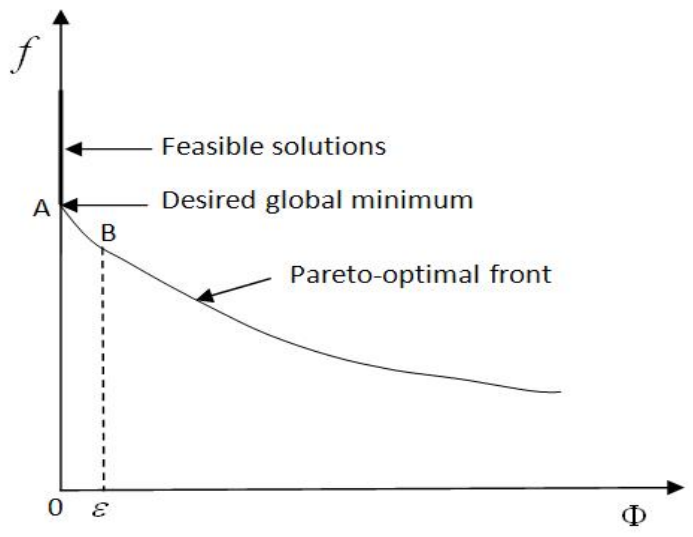


Figure 3.1 The Pareto-optimal front, feasible solutions and desired constrained minimum for a bi-objective constraint handling optimization problem

3.3 A MULTI-OBJECTIVE CONSTRAINT-HANDLING METHOD WITH A DYNAMIC NEIGHBOURHOOD PSO ALGORITHM

3.3.1 PSO Algorithm

The generic local variant model PSO formulation in Equation (2.9) and Equation (2.10) are adopted. If a maximum velocity $v_{\max}$ is used, PSO algorithm can be rewritten as:

$$\begin{aligned} v_{id}^{(t+1)} &= \chi[wv_{id}^{(t)}+c_1r_{1id}^{(t)}(pBest_{id}^{(t)}-x_{id}^{(t)})+c_2r_{2id}^{(t)}(lBest_{id}^{(t)}-x_{id}^{(t)})] \\ v_{id}^{(t+1)} &= v_{\max}, \qquad if\ v_{id}^{(t+1)} > v_{\max} \\ v_{id}^{(t+1)} &= -v_{\max}, \qquad if\ v_{id}^{(t+1)} < -v_{\max} \end{aligned} \tag{3.2}$$

$$x_{id}^{(t+1)} = x_{id}^{(t)} + v_{id}^{(t+1)} \tag{3.3}$$

Although Clerc and Kennedy [94] suggested the use of a constriction coefficient χ to the velocity formula and showed that the constriction coefficient can converge without using $v_{\max}$, their suggestion is based on the unconstrained optimization experiments. In order to ensure convergence and explore a wider area, in this research, both χ and $v_{\max}$ will be used.

Neither the original PSO algorithm nor its variations have a mechanism to incorporate constraint-handling with the algorithms. The parameters suited better for unconstrained problems may not be suitable for constrained problems. In order to integrate constraint handling with PSO, we introduce a few selection rules to determine the particles' behaviour in the next section.

3.3.2 Selection Rules

In the PSO algorithm, the main task is to determine a particle's best past location *pBest* and which particle is the best particle *lBest* among a neighbourhood. For a single objective optimization problem, this can be easily determined by the objective function. Considering our optimization model is now a bi-objective unconstrained problem, the notion of dominance comparison can be adopted [1]. The following selection rules are defined:

- Non-dominated particles are better than dominated ones.
- When two particles do not dominate each other, a particle with lower Φ (constraint violations) is better than a particle with higher Φ .

These two rules will be used in comparing particles.

3.3.3 Performance-Based Dynamic Neighbourhood Topology

Neighbourhood topology determines how particles are allocated in a neighbourhood and how particles communicate with each other. Several neighbourhood topologies have been proposed by Kennedy et al [95, 96] . It was found that von Neumann topology (north, south, east and west, of each particle placed on a two dimensional lattice) is an overall winner among many different communication topologies [96]. However, the von Neumann topology is difficult to implement.

Two most commonly used neighbourhood topologies are listed as follows:

- Circle (ring) neighbourhood topology: each individual is connected to its k immediate neighbours only, for example, if k is 2, a particle with index i will have the particle index $i-1$ and particle $i+1$ as its neighbours. It is realized that a particle and its k neighbours are not geographically close neighbours in both search space and objective space. They are conceptually neighbours according to their indexes.
- Star neighbourhood topology: every individual is connected to every other individual. This is a communication intensive topology since each particle has all other particles in the swarm as its neighbours. The star topology is a global model of PSO.

Figure 3.2 illustrates the circular ring ((a)) and the star ((b)) neighbourhood topologies. The circular ring topology tends to allow for broader exploration of the problem space. When one particle finds a promising region, only its immediate neighbours will initially be drawn to that area. No other particles in the swarm will know about that region unless their own immediate neighbours move there. In the star topology, if one particle finds a promising region, all other particles of the swarm are immediately drawn to it. As a result, the swarm generally converges more quickly but sometimes is trapped in a local optimal point in the space. Generally, the circular ring topology propagates information slowly and the star topology propagates information quickly [44, 95]; the circular ring topology can explore broader spaces than the star topology.

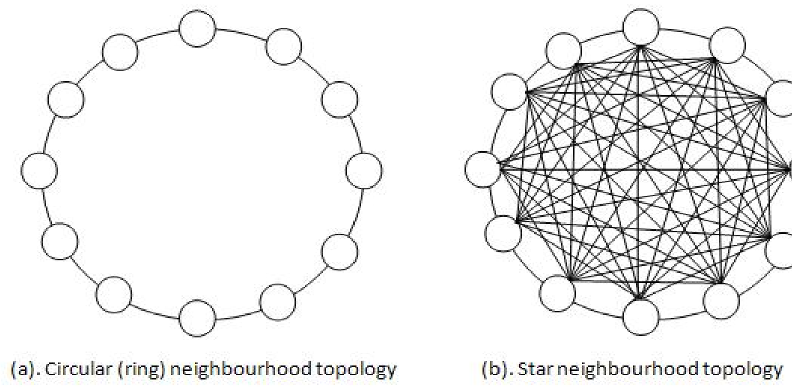

(a). Circular (ring) neighbourhood topology
(b). Star neighbourhood topology

Figure 3.2 Two most common neighbourhood topologies for PSO

The circle (ring) topology is adopted in our implementation. In order to improve the computation efficiency, a performance-based dynamic circular ring topology is proposed below.

Performance-based dynamic circular ring neighbourhood topology

Hypothesis: Allocating particles that have similar performance in a neighbourhood is more efficient than allocating particles randomly in a neighbourhood in PSO.

This hypothesis is inspired by the human social networks and the theory of sociometry (the study and measurement of interpersonal relationships in a group of people) [97]. In human social activities, people with the same or similar degrees of interests or performance generally communicate more efficiently. For example, in a university's scenario, an academic's career includes a few steps including Level A (Associate Lecturer), Level B (Lecturer), Level C (Senior Lecturer), Level D (Associate Professor) and Level E (Professor). Level A academics adopt Level B's experiences to promote to Level B; Level B academics learn experiences from Level C to move onto

Level C. And so forth. Each individual learns from his/her closer (in terms of academic performance) neighbours.

Performance rules: Based on the above hypothesis, the particles with a similar level of performance should be allocated in a neighbourhood. The next question is how to evaluate the particles' performance. A modified constraint dominance concept is adopted as performance rules for our specific multi-objective constraint-handling optimization problems, as follows:

- If two particles are both feasible, that is, $\Phi \le \varepsilon$, the one with the lower f wins.
- If the above is not true, the particle with lower Φ wins (this covers the situation where one particle is feasible and the other is not).

The performance rules will be used for sorting particles in the dynamic neighbourhood topology.

Dynamic neighbourhood topology: Initially, each individual is connected to its k immediate neighbours only (same as the circular ring topology). After each iteration, all particles in the swarm are sorted according to the performance rules. Once sorted, the particles in the swarm are reindexed. Although a particle i still has its k immediate neighbours connected, these k particles may not be those particles in the last iteration. They are particles close to each other in terms of their performance. For our multi-objective constraint-handling model (Equation (3.1)), this means that a particle will have its geographically closer particles (in solution space) as its neighbours.

Figure 3.3 illustrates the idea of the dynamic neighbourhood topology. Originally, the five particles (A, 1), (B, 2), (C, 3), (D, 4), and (E, 5) (where the first item in the brackets is the particle object, the second item in the brackets is the index) form a circular ring topology. Supposing the neighbourhood size is 2, the five groups of neighbourhood are: ABC, BCD, CDE, DEA and EAB. After sorting, the new groups of neighbourhood become: ABE, BEC, ECD, CDA, and DAB. The sorting operation has actually changed the particles' indexes.

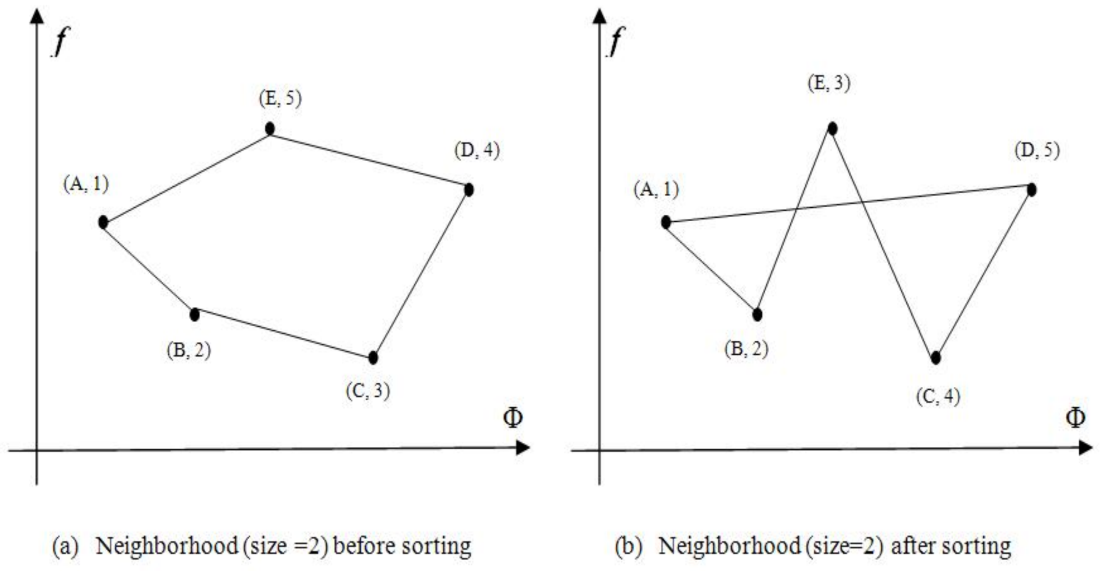

f
(E, 5)
(D, 4)
(A, 1)
(B, 2)
(C, 3)
Φ
(a) Neighborhood (size =2) before sorting
f
(E, 3)
(D, 5)
(A, 1)
(B, 2)
(C, 4)
Φ
(b) Neighborhood (size=2) after sorting

Figure 3.3 Illustration of the dynamic neighbourhood topology

3.3.4 The Proposed Algorithm

Figure 3.4 and Figure 3.5 illustrate the proposed algorithm which integrates the multi-objective constraint-handling method with PSO algorithm. Figure 3.4 is the data flow diagram and Figure 3.5 is the corresponding pseudo code. Compared with the original PSO, the proposed algorithm has the following features:

- Whenever calculating fitness, both objectives f and Φ need to be evaluated;
- A particle's best neighbour particle (*lBest*) is determined by the following steps:
  - Find all the non-dominated particles in the neighbourhood;
  - If there is only one non-dominated particle in the neighbourhood, select it as *lBest*; otherwise, select one with the lowest Φ as *lBest* (follows selection rules).
- A particle's personal best, *pBest,* is determined by the selection rules, that is, if a particle's new location is better than its best previous location, the *pBest* is updated.
- A minor perturbation with the probability of p is introduced after calculating the next particle position. The aim for using perturbation is to keep population diversity and to prevent premature convergence.
- After each iteration, apply the sorting algorithm to reindex particles according to their performance (follows performance rules). The particles with similar performance will be allocated in the neighbourhood.

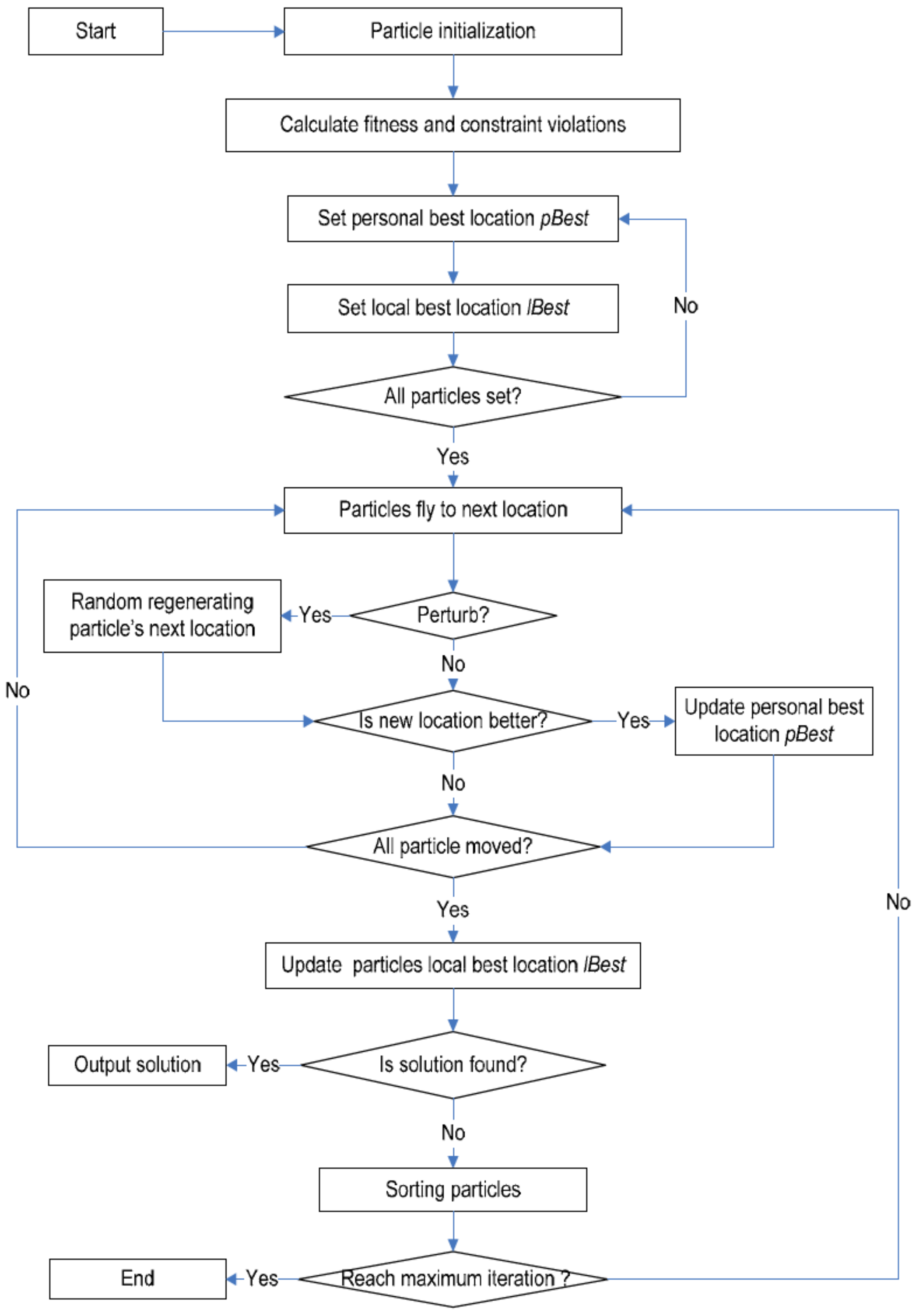
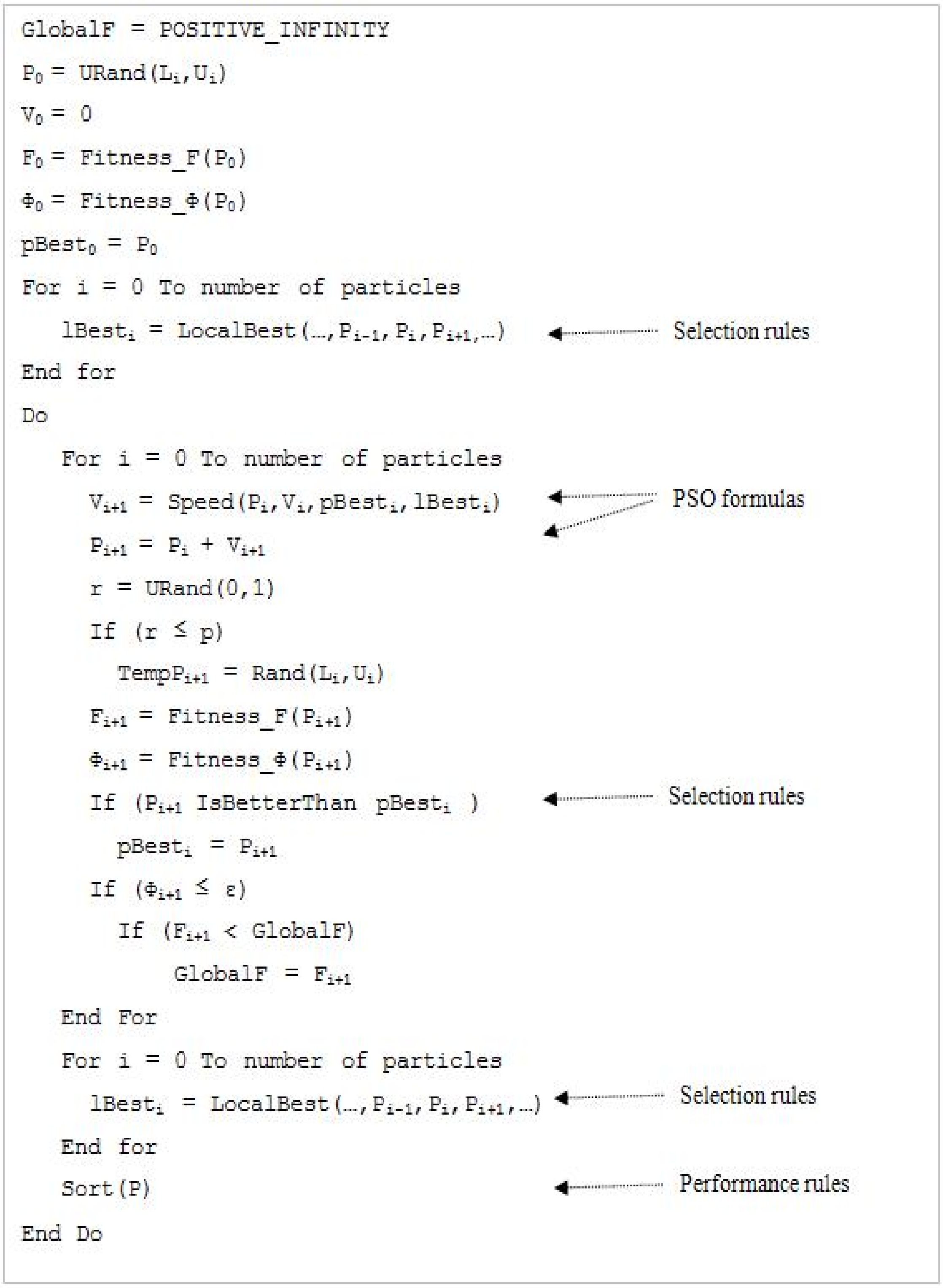

Start
Particle initialization
Calculate fitness and constraint violations
Set personal best location *pBest*
Set local best location *lBest*
All particles set?
No
Yes
Particles fly to next location
Random regenerating particle's next location
Yes
Perturb?
No
Is new location better?
Yes
Update personal best location *pBest*
No
All particle moved?
No
Yes
Update particles local best location *lBest*
Output solution
Yes
Is solution found?
No
Sorting particles
End
Yes
Reach maximum iteration ?
No

Figure 3.4 Data flow diagram of the proposed multi-objective constraint-handling method with PSO algorithm



```
GlobalF = POSITIVE_INFINITY
$P_0$ = URand($L_i$,$U_i$)
$V_0$ = 0
$F_0$ = Fitness_F($P_0$)
$\Phi_0$ = Fitness_$\Phi$($P_0$)
$pBest_0$ = $P_0$
For i = 0 To number of particles
   $lBest_i$ = LocalBest(…,$P_{i-1}$,$P_i$,$P_{i+1}$,…)      Selection rules
End for
Do
   For i = 0 To number of particles
     $V_{i+1}$ = Speed($P_i$,$V_i$,$pBest_i$,$lBest_i$)      PSO formulas
     $P_{i+1}$ = $P_i$ + $V_{i+1}$
     r = URand(0,1)
     If (r $\le$ p)
       Temp$P_{i+1}$ = Rand($L_i$,$U_i$)
     $F_{i+1}$ = Fitness_F($P_{i+1}$)
     $\Phi_{i+1}$ = Fitness_$\Phi$($P_{i+1}$)
     If ($P_{i+1}$ IsBetterThan $pBest_i$ )      Selection rules
       $pBest_i$ = $P_{i+1}$
     If ($\Phi_{i+1}$ $\le$ $\varepsilon$)
       If ($F_{i+1}$ < GlobalF)
           GlobalF = $F_{i+1}$
   End For
   For i = 0 To number of particles
     $lBest_i$ = LocalBest(…,$P_{i-1}$,$P_i$,$P_{i+1}$,…)      Selection rules
   End for
   Sort(P)      Performance rules
End Do
```

Figure 3.5 Pseudo code of the proposed multi-objective constraint-handling method with PSO algorithm

3.4 NUMERICAL OPTIMIZATION SIMULATION

There are thirteen well-known numerical benchmark functions named G1, G2, …, G13. These functions can be found from many papers such as in references [22, 23, 37]. They are also included in the Appendix I of this thesis. These functions have been popularly used for optimization algorithm testing for years because they represent a wide variety of optimization problems including linear and nonlinear in objective and constraint functions, equality and inequality constraints, large and small dimensions, large and small search spaces, large and small feasible regions,

3.4.1 Test Functions

The numerical test functions and their features are listed in Table 3-1 (taken from [70]), where *n* is the number of decision variables (that is, dimensions), LI is the number of Linear-Inequality constraints, NI is the number of Nonlinear-Inequality constraints, LE is the number of Linear-Equality constraints and NE is the number of Nonlinear-Equality constraints.

The relative size of feasible space ρ suggested by Michalewicz and Schoenauer [49] in Table 3-1 is the ratio between the feasible and the total search (feasible and infeasible) region of each of these problems. It is calculated by the following expression: $\rho = |F| / |S|$, where $|F|$ is the number of feasible solutions and $|S|$ is the total number of solutions randomly generated. In Table 3-1, S = 1,000,000 random solutions [70].

Table 3-1 The 13 constrained nonlinear optimization test functions

TF	n	Type	ρ	LI	NI	LE	NE
G1	13	Quadratic	0.0003%	9	0	0	0
G2	20	Nonlinear	99.9973%	1	1	0	0
G3	10	Polynomial	0.0026%	0	0	0	1
G4	5	Quadratic	27.0079%	0	6	0	0
G5	4	Cubic	0.0000%	2	0	0	3
G6	2	Cubic	0.0057%	0	2	0	0
G7	10	Quadratic	0.0000%	3	5	0	0
G8	2	Nonlinear	0.8581%	0	2	0	0
G9	7	Polynomial	0.5199%	0	4	0	0
G10	8	Linear	0.0020%	3	3	0	0
G11	2	Quadratic	0.0973%	0	0	0	1
G12	3	Quadratic	4.7697%	0	729	0	0
G13	5	Nonlinear	0.0000%	0	0	1	2

3.4.2 Parameters

For each case, 30 independent runs have been performed. PSO parameters are: $c_1 = c_2 = 2.0$; $\chi = 0.63$; $V_{\max} = 0.5 \bullet (U - L)$; number of particles is 100; the maximum iteration $i_{\max}$ is set to 10,000; the inertia weight $w = 0.25 \bullet (1 - i / i_{\max})$; the perturbation probability $p = 0.1$; a potential solution is considered feasible when its $\Phi < \varepsilon = 1.0E-05$; the tolerance allowed for an equality constraint is $\delta = 1.0E-03$.

3.4.3 Results and Discussion

The experiment results are presented in Table 3-2 and Figure 3.6 to Figure 3.18. Table 3-2 consists of the best results, the mean results and the standard deviations found in 30 independent runs for each test function. Figure 3.6 to Figure 3.18 illustrate the algorithm convergence from the best runs for each function.

According to Table 3-2, the proposed multi-objective constraint-handling PSO algorithm can find solutions to most benchmark functions. From the quality search point of view (the best results found), the results match the well-known solution well (or even better) in eight out of thirteen functions, that is, G3, G5, G6, G8, G9, G11,G12 and G13. The results are close to the well-known solutions in function G4, G7 and G10. The best results found for function G1 and G2 are not very satisfactory. Regarding to algorithm consistency, Table 3-2 demonstrates most standard deviations are small in relation to their magnitudes.

It is noticed the algorithm works well for function G5, G10 and G13. These three functions are considered very complex problems because G5 has both inequality and equality constraints and these three functions have very small feasible regions (refer to Table 3-1).

Figure 3.6 to Figure 3.18 demonstrate the algorithm can converge very fast (in less than 100 iterations) in seven functions, that is, G3, G6, G7, G8, G9, G11 and G12. The algorithm converges reasonably fast (between 2000 - 4000 iterations) in four functions, that is, G4, G5, G10 and G13. The algorithm converges slowly in two functions, that is, G1 and G2 (more than 10000 iterations).

The tolerance allowed for equality constraints δ and for feasibility criterion ε has great impact on the algorithm performance. A larger δ and a larger ε will make the search easier. In our simulation, we selected a reasonably small $\delta = 1.0E-03$ and fairly small $\varepsilon = 1.0E-05$ to cover a wide range of test functions.

Table 3-2 Best, mean and standard deviation results from 30 independent runs

TF	*Optimal*	*Best Result*	*Mean Result*	*Std. Dev.*
G1	-15.00000	-14.0711764	-12.16478164	9.24E-01
G2	-0.803619	-0.61760436	-0.46054886	6.07E-02
G3	-1.000000	-1.005059969	-1.004246441	1.83E-03
G4	-30665.539	-30663.17206	-30658.01627	2.95E-00
G5	5126.4981	5126.484102	5127.57779	3.71E-00
G6	-6961.81388	-6961.826142	-6961.826052	6.33E-05
G7	24.306209	24.30753104	24.60382281	7.59E-01
G8	-0.095825	-0.095825041	-0.095825041	4.16E-17
G9	680.630057	680.630046	680.6300462	1.12E-07
G10	7049.3307	7049.226787	7176.001082	141.7E-00
G11	0.750000	0.748990016	0.748990001	8.45E-10
G12	-1.000000	-1.000000000	-1.000000000	0.00E-00
G13	0.053950	0.053962476	0.133453421	8.89E-02

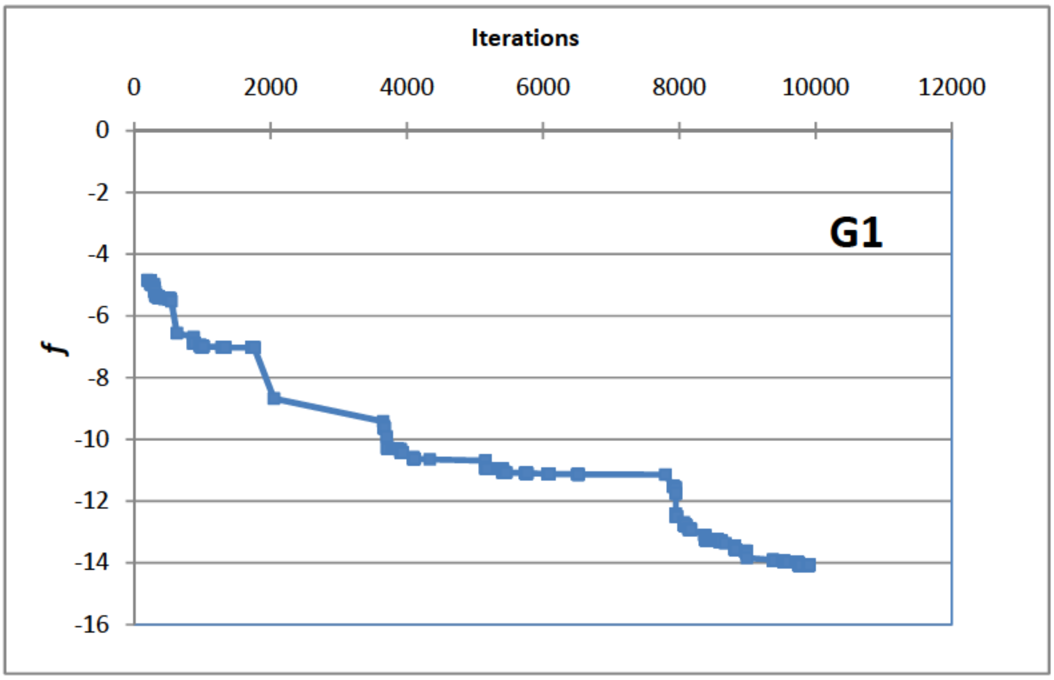


Figure 3.6 Convergence graph of best solution for function G1

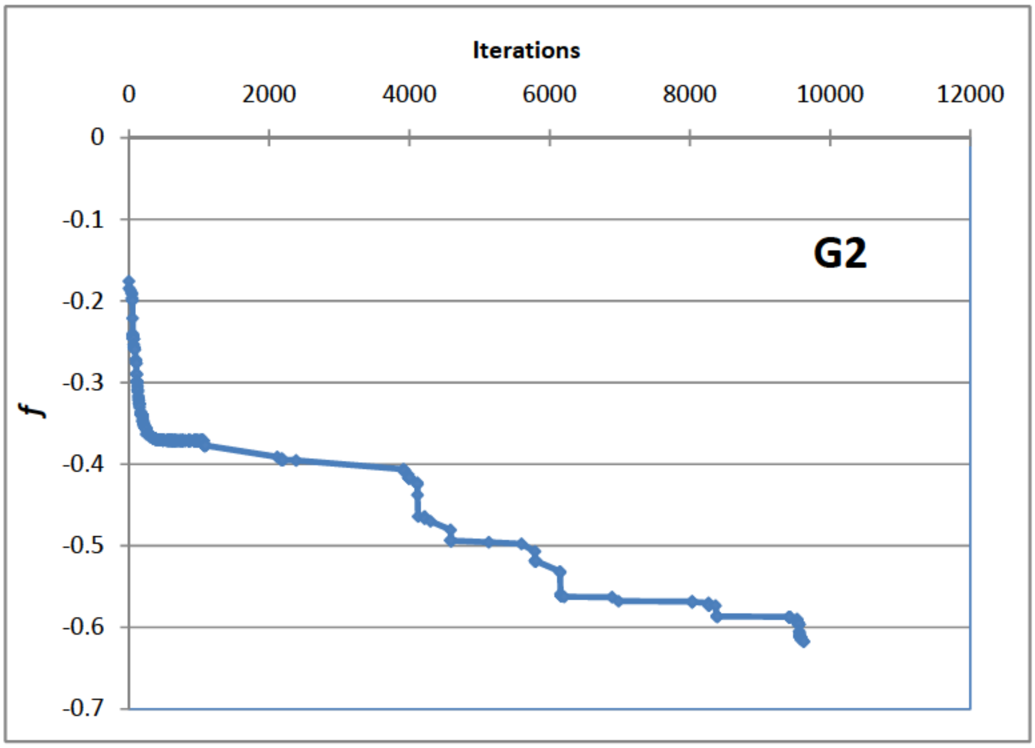


Figure 3.7 Convergence graph of best solution for function G2

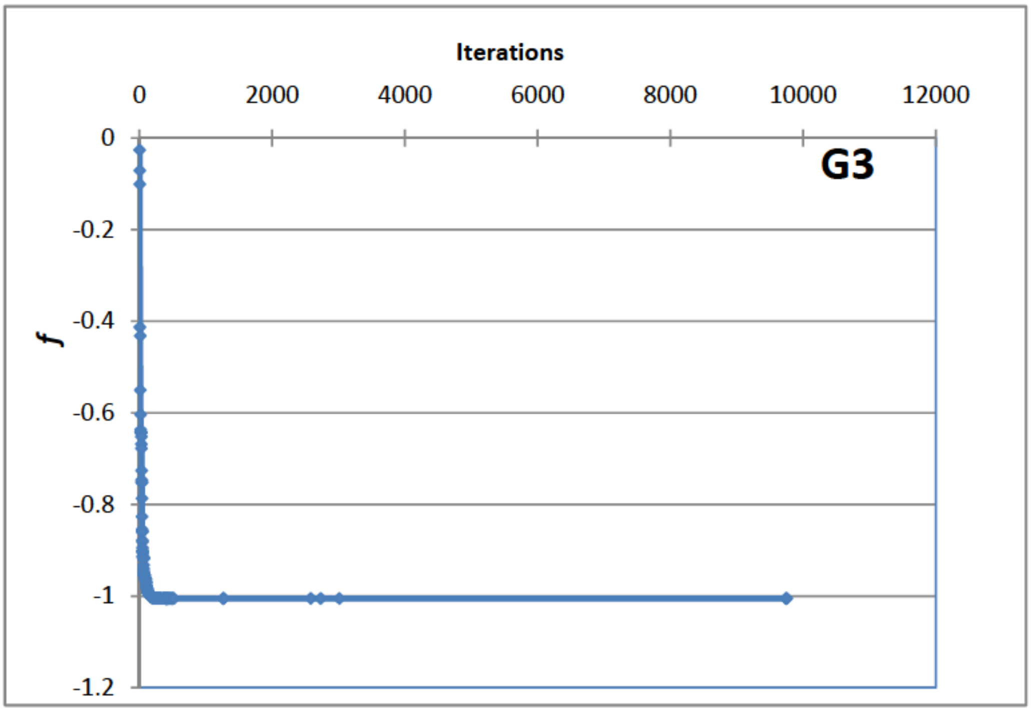


Figure 3.8 Convergence graph of best solution for function G3

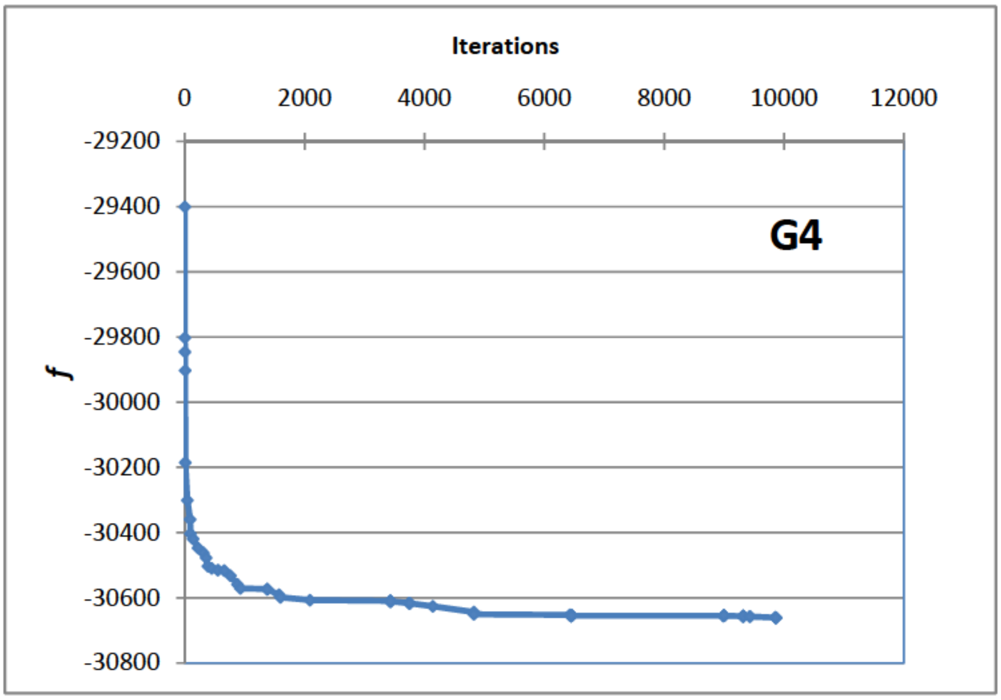


Figure 3.9 Convergence graph of best solution for function G4

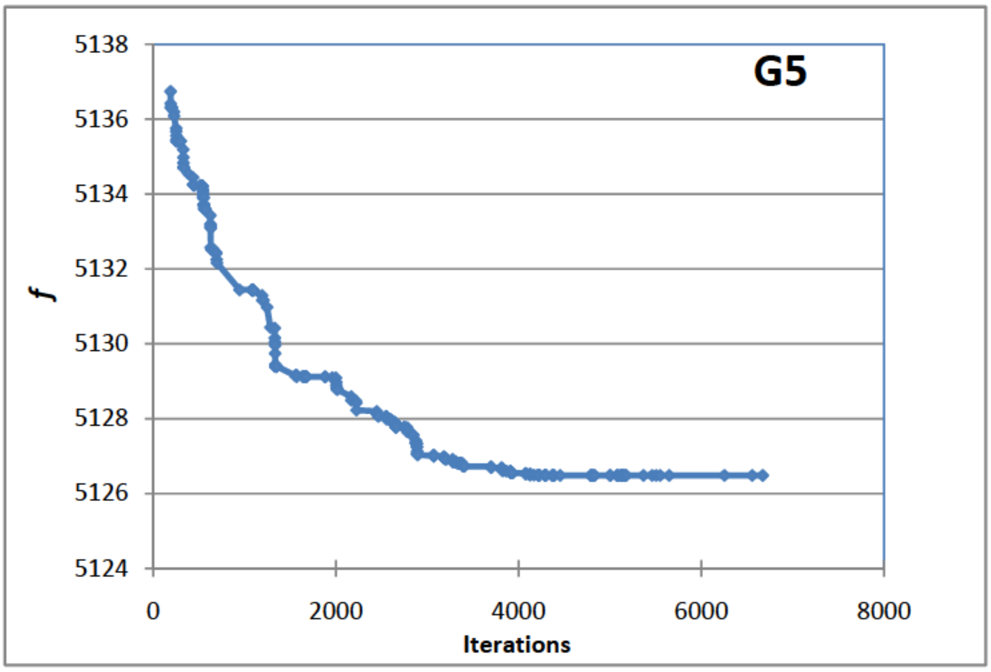


Figure 3.10 Convergence graph of best solution for function G5

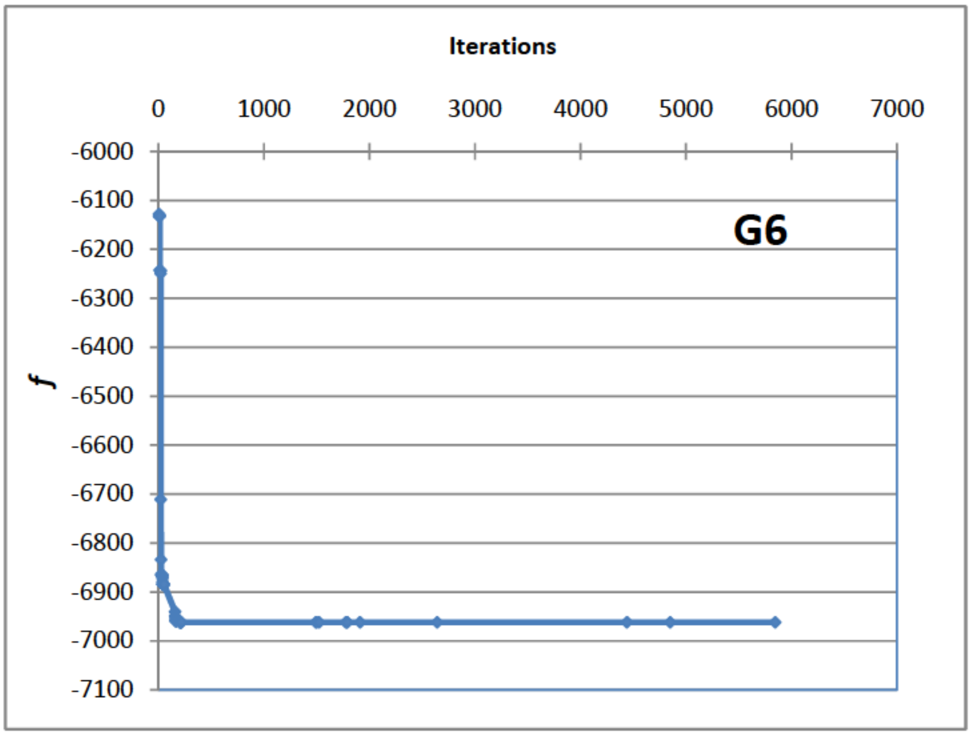


Figure 3.11 Convergence graph of best solution for function G6

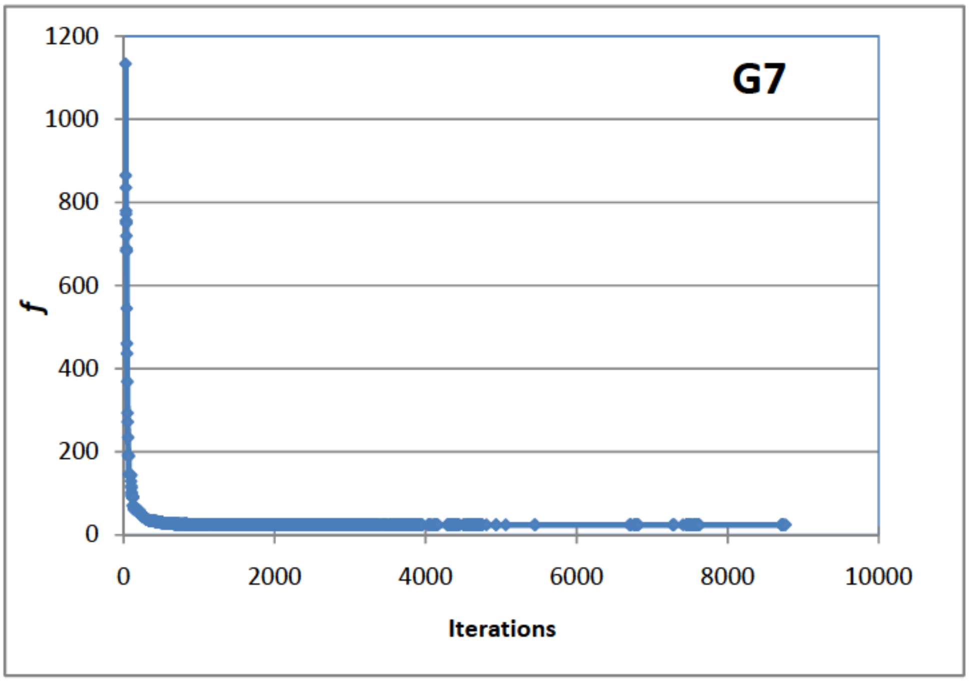


Figure 3.12 Convergence graph of best solution for function G7

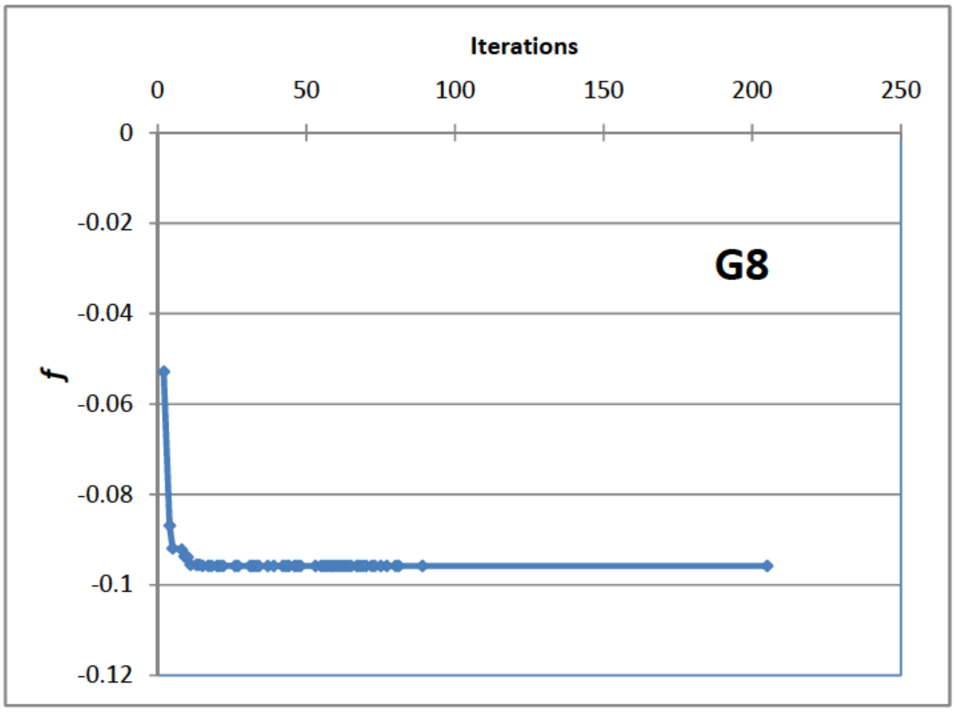


Figure 3.13 Convergence graph of best solution for function G8

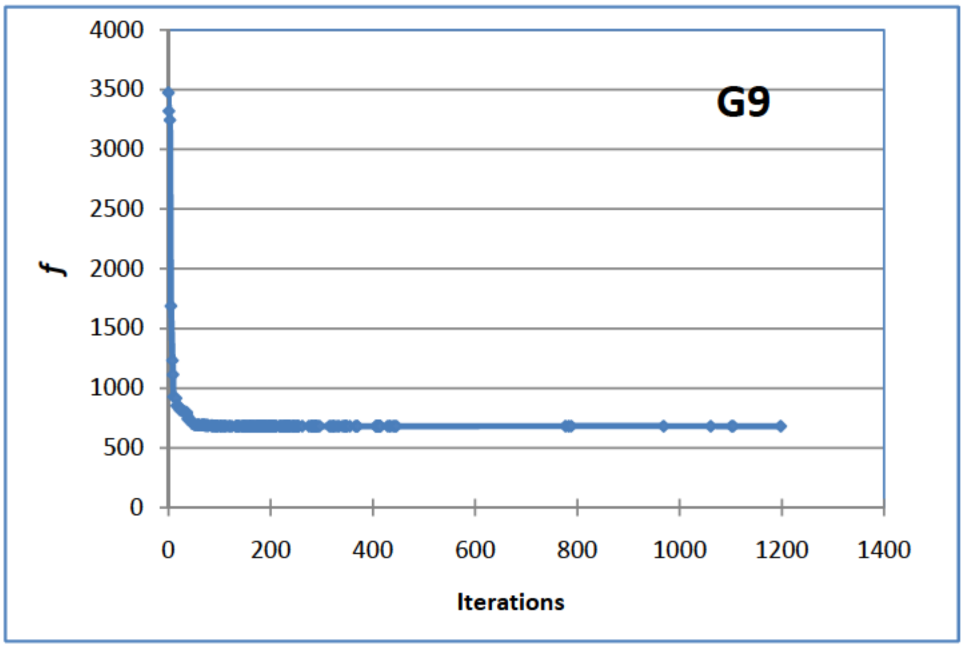


Figure 3.14 Convergence graph of best solution for function G9

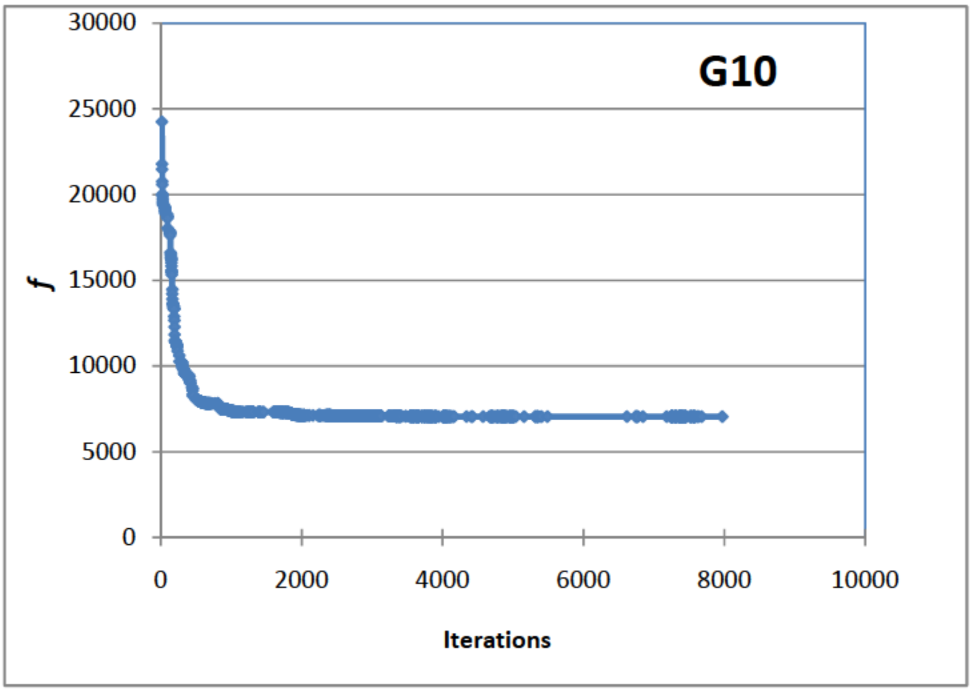


Figure 3.15 Convergence graph of best solution for function G10

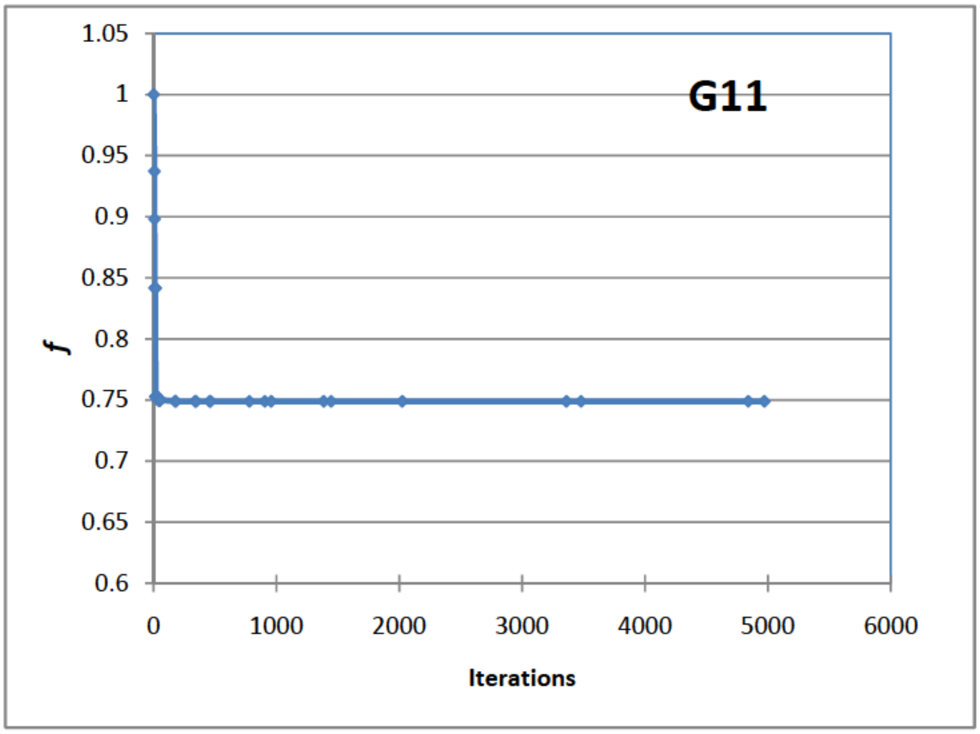


Figure 3.16 Convergence graph of best solution for function G11

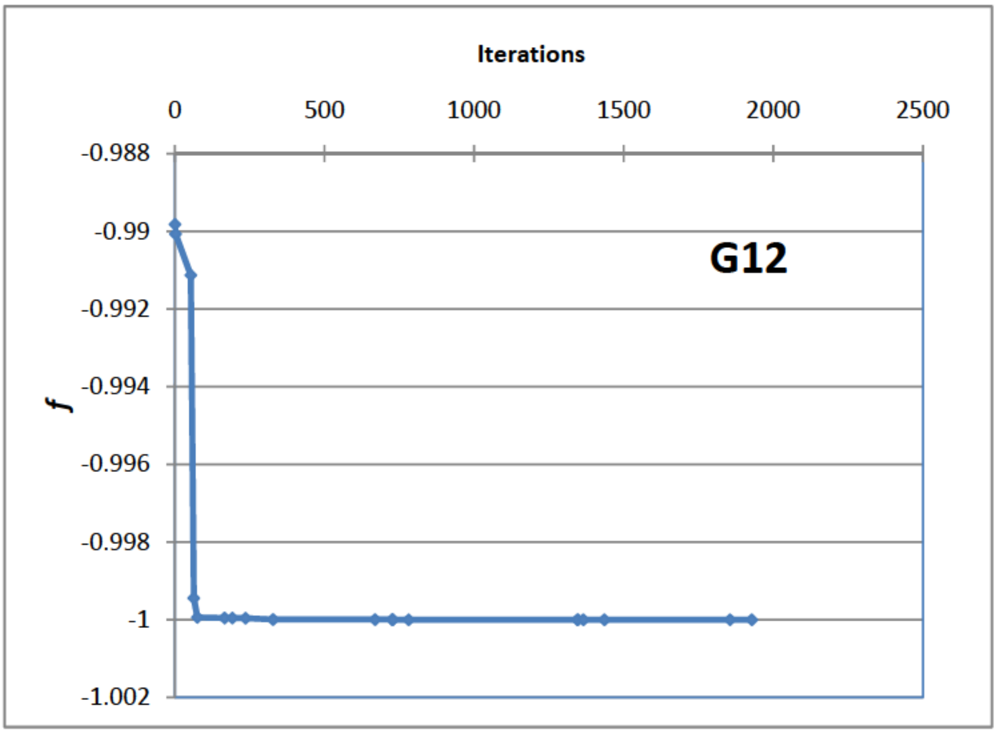


Figure 3.17 Convergence graph of best solution for function G12

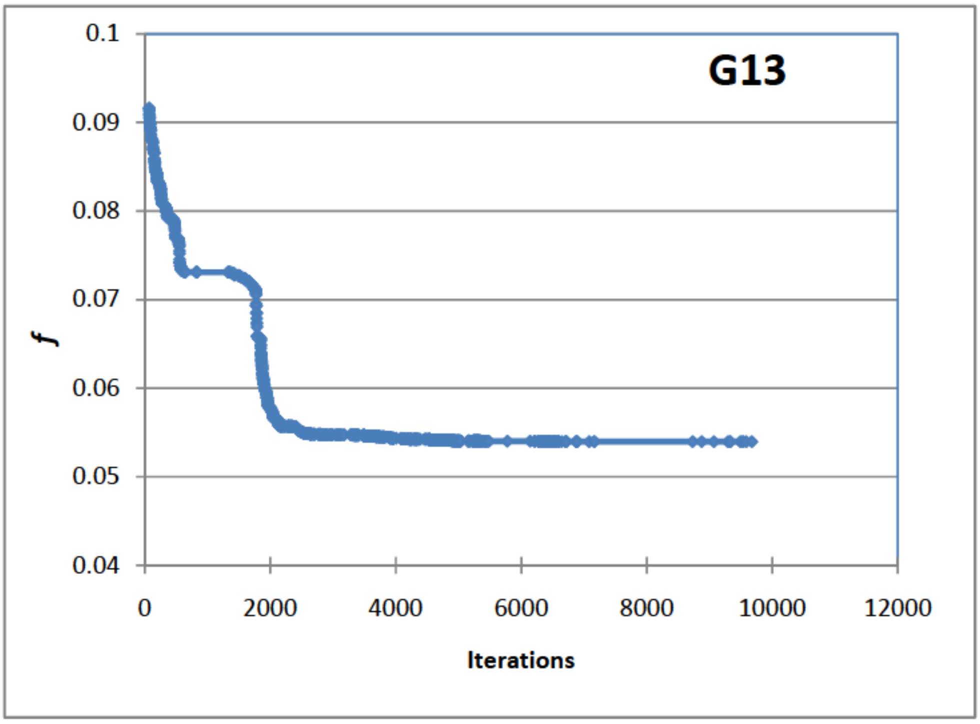


Figure 3.18 Convergence graph of best solution for function G13

3.5 COMPARATIVE STUDY

3.5.1 Quality and Consistency Comparison

Another similar study found at the same time is by Flores-Mendoza and Mezura-Montes [98] who compared their approach with the state-of-the-art algorithms and claimed their approach is effective. Thus, a comparative study is performed to compare our approach with [98]. Table 3-3 lists the comparison results. For easy identification, the results from the approach presented in this chapter is referred to "This thesis", and results from [98] is referred to "Reference".

Table 3-3 demonstrates that the proposed approach is able to provide similar search results to or better search results than those in [98] in ten functions (G3, G5, G6, G7, G8, G9, G10, G11, G12 and G13) (refer to Best and Mean columns). The proposed approach also achieved better consistent results in above ten functions (refer to the Std. Dev. column). In particular, the proposed approach outperformed in complex problems G5, G10 and G13. However, our approach is unable to perform better in function G1 and G2. Since both G1 and G2 are high dimensional problems, it seems that the proposed approach needs to be improved for solving high dimensional optimization problems. This may be explained by the "No free lunch" theorem [99].

In our simulation experiments, 100 particles with a maximum 10000 iterations are used for all functions. Although many functions can converge in less than 2000 iterations (refer to convergence graphs Figure 3.6 to Figure 3.18), the maximum number of iterations is used for trying to cover those functions with slow convergence rate like G1, G2 and G5. It looks that our approach used a larger number of function evaluations (1,000,000) than the compared reference (160,000). The computation cost is not a significant problem in our multithreaded object-oriented implementation (details in Chapter 4).

Table 3-3 Best, mean and standard deviation results comparison from 30 independent runs

TF	Optimal	Case	Best	Mean	Std. Dev.
G1	-15.00000	This thesis	-14.0711764	-12.16478164	9.24E-01
		Reference	**-15**	**-15**	**0.000**
G2	-0.803619	This thesis	-0.61760436	-0.46054886	6.07E-02
		Reference	**-0.802629**	**-0.713879**	**0.046231**
G3	-1.000000	This thesis	**-1.005059969**	**-1.004246441**	**1.83E-03**
		Reference	-0.641	-0.154	0.170
G4	-30665.539	This thesis	-30663.17206	-30658.01627	2.95E-00
		Reference	**-30665.539**	**-30665.539**	**7.4E-12**
G5	5126.4981	This thesis	**5126.484102**	**5127.57779**	**3.71E-00**
		Reference	5126.498	5135.521	12.385
G6	-6961.81388	This thesis	**-6961.826142**	**-6961.826052**	**6.33E-05**
		Reference	-6961.814	-6961.814	2.810E-05
G7	24.306209	This thesis	**24.30753104**	**24.60382281**	**7.59E-01**
		Reference	24.366	24.691	0.227
G8	-0.095825	This thesis	**-0.095825041**	**-0.095825041**	**4.16E-17**
		Reference	-0.095825	-0.095825	4.234E-17
G9	680.630057	This thesis	**680.630046**	**680.6300462**	**1.12E-07**
		Reference	680.638	680.674	0.030
G10	7049.3307	This thesis	**7049.212219**	**7176.001082**	**141.694**
		Reference	7053.963	7306.466	222.824
G11	0.750000	This thesis	**0.748990016**	**0.748990001**	**8.45E-10**
		Reference	0.749	0.753	6.537E-03
G12	-1.000000	This thesis	**-1.000000000**	**-1.000000000**	**0.00E-00**
		Reference	-1.000	-1.000	0.000
G13	0.053950	This thesis	**0.053962476**	**0.133453421**	**8.89E-02**
		Reference	0.066845	0.430408	0.239807

3.5.2 Dynamic Neighbourhood and Static Neighbourhood Comparison

In order to see the difference between the dynamic neighbourhood topology and the static neighbourhood topology, two experiments have been designed as follows:

Target-based Experiment (TBE): For each optimization problem, set a target value for its fitness function, and then calculate the number of iterations needed to reach the target for both dynamic neighbourhood and static neighbourhood topologies. To implement it, a criterion C can be defined to indicate the minimum distance from a solution f to the predefined target T. For example, the numerical function G5 has a well-known solution (target) $T = 5126.4981$, if the criterion $C = 0.001$ is selected, the iteration stops once a solution f meets the criterion $|f - T| \leq C$. In case there is no solution meeting the criterion found, a maximum number of iterations is needed.

Iteration-based Experiment (IBE): For each optimization problem, set a maximum number of iterations, and then calculate the best results found at the end of maximum iterations.

Because the objective for these two experiments is to evaluate the performance between the dynamic neighbourhood topology and the static neighbourhood topology, the algorithm for both topologies are the same. Any function out of the thirteen benchmark functions can be selected for this purpose. The experiment results based on the TBE and IBE for the five numerical functions - G5, G6, G7, G9 and G10 are listed below. The results for other functions are not included in this chapter since the five functions should be sufficient to demonstrate the influence of the idea.

Table 3-4 consists of the lowest, average and highest number of iterations needed from 30 independent runs based on TBE. The criterion value *C* = 0.001 for G5, G6 and G9; *C* = 0.01 for G7 and *C* = 0.1 for G10. The reason is that both G7 and G10 have a target minimum which is difficult to achieve. If a very small *C* is selected, it is possible that the criterion can never be reached through all iterations. According to Table 3-4, the average number of iterations needed (4379 for G5, 886 for G6, 9310 for G7, 303 for G9 and 9460 for G10) in the dynamic neighbourhood topology are less than those from the static neighbourhood (6952 for G5, 8503 for G6, 9914 for G7, 8968 for G9 and 9889 for G10). Particularly, the algorithm in the dynamic neighbourhood topology converges faster than in the static neighbourhood topology in function G6 and G9 where the maximum number of iterations (1827 for G6 and 490 for G9) in the dynamic neighbourhood topology are less than the minimum numbers of iterations (5782 for G6 and 3731 for G9) in the static neighbourhood topology.

Table 3-5 includes the best, the average and the worst results found from 30 independent runs based on IBE for the test function G5, G6, G7, G9 and G10. The maximum iteration is set to 4000. From the Table 3-5, the best, average and the worst results found from the dynamic neighbourhood topology are all better than the results achieved from the static topology.

Figure 3.19 to Figure 3.23 illustrate the convergence graphs of the average results found from these two neighbourhood topologies based on the TBE. As indicated from these results, **the proposed dynamic neighbourhood topology can find solutions faster than the static neighbourhood topology. The comparative experiment results support the hypothesis proposed in Section 3.3.3.**

Table 3-4 Lowest, average and highest iteration needed for dynamic and static neighbourhood topology from 30 independent runs

TF	Optimal (Target)	Criteria	Neighbourhood Type	Lowest	Average	Highest
G5	5126.4981	0.001	Dynamic	**105**	**4379**	**9990**
			Static	352	6952	9999
G6	-6961.81388	0.001	Dynamic	**113**	**886**	1827
			Static	5782	8503	9626
G7	24.306209	0.010	Dynamic	**3609**	**9310**	**9999**
			Static	8344	9914	9999
G9	680.630057	0.001	Dynamic	**200**	**303**	**490**
			Static	3731	8968	9999
G10	7049.3307	0.100	Dynamic	**5905**	**9460**	**9999**
			Static	9227	9889	9999

Table 3-5 Best, average and worst results found for dynamic and static neighbourhood topology from 30 independent runs (maximum iteration = 4000)

TF	Optimal	Neighbourhood Type	Best	Average	Worst
G5	5126.4981	Dynamic	**5126.484102**	**5132.648157**	**5178.538574**
		Static	5126.484123	5133.324432	5201.465006
G6	-6961.81388	Dynamic	**-6961.826137**	**-6961.826003**	**-6961.825799**
		Static	-6961.824322	-6960.091727	-6948.589191
G7	24.3062090	Dynamic	**24.33202325**	**25.59205796**	**27.64933147**
		Static	25.8460636	32.19146368	38.767848
G9	680.630057	Dynamic	**680.6300462**	**680.6300466**	**680.6300474**
		Static	680.6621935	681.3825028	683.5746460
G10	7049.3307	Dynamic	**7051.067080**	**7285.291824**	**8109.872342**
		Static	7105.547735	7665.149708	8945.864704

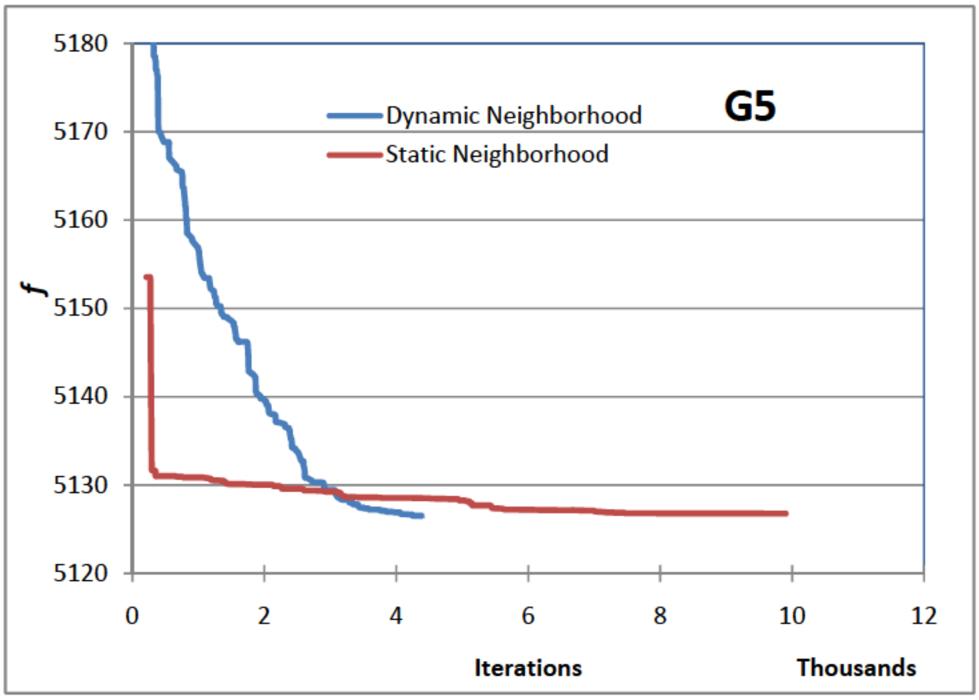


Figure 3.19 Target-based experiment convergence graph of average solution in different neighborhood topoloies for G5

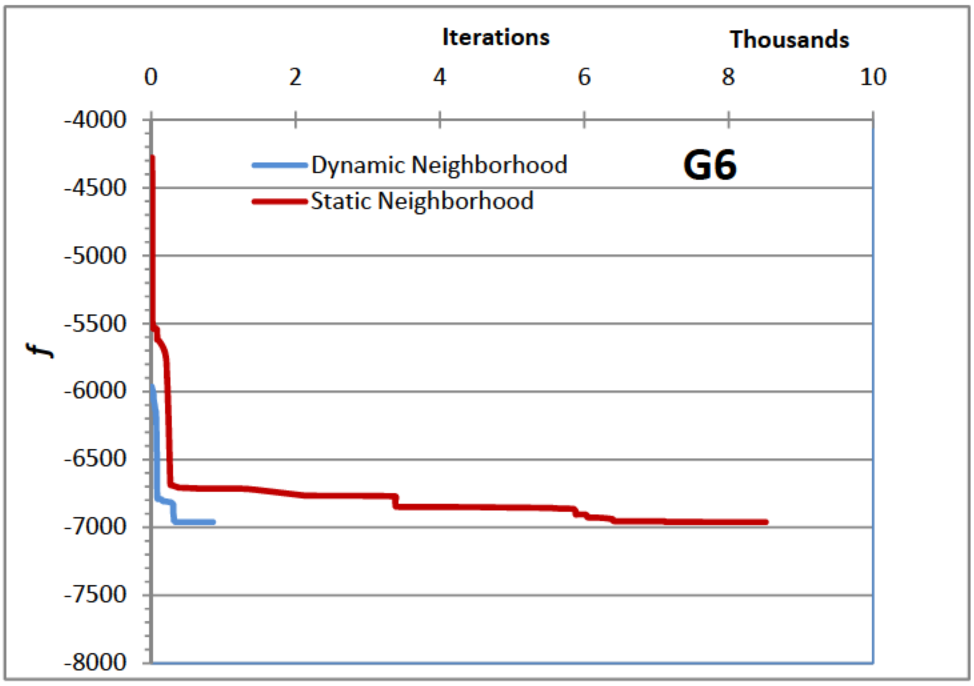


Figure 3.20 Target-based experiment convergence graph of average solution in different neighborhood topoloies for G6

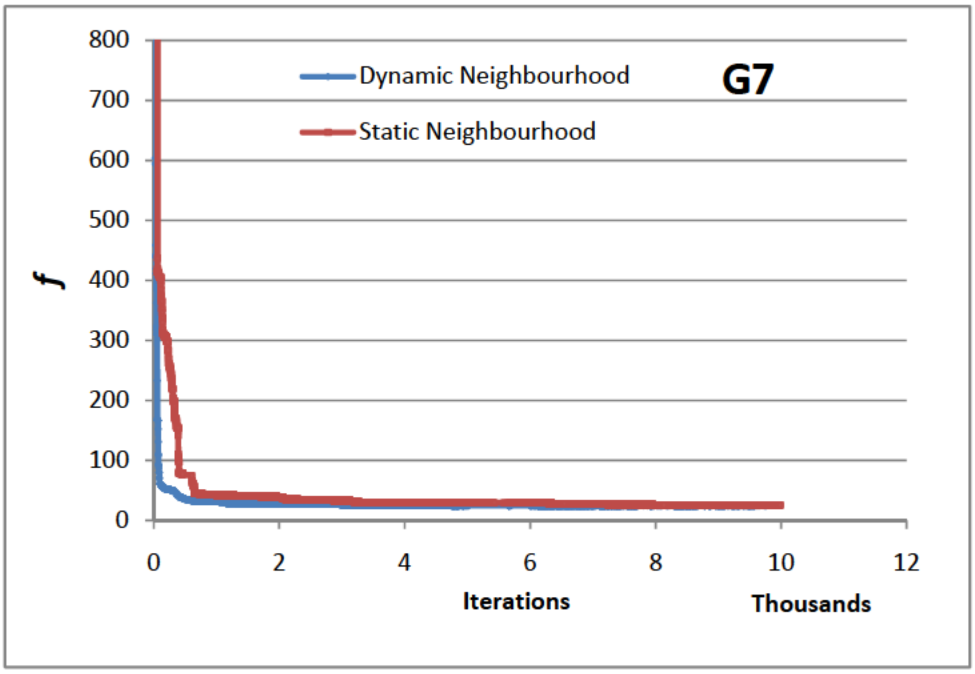


Figure 3.21 Target-based experiment convergence graph of average solution in different neighborhood topoloies for G7

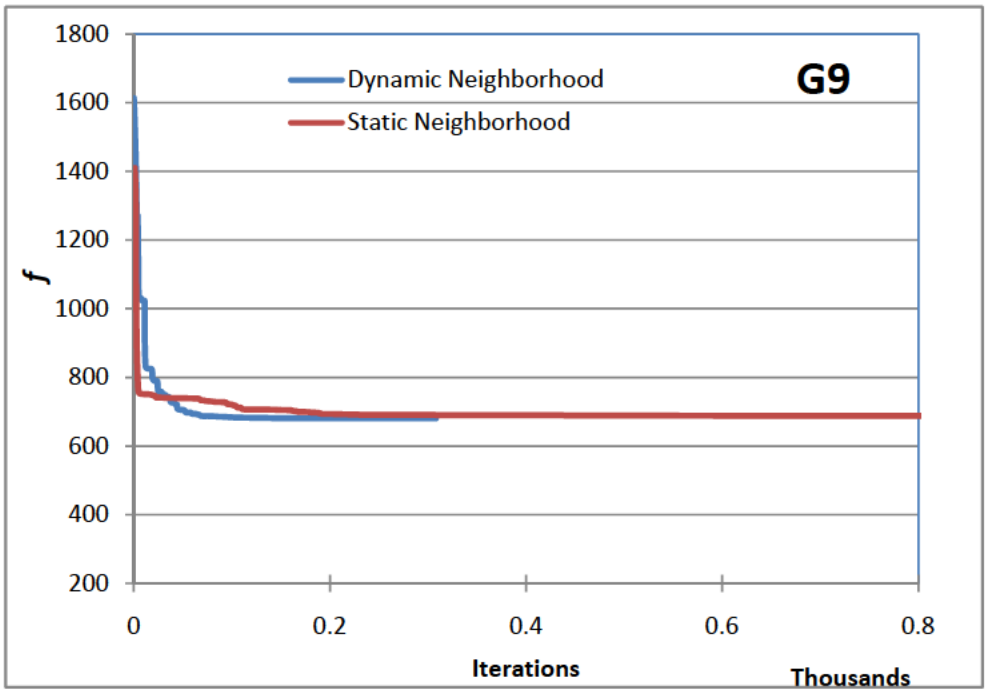


Figure 3.22 Target-based experiment convergence graph of average solutions in different neighborhood topoloies for G9

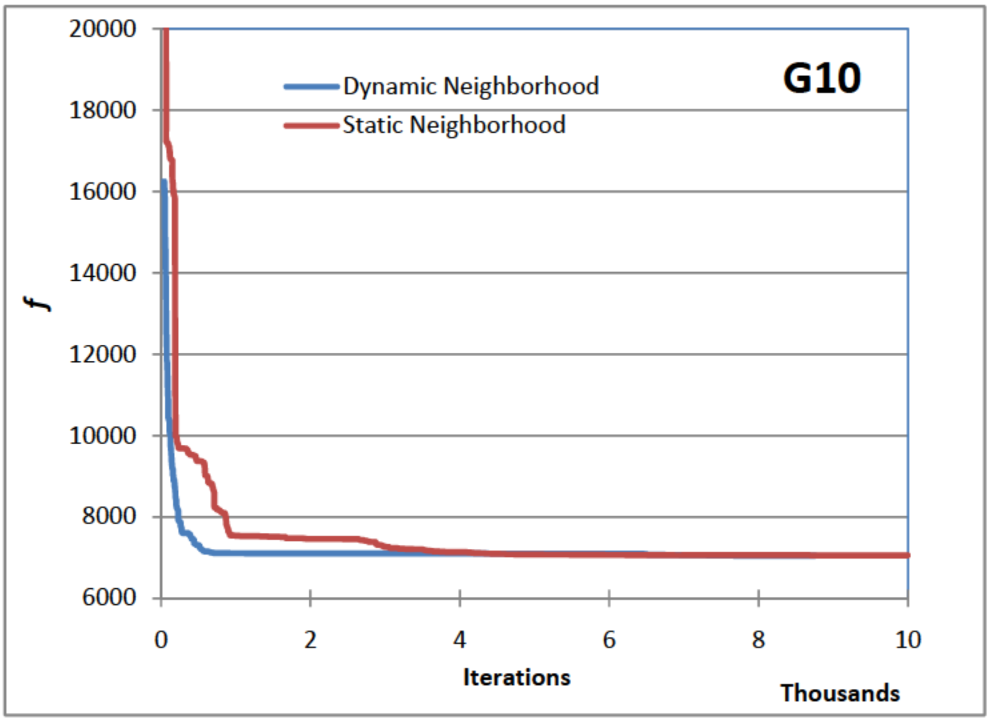


Figure 3.23 Target-based experiment convergence graph of average solutions in different neighborhood topoloies for G10

It appears that the algorithm with the performance-based neighbourhood topology converges faster than the algorithm with the static neighbourhood topology. The reason is as follows. In the original circular ring topology, a particle always has its k neighbours connected during whole iterations. If one particle p_1 discovers a good solution, it informs its immediate neighbour p_2 and p_3. If p_2 and p_3 are far from p_1, due to the maximum velocity applied in the PSO algorithm, it will take time for p_2 and p_3 to fly toward p_1. Furthermore, p_2 and p_3 are constrained by their other neighbours, the total force applied on p_2 and p$_3$ is in an unordered direction which slows down the process. In the dynamic neighbourhood topology, particles are sorted according to their performance. They are geographically closer in solution space under our performance rules. A particle is always attracted by those at the front. The total force applied to each particle is in the same (or similar) direction. This will speed up the search process.

This can be explained in the university scenario. A professor discovers a good idea on how to get an ARC (Australia Research Council) discovery grant. If the professor's immediate neighbours are on a lower level, say lecturers, then the professor may instruct his/her neighbours to apply for the ARC grant by using his/her idea. Since the lectures' capability (velocity in PSO) is limited and they are constrained by other commitments such as teaching, the possibility of success for the lecturers to get the ARC grant is not very high. On the other hand, if the professor's immediate neighbours are other professors or associate professors with the same goal focusing on research, the communication between the professor and his/her neighbours would be more efficient and lead to a greater likelihood of success.

3.6 SUMMARY

A multi-objective constraint-handling method with a dynamic neighbourhood PSO algorithm has been proposed for tackling single objective constrained optimization problems. By adopting a multi-objective constraint-handling method, a single objective constrained optimization problem is converted to a bi-objective unconstrained optimization problem. Then the concept of Pareto dominance from multi-objective optimization techniques is used. An adaptive inertia weight factor and a minor perturbation are adopted to improve the convergence and the diversity. The simulation results for the well-known benchmark functions have demonstrated the proposed approach is effective and efficient in quality search and consistency.

Compared with the recent research results, the proposed approach is able to provide similar good or better quality and consistent results in ten out of thirteen functions. In particular, the proposed approach outperforms function G5, G10 and G13 which are considered complex problems. However, our approach is unable to perform better in function G1 and G2. Since both G1 and G2 are high dimensional problems, it seems that the proposed approach needs to be improved for solving high dimensional optimization problems. This may be explained by the "No free lunch" theorem [99].

Based on the experiment results from the five test functions G5, G6, G7, G9 and G10, the proposed performance-based dynamic neighbourhood has proved to be able to find solutions faster than the static topology. It reveals that the performance improvement by the dynamic neighbourhood topology is significant for some functions like G6 and G9. For functions G5, G7 and G10, although the improvement is not obvious, it still outperforms the static neighbourhood topology. The Iteration-based experiment results demonstrate that the dynamic neighbourhood topology can find better results than static neighbourhood topology. These results support the hypothesis that "Allocating particles that have a similar performance in a neighbourhood is more efficient than allocating particles randomly in a neighbourhood in PSO". We argue that communication is more effective between individuals which are in the same or similar performance group.

This study is one of the few attempts to adopt the multi-objective constraint-handling method into PSO algorithm.

Chapter 4

CONSTRAINED MULTI-OBJECTIVE OPTIMIZATION USING PSO ALGORITHM

4.1 INTRODUCTION

Chapter 3 presented a multi-objective constraint-handling method incorporating with PSO algorithm for single objective constrained optimization problems. The target for this chapter is to solve constrained multi-objective optimization problems using PSO algorithm.

Most real-world search and optimization problems involve multiple objectives (MO) that need to be achieved simultaneously. The presence of the constraints brings difficulties in optimization since the search space has to be restricted in feasible regions. In solving MO problems, three goals need to be achieved [1]: find a set of solutions as close as possible to the true Pareto-optimal front; find a set of solutions as diverse as possible; and find a set of solutions as many as possible.

Population-based optimization techniques such as GAs, PSO, DE and ACO have been a popular choice for MO problems. The main reason is that these algorithms are capable of finding a set of Pareto-optimal solutions in a single run. With the success of the PSO in single objective optimization, researchers are motivated to extend the use of PSO in

MO problems. In order to apply PSO algorithm in multi-objective optimization problems, two main decisions need to be made: the first is how to determine a particle's personal best location; and the second is how to select the best particle among a neighbourhood (*gBest* or *lBest*). Up to now, a number of strategies (more than twenty five [87]) that adopt PSO algorithm in MO problems have been proposed. Some examples are: dynamic neighbourhood [79], grid method [80], weighted aggregation [81], multi-swarm [82], dominated tree [83], fitness sharing [85], maximin strategy [100] and Huo et al [86]. A detailed review can be found in [87]. However, most of these approaches are constraint free. How to integrate constraint-handling methods with the multi-objective PSO has motivated this research.

There are mainly two papers that have addressed the constrained multi-objective optimization by using PSO algorithm [88] [89]. However, they have some drawbacks as stated in Chapter 2.

In this chapter, we also adopt the constraint dominance concept, and propose an easy-to-implement PSO algorithm for tackling constrained multi-objective optimization problems. The proposed approach defines two sets of rules for determining the cognitive and social components of the PSO algorithm. The simulation results to the four constrained multi-objective optimization problems will be presented.

The rest of the chapter is organized as follows. Section 4.2 presents the proposed approach for constrained multi-objective optimization problems including the selection rules and the modified PSO algorithm. Section 4.3 presents the experiment results to the four test functions. Section 4.4 discusses the performance issues of the proposed approach. Section 4.5 summarizes the chapter.

4.2 PROPOSED APPROACH

4.2.1 Problem Description and Constraint Dominance

As stated in Chapter 1, a general multi-objective constrained optimization problem consists of a decision vector $x=(x_1,x_2,...,x_n)^T$, an objective function vector $f(x)=(f_1(x),f_2(x),...,f_k(x))$ and a constraint function vector $g(x)=(g_1(x),g_2(x),...,g_m(x))$. The problem can be stated as to find $x*$ which

$$\left.\begin{array}{lll} \text{minimize} & f_j(x), & j=1,2,\ldots k \\ \text{subject to} & g_i(x)\le 0, & i=1,2,\ldots m \end{array}\right\}$$

where k is the total number of objective functions and m is the total number of constraints.

From Chapters 2 and 3, the feasibility of a solution can be assessed by its total amount of constraint violations, described by

$$\Phi\ (x)=\sum_{i=1}^{m}\max(0,g_i(x))$$

If a solution satisfies all constraints, $\Phi(x)$ returns a zero; the solution is feasible. Otherwise, $\Phi(x)$ returns a positive number; the solution is infeasible. Considering an absolute equality is difficult to achieve in implementation, we can use a feasibility criterion ε to evaluate a solution's feasibility, that is, a solution $\boldsymbol{x}$ is considered feasible if its $\Phi(x)\le\varepsilon$ (ε is small positive number).

The constraint dominance concept is adopted in the proposed approach. The definition can be found in Chapter 1.

4.2.2 Selection Rules

Based on the constraint dominance concept, we propose the selection rules for determining *pBest* and *lBest* of the PSO algorithm, as follows.

Rule Set 1: Personal best particle updating rules

Suppose a particle's new location is *pNew* and its personal best location in the history is *pBest:*

- If both *pNew* and *pBest* are feasible, and if *pNew* dominates *pBest*, update *pBest* with *pNew*;
- If *pNew* is feasible and *pBest* is not feasible*,* update *pBest* with *pNew*;
- If *pNew* is not feasible and *pBest* is feasible, *pBest* is not updated;
- If both *pNew* and *pBest* are infeasible, and if *pNew* has a lower constraint violations (Φ) than *pBest* has, update *pBest* with *pNew*.

The Rule Set 1 is summarized in Table 4-1.

Table 4-1 Personal best particle updating rules

pNew	*pBest*	*pNew* dominates *pBest*?	*pNew* has lower Φ than *pBest*?	Next *pBest*
feasible	feasible	yes		*pNew*
feasible	infeasible			*pNew*
infeasible	feasible			*pBest*
infeasible	infeasible		yes	*pNew*

Rule Set 2: Local (or global) best particle selection rules:

Among a neighbourhood, select the best performed *k* (*k* is the number of objective functions) particles $lBest_i$ $(i= 1$ to $k)$ in each objective function:

- If all $lBest_i$ $(i= 1$ to $k)$ are feasible, randomly select one as *lBest.* In this way, a particle may follow $lBest_1$ at a time, and follow $lBest_2$ at another time. All $lBest_i$ $(i= 1$ to $k)$ will get the same probability to be selected as *lBest;*
- If there exist feasible particles and infeasible particles in all $lBest_i$ $(i= 1$ to $k)$, the *lBest* is randomly selected from all the feasible particles. The infeasible particles are disregarded;
- If there exist no feasible particles in all $lBest_i$ $(i= 1$ to $k)$, the particle with the lowest constraint violations is selected as *lBest.*

For two objective optimization problems, for example, if $lBest_1$ is the best particle in objective f_1 and particle $lBest_2$ is the best particle in objective f_2. An arbitrary particle *p* in the same neighbourhood will have its *lBest* determined by the rules listed in Table 4-2.

Table 4-2 Local best particle selection rules

$lBest_1$	$lBest_2$	*lBest*
feasible	feasible	$rand(lBest_1, lBest_2)$
feasible	infeasible	$lBest_1$
infeasible	feasible	$lBest_2$
infeasible	infeasible	one with the lower Φ

The proposed criterion-based rules for *pBest* and *lBest* have the following features:

- The feasibility is on the top priority;

- The *pBest* evolves towards the bottom-left direction in the objective space (assume all objectives are being minimized) since the Pareto dominance concept is adopted;
- The *lBest* makes effort to extend the spread along the Pareto-optimal front since the best particle in one objective function is followed.

- The rules can be applied to problems that have any number of objective functions.

4.2.3 Algorithm

Table 4-3 is the structure of the proposed PSO algorithm for constrained multi-objective optimization problems.

Table 4-3 Structure of the modified PSO algorithm for constrained multi-objective optimization problems

```
01: Initialize particles
02: Calculate fitness values of particles under each objective
03: Calculate constraint violations of each particle
04: Set current locations as personal best locations
05: Set local best location for each particle according to Rule Set 2
06: Do
07:   For each particle
08:       Calculate new velocity by PSO formula
09:       Calculate new location by PSO formula
10:       Update personal best location according to Rule Set 1
11:   End For
12:  Set local best location for each particle according to Rule Set 2
13:  End Do
```

4.3 EXPERIMENTS

4.3.1 Test Functions

Four functions (taken from [1]), named BNH, TNK, SRN and OSY have been selected for testing the proposed approach. Function BNH has a continuous convex Pareto-optimal set. Function TNK has a discontinuous, convex and nonconvex Pareto-optimal set. Both function SRN and function OSY have the continuous linear (can be considered convex or nonconvex) Pareto-optimal sets. The TNK and OSY functions are considered difficult problems since TNK has a discontinuous Pareto-front and OSY is a high dimensional and highly constrained problem. They are described in Equations (4.1) to (4.4), as follows.

BNH:

$$\begin{cases} \text{Minimize} & f_1(x) = 4x_1^2 + 4x_2^2, \\ & f_2(x) = (x_1 - 5)^2 + (x_2 - 5)^2, \\ \text{subject to} & g_1(x) = (x_1 - 5)^2 + x_2^2 - 25 \le 0, \\ & g_2(x) = 7.7 - (x_1 - 8)^2 - (x_2 + 3)^2 \le 0, \\ & 0 \le x_1 \le 5, \qquad 0 \le x_2 \le 3. \end{cases} \tag{4.1}$$

TNK:

$$\begin{cases} \text{Minimize} & f_1(x) = x_1, \\ & f_2(x) = x_2, \\ \text{subject to} & g_1(x) = 1 - x_1^2 - x_2^2 + 0.1\cos(16\arctan\frac{x_1}{x_2}) \le 0, \\ & g_2(x) = (x_1 - 0.5)^2 + (x_2 - 0.5)^2 - 0.5 \le 0, \\ & 0 \le x_1, x_2 \le \pi. \end{cases} \tag{4.2}$$

SRN:

$$\begin{cases} \text{Minimize} & f_1(x) = 2 + (x_1 - 2)^2 + (x_2 - 1)^2, \\ & f_2(x) = 9x_1 - (x_2 - 1)^2, \\ \text{subject to} & g_1(x) = x_1^2 + x_2^2 - 225 \le 0, \\ & g_2(x) = x_1 - 3x_2 + 10 \le 0, \\ & -20 \le x_1, x_2 \le 20. \end{cases} \tag{4.3}$$

OSY:

$$\begin{cases} \text{Minimize} & f_1(x) = -[25(x_1-2)^2 + (x_2-2)^2 + (x_3-1)^2 + (x_4-4)^2 + (x_5-1)^2], \\ & f_2(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2, \\ \text{subject to} & g_1(x) = 2 - x_1 - x_2 \le 0, \\ & g_2(x) = -6 + x_1 + x_2 \le 0, \\ & g_3(x) = -2 - x_1 + x_2 \le 0, \\ & g_4(x) = -2 + x_1 - 3x_2 \le 0, \\ & g_5(x) = -4 + (x_3-3)^2 + x_4 \le 0, \\ & g_6(x) = 4 - (x_5-3)^2 - x_6 \le 0, \\ & 0 \le x_1, x_2, x_6 \le 10, \quad 1 \le x_3, x_5 \le 5, \quad 0 \le x_4 \le 6. \end{cases} \tag{4.4}$$

4.3.2 Results

For multi-objective optimization, the number of non-dominated solutions is directly linked to the population size. Therefore, a large size of population is required.

Twenty independent runs have been performed for each case. PSO parameters are: $c_1 = 1.0$; $c_2 = 2.0$; $V_{\max} = 0.5 \bullet (U{-}L)$, where U and L are the upper and lower boundary for the decision variables; the population size is 200 for BNH and SRN; the population size is 500 for TNK and OSY; the maximum iteration $i_{\max}$ for all four cases are 1000; the inertia weight $w = 0.1$; $\chi = 0.63$; a potential solution is considered feasible when its $\Phi < \varepsilon = 1.0E{-}05$

; Neighbourhood topology is set to the circular ring local model with the neighbourhood size 2.

Figures 4.1 to Figure 4.8 illustrate the theoretical Pareto-optimal fronts and the simulated Pareto-optimal fronts from the best runs for the four test cases. The best run means that the Pareto-optimal front has a small Spacing/Spread (S/D) value. The theoretical Pareto-optimal fronts are deliberately included for a **visual** comparison because there are not many research data available for constrained multi-objective test problems.

156 out of 200 non-dominated solutions are found for BHN function. 111 out of 500 non-dominated solutions are found for TNK function. 161 out of 200 non-dominated solutions are found for SRN function and 56 out of 500 non-dominated solutions are found for OSY function. The algorithm achieved a reasonably good number of non-dominated solutions for the first three functions. However, the number of non-dominated solutions found for the last function (OSY) is small due to the complexity of the problem.

By observing the simulated Pareto-optimal fronts and the theoretical Pareto-optimal fronts [1], the proposed approach is able to converge to the Pareto-optimal solutions effectively. The final solution curves are reasonably dispersed.

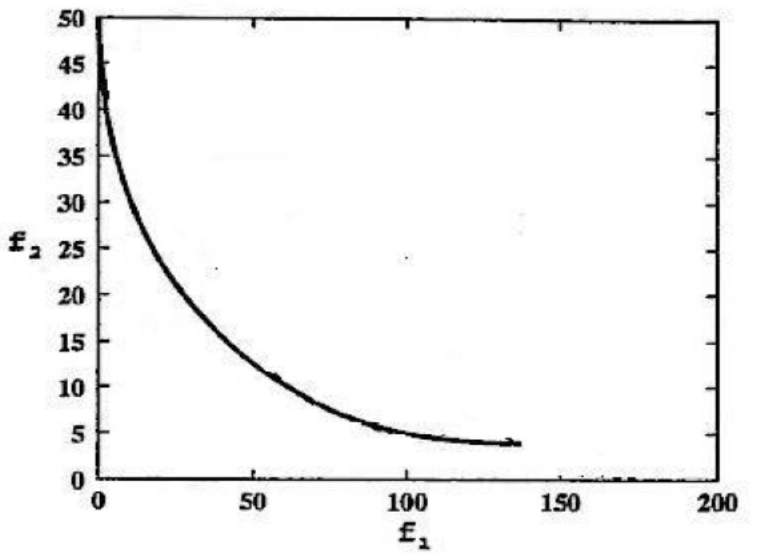


Figure 4.1 Theoretical Pareto-optimal front for test problem BNH (taken from [1])

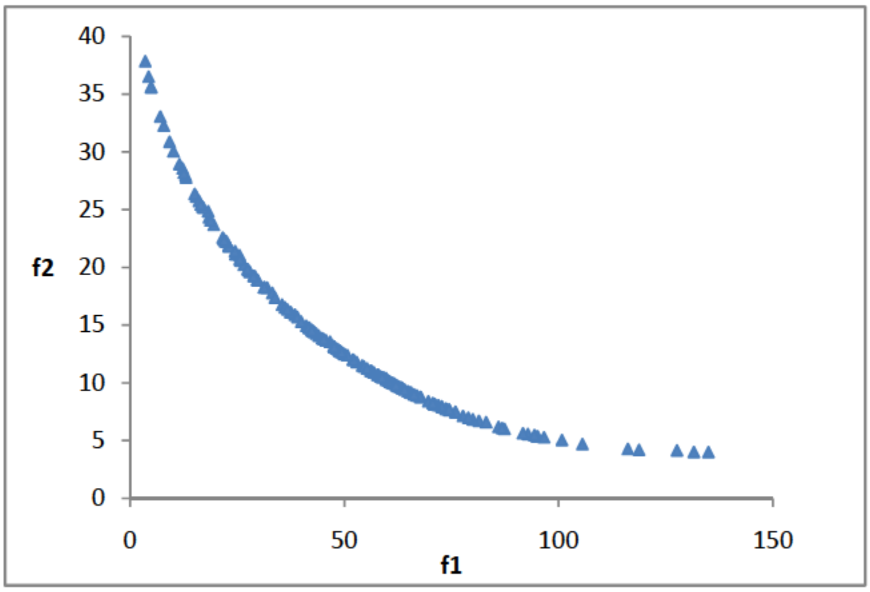


Figure 4.2 Simulated Pareto-optimal front for test problem BNH

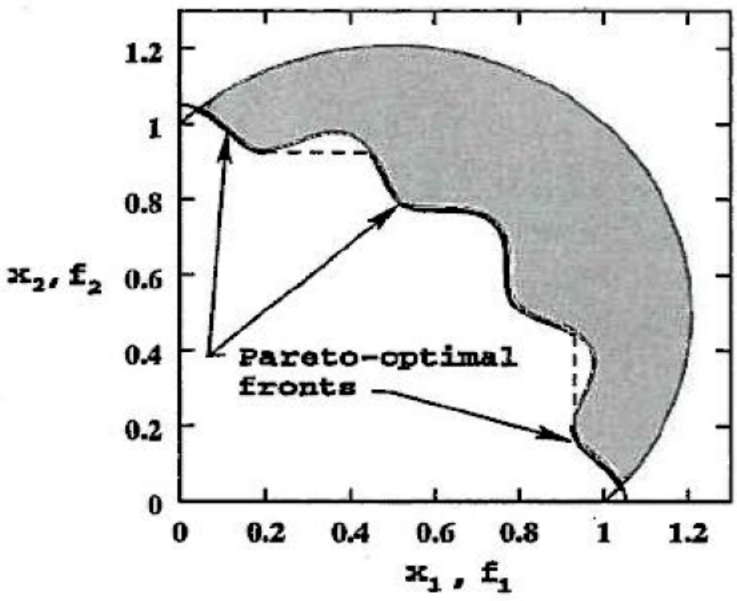


Figure 4.3 Theoretical Pareto-optimal front for test problem TNK (taken from [1])

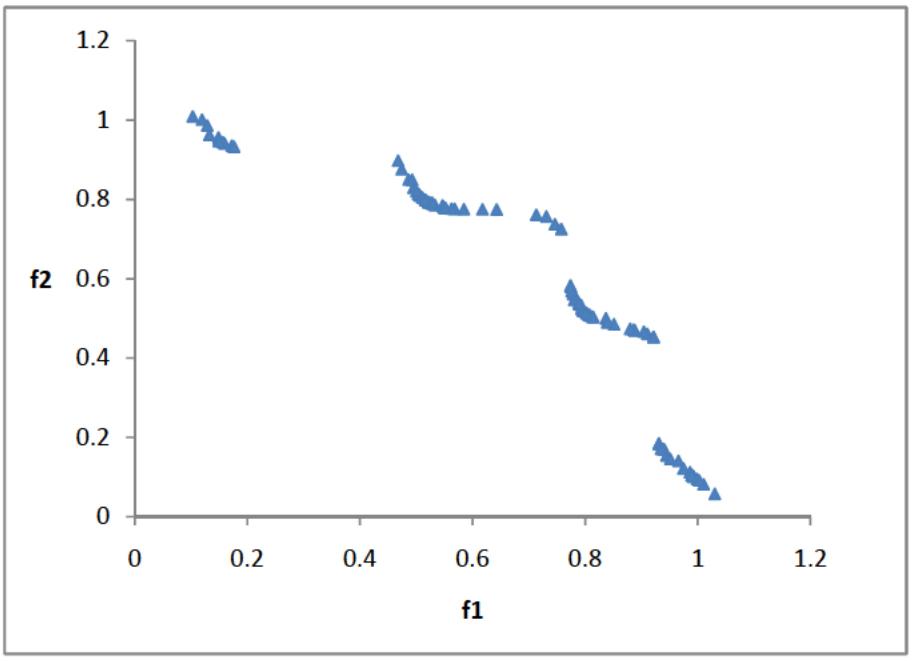


Figure 4.4 Simulated Pareto-optimal front for test problem TNK

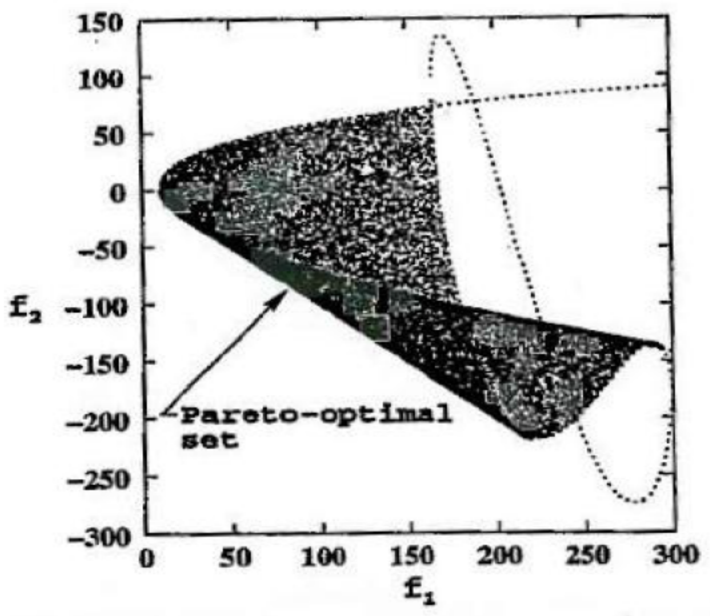


Figure 4.5 Theoretical Pareto-optimal front for test problem SRN (taken from [1])

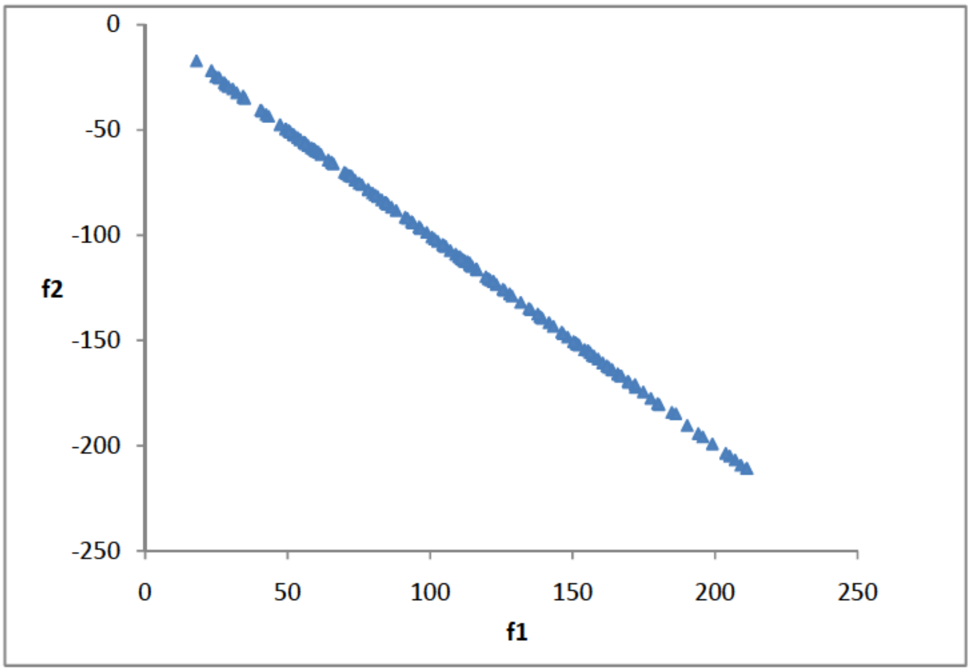


Figure 4.6 Simulated Pareto-optimal front for test problem SRN

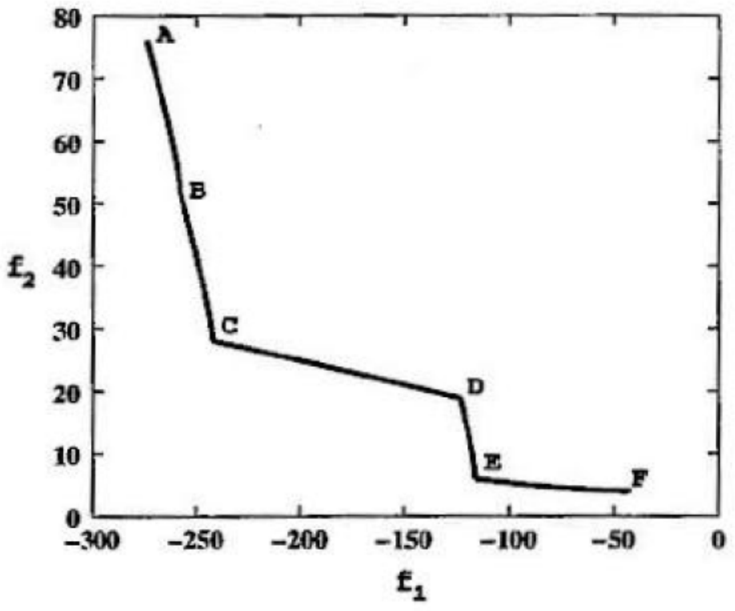


Figure 4.7 Theoretical Pareto-optimal front for test problem OSY (taken from [1])

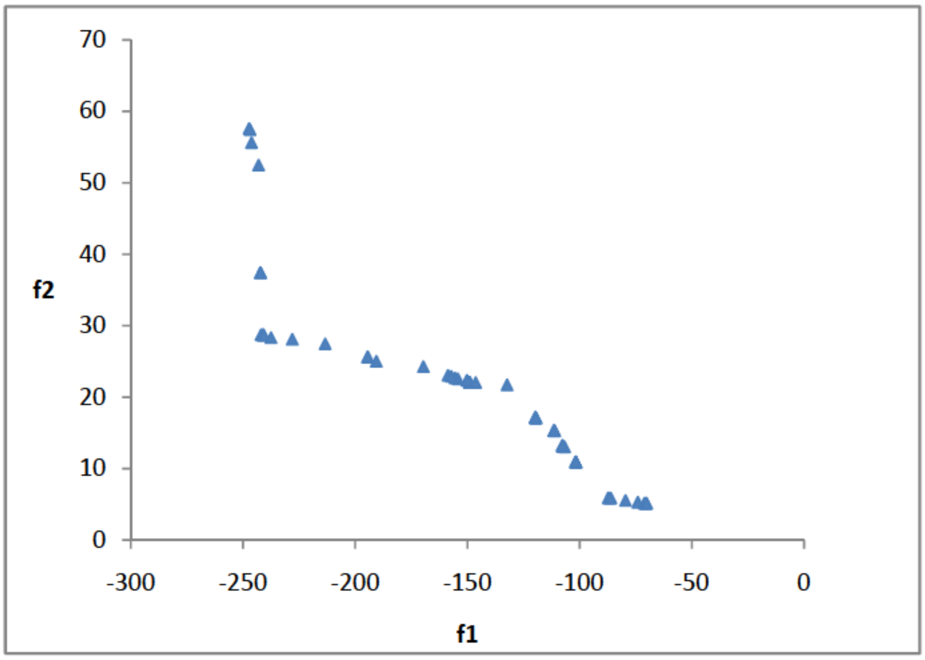


Figure 4.8 Simulated Pareto-optimal front for test problem OSY

4.4 PERFORMANCE EVALUATION

Three metrics can be used for evaluating the performance of a multi-objective optimization algorithm, as follows.

Spacing (SP) (as cited in [1]) measures how well distributed (spaced) the solutions in the non-dominated set found. The formula is presented in Equation (4.5),

$$S = \sqrt{\tfrac{1}{n}\sum_{i=1}^{n}(d_i - \overline{d})^2} \tag{4.5}$$

where n is the number of solutions in the obtained non-dominated set, $d_i = \min_{k\in n \wedge k\neq i} \sum_{m=1}^{M} | f_m^i - f_m^k |$ and $\overline{d}$ is the mean value of the above distance measure $\overline{d} = \sum_{i=1}^{n} d_i / n$; M is the number of objective functions. When the solutions are near uniformly spaced, the corresponding distance measure will be small. Thus, an algorithm finding a set of non-dominated solutions having a smaller spacing S is better.

Maximum Spread [61] gives a value which represents the maximum extension between the farthest solutions in the non-dominated set found. The formula is presented in Equation (4.6). A bigger value indicates better performance.

$$D = \sqrt{\sum_{m=1}^{M} (\max_{i=1}^{n} f_m{}^i - \min_{i=1}^{n} f_m{}^i)^2} \tag{4.6}$$

f_m is the m-th objective function value.

Generational Distance (as cited in [1]) is a metric to find the average distance of the non-dominated set of solutions from the real Pareto optimal set. The formula is given in Equation (4.7),

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{4.7}$$

where d_i is the Euclidean distance between solution i from the set of n non-dominated solutions found and the closest element from the real Pareto optimal set. A smaller value indicates the solutions found are closer to the real Pareto front.

Experiments have been conducted based on the first two metrics, that is, "Spacing" and "Spread". Table 4-4 presents the experiment results for the four test cases. Since a smaller "Spacing" value and a larger "Spread" value are expected, the minimum *S* and the maximum *D* are listed in the table. The average values and the standard deviations are also included.

The "Generational Distance" is not evaluated at this stage because of the lack of the real Pareto-optimal front data.

Due to the complexity and different user demands, there are not many data available for performance comparison. We include the "Spacing" data (EM-MOPSO) from [89] in Table 4-5 for a brief comparison. No "Max Spread" data available from [89]. The NSGA-II data in Table 4-5 is also taken from [89].

Table 4-4 Spacing and Maximum Spread for testing cases based on 20 runs

Criteria	Item	BNH	TNK	SRN	OSY
	Min.	0.599672	0.004032	1.089612	1.040084
S	Avg.	0.979688	0.012609	1.587708	2.943298
	Std.	0.258917	0.003361	0.423624	1.057702
	Max.	143.5117	1.403277	302.6255	225.3124
D	Avg.	136.9071	1.362672	276.0655	155.4652
	Std.	3.775605	0.031040	16.26418	21.64976

Table 4-5 Statistic results by different approaches

		This approach	NSGA-II	EM-MOPSO
	Min.	0.599672	0.6408	0.6357
BNH	Avg.	0.979688	0.7756	0.6941
	Std.	0.258917	0.0727	0.0385
	Min.	1.089612	1.3402	1.0768
SRN	Avg.	1.587708	1.5860	1.2439
	Std.	0.423624	0.1337	0.1055

According to Table 4-5, the proposed approach obtained the best “Spacing” value in function BNH. For function SRN, the proposed approach outperformed NSGA-II in the best “Spacing” value but slightly worse than EM-MOPSO. For algorithm consistency, the proposed approach did not outperform the other two approaches in both functions but the standard deviation values obtained are less than 0.5.

It is realized that the “Spacing” value can only be applied in consecutive solutions which may not be suitable for those problems that have discontinuous solutions such as function TNK. In such cases, other performance metrics may be taken into consideration.

4.5 SUMMARY

Population-based evolutionary techniques have been a popular choice for multi-objective optimization problems. The presence of constraints brings difficulties since the search space has to be restricted in a feasible region. Most existing multi-objective PSO proposals did not consider the constraints. Integrating constraint-handling mechanisms with multi-objective PSO is a challenging topic.

This chapter has proposed a modified PSO algorithm for solving constrained multi-objective optimization problems. Based on the constraint dominance concept, the proposed approach defines two sets of rules for determining the cognitive and social components of the PSO algorithm. The advantages of the proposed approach are: it can be applied in solving problems that have any number of objective functions; it is simple to understand and easy-to-implement; it is relatively computationally inexpensive since no external archive is used.

The simulation results to the four constrained multi-objective optimization problems have demonstrated the proposed approach is able to find the Pareto-optimal solutions effectively. Performance evaluation has shown the proposed approach achieved reasonably good results in two metrics -“Spacing” and “Spread”.

The proposed approach is one of the few attempts that using PSO in constrained multi-objective optimization problems.

Chapter 5
PROGRAM DESIGN, IMPLEMENTATION AND MORE RESULTS

5.1 INTRODUCTION

This chapter consists of two parts. The first part describes the design and implementation issues for the proposed approaches in Chapters 3 and 4. The second part presents the simulation results for three engineering design optimization problems. A special case, that is, when a predefined goal is known, the optimization task is to identify the design variables that achieve the goal, will be discussed.

A basic advantage of all EAs lies in the inherent parallelism of the algorithms. Parallel implementations of EAs are easily scalable to large populations, thus providing a good potential to exploit even massively hardware [101]. Parallel computing involves using multiple processing elements simultaneously to solve parallel execution problems. Multithreaded programming principle is able to simulate the parallel processes in a single processing element. In order to improve the computing performance, the multithreaded object-oriented programming principles are adopted in the program design and implementation which will be presented in the first part of this chapter.

In some real world applications, it is often that the optimization problems have predefined targets (or goals) and the optimization task is to identify the decision variables that are needed to attain the predefined targets. For example, in a budget allocation application, the total budget is pre-specified; an optimization task could be used to find a rational money allocation to different budgetary items which minimize the deviations from the total budget. A simple scenario is: Suppose you have a total budget *T*, you are going to allocate this budget *T* to the budgetary items $x_1, x_2, ..., x_n$. The objective is to find $x = (x_1, x_2, ...x_n)$ to achieve $T = \sum_1^n x_i$ or to minimize $|T - \sum_1^n x_i|$ subject to satisfying a number of constraints such as $x_1 + x_3 \leq T_1$. That is, if a solution with the desired target exists, the optimization task is to identify that particular solution. If there exists no solution which achieves pre-specified targets, the optimization task is to find solutions which minimize deviations from the targets. This is referred to as goal programming principle [1].

The goal programming generally involves multiple goals that need to be achieved. When the multiple goals conflict with each other, the optimization is related to the multi-objective optimization. Consider the goal programming as a constraint satisfaction process, if one goal is picked up as the objective function and other goals to be treated as constraints, a goal programming problem can be transformed into a single objective constrained optimization problem. The proposed multi-objective constraint-handling method in Chapters 3 should be applicable.

By modifying the algorithm proposed in chapter 3, the second part of this chapter will present a goal-oriented multi-objective constraint-handling method incorporated with the PSO algorithm for tackling optimization problems that have predefined goals. The simulation results to the three well known engineering design optimization problems will be presented and discussed.

The rest of the chapter is organized as follows: Section 5.2 presents a multithreaded Object-Oriented approach for PSO implementation including UML modelling. Section 5.3 presents the goal-oriented multi-objective constraint-handling method and simulation results to the three engineering design optimization problems. Section 5.4 is the summary of the chapter.

5.2 A MULTITHREADED OBJECT-ORIENTED APPROACH FOR PSO IMPLMENTATION

5.2.1 Parallel Computing and Evolutionary Algorithms

Generally, two types of parallelization exist for evolutionary algorithms – individual parallelization and population parallelization. For individual parallelization, all individuals evolve simultaneously according to some artificial rules such as operators in GAs or mathematical formulas in PSO algorithm. The individuals may or may not need to communicate with the others during evolution. Population parallelization involves multiple populations (population contains individuals) that evolve simultaneously. The co-evolution model [102], the multi-swarm model [103] and the multiple-independent-run are typical examples for population parallelization.

Two main parallel models have been followed for parallel computing. In *fine-grained* model, few individuals (or populations) are assigned to single processors and information exchange among the processors is frequent. On the contrary, in *coarse-grained* model, larger subpopulations are assigned to single processors and information exchange is rather rare. Research on parallel EAs has quickly shown that fine-grained parallelization results in a very significant communication overhead. Therefore, the focus has mostly turned to coarse-grained parallelization schemes [27].

5.2.2 Multithreaded Object-Oriented Programming in Brief

In computer science contexts, threads are a way for a program to split itself into two or more simultaneously running tasks. The multiple threads can be distributed to a multiprocessor (multiple CPUs) system such as a supercomputer or a cluster computing system. Moreover, multithreading allows a single-processor system to act like a multiprocessor system in order to allow a computer with a single CPU to simulate concurrency. This feature greatly benefits computing tasks that can be easily split into multiple tasks and execution simultaneously. Multithreaded programs often run faster and are more user-friendly than those programs written in sequential and single-threaded programs.

Figure 5.1 illustrates how multithreading executes in a multiprocessor system and in a single-processor system. With a multi-processor system, the multiple threads are distributed to each processor. Each thread is independent of others. In a single-processor system, the multiple threads share the CPU time. The CPU devotes a small amount of time to one task, and then devotes a small amount of time to another task. In Java programming language, the multiple threads are scheduled by the Java Virtual Machine (JVM) which interprets compiled Java binary code for a computer processor so that it can perform a Java program instructions [104] .

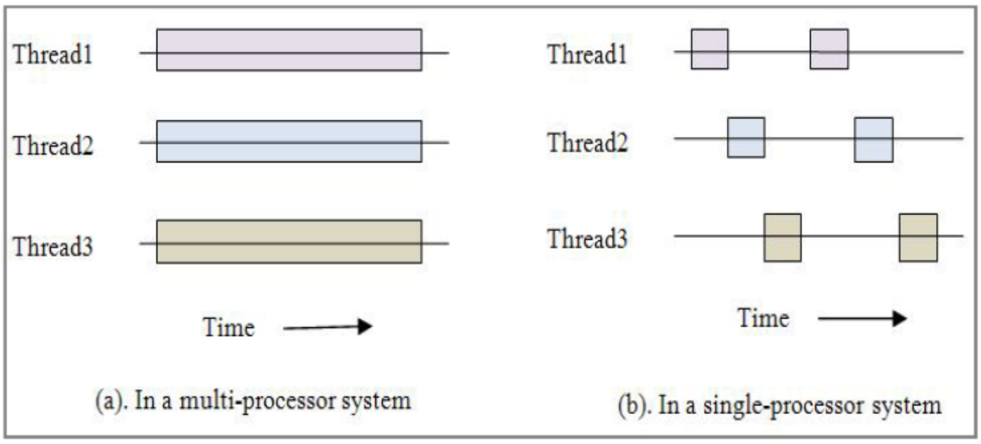

Thread1
Thread2
Thread3
Time
(a). In a multi-processor system
Thread1
Thread2
Thread3
Time
(b). In a single-processor system

Figure 5.1 Executing multiple threads in a multi-processor system and in a single-processor system

Object-oriented programming uses "objects" and their interactions to design applications and computer programs. Software designed in the object-oriented approach has better adaptability, reusability and robustness. Chief principles of the object-oriented approach are encapsulation, modularity and abstraction [105]. The techniques to support the principles are typing, interfaces, inheritance, polymorphism, classes and objects. Figure 5.2 summarizes the design goals, the corresponding principles and techniques used for an object-oriented approach. We adopt the OO design in order to maximize the code reusability and adaptability.

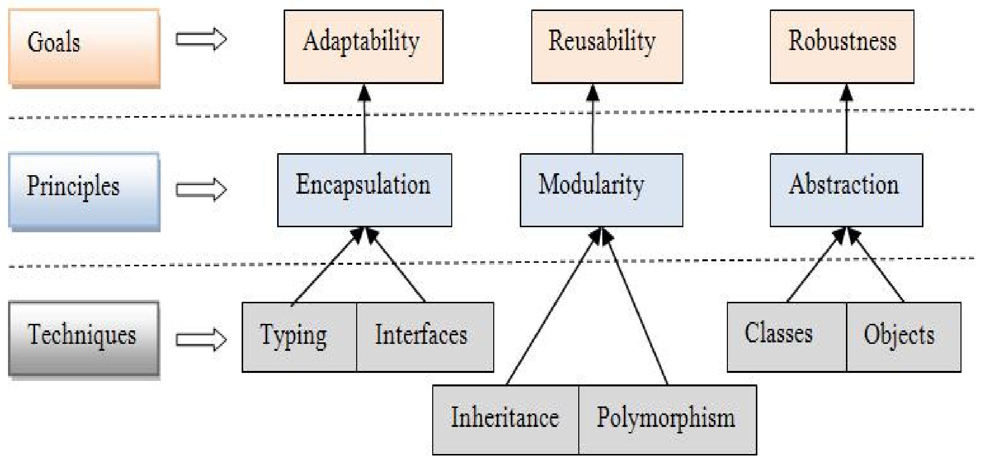


Figure 5.2 The goals, principles, and techniques of object-oriented design

5.2.3 System Design and Implementation

5.2.3.1 Design Objectives

Several objectives need to be considered in system design.

- The system should be applicable to different optimization problems.
- The programs should be easily adapted to different strategies in order to compare with other approaches.
- When generating individuals (particles for PSO) randomly, the individuals that lie on the variable boundary should be included. It is emphasized here since many random functions such as Java's `Random` function generating numbers that include the lower boundary value but **exclude** the upper boundary values.
- The multiple-independent-run can be executed in parallel. The coarse-grained parallel model would suffice since there is no information exchange between populations.

5.2.3.2 UML Modelling

Unified Modelling Language (UML) [106] is a standardized general-purpose modelling language in the field of software engineering which has been widely used in object-oriented design. To help understanding, a few UML notations are listed in Table 5-1.

Figure 5.3 illustrates the class structure in UML model. Six Java classes and one Java interface have been defined for the system. Two important interfaces "`Runnable`" and "`Comparable`" from the Java Application Programming Interface (API) [107] are also included in the UML model for easy illustration. Other Java classes from Java API are omitted in the UML model since they are considered trivial. The main responsibilities for each class or interface are specified as follows:

Table 5-1 UML relationship notations

Relationship	Symbol	Meaning
Interface Implementation	---------▷	Denotes one class must implement all methods defined by the interface
Aggregation	◇————	Denotes that objects of one class contain references to objects of another class
Dependency	---------〉	Denotes methods of one class uses an object of the other class in some way

- `StartMain`: This class is the entry point where the program starts to run. It creates objects of the application (that is, which optimization problem to run), generates multithreads for multiple independent runs, and starts building swarms to evolve for computation.

- `ApplicationInterface`: This interface defines common methods that an application must implement. It uses interface types to make code more reusable.
- `TheApplication`: This is the class for a specific optimization problem. Since each problem has different dimension, it contains problem attributes like the number of dimensions and the boundary constraints. This class implements the "`ApplicationInterface`" interface.
- `Randoms`: Like any population-based computation, PSO needs to uniformly generate particles randomly. The "`Randoms`" class provides methods to generate double values between a lower boundary constraint and an upper boundary constraint.
- `BoundaryConstraint`: "`BoundaryConstraint`" class is simply the limits to (the minimum and maximum of) a number range. Its "`boundaryType`" attribute indicates how a value that exceeds the boundaries is to be modified such that it is again within the boundaries.
- `Particle`: This class represents particle objects. A particle has its ID, velocity, coordinators (that is, location in a D-dimensional space), fitness and constraint violations. Each particle object also contains a reference to its best local particle of the neighbourhood. The class aggregates "`TheApplication`" and "`BoundaryConstraint`" classes. It also implements "Comparable" interface to define how to compare particles.
- `Swarm`: The "`Swarm`" class contains the main routine of PSO algorithm. It implements the "`Runnable`" interface since multiple swarms will be used to achieve multiple-independent-run which is always needed for any evolutionary

algorithms (that is, we should never report results by one run). The "`Swarm`" class initializes particles, determines particles' behaviours, calculates particles next positions, and conducts other necessary functions such as output data.

- `Runnable`: It is a standard Java interface from Java API. The "`Runnable`" interface should be implemented by any class whose instances are intended to be executed by a thread. `Comparable`: The "`Comparable`" interface imposes a total ordering on the objects of each class that implements it. By implementing its "`comparTo`" method, the objects can be sorted according to the rules specified. Since it is often that particles need to be compared in PSO algorithm, adopting this approach will make the comparison a lot easier.

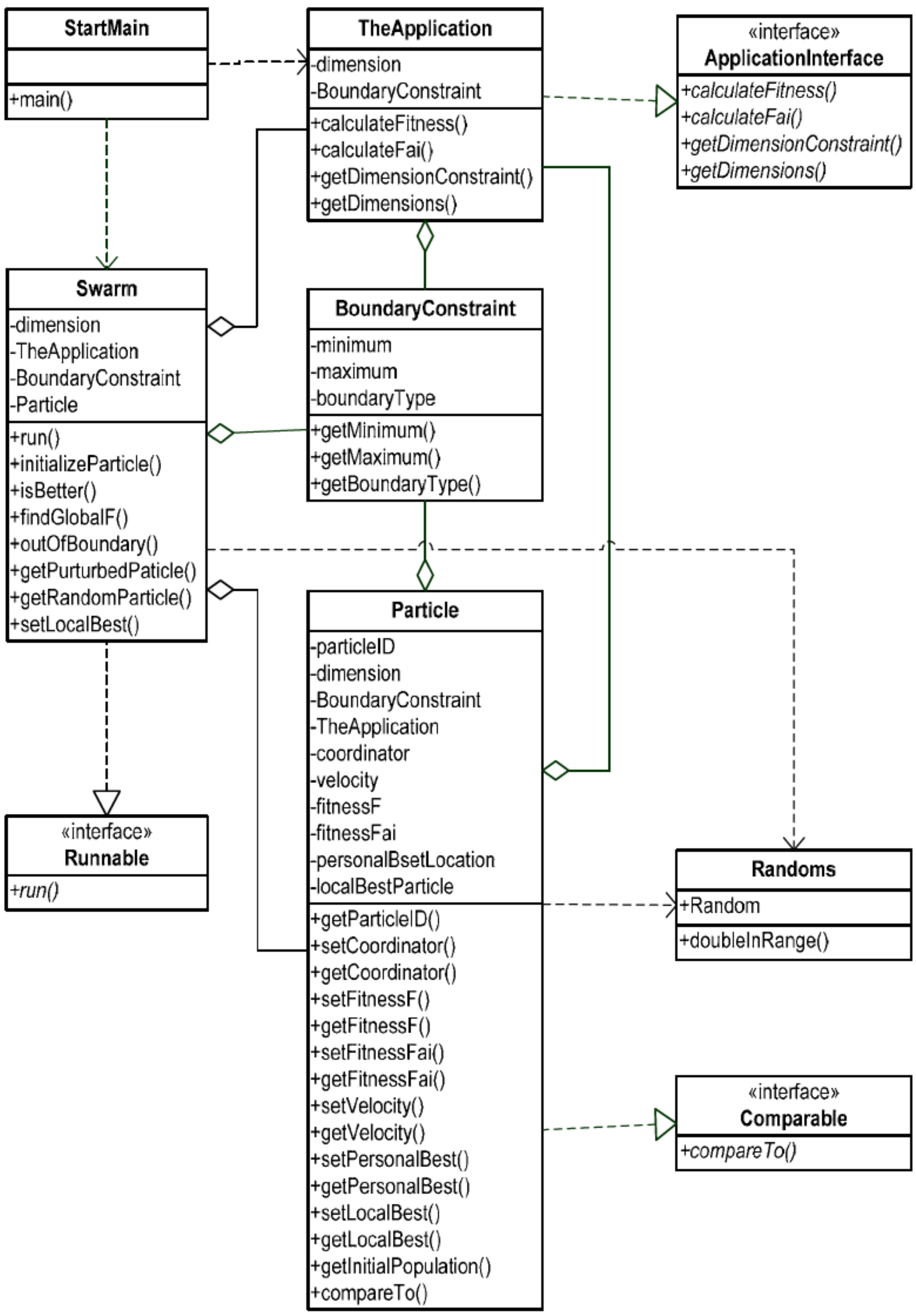


Figure 5.3 Class structure in UML model

5.2.3.3 Multithreaded Execution Process

The multiple-independent-run has been implemented in multithreads in Java. Each thread creates an independent swarm and the swarm evolves towards an optimal solution according to the PSO algorithm. After the maximum number of evolutions reached or the quit criterion is met, the best solution from each swarm is stored in a data file for post analysis. The post analysing includes calculating the best result, the mean result, the worst result, the standard deviations and reporting those results. Since no information exchange is needed between the swarms, the execution is a coarse-grained parallel process. Figure 5.4 illustrates the execution process.

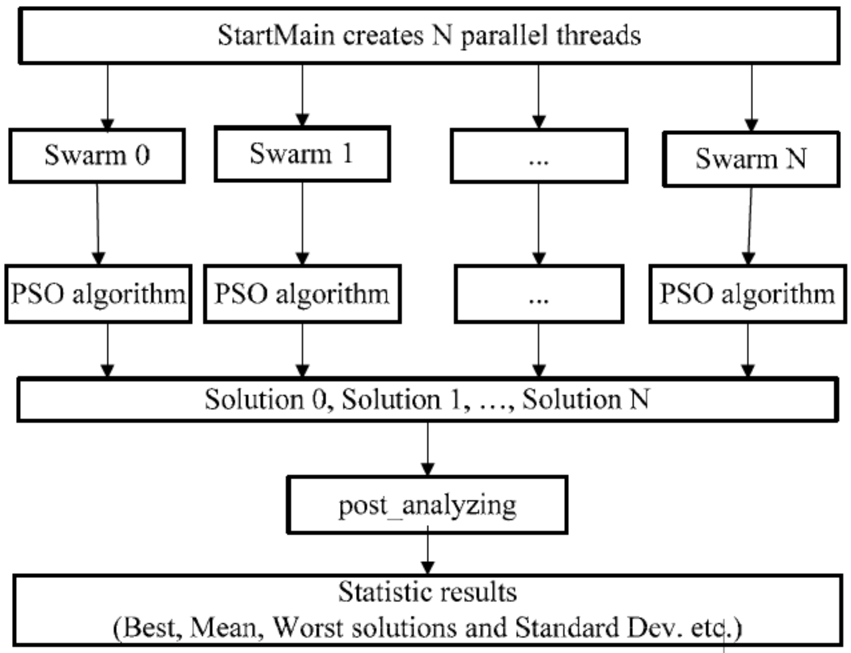

StartMain creates N parallel threads
Swarm 0
Swarm 1
...
Swarm N
PSO algorithm
PSO algorithm
...
PSO algorithm
Solution 0, Solution 1, …, Solution N
post_analyzing
Statistic results
(Best, Mean, Worst solutions and Standard Dev. etc.)

Figure 5.4 Parallel execution process diagram

5.2.4 Discussion

There are a number of benefits of using a Multithreaded Object-Oriented approach for EA implementation. Firstly, it is easy to extend to different applications. Object-Oriented design is modular. The structure imposed by modularity enables software reusability. For example, in the model illustrated in Figure 5.3, the `TheApplication` class is the only class that needed to be replaced for different applications. The implementation to the `TheApplication` class is simple since the pattern has been specified in the `ApplicationInterface` interface. Secondly, it is easy to adapt to different computing strategies. For example, if the comparison criteria for particles are changed, the user only needs to modify the `comparTo` method in the `Particle` class; if the user wants to change the random strategy such as uniform distribution or normal distribution, the `Randoms` class needs to be modified only. Lastly, since Java offers some advantages such as platform independent, friendly graphical user interface and multithreading, the approach can make the parallel execution in any platform more convenient.

However, compared with other traditional scientific programming languages like FORTRAN [108] or C [109], Java is not considered a high performance programming language in numerical computing [110]. For those problems that need intensive computation (e.g., execution takes months or years), approaches other than Java-based may perform better.

5.3 ENGINEERING DESIGN OPTIMIZATION

5.3.1 Problem Transformation and Formulation

Let's take a goal programming problem that has two objective functions as an example to illustrate the idea. Suppose $f_1(x)$ has a predefined target T_1 , and $f_2(x)$ has a predefined target T_2 , the optimization task can be stated as to find $x^* = (x_1, x_2, ..., x_n)$ that satisfy $f_1(x) = T_1$ and $f_2(x) = T_2$

or

$$\begin{cases} \text{minimize} & F_1(x) = \; | f_1(x) - T_1 | \\ \text{subject to} & f_2(x) \; = \; T_2 \qquad \text{and} \\ & g_i(x) \le 0, \qquad i = 1, 2, ..., m; \end{cases} \tag{5.1}$$

Recall the single objective constrained problems as in Equation (1.2), the only difference in Equation (5.1) from Equation (1.2) is that one more equality constraint is added. This constraint can be merged into other constraints as in $g_i(x)$ which make a total $m+1$ number of constraints, and then a goal programming problem can be rewritten as

$$\begin{cases} \text{minimize} & F(x) = \; | f(x) - T | \\ \text{subject to} & g_i(x) \le 0, \qquad i = 1, 2, ..., m+1; \end{cases} \tag{5.2}$$

Recall the multi-objective constraint-handling method, Equation (5.2) can be transformed into

$$\begin{cases} \text{minimize} & F'(x) = (F(x), \Phi(x)) \\ \text{where} & F(x) = f(x) - T \quad \text{and} \quad \Phi(x) = \sum_{i=1}^{m} \max(0,\ g_i(x)) \\ \text{goals} & f(x) - \mathrm{T} \le \Delta \ \text{ and } \ \Phi(x) \le \varepsilon \end{cases} \tag{5.3}$$

Where Δ and ε are two small positive numbers which indicate how close a solution to the predefined targets. Please note there is no need to use $| f(x) - \mathrm{T} | \le \Delta$ because any better-than-target solutions (that is, $f(x) < \mathrm{T}$) are allowed to be found for the model.

Figure 5.5 illustrates the expected solution area (goal area) for the model in Equation (5.3) in the objective space.

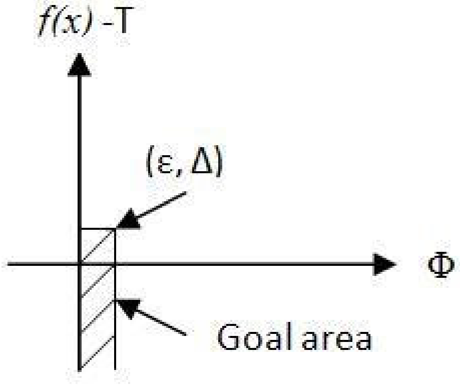


Figure 5.5 The goal area for a bi-objective constraint-handling optimization problem

In terms of the PSO algorithm, the particles should fly toward the goal area. Once the goals are achieved, no more evolution is needed. In case there is no solution satisfying both goals, the priority is given to $\Phi(x) \le \varepsilon$, the solution is found from all particles where their $\Phi(x) \le \varepsilon$.

5.3.2 A Goal-Oriented Multi-objective Constraint-handling Method with PSO Algorithm

Figure 5.6 illustrates the goal-oriented multi-objective constraint-handling method with the PSO algorithm. It is modified from the multi-objective constraint-handling method with PSO algorithm proposed in Chapter 3. PSO variant adopted in this section is the same as the one in Chapter 3, that is, as in Equation (3.2) and Equation (3.3). The selection rules are the same as the ones in Chapter 3. The modified parts are highlighted in Figure 5.6.

Compared with the previous multi-objective constraint-handling PSO with no predefined goals, the goal-oriented multi-objective constraint-handling method via PSO algorithm has the following differences:

- Whenever calculating fitness, both objectives $F(x) = f(x) - T$ and $\Phi(x)$ need to be evaluated. The first objective function now becomes $f(x) - T$ rather than $f(x)$.
- After each iteration, check whether the goal is obtained; if obtained, the iteration ends. This means that the particles do not need to go through the maximum iterations. Once the target is met, it stops evolving.

```
GlobalF = POSITIVE_INFINITY;
$P_0$ = URand ($L_i$, $U_i$)
$V_0$ = 0
$F_0$ = Fitness_F($P_0$)
$\Phi_0$ = Fitness_$\Phi$($P_0$)
$pBest_0$ = $P_0$
For i = 0 To number of particles
  $lBest_i$ = LocalBest(…, $P_{i-1}$, $P_i$, $P_{i+1}$, …)
End for
Do
  For i = 0 To n
     $V_{i+1}$ = Speed ( $P_i$, $V_i$, $pBest_i$, $lBest_i$)
     $P_{i+1}$ = $P_i$ + $V_{i+1}$
     r = URand (0,1)
     If ($r \leq p$)
       $TempP_{i+1}$ = Rand ($L_i$, $U_i$)
     $F_{i+1}$ = Fitness_F($P_{i+1}$)
     $\Phi_{i+1}$ = Fitness_$\Phi$($P_{i+1}$)
     If ($P_{i+1}$ isBetterThan $pBest_i$ )
       $pBest_i$ = $P_{i+1}$
     If ($\Phi_{i+1} \leq \varepsilon$)
       If ($F_{i+1}$ < GlobalF)
          GlobalF = $F_{i+1}$
   End For
   If (GlobalF <= $\Delta$ )
      Output GlobalF and $P_{i+1}$
      Stop
   For i = 0 To n
      $lBest_i$ = LocalBest(…, $P_{i-1}$, $P_i$, $P_{i+1}$, …)
   End for
End Do
```

Figure 5.6 Pseudo code of the goal-oriented multi-objective constraint-handling method with PSO algorithm

5.3.3 Results

Three engineering optimization examples are selected for simulation. The neighbourhood topology is set to ring topology with the neighbour size of 2. For each case, 30 independent runs (note: it is generally a convention to run programs 30 times to evaluate the consistency) have been performed. PSO parameters are: $w=0$; $c_1=c_2=2$; $\chi=0.63$; $V_{\max}=0.5\bullet(U-L)$, where U and L are the upper and lower limits for decision variable x; $p=0.1\%$; *number of particles* is *100*; the maximum iteration is set to *10000.* The feasibility tolerance allowed $\varepsilon=1.0E-09$.

In order to see the improvement made from different approaches, results from three other recent approaches are included in this section. The three approaches, CPSO [78], HPW-PSO [74] and Coello and Montes [69] are selected for comparison. The reasons for choosing these three approaches are: both CPSO and HPW-PSO are based on PSO algorithm as we adopted. However, they use different constraint handling methods. CPSO adopted a penalty function method and HPW-PSO adopted a preserving and searching for feasibility method. The Coello and Montes' approach uses the similar multi-objective constraint handling method as we adopted, but their implementation is through genetic algorithm. Therefore, we can evaluate the algorithm in different aspects.

5.3.3.1 Results for Welded Beam Design Problem

The Welded Beam Design problem (E01) is described in Appendix II. The goal value for this problem is set to 1.724852 which is best-known. The best solution found from our simulation and the other three approaches are listed in Table 5-2. In our approach, the best result found is **1.724852321**, which is close to the best-known result 1.724852; the mean result for 30 independent runs is **1.724861948**; and the standard deviation is **2.05462E-05**.

According to Table 5-2, the solution found from our approach is better than those achieved by the other three approaches (the result presented in CPSO [78] seems incorrect because they used $\tau_{\max} = 13,000 psi$ rather than $\tau_{\max} = 13,600 psi$). The statistics data demonstrate our approach performs excellently in both quality search and consistency.

Table 5-2 Optimal solution of welded beam design

Design	Best solution found			
variables	Proposed	CPSO	HPW-PSO	Coello and Montes
x_1	0.205729642	0.202369	0.24436898	0.205986
x_2	3.470488637	3.544214	6.21751974	3.471328
x_3	9.036623843	9.048210	8.29147139	9.020224
x_4	0.205729643	0.205723	0.244436898	0.206480
$g_1(x)$	-1.67E-09	-12.839796	-5741.1769331	-0.103050
$g_2(x)$	-1.32E-05	-1.247467	-0.00000067	-0.231748
$g_3(x)$	-6.08E-10	-0.001498	0.00000000	-0.000495
$g_4(x)$	-3.43298378	-3.429347	-3.02295458	-3.430043
$g_5(x)$	-8.07E-02	-0.079381	-0.11936898	-0.080986
$g_6(x)$	-2.36E-01	-0.235536	-0.23424083	-0.235514
$g_7(x)$	-2.47E-04	-11.681355	-0.00030900	-58.64688
$f(x)$	1.724852321	1.728024	2.3809565827	1.728226

It needs to be mentioned that our solutions are generated based on 100 particles and 10,000 maximum iterations, which forms a total maximum number of 1,000,000 function evaluations. In fact, since a goal oriented programming concept is adopted in the program, the total number of function evaluations, in most cases, is less than this maximum number setting. An experiment result will be presented later in this section.

Table 5-3 Statistic results for different approaches (welded beam design)

Case	Approach	Best	Mean	Std. Dev.
Case 1	Proposed	1.72485231	1.73612022	2.46E-02
	CPSO	1.72802400	1.74883100	1.29E-02
Case 2	Proposed	1.72485231	1.73612022	2.46E-02
	HES-PSO	1.72485084	NA	NA
Case 3	Proposed	1.72485747	1.76521069	4.40E-02
	Coello and Montes	1.72822600	1.79265400	7.47E-02

Case 1 and Case 2: 40 particles, 5000 iterations; Case 3: 40 particles, 2000 iterations

To compare with others, we simulated our approach by adjusting the total maximum number of function evaluations to 200,000 (to match CPSO[78]), 30,000 (to match HPW-PSO[74]) and 80,000 (to match Coello and Montes[69]) respectively, based on the 30 independent runs, the simulation results are listed in Table 5-3.

From Table 5-3, the proposed approach performs better in quality search (best solution found) than any of the other three approaches. The mean results obtained by our approach are also better than HPW-PSO and Coello and Monster's. In consistency (Standard Dev.), our approach performs better than Coello and Montes's approach and slightly worse than the other two. However, they are still small and acceptable.

5.3.3.2 Results for Pressure Vessel Design Problem

The Pressure Vessel Design problem (E02) is described in Appendix II. The goal value for this problem is set to 6059.946. The best solution found from our approach and the other three approaches are listed in Table 5-4. The best result found from our approach is

5971.4003, which is a better solution than any other solution (the best-known result is 6059.94634 before this research); the mean result for 30 independent runs is **6049.1590**; and the standard deviation is **22.841537**. As we can see, our approach performs well in quality search and reasonable consistency. Same as before, the best results are generated with a particle size of 100 and the maximum iteration is 10000.

Again, Table 5-5 is a collection of data obtained by comparing with other approaches.

From Table 5-5, the proposed approach in this chapter performs better in quality search than the other three. The mean results and standard deviations obtained by our approach are not better than the others. By looking up the much better solution generated from the original setting (100 particles, 10000 maximum iterations), it looks like our approach need more iterations to achieve more consistent results for this problem.

Table 5-4 Optimal solution of pressure vessel design

Design	Best solution found			
variables	Proposed	CPSO	HPW-PSO	Coello and Montes
x_1	0.79641436	0.812500	0.81250000	0.812500
x_2	0.39944943	0.437500	0.43750000	0.437500
x_3	41.0039194	42.091266	42.0984456	42.097398
x_4	190.801191	176.746500	176.636595	176.654047
$g_1(x)$	-5.04E-03	-0.000139	0.00000000	-0.000020
$g_2(x)$	-8.27E-03	-0.035949	-0.03588083	-0.035891
$g_3(x)$	-595.450105	-116.382700	0.00000000	-27.886075
$g_4(x)$	-49.1988089	-63.253500	-63.3634042	-63.345953
$f(x)$	5971.4003	6061.0777	6059.7143	6059.94634

Table 5-5 Statistic results for different approaches (pressure vessel design)

Case	Approach	Best	Mean	Std. Dev.
Case 1	Proposed	5990.105542	6160.11071	145.152
	CPSO	6061.077700	6147.13320	86.4545
Case 2	Proposed	5990.105542	6160.11071	145.152
	HES-PSO	6059.131300	NA	NA
Case 3	Proposed	6035.857686	6362.82540	266.134
	Coello and Montes	6059.946300	6177.25330	130.930

Case 1 and Case 2: 40 particles, 5000 iterations; Case 3: 40 particles, 2000 iterations

5.3.3.3 Results for Spring Design Problem

The Spring Design problem (E03) is described in Appendix II. The goal value for this problem is set to 0.012665. The best solution found from our approach and other three approaches are listed in Table 5-6. The best result found from our approach is **0.012665236**, which is very close to the best-known solution 0.012665 and better than any of the three comparing approaches; the mean result from our approach is **0.012714543**; and the standard deviation is **6.28E-05**. These data demonstrate our approach performs really well in both quality search and consistency.

Table 5-7 is a collection of data obtained by comparing with the other approaches. The data from Table 5-7 indicate the proposed approach in this research performs better than or very similar in quality search to the three other approaches (note the HPW-PSO[74] has second constraint nearly broken up). The mean results and the standard deviations obtained by our approach are slightly worse than the others. However, they are fairly small and in an acceptable range.

Table 5-6 Optimal solution of tension/compression string design

Design variables	Best solution found			
	Proposed	CPSO	HPW-PSO	Coello and Montes
x_1	0.051702169	0.051728	0.05169040	0.051989
x_2	0.357033166	0.357644	0.35674999	0.363965
x_3	11.27049760	11.244543	11.2871260	10.890522
$g_1(x)$	-4.07E-8	-0.000845	-0.0000045	-0.000013
$g_2(x)$	5.14E-9	-1.26E-05	0.00000009	-0.000021
$g_3(x)$	-4.05440797	-4.051300	-4.0538266	-4.061338
$g_4(x)$	-0.72750978	-0.727090	-0.7277064	-0.722698
$f(x)$	0.012665236	0.0126747	0.01266528	0.012681

Table 5-7 Statistic results for different approaches (tension/compression string design)

Case	Approach	Best	Mean	StD. Dev.
Case 1	Proposed	0.012666062	0.012812823	2.28E-04
	CPSO [78]	0.012674700	0.012730000	5.20E-05
Case 2	Proposed	0.012667195	0.013350094	7.50E-04
	HPW-PSO[74]	0.012665281	0.012702330	4.12E-05
Case 3	Proposed	0.012666626	0.012964163	3.67E-04
	Coello and Montes[69]	0.012681000	0.012742000	5.90E-05

Case 1 and Case 2: 40 particles, 5000 iterations; Case 3: 40 particles, 2000 iterations

5.3.4 Discussion

As mentioned before, the best solutions generated from our approach are based on 100 particles and 10,000 maximum iterations, which form a total number of 1,000,000 function evaluations. Although the maximum number of function evaluation looks larger, the actual numbers of function evaluation are less than this maximum because a goal oriented programming concept is adopted in the program. That is, once the goal is reached, evolution stops. This approach suits the real world applications better. Therefore, we have performed an experiment to illustrate how many minimum iterations/generations are needed to reach the best-known solutions or to achieve even better solutions. Table 5-8 is a summary of the experiment results based on 30 independent runs and with a particle population size of 100. In Table 5-8, a goal result is the existing best-known result. The minimum tolerance allowed Δ indicates how close a solution is to a goal solution. If a solution is less than the goal solution (better than the best-known) or if its $f - goal \le \Delta$ (close enough to the goal), evolution stops. Since the problem E02 has a larger magnitude than E01 and E03, the Δ is set to a relative larger value.

According to the Table 5-8, the lowest number of iterations needed to reach the goal solution for the E01, E02 and E03 problems are 610, 680 and 183 respectively; the average number of iterations needed are 1597, 4210 and 1158 respectively. This mechanism makes the computation cost more reasonable. Under our multithreaded Java programming implementation, the computation costs to these three engineering problems are not a big issue. The feasibility tolerance ε impacts on the results. A larger ε can sometime make the constraints not fully satisfied but the search will be easier. We used a fairly small $\varepsilon = 1.0E-09$ to ensure the constraints are fully satisfied.

Table 5-8 Iterations needed to reach the best-known solution

Problem	*Goal (Best-Known)*	*Tolerance allowed Δ*	*Lowest**	*Average**	Highest*
E01	1.724852	1.0E-04	610 (61000)	1597 (159700)	3324 (332400)
E02	6059.946	1.0E-01	680 (68000)	4210 (421000)	9998 (999800)
E03	0.012665	1.0E-04	183 (18300)	1158 (115800)	4590 (459000)

* indicate lowest iteration, average iteration and highest iteration needed (number in brackets indicates the number of function evaluation)

There is an issue in applying the goal-oriented multi-objective constraint-handling method via PSO algorithm in goal programming discipline. The goal programming problems normally involve multiple goals to be satisfied. How to pick up one goal acting on the objective function and treat other goals as constraints is crucial for optimization success. Referring to Equation (5.1), if $f_2(x) = T_2$ or $|f_2(x) - T_2| \le \delta$ is treated as a constraint, there is a possibility that this constraint can never be satisfied if the equality tolerance δ is small. In practice, some high level knowledge like preference (or priority) may help in choosing objective function and constraint functions. Otherwise, multi-objective optimization techniques should be used.

5.4 SUMMARY

This chapter has presented a multithreaded object-oriented approach for PSO implementation. Based on the multithreading concept and the object-oriented programming principles, the UML modelling has been introduced. The multiple-independent-run has been implemented in the coarse-grained parallel execution process. The proposed multithreaded object-oriented approach has a number of advantages, for example, it is easy to extend to different applications, and easy to adapt to different computing strategies.

This chapter has also presented the engineering design applications using the proposed goal-oriented multi-objective constraint-handling method via PSO algorithm. By picking up one goal as the objective and treating other goals as constraints, a goal programming problem can be transferred into a single objective constrained problem. The simulation results to the three well-known engineering design problems demonstrate that the proposed approach is effective and efficient in finding the consistent solutions for the three engineering design optimization problems.

In order to evaluate the algorithm performance, the population size and the maximum number of iterations have been adjusted to match three existing approaches for comparison. In relation to the search quality, the proposed approach has achieved better or very similar results on the three well-known engineering problems compared with existing approaches. Remarkably, a best ever solution has been found for the *pressure vessel design problem*. In relation to the algorithm consistency, the proposed approach performs better or similar in two (E01 and E03) out of three engineering design problems than the compared approaches. For E02 (*pressure vessel design problem*), the mean results and standard deviations obtained by the proposed approach are not better than the others. However, if the population and maximum number of iterations increase to the original values proposed (that is, 100 particles and 10000 maximum iterations), much better results can be achieved.

Since the goals can be used as the exit criteria, each particle does not need to go through the whole number of iterations. This will make the computation cost more reasonable.

Goal programming problems can be regarded as constraint satisfaction problems. In applying the proposed approach in goal programming discipline, one crucial task is to pick up one goal as objective function and treat other goals as constraints. If not properly selected, there is a possibility that the constraints can never be satisfied. Some high level knowledge like preference (or priority) may help in choosing the objective function and constraint functions. Otherwise, multi-objective optimization techniques should be considered for goal programming problems.

Chapter 6

POWER GENERATION UNIT LOADING OPTIMIZATION

6.1 INTRODUCTION

In December 2008, the Australian government introduced a national Emission Trading Scheme (ETS) in order to reduce carbon pollution which is causing climate change and is resulting in higher temperatures, more droughts, rising sea levels and more extreme weather [111] . Many companies and organizations have responded in compliance with the ETS. Since the largest source of greenhouse gas emissions (69.6% in 2006) [111] is contributed by the energy sector, power generation unit efficiency will be of great importance in the evolving carbon constrained economy.

Power plant efficiency improvement activities can be classified into two categories – plant modification (often irreversible) and operational improvement (often reversible). Traditional performance improvement activities have been often linked to plant modifications and large capital investment. Those performance improvement modifications are not, as people think, risk free. Even successfully done, they do not always materialize the promised benefits. For example, when a unit is upgraded to suit higher load, it will not be as efficient when load demand is low. Frequent changes in market strategy often require reversible changes such as operational changes rather than irreversible plant modifications. On the other hand, operational improvement is low-risk, low-cost and often with instant benefits. However, due to the various reasons, efficiency improvement through optimal operations has not been given the necessary attention.

The research on power generation unit loading optimization is to improve the operations to address the issue. It must be noted that any improvement gains including plant modification can only be materialized through operation. Optimal operation is the key to power generation performance.

The unit loading optimization problem has been studied in a branch called "Economic Load Dispatch" over the years. Many of the models only consider one objective, that is, heat consumption (as production cost), and using traditional deterministic approaches [112-115]. Due to public awareness of environmental protection, society demands adequate and secure electricity not only at the cheapest possible price but also at minimum levels of pollution [116]. Minimizing atmospheric pollution will be one of the major challenges for electric utilities. Several attempts have been made in using the stochastic metaheuristic methods for this application [116-118]. Zhao and Cao [117] proposed to use PSO algorithm and fuzzy rules to solve this multi-objective optimization problem. Basu et al [116, 118] reported their approach of using evolutionary programming and fuzzy satisfying method for this problem. Both approaches demonstrated the capabilities of using metaheuristics for the multi-objective load dispatch optimization problem.

As an example, this chapter presents two PSO-based approaches for solving the power generation unit loading optimization problem. The first approach (Model 1) treats emission as additional constraints and considers the application a single objective optimization problem. The second approach (Model 2) treats emission as an additional objective and considers the application as a multi-objective optimization problem. For evaluation purpose, two constraint-handling mechanisms, that is, multi-objective constraint-handling method and preserving feasibility constraint-handling method, are compared for the first model. The simulation results based on a coal-fired power plant is performed and the results will be presented and discussed.

The rest of the chapter is organized as follows. The problem modelling is presented in Section 6.2. It consists of problem description, specification and model formulation. Section 6.3 presents the proposed approach for the Model 1 including two constraint-handling mechanisms incorporating with the PSO algorithm. Section 6.4 presents the approach for the Model 2. The simulation results and a comparison study of the two different constraint-handling methods will be presented in Section 6.5. Section 6.6 is a summary of the chapter.

6.2 MODELLING

6.2.1 General Description

A typical coal-fired power generation unit mainly consists of three components - a boiler, a steam turbine and a generator. The boiler burns fuel (coal) to produce heat. The heat turns water into steam. The turbine is driven by the steam to transform heat into mechanical energy. The generator converts mechanical energy into electrical energy. Figure 6.1 illustrates this typical coal-fired power generation unit.

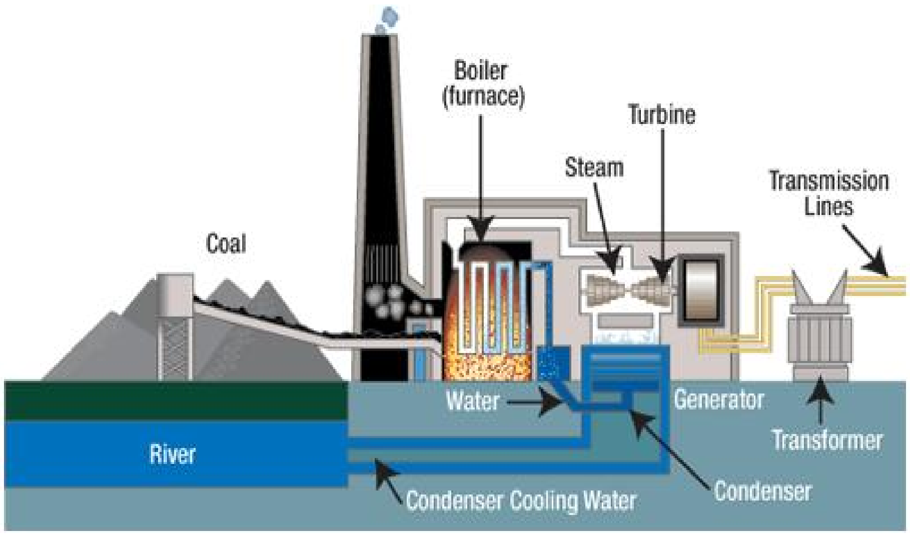


Figure 6.1 A typical coal-fired power generation unit
(image from: http://www.tva.gov/power/images/coalart.gif)

A power generation plant usually has a number of units that work together. Generally, a power generation company has a *m*-year (or *m*-month) overhaul system, that is, each time, a unit goes through a major overhaul in turn and every *m* years (or *m*-months) the plant completes an overhaul cycle. The unit which was overhauled most recently would have the highest thermal efficiency and the one close to an overhaul will have the lowest thermal efficiency. Units with higher thermal efficiency will consume less fuel and cause less environmental harm while units with lower thermal efficiency will consume more fuel and lead to higher environmental harm. In the normal operation range, unit thermal efficiency increases (or heat rate decreases) as load increases. The thermal efficiency for each unit is different depending on when the unit is last overhauled, what kind of problems it developed, what modifications it went through, and what operation mode a unit is operating under (such as mill pattern). The optimized loading can be achieved based on the units' thermal efficiency and emission characteristics, that is, heat rate/NO_x vs. load, for a given plant condition.

The generation of electricity from fossil fuel release several contaminants, such as SO_2, NO_x and CO_2, into the atmosphere. Among these contaminants, nitrogen oxides NO_x are contributed largely by the power stations and they are strongly requested to be reduced by the Environmental Protection Agency [119]. In this research, NO_x emission is taken as a selected index for environment conservation. However, the methodology can be easily extended to other contaminates such as CO_2.

There are two objectives for the power generation loading optimization problem. One is to minimize the total heat consumption (fuel consumption) and another is to minimize the total NO_x emission. It is desirable that the unit with higher thermal efficiency (lower heat rate) receives higher workload and the unit with lower thermal efficiency (higher heat rate) receives lower workload.

6.2.2 Specification

Table 6-1 introduces the nomenclature of the power generation loading optimization problem. A specification to these terms is followed.

Table 6-1 Nomenclature of power generation loading optimization

Symbol	Meaning
M_{total}	total power demand by the market, total workload (MW)
$M_{\min}$	lowest workload (MW)
$M_{\max}$	highest workload (MW)
Q	Total NO_x emission for all units at a given load (g/m^3)
F	total units heat consumption (MJ / h)
a	coefficients of the polynomial to heat rate function
b	coefficients of the polynomial to emission curve function
f	unit heat rate, is the heat consumption for generating per unit electricity (KJ/KW.h)
g	output demand constraint function (MW)
h	heat consumption per hour to a unit at a given load (MJ / h)
i	generation unit index (subscript)
k	order of polynomial function (superscript)
n	number of generation unit
P	maximum NO_x emission license limit to each unit (g/m^3)
q	NO_x emission level to a unit at a given load (g/m^3)
r	NO_x emission constraint function (g/m^3)
x	workload allocated to a unit (MW)
δ	minimum error criterion for equality constraint

Specification:

- For a given condition, a unit's heat rate f_i is a function of the unit load x_i which can be expressed by a polynomial format. This function is obtained from field testing and unit modelling. The general expression for the heat rate function for unit i is

$$f_i(x_i) = a_{ik}\, x_i^k + a_{i(k-1)}\, x_i^{(k-1)} + \ldots + a_{i1}\, x_i + a_{i0} \tag{6.1}$$

- A unit's heat consumption h_i at a given load x_i is calculated by

$$h_i = x_i\, f_i(x_i) \tag{6.2}$$

- Each unit has its own NO$_x$ emission curve q. It is generally a linear function in the normal operation range, which is obtained from the field testing and unit modelling.

$$q_i(x_i) = b_{i1}\, x_i + b_{i0} \tag{6.3}$$

- The total heat consumption is the sum of all units' heat consumption, which can be expressed as the following

$$F(x) = \sum_{i=1}^{n} h_i = \sum_{i=1}^{n} x_i f_i(x_i) \tag{6.4}$$

- The total workload is the total power generated by all units at a given time.

$$M_{total} = \sum_{i=1}^{n} x_i \tag{6.5}$$

- The NO$_x$ gas emission for each unit has to be restricted within a license limit *P*.

$$q_i(x_i) \le P \quad (i = 1, 2, \ldots n) \tag{6.6}$$

- The total NO_x gas emission is

$$Q(x_i) = \sum_{i=1}^{n} q_i(x_i) \tag{6.7}$$

- For stable operation, the workload for each unit must be restricted within its lower and upper limits. This is the range where a unit load can be readily adjusted without excessive human intervention. For example, a unit is operating between 60% to 100% load without the need of mill change.

$$M_{i\min} \le x_i \le M_{i\max} \quad (i = 1, 2, \ldots n) \tag{6.8}$$

Figure 6.2 illustrates a multi-unit power generating model.

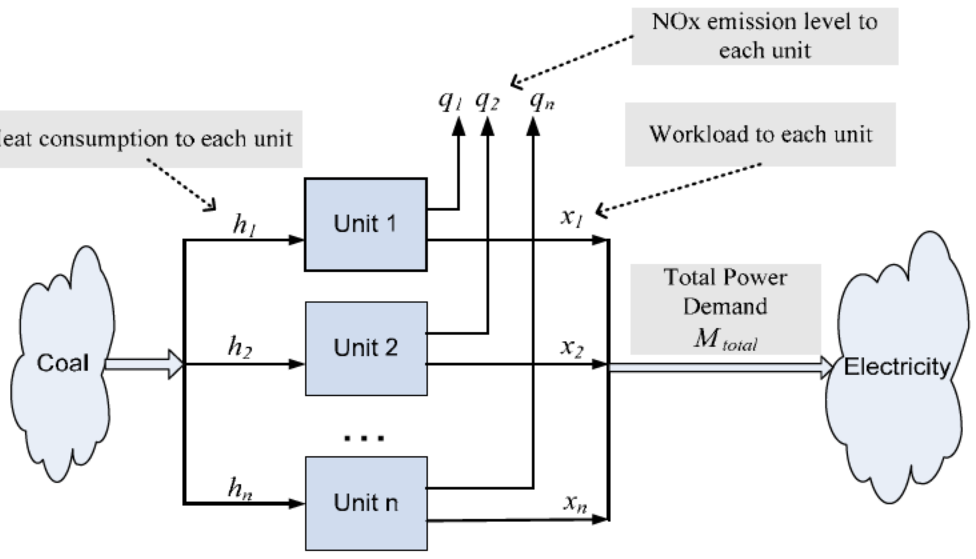


Figure 6.2 A multi-unit power generation model

6.2.3 Formulation

Several constraints should be taken into consideration. The first one is that the total power generated should meet the market demand at a given time. Considering that the data types have to be implemented in double precision, this constraint can be rewritten as

$$g_1(x) = \left| \sum_{i=1}^{n} x_i - M_{total} \right| < \delta \, . \tag{6.9}$$

The second set of constraints is the NO$_x$ gas emission. For countries like Australia, there is an environmental licence limit applied in practice. The licence specifies the maximum amount of NO$_x$ gas emission allowed for each thermal unit. In this case, the constraints can be written as

$$r_i(x) = q_i(x_i) - P \leq 0 \quad (i = 1, 2, \ldots n) \, . \tag{6.10}$$

If there is no environmental licence applied, these constraints can be disregarded.

The third constraint is the unit capacity constraint which can be modelled as the boundary constraint in the optimization.

The objective for the power generation loading optimization is to find the optimal unit load distribution so as to minimize the total heat consumption $F(x)$ and the total NO$_x$ gas emission $Q(x)$. The optimization problem can take two different models – the single objective constrained model and the multi-objective constrained model, as described in Equations (6.11) and (6.12).

Model 1 – Single objective constrained model

$$\begin{cases} \text{Minimize} & F(x) = \sum_{i=1}^{n} h_i = \sum_{i=1}^{n} x_i f_i(x_i) \\ \text{subject to} & g_1(x) = \left|\sum_{i=1}^{n} x_i - M_{total}\right| - \delta \le 0 \\ & r_i(x) = q_i(x_i) - P \le 0 \quad (i = 1, 2, \ldots n) \\ & \\ \text{where} & f_i(x_i) = a_{ik} x_i^{k} + a_{i(k-1)} x_i^{(k-1)} + \ldots + a_{i1} x_i + a_{i0} \\ & M_{i\min} \le x_i \le M_{i\max} \ (i = 1, 2, \ldots n) \end{cases} \tag{6.11}$$

Model 2 – Multi-objective constrained model

$$\begin{cases} \text{Minimize} & F(x) = \sum_{i=1}^{n} h_i = \sum_{i=1}^{n} x_i f_i(x_i) \\ & Q(x) = \sum_{i=1}^{n} q_i(x_i) \\ \text{subject to} & g_1(x) = \left|\sum_{i=1}^{n} x_i - M_{total}\right| - \delta \le 0 \\ & r_i(x) = q_i(x_i) - P \le 0 \quad (i = 1, 2, \ldots n) \\ & \\ \text{where} & f_i(x_i) = a_{ik} x_i^{k} + a_{i(k-1)} x_i^{(k-1)} + \ldots + a_{i1} x_i + a_{i0} \\ & q_i(x_i) = b_{i1} x_i + b_{i0} \\ & M_{i\min} \le x_i \le M_{i\max} \ (i = 1, 2, \ldots n) \end{cases} \tag{6.12}$$

The difference between the Model 1 and Model 2 is that the Model 2 has an additional objective – minimizing NO_x gas emission.

6.3 APPROACHES FOR SINGLE OBJECTIVE CONSTRAINED MODEL

In order to evaluate the algorithm performance, two constraint-handling mechanisms, that is, the multi-objective constraint-handling method and the preserving feasibility constraint-handling method, have been adopted in this application. The following sections describe the algorithms.

6.3.1 Multi-objective Constraint-Handling Method Incorporating with PSO Algorithm

As stated in the earlier chapters, a multi-objective constraint-handling method treats constraints as separate objectives in which a constrained optimization problem can be transformed into a bi-objective problem. The first objective is to optimize the original objective function and the second objective is to minimize

$$\Phi(x) = \sum_{i=1}^{m} \max(0, g_i(x))$$

By adopting this concept, the power generation loading optimization problem in Model 1 can be transformed into

$$\begin{cases} \text{Minimize} \quad F_1(x) = (F(x), \Phi(x)) \\ \text{where} \quad F(x) = \sum_{i=1}^{n} x_i f_i(x_i) \\ \qquad \Phi(x) = \max(0,\ g_1(x)) + \sum_{i=1}^{n} \max\ (0,\ r_i(x)) \\ \qquad g_1(x) = \ \left| \sum_{i=1}^{n} x_i - M_{total} \right| \text{ - } \delta \\ \qquad r_i\ (x)\ =\ q_i(x_i)\ -\ P \quad (i = 1,\ 2, \ldots,\ n) \end{cases} \tag{6.13}$$

The problem in Equation (6.13) becomes a bi-objective unconstrained optimization problem. The method proposed in Chapter 3 can be applied directly. There is no need to mention the algorithm again.

6.3.2 Preserving Feasibility Method Incorporating with PSO Algorithm

The preserving feasibility method [14] assumes that the constraints are all linear and the start points are all feasible. When initializing, particles can be generated within the entire search space but only those which are in feasible space (satisfy all the constraints) are kept for processing. However, although initial particles are all in the feasible space, during flying, they may get out of the feasible space to become infeasible due to improper parameter settings. In order to maintain the population size, it would be better to get these infeasible particles repaired rather than rejecting them. Unfortunately, there are no standard repairing algorithms for every situation. The repairing infeasibility methods lie in their problem dependence [37]. In this research, an infeasible particle is repaired by replacing the infeasible particles with a closer, first-found feasible particle. The algorithms are illustrated Figure 6.3 and Figure 6.4.

Figure 6.3 is a graphical illustration of the repairing algorithm. P_s is an infeasible particle, P_r is a feasible reference particle, Z_1, Z_2… are those attempt particles between P_s and P_r, Z_n is the first-found feasible particle between P_s and P_r. Z_n will be used as a repaired particle of P_s. Figure 6.4 is the repairing algorithm.

Table 6-2 is the Pseudo code of the preserving feasibility constraint-handling method with PSO algorithm. Compared with the original PSO algorithm, two modifications have been made:

1. All particles are repeatedly initialized until they are feasible, that is, to satisfy all constraints.

2. During flying (iteration), if particles are not feasible, repair them to be feasible. Then calculate the fitness.

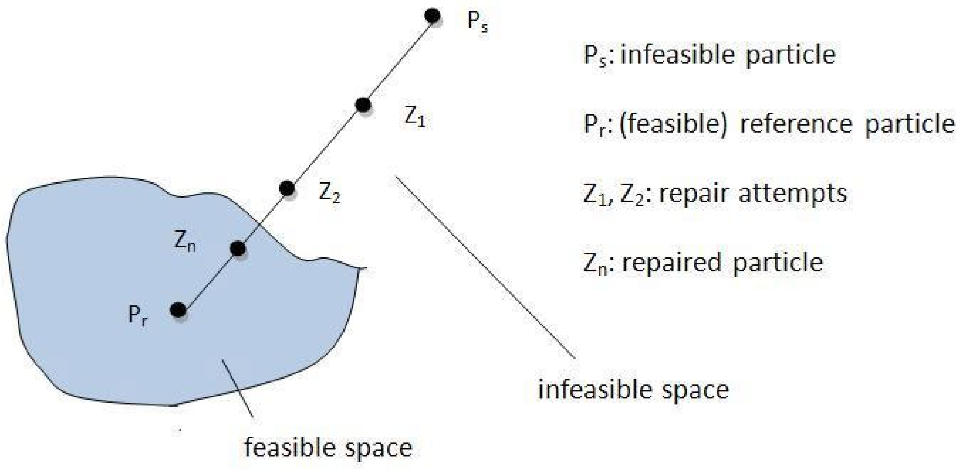

P_s
Z_1
Z_2
Z_n
P_r
P_s: infeasible particle
P_r: (feasible) reference particle
Z_1, Z_2: repair attempts
Z_n: repaired particle
infeasible space
feasible space

Figure 6.3 The graphic illustration of the repairing algorithm

```
1.  Select a reference particle $P_r$.
2.  Create a sequence of candidate particles
    $Z_i$ between $P_r$ and $P_s$:
              $Z_i = a_i P_s + (1 - a_i) P_r$
    where $0 < a_i < 1$, is generated at random
    The process stops when the first feasible $Z_n$ is found.
3.  With $Z_n$ the repaired particle of $P_s$,
   go back to the PSO algorithm for fitness calculation
```

Figure 6.4 The infeasibility repairing algorithm

Table 6-2 Pseudo code of the preserving feasibility constraint-handling method with PSO algorithm

```
01: For i = 0 to population size
02:     Do
03:             Initialize particle
04:     While particle is not feasible
05: End For
06: For each particle
07:     Calculate fitness F
08:     Calculate constraint violations Φ
09:     Set current locations as personal best locations
10:     Set local best location for each particle according selection rule
11: End For
12: Do
13:      For each particle
14:             Calculate new velocity by PSO formula
15:             Calculate new location by PSO formula
16:             Repair particle if not feasible
17:             Update  personal best location according to selection rule
18:      End For
19:  Set local best location for each particle according to selection rule
20:  End Do
```

6.4 APPROACH FOR MULTI-OBJECTIVE CONSTRAINED MODEL

Again, the feasibility of a solution can be evaluated by its constraint violations $\Phi = \sum_{i=1}^{m} \max(0, g_i(x))$. If $\Phi \le \varepsilon$, where ε is the tolerance allowed for feasibility, the solution is feasible. Otherwise, ε is a positive number indicating how far the solution is from the feasible region. The smaller the constraint violation value, the closer the solution to the feasible region. Model 2 in Equation (6.12) can be rewritten as

$$\begin{cases} \text{Minimize} & F(x) = \sum_{i=1}^{n} h_i = \sum_{i=1}^{n} x_i f_i(x_i) \\ & Q(x) = \sum_{i=1}^{n} q_i(x_i) \\ \text{subject to} & \Phi(x) = \max(0,\ g_1(x)) + \sum_{i=1}^{n} \max\ (0,\ r_i(x)) \ \le \varepsilon \\ \text{where} & g_1(x) = \ |\sum_{i=1}^{n} x_i - M_{total}| \ - \delta \\ & r_i\ (x) = q_i(x_i) - P \qquad (i = 1,\ 2, \ldots,\ n) \end{cases} \tag{6.14}$$

The problem in Equation (6.14) is a multi-objective (two objectives) constrained (one constraint) optimization problem. The method proposed in Chapter 4 can be applied directly.

6.5 SIMULATION RESULTS AND DISCUSSION

6.5.1 Unit Heat Rates and Unit Gas Emission Curves

A local power plant has four 360MW and a total generation capacity of 1440MW. It has a four-year overhaul system, that is, each year, a unit goes through a major overhaul in turn and every four year the plant completes an overhaul cycle.

The boundary constraints $M_{\min}$ and $M_{\max}$ for each unit are 220 (MW) and 360 (MW). The total load output of the power station ranges from $4 \times 220 = 880$ (MW) as the minimum to $4 \times 360 = 1440$ (MW) as the maximum. It would be better to simulate a series of output (a series of M_{total}) in order to allow the power plant to choose from the optimal results according to the market demand.

The heat rate functions and the NO_x emission functions for the four generator units are provided from a local power plant setting. The heat rate functions are in the polynomial format with the power of two. The NO_x emission functions are linear. Table 6-3 lists the sample functions. These functions can be modified when the units' performance are changed. Due to commercial reasons, the functions have been slightly modified.

Table 6-3 Unit heat rate and NO_x emission functions

Unit No.	Heat Rate	NO_x Emission
1	$f(x_1) = 0.0023\,x_1^2 - 3.7835\,x_1 + 9021.7$	$q(x_1) = 0.0036\,x_1 - 0.1717$
2	$f(x_2) = 0.0238\,x_2^2 - 9.7773\,x_2 + 9432.6$	$q(x_2) = 0.0031\,x_2 - 0.0226$
3	$f(x_3) = 0.0187\,x_3^2 - 5.3678\,x_3 + 10240.0$	$q(x_3) = 0.0036\,x_3 - 0.1252$
4	$f(x_4) = 0.0120\,x_4^2 - 5.7450\,x_4 + 9231.7$	$q(x_4) = 0.0039\,x_4 - 0.1706$

6.5.2 Parameter Setting

The minimum error criterion for equality constraint is selected as $\delta = 1.0E-03$. The NO$_x$ license limits P is 1.3 g/m^3. PSO neighbourhood topology is set to static ring topology with the neighbour size of 2. PSO parameters are: $w = 0$; $c_1 = c_2 = 2$; $\chi = 0.63$; $V_{\max} = 0.5 \times (M_{\max} - M_{\min})$; population size is 40 for Model 1 and 500 for Model 2; the maximum iteration is set to 10,000 for Model 1 and 1000 for Model 2. The feasibility tolerance allowed $\varepsilon = 1.0E-08$, that is, if a total amount of constraint violation $\Phi \leq \varepsilon$, the solution is considered feasible.

For each total load output M_{total}, the program runs ten times with the best solution recorded. For Model 1, the best solution means the feasible solution that has the lowest heat consumption. For Model 2, the best solution is a set of Pareto-optimal solutions that has a small "Spacing/Spread" (refer to Chapter 4) value.

6.5.3 Results

6.5.3.1 Results for Single Objective Constrained Model (Model 1)

These results are generated by the multi-objective constraint-handling method incorporating with PSO algorithm. The preserving feasibility constraint-handling approach will be discussed in section 6.5.4.

Table 6-4 and Figure 6.5 present the simulation results to the whole range of the generation capacity. For each total output demand, the optimal workloads to the four generators have been found based on their efficiency functions as listed in Table 6-3. After optimization, the unit with higher thermal efficiency will receive a higher workload (such as Unit 1) while the unit with lower thermal efficiency will receive a lower workload (such as Unit 3). In practice, when the total output load changes, the optimal load allocation can be found from these data.

Table 6-4 Optimized workload distribution for Model 1

M_{total} (MW)	Unit 1 (MW)	Unit 2 (MW)	Unit 3 (MW)	Unit 4 (MW)
880	220.0000	220.0000	220.0000	220.000
900	235.1297	222.3830	220.0131	222.4746
950	273.1694	220.7023	220.0023	236.1254
1000	326.7896	230.9750	220.0002	222.2353
1050	359.6199	224.8249	221.3988	244.1572
1100	359.9975	227.8653	221.2147	290.9222
1150	359.8885	235.5493	221.1815	333.3800
1200	359.9999	269.2535	247.5140	323.2334
1250	359.9937	326.3954	221.1032	342.5064
1300	358.9863	339.9221	241.0938	359.9969
1350	359.9939	325.3112	305.7268	358.9679
1400	358.4181	353.5454	328.0362	359.9999
1440	360.0000	360.0000	360.0000	360.0000

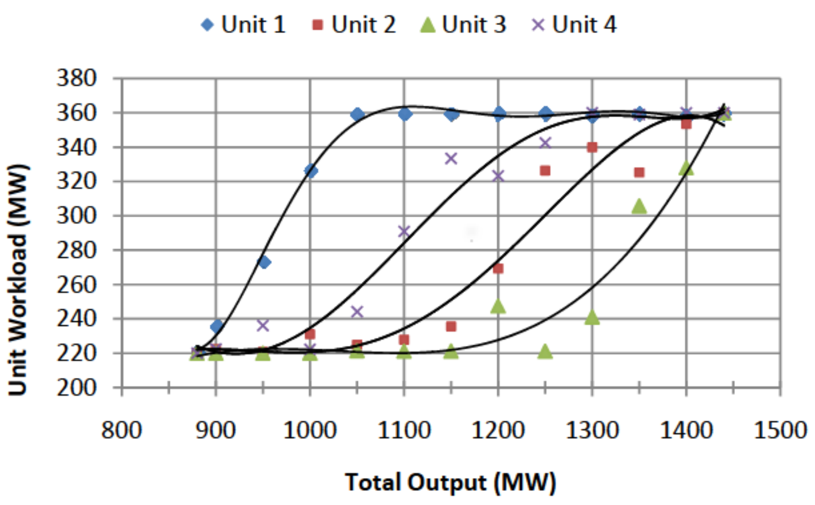


Figure 6.5 Optimal unit loading distribution for the whole range of the generation capacity for model 1

6.5.3.2 Results for Multi-objective Constrained Model (Model 2)

For a specific output demand, the optimization result should be a set of Pareto-optimal solutions. Figure 6.6 and Figure 6.7 present the simulation results for $M_{total} = 1000$ MW and $M_{total} = 1200$ MW.

It needs to be mentioned that the objective F (Heat Consumption) conflicts with and the objective Q (NO_x emission). NO_x is a product of combustion. NO_x increases as combustion temperature and oxygen increases. While the better combustion consumes less fuel and produces better efficiency and lower heat rate; and better combustion generates high NO_x levels as the flame would be hotter and oxygen would be more.

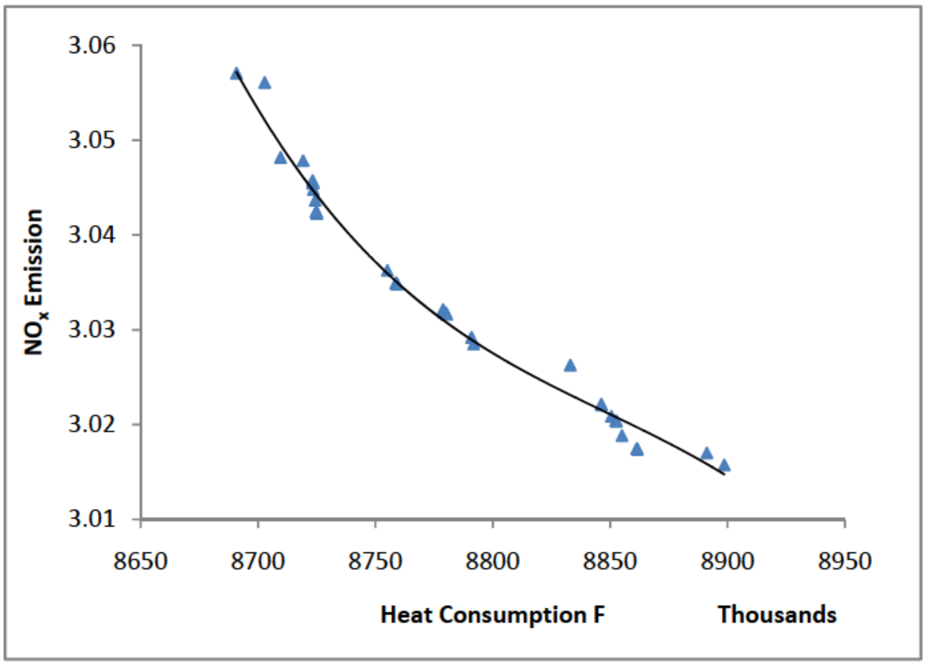


Figure 6.6 Simulated Pareto-front for Model 2 ($M_{total} = 1000$ MW)

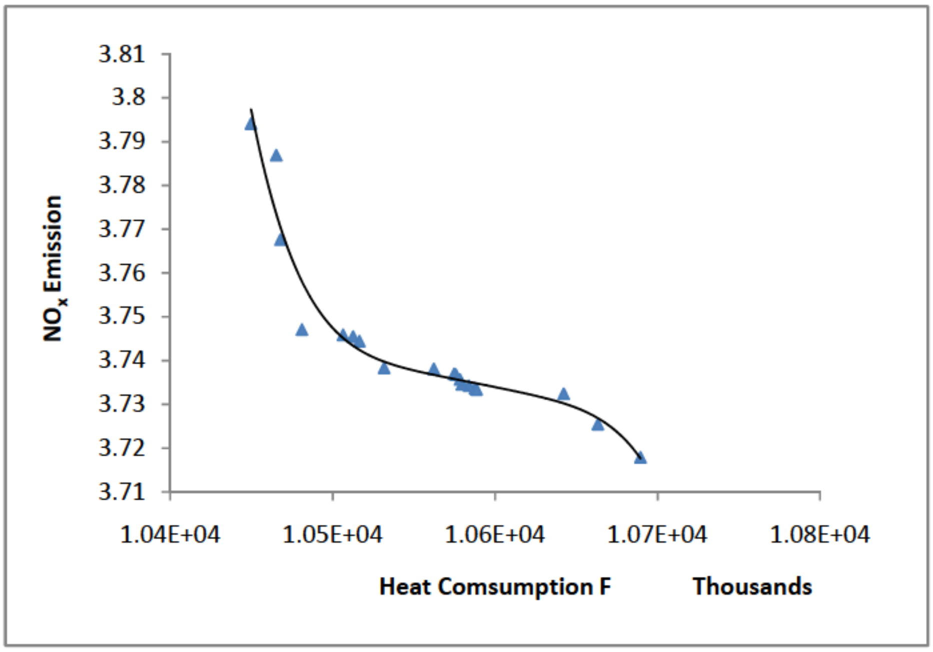


Figure 6.7 Simulated Pareto-front for Model 2 ($M_{total} = 1200$ MW)

6.5.4 Discussion

In order to see the significance of the loading optimization, the money saving from the optimization can be calculated. Heat consumption can be obtained from the objective function for an average allocation (before optimization applied) and an optimized allocation (after the optimization). The heat consumption saving is calculated for comparing the difference between the two. The formula is:

$$H_{saving} = \sum_{i=1}^{4} x_{avg} f_i(x_{avg}) - \sum_{i=1}^{4} x_i f_i(x_i) \tag{6.13}$$

where the $x_{avg} = M_{total} / 4$, the f_i is the heat rate curve listed in Table 6-3.

The heat consumption saving can be calculated by using fuel heating value or the calorific value and the price of fuel. Suppose the fuel (coal) price is $\cos t = \$30$ per ton, the calorific value $c = 26$ MJ/kg, the annual money saving from the optimization can be calculated by

$$M_{saving} = \frac{H_{saving}}{c \times 1000} \times \cos t \times 24 \times 360 \tag{6.14}$$

The result is illustrated in Figure 6.8

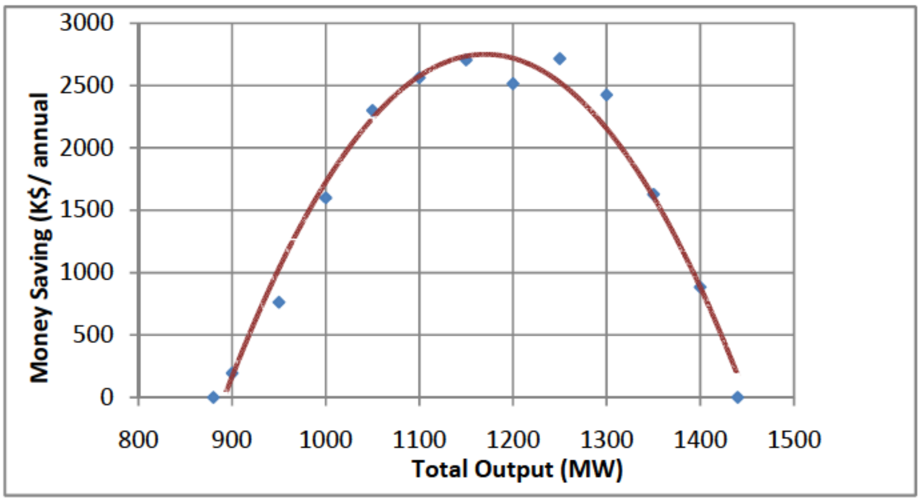


Figure 6.8 Annual money saving from loading optimization
(Calorific value = 26 MJ / kg, fuel price = $30 /per ton)

The curve in Figure 6.8 indicates that most benefits from load optimization are made around 1100MW-1300MW in excess of annual fuel saving of two million dollars while no gain is obtained on minimum and maximum loading conditions, which is logical as no options for loading at both ends. In reality, it is impossible to always operate the plant in such a desirable way, that is, we cannot guarantee all four units keep running for a whole year without stopping. Assume there is a 50% chance of possible loading optimization, the benefits will be halved and fuel savings will be around one million dollars per year.

In order to compare the two constraint-handling methods for Model 1, two experiments have been conducted to evaluate the computation time for each individual run. PSO parameters for both approaches are the same. The 40 particles, 10000 maximum iterations have been used for both experiments. Based on ten independent runs, the minimum time, maximum time and the average time spent for $M_{total} = 1000\ \text{MW}$ are listed in Table 6-5.

Table 6-5 Time spent for two constraint-handling approaches for Model 1
(Based on 10 independent runs)

CPU time spent	*Approach I (ms)	**Approach II (ms)
Minimum	31	3016
Maximum	156	4204
Average	68.9	3925.3

* Multi-Objective Constraint-Handling with PSO
** Preserving Feasibility Constraint-handling method with PSO

Table 6-5 demonstrates that the multi-objective based constraint-handling method is much faster than the preserving feasibility method with PSO. The main reason is that the preserving feasibility approach assumes all particles starting at the feasible space which require a long initialization process. In other words, the evolution will not start until all particles are in the feasible space. It may be impractically too long or impossible for the problems that have large search spaces and with small feasible spaces. The multi-objective constraint-handling approach, however, does not require the particles to be in the feasible space at the beginning. The initialization does not need to check if the particles satisfy all constraints which make the initialization easier and faster.

Regarding to the two models, if there is an environment licence limit applied, either Model 1 or Model 2 will suffice. Otherwise Model 2 can be adopted.

6.6 SUMMARY

Power generation unit efficiency will be of greater practical importance in the coming pollution constrained economy in terms of fuel saving and minimizing environmental harm. This chapter has presented a real world application - Power Generation Unit Loading Optimization using PSO algorithm.

Based on the problem description and specification, the two optimization models, that is, the single objective constrained model (Model 1) and the multi-objective constrained model (Model 2), have been presented. The multi-objective constraint-handling method incorporating with PSO algorithm (proposed in Chapter 3) has been adopted for the single objective constrained model and the selection rules based constraint-handling method with PSO algorithm (proposed in Chapter 4) has been adopted for the multi-objective constrained model.

A simulation to a four-unit coal-fired local power plant has been conducted. The simulation results reveal the capability, effectiveness and efficiency of applying the proposed approaches in the power industry. The simulation results have also demonstrated that the room for loading optimization is significant.

In order to compare the two constraint-handling methods, two experiments have been conducted to evaluate the computation time. The experiment results have demonstrated that the multi-objective constraint-handling based approach is more efficient than the preserving feasibility constraint-handling approach in terms of CPU time consumed. The main reason is that the preserving feasibility approach assumes all particles starting at the feasible space which require a long initialization process. Since the multi-objective constraint-handling method has no problem-dependent parameters like those applied in the penalty function based constraint-handling approach, it makes it easier to extend to a wide variety of applications.

Regarding to the two models, if there is an environment licence limit applied in practice, either Model 1 or Model 2 will suffice. Otherwise Model 2 can be adopted subject to disregard the constraints on environment licence limit.

Chapter 7

CONCLUSIONS AND FUTURE RESEARCH

7.1 CONCLUSIONS

The population-based evolutionary algorithms have the ability to capture multiple optimal solutions in one single simulation run which leads to a high computing performance. The flexible representations make the algorithms appropriate to be used in a wide variety of problem domains.

The original versions of the EAs have no mechanisms to deal with constraints. Among a number of constraint-handling methods available, the multi-objective constraint-handling method has a number of advantages. Firstly, it has no problem-dependent parameters like those applied in the penalty function based approaches, which makes the approach applicable for a wide variety of applications. Secondly, the multi-objective constraint-handling method does not require that all individuals start evolving at the feasible region like those applied in the preserving feasibility methods, which makes the initialization a lot faster and ensures that the evolution will start. Thirdly, the multi-objective constraint-handling method does not reject the infeasible solutions during evolution. Recent research shows that maintaining infeasible solutions during evolution can improve the computing performance [51] and can transform some EA-hard problems into EA-easy problems [52].

To tackle single objective constrained optimization problems, a multi-objective constraint-handling method incorporating a dynamic neighbourhood PSO algorithm has been proposed in this thesis. A modified PSO algorithm for tackling multi-objective constrained optimization problems has also been proposed. The simulation results for the numerical benchmark functions have demonstrated the proposed approaches are effective and efficient in finding the consistent quality solutions.

As applications, three well-known engineering design optimization problems and the power generation loading optimization problem have been investigated. The simulation results to the applications have revealed the capability, effectiveness and efficiency of applying the proposed approaches in constrained optimization.

In summary, the overall goal this thesis, that is, to investigate the PSO algorithm in constrained optimization and its application in power generation, has been successfully achieved.

The major contributions and results in this thesis can be summarized in the following:

A multi-objective constraint-handling method incorporating PSO algorithm has been proposed. The simulation results to the thirteen well-known benchmark functions demonstrate that the proposed approach is effective and efficient in most (eleven out of thirteen) functions. The algorithm did not receive satisfactory results for two out of thirteen test functions G1 and G2. It seems that the proposed approach needs to be improved for solving high dimensional optimization problems. This may be explained by the "No free lunch" theorem [99].

- **A novel performance-based dynamic neighbourhood topology has been proposed.** The proposed performance-based dynamic neighbourhood has proved to be able to converge faster than the static neighbourhood topology. The method has potential to be adopted in those applications that are computational intensive.
- **An effective multi-objective PSO algorithm has been proposed.** Most existing multi-objective PSO proposals do not address the constraints. Integrating constraint-handling mechanisms with multi-objective PSO is a challenging topic. This thesis is one of the few attempts to integrate constrain-handling methods with the multi-objective PSO algorithms. The simulation results to the four constrained multi-objective optimization problems demonstrated the proposed approach is able to find the Pareto-optimal solutions effectively.
- **As a variant of the proposed approach, a goal-oriented multi-objective constraint-handling method via PSO algorithm has been introduced for tackling those problems that have predefined goals.** Since the goals can be used as the exit criteria, each particle does not need to go through a whole iterations. This makes the computation more cost-effective.
- **A real world application, Power Generation Loading Optimization gas been presented.** The power generation loading optimization problem is of practical importance in the evolving pollution constrained power industry in terms of fuel saving and minimizing environmental harm. This thesis has presented two optimization models derived from the practice of power industry, and applies the

methods proposed in the thesis in solving these problems. The simulation results have shown that there is a large room for loading optimization. The project has great commercialisation potential.

7.2 FUTURE RESEARCH OUTLOOK

In concluding this thesis, an outlook on future research can be foreseen as follows.

- Except for GAs, algorithms like PSO, DE and ACO have not been well studied in solving constrained multi-objective optimization problems. The multi-objective constrained optimization is a fertile area for future research.
- The multithreading technique makes parallel execution possible which leads to high performance computation. Using the multithreading technique to implement other parallel processes for EAs, or distributing the parallel processing tasks to distributed hardware for large computational-intensive applications will be an interesting topic.
- Extending the proposed method in this thesis for tackling goal programming problems should yield useful outcomes. When a goal programming problem is regarded as a constraint satisfaction problem, the approach proposed in this thesis may be used. The detailed model needs to be identified in the future.
- For the power generation loading optimization application, minimizing emissions from other contaminants such as SO_2, and CO_2 should be taken into consideration. The current work undertaken only considered the emission from NO_x.

APPENDIX I CONSTRAINED NUMERICAL OPTIMIZATION TEST FUNCTIONS

G1: *Minimize*

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

s.t.

$$g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \le 0,$$
$$g_2(x) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \le 0,$$
$$g_3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \le 0,$$
$$g_4(x) = -8x_1 + x_{10} \le 0,$$
$$g_5(x) = -8x_2 + x_{11} \le 0,$$
$$g_6(x) = -8x_3 + x_{12} \le 0,$$
$$g_7(x) = -2x_4 - x_5 + x_{10} \le 0,$$
$$g_8(x) = -2x_6 - x_7 + x_{11} \le 0,$$
$$g_9(x) = -2x_8 - x_9 + x_{12} \le 0.$$

- Number of variables: 13 variables.
- Search Space: $0 \le x_i \le u_i, \quad i = 1,2,...,13, \quad u = (1,1,...,1,100,100,100,1)$
- The global minima: $x^* = (1,1,...,1,3,3,3,1), f(x^*) = -15.$

G2: *Maximize*

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

s.t.

$$g_1(x) = -\prod_{i=1}^{n} x_i + 0.75 \le 0,$$
$$g_2(x) = \sum_{i=1}^{n} x_i - 7.5n \le 0.$$

- Number of variables: n variables.

- Search Space: $0 \le x_i \le 10, \quad i = 1, 2, ...n.$
- Best known: at $n = 20, f(x^*) = 0.803619.$

G3: *Maximize*

$$f(x) = (\sqrt{n})^n \prod_{i=1}^{n} x_i,$$

s.t.

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

- Number of variables: *n* variables.
- Search Space: $0 \le x_i \le 1, \quad i = 1, 2, ...n.$
- The global minima: at *n =10*, $x^* = (1/n^{0.5}, ...1/n^{0.5}), f(x^*) = 1.$

G4: *Minimize*

$$f(x) = 5.3578547 x_3^2 + 0.8356891 x_1 x_5 + 37.293239 x_1 - 40792.141,$$

s.t.

$$g_1(x) = 85.334407 + 0.0056858 x_2 x_5 + 0.0006262 x_1 x_4 - 0.0022053 x_3 x_5 - 92 \le 0,$$
$$g_2(x) = -85.334407 - 0.0056858 x_2 x_5 - 0.0006262 x_1 x_4 + 0.0022053 x_3 x_5 \le 0,$$
$$g_3(x) = 80.51249 + 0.0071317 x_2 x_5 + 0.0029955 x_1 x_2 + 0.0021813 x_3^2 - 110 \le 0,$$
$$g_4(x) = -80.51249 - 0.0071317 x_2 x_5 - 0.0029955 x_1 x_2 - 0.0021813 x_3^2 + 90 \le 0,$$
$$g_5(x) = 9.300961 + 0.0047026 x_3 x_5 + 0.0012547 x_1 x_3 + 0.0019085 x_3 x_4 - 25 \le 0,$$
$$g_6(x) = -9.300961 - 0.0047026 x_3 x_5 - 0.0012547 x_1 x_3 - 0.0019085 x_3 x_4 + 20 \le 0.$$

- Number of variables: 5 variables.
- Search Space: $l_i \le x_i \le u_i, \ i = 1, ..., 5,,$

$$l = (78, 33, 27, 27, 27), \quad u = (102, 45, 45, 45, 45).$$

- The global minima:

$$x^* = (78, 33, 29.995, 45, 36.7758), \quad f(x^*) = -30665.539.$$

G5: *Minimize*

$$f(x) = 3x_1 + 10^{-6} x_1^3 + 2x_2 + \frac{2}{3} \times 10^{-6} x_2^3,$$

s.t.

$$g_1(x) = x_3 - x_4 - 0.55 \le 0,$$
$$g_2(x) = x_4 - x_3 - 0.55 \le 0,$$

$$h_1(x) = 1000\ [\sin(-x_3 - 0.25) + \sin(-x_4 - 0.25)] + 894.8 - x_1 = 0,$$
$$h_2(x) = 1000\ [\sin(x_3 - 0.25) + \sin(x_3 - x_4 - 0.25)] + 894.8 - x_2 = 0,$$
$$h_3(x) = 1000\ [\sin(x_4 - 0.25) + \sin(x_4 - x_3 - 0.25)] + 1294.8 = 0.$$

- Number of variables: 4 variables.
- Search Space: $l_i \le x_i \le u_i,\ \ i = 1,...,4,$

$$l = (0, 0, -0.55, -0.55),\ \ u = \left(1200,\ 1200,\ 0.55,\ 0.55\right).$$

- Best known:

$$x^* = (679.9453, 1026, 0.118876, -0.3962336),\quad f(x^*) = 5126.4981.$$

G6: *Minimize*

$$f(x) = (x_1 - 10)^3 + (x_2 - 20)^3,$$

s.t.

$$g_1(x) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \le 0,$$
$$g_2(x) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \le 0.$$

- Number of variables: *2* variables.
- Search Space: $l_i \le x_i \le 100,\ \ i = 1, 2\ \ and\ l = (13, 0).$
- The global minima: $x^* = (14.095, 0.84296),\ f(x^*) = -6961.81388.$

G7: *Minimize*

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

s.t.

$$g_1(x) = 4x_1 + 5x_2 - 3x_7 + 9x_8 - 105 \le 0,$$
$$g_2(x) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \le 0,$$
$$g_3(x) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \le 0,$$
$$g_4(x) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \le 0,$$
$$g_5(x) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \le 0,$$
$$g_6(x) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \le 0,$$
$$g_7(x) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \le 0,$$
$$g_8(x) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \le 0,$$

- Number of variables: 10 variables.
- Search Space: $-10 \le x_i \le 10,\ i = 1,2,...,10.$
- The global minima:

$$x^* = (2.171996,\ 2.363683,\ 8.773926,\ 5.095984,\ 0.9906548,\ 1.430574, 1.321644,\ 9.828726,\ 8.280092,\ 8.375927),$$
$$f(x^*) = 24.3062091.$$

G8: *Maximize*

$$f(x) = \frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)},$$

s.t.

$$g_1(x) = x_1^2 - x_2 + 1 \le 0,$$
$$g_2(x) = 1 - x_1 + (x_2 - 4)^2 \le 0.$$

- Number of variables: 2 variables.
- Search Space: $0 \le x_i \le 10,\ i = 1,2.$
- The global minima: $x^* = (1.2279713,\ 4.2453733),\quad f(x^*) = 0.095825.$

G9: *Minimize*

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

s.t.

$$g_1(x) = 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 - 127 \le 0,$$
$$g_2(x) = 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 - 282 \le 0,$$
$$g_3(x) = 23x_1 + x_2^2 + 6x_6^2 - 8x_7 - 196 \le 0,$$
$$g_4(x) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \le 0.$$

- Number of variables: 7 variables.
- Search Space: $-10 \le x_i \le 10,\ i = 1,2,...,7.$
- The global minima:

$$x^* = (2.330499,\ 1.951372,\ -0.4775414, 4.365726,\ -0.6244870,\ 1.038131,\ 1.594227),$$
$$f(x^*) = 680.6300573.$$

G10: *Minimize*

$$f(x) = x_1 + x_2 + x_3,$$

s.t.

$$g_1(x) = -1 + 0.0025(x_4 + x_6) \le 0,$$
$$g_2(x) = -1 + 0.0025(-x_4 + x_5 + x_7) \le 0,$$
$$g_3(x) = -1 + 0.01(-x_5 + x_8) \le 0,$$
$$g_4(x) = 100x_1 - x_1x_6 + 833.33252x_4 - 83333.333 \le 0,$$
$$g_5(x) = x_2x_4 - x_2x_7 - 1250x_4 + 1250x_5 \le 0,$$
$$g_6(x) = x_3x_5 - x_3x_8 - 2500x_5 + 1250000 \le 0.$$

- Number of variables: 8 variables.
- Search Space: $l_i \le x_i \le u_i, \ i = 1,...,8,$

$$l = (100,\ 1000,\ 1000,\ 10,\ 10,\ 10,\ 10,\ 10),$$
$$u = (10000,\ 10000,\ 10000,\ 1000,\ 1000,\ 1000,\ 1000,\ 1000).$$

- The global minima:

$$x^* = (579.3167,\ 1359.943, 5110.071,\ 182.0174,$$
$$295.5985,\ 217.9799,\ 286.4162,\ 395.5979),$$
$$f(x^*) = 7049.3307.$$

G11: *Minimize*

$$f(x) = x_1^2 + (x_2 - 1)^2,$$

s.t.

$$h_1(x) = x_2 - x_1^2 = 0.$$

- Number of variables: 2 variables.
- Search Space: $-1 \le x_i \le 1, \ i = 1,2.$
- The global minima: $x^* = \pm(1/2^{0.5}, 1/2), \ f(x^*) = 0.75.$

G12: *Maximize*

$$f(x) = 1 - 0.01\,[(x_1 - 5)^2 + (x_2 - 5)^2 + (x_3 - 5)^2],$$

s.t.

$$g_{i,j,k}(x) = (x_1 - i)^2 + (x_2 - j)^2 + (x_3 - k)^2 - 0.0625 \le 0,$$
where $i, j, k = 1,2,...,9.$

- Number of variables: 3 variables.
- Search Space: $0 \le x_i \le 10, \;\; i = 1,2,3.$
- The global minima: $x^* = (5,5,5), \quad f(x^*) = 1.$

G13: *Minimize*

$$f(x) = e^{x_1 x_2 x_3 x_4 x_5},$$

s.t.

$$h_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0,$$
$$h_2(x) = x_2 x_3 - 5 x_4 x_5 = 0,$$
$$h_3(x) = x_1^3 + x_2^3 + 1 = 0.$$

- Number of variables: 5 variables.
- Search Space: $l_i \le x_i \le u_i, \;\; i = 1,\ldots,5,$

  $$l = -u, \;\; u = (2.3,\; 2.3,\; 3.2,\; 3.2,\; 3.2).$$

- The global minima:

  $$x^* = \left(-1.717143,\; 1.595709, 1.827247,\; -0.7636413,\; -0.763645\right),$$
  $$f(x^*) = 0.0539498.$$

APPENDIX II ENGINEERING DESIGN OPTIMIZATION PROBLEMS

E01: Welded Beam Design Problem

As shown in Figure I, a welded beam is designed for minimum cost subject to constraints of shear stress (τ), bending stress in the beam (σ), buckling load on the bar (P_c), end deflection of the beam (δ), and side constraints. There are four design variables: the thickness of the weld $h = x_1$, the length of the welded joint $l = x_2$, the width of the beam $t = x_3$ and the thickness of the beam $b = x_4$.

Please note the welded beam problem included in this chapter is not exactly the same as the original version proposed by Reklaitis et al in 1983 [120]. There are five constraints in the original version. For some reason, many researchers have studied another version of this problem as following with seven constraints. We include this version for comparison purposes.

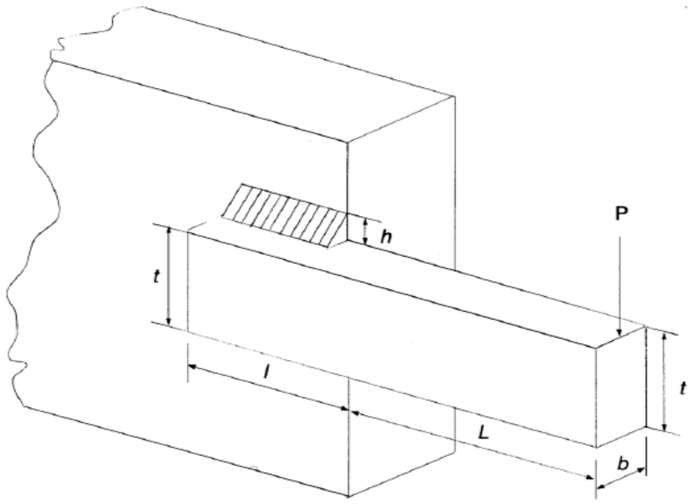


Figure I Welded beam design

The formal statement of the problem is the following:

minimize

$$f(x) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14.0 + x_2)$$

subject to

$$\begin{aligned} g_1(x) &= \tau(x) - \tau_{\max} \le 0 \\ g_2(x) &= \sigma(x) - \sigma_{\max} \le 0 \\ g_3(x) &= x_1 - x_4 \le 0 \end{aligned}$$

$$\begin{aligned} g_4(x) &= 0.10471 x_1^2 + 0.04811 x_3 x_4 (14 + x_2) - 5 \le 0 \\ g_5(x) &= 0.125 - x_1 \le 0 \\ g_6(x) &= \delta(x) - \delta_{\max} \le 0 \\ g_7(x) &= P - P_c(x) \le 0 \end{aligned}$$

where:

$$\tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau'' \frac{x_2}{2R} + (\tau'')^2}$$

$$\tau' = \frac{P}{\sqrt{2} x_1 x_2}$$

$$\tau'' = \frac{MR}{J}, \quad M = P\left(L + \frac{x_2}{2}\right)$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}$$

$$J = 2\left\{ \frac{x_1 x_2}{\sqrt{2}} \left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2} \right)^2 \right] \right\}$$

$$\delta(x) = \frac{4PL^3}{E x_3^3 x_4}, \qquad \sigma(x) = \frac{6PL}{x_4 x_3^2}$$

$$P_c(x) = \frac{4.013\sqrt{(E G x_3^2 x_4^6)/36}}{L^2} \left(1 - \frac{x_3}{2L} \sqrt{\frac{E}{4G}} \right)$$

$$P = 6000\ lb,\ L = 14\ in.,\ E = 30 \times 10^6\ psi,\ G = 12 \times 10^6\ psi$$

$$\tau_{\max} = 13{,}600\ psi,\ \sigma_{\max} = 30{,}000\ psi,\ \delta_{\max} = 0.25\ in.$$

The ranges for the design variables are given as follows:

$$0.1 \le x_1 \le 2.0, \quad 0.1 \le x_2 \le 10,$$
$$0.1 \le x_3 \le 10, \quad 0.1 \le x_4 \le 2.0.$$

E02: Pressure Vessel Design Problem

The pressure vessel design problem, shown in Figure II, is a cylindrical vessel capped at both ends by hemispherical heads. The objective is to minimize the total cost, including the cost of the materials forming the welding. There are four design variables: Thickness of the shell $T_s = x_1$, thickness of the head $T_h = x_2$, the inner radius $R = x_3$, and the length of the cylindrical section of the vessel $L = x_4$, T_s and T_h are discrete values which are integer multiples 0.0625 (inch), in accordance with the available thickness of rolled steel plates, R and L are continuous.

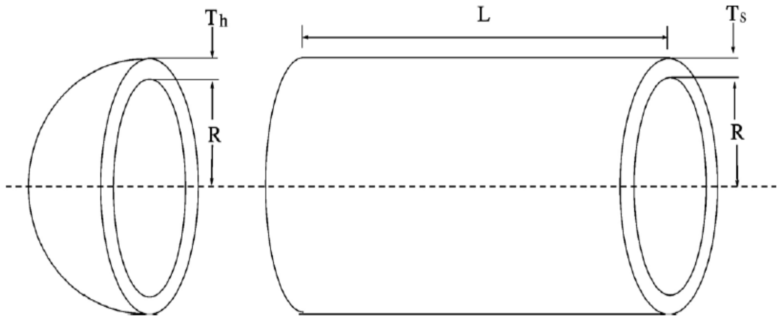


Figure II Pressure vessel design

The optimization problem can be expressed as follows:

minimize

$$f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$$

subject to

$$g_1(x) = 0.0193x_3 - x_1 \le 0$$
$$g_2(x) = 0.00954x_3 - x_2 \le 0$$
$$g_3(x) = 1296000 - \pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 \le 0$$
$$g_4(x) = x_4 - 240 \le 0$$

where the design variables have to be in the following ranges:

$0.0625 \le x_1 \le 6.1875, \quad 0.0625 \le x_2 \le 6.1875,$
$10 \le x_3 \le 200, \quad 10 \le x_4 \le 200.$

E03: Spring Design Problem

As shown in Figure III, this problem consists of minimizing the weight of a tension/compression spring, subject to constraints of minimum deflection, shear stress, surge frequency, and limits on outside diameter and on design variables. The three design variables are: the wire diameter $d = x_1$, the mean coil diameter $D = x_2$ and the number of active coils $N = x_3$.

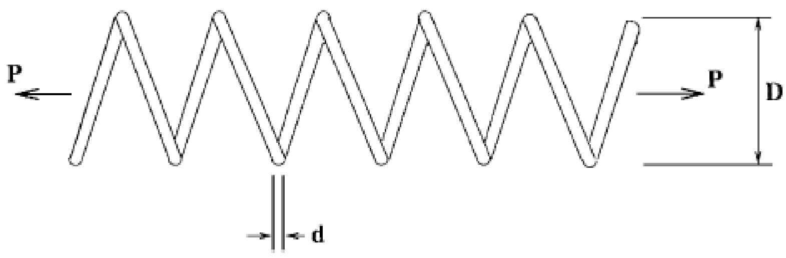


Figure III Spring design

The formal statement of the problem is as follows:

minimize

$$f(x) = (x_3 + 2)x_2 x_1^2$$

subject to

$$g_1(x) = 1 - \frac{x_2^3 x_3}{71785 x_1^4} \le 0$$

$$g_2(x) = \frac{4x_2^2 - x_1 x_2}{12566(x_2 x_1^3 - x_1^4)} + \frac{1}{5108 x_1^2} - 1 \le 0$$

$$g_3(x) = 1 - \frac{140.45 x_1}{x_2^2 x_3} \le 0$$

$$g_4(x) = \frac{x_2 + x_1}{1.5} - 1 \le 0$$

The boundaries of the design variables are as follows:

$$0.05 \le x_1 \le 2, \quad 0.25 \le x_2 \le 1.3, \quad 2 \le x_3 \le 15.$$

BIBLIOGRAPHY

[1] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*: John Wiley & Sons, 2001.

[2] K. Deb, *Optimization for Engineering Design: Algorithms and Examples*. New Delhi: Prentice-Hall, 1995.

[3] D. Buche, "Multi-Objective Evolutionary Optimization of Gas Turbine Components," Zurich: PhD thesis, Swiss Federal Institute of Technology, 2003.

[4] X. Yao, "Evolutionary Computation, A Gentle Introduction," in *Evolutionary Optimization*, R. Sarker, M. Mohammadian, and X. Yao, Eds.: Kluwer Academic Publishers, 2002, pp. 27-53.

[5] J. C. Spall, *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*: Wiley, 2003.

[6] Wikipedia, "Global Optimization," http://en.wikipedia.org/wiki/Global_optimization, Retrieved on 2008-8-13.

[7] A.-R. Hedar and M. Fukushima, "Derivative-Free Filter Simulated Annealing Method for Constrained Continuous Global Optimization," *Journal of Global Optimization,* vol. 35(4), pp. 521-549, 2006.

[8] A.-R. Hedar, E. Hamdy, and M. Fukushima, "Tabu Programming Method: A New Meta-Heuristics Algorithm Using Tree Data Structures for Problem Solving," http://www.amp.i.kyoto-u.ac.jp/tecrep/ps_file/2008/2008-004.pdf, Retrieved on 2008-11-2.

[9] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science,* vol. Volume 220, pp. 671-680, 1983.

[10] B. A. Berg, *Markov Chain Monte Carlo Simulations and Their Statistical Analysis*. Singapore: World Scientific, 2004.

[11] B. Bharath and V. Borkar, "Stochastic Approximation Algorithms: Overview and Recent Trends," *Sadhana* vol. 24, pp. 425-452, 1999.

[12] E. Camponogara and S. N. Talukdar, "A Genetic Algorithm For Constrained and Multiobjective optimization," in *Proceeding of 3rd Nordic Workshop on Genetic Algorithms and Their Applications,* Vaasa, Finland, 1995, pp. 49-62.

[13] D. E. Goldberg, *Genetic Algorithms for Search, Optimization, and Machine Learning*: Addison-Wesley, 1989.

[14] Z. Michalewicz and C. C. Janikow, "GENOCOP: A Generic Algorithm for Numerical Optimization Problems with Linear Constraints," *Communication of the ACM,* 1996.

[15] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *Proceedings of IEEE International Conference on Neural Networks*. vol. 4 Perth, Australia: IEEE Service Centre, 1995, pp. 1942-1948.

[16] J. Kennedy, R. Eberhart, and Y. Shi, *Swarm Intelligence*: San Francisco: Morgan Kaufmann Publisher, 2001.

[17] M. Dorigo and T. Stutzle, *Ant Colony Optimization*: MIT Press, 2004.

[18] K. V. Price, R. M. Storn, and J. A. Lanpinen, *Differential Evolution: A Practical Approach to Global Optimization*: Springer, 2005.

[19] K. E. Parsopoulos and M. N. Vrahatis, "Recent Approaches to Global Optimization Problems Through Particle Swarm Optimization," *Natural Computing,* vol. 1, pp. 235-306, 2002.

[20] Y. Shi and R. Eberhart, "Parameter Selection in Particle Swarm Optimization," in *The 7th Ann. Conf. on Evolutionary Programming*, San Diego, CA, 1998.

[21] H. Fan, "A Modification to Particle Swarm Optimization Algorithm," *Engineering Computations,* vol. vol.19, pp. 970-989, 2002.

[22] S. Koziel and Z. Michalewicz, "Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization," *Evolutionary Computation,* vol. vol. 7, no.1, pp. 19-44, 1999.

[23] T. P. Runarsson and X. Yao, "Stochastic Ranking for Constrained Evolutionary Optimization," *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION,* vol. VOL. 4,, pp. 284-294, September 2000 2000.

[24] F. Glover, "Future Paths for Integer Programming and Links to Artificial Intelligence," *Computers and Operations Research,* vol. 13(5), pp. 533-549, 1986.

[25] A.-R. H. A. Ahmed, "Studies on Metaheuristics for Continuous Global Optimziation Problems," PhD thesis, Kyoto University, Japan, 2004.

[26] Wikipedia, "Metaheuristic," http://en.wikipedia.org/wiki/Metaheuristic, Retrieved on 2008-8-25.

[27] M. Dorigo, M. Birattari, and T. Stutzle, "Ant Colony Optimization, Artificial Ants as a Computational Intelligence Technique," *IRIDIA - Technical Report Series, TR/IRIDIA/2006-023,* 2006.

[28] M. Dorigo, "Optimization, Learning and Natural Algorithms (in Italian)," in *Ph.D. dissertation, Dipartimento di Elettronica*: Politecnico di Milano, 1992.

[29] R. Storn and K. Price, "Differential Evolution - A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces," *Technical Report TR-95-912, ICSI,* 1995.

[30] C. A. C. Coello, "A Survey of Constraint Handling Techniques used with Evolutionary Algorithms, Technical Report Lania-RI-9904, Laboratorio, Nacional de Informtica Avanzada," 1999.

[31] C. A. C. Coello, "Constraint-Handling using an Evolutionary Multiobjective Optimization Technique," *Civil Engineering and Environmental Systems,* vol. 17, pp. 319-346, 2000.

[32] Ruhul Sarker, Masoud Mohammadian, and X. Yao, "Evolutionary Optimization," in *International Series in Operations Research & Management Science*, F. S. Hillier, Ed., 2002.

[33] Wikipedia, "Genetic Algorithm," http://en.wikipedia.org/wiki/Genetic_algorithm, Retrieved on 2008-10-14.

[34] J. Holland, *Adaptation in Natural and Artificial Systems*: University of Michigan Press, 1975.

[35] D. Goldberg and K. Deb, "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms," in *Foundations of Genetic Algorithms*: Morgan Kaufmann, 1991, pp. 69-93.

[36] TJHSST Science and Technology, "Genetic Algorithms ": http://www.tjhsst.edu/~ai/AI2001/GA.HTM, Retreived on 2008-11-2.

[37] Z. Michalewicz and M. Schmidt, "Evolutionary Algorithms and Constrained Optimization," in *Evolutionary Optimization*, R. Sarker, M. Mohammadian, and X. Yao, Eds.: Kluwer Academic Publishers, 2002, pp. 57-86.

[38] R. Storn and K. Price, "DE - A Simple and Efficient Heuristic for Global Optimization over Continuous Space," *Journal of Global Optimization,* vol. 11, pp. 341-359, 1997.

[39] K. Price, "An introduction to DE," in *New Ideas in Optimization*, D. Corne, M. Dorigo, and G. Glover, Eds. London: McGraw-Hill, 1999, pp. 78-108.

[40] J. Lampinen, "A Constraint Handling Approach for the Differential Evolution Algorithm," in *Proceedings of IEEE World Congress on Computational Intelligence,* Honolulu, Hawaii, 2002, pp. 1468 - 1473

[41] H.-Y. Fan, J. Lampinen, and Y. Levy, "An Easy-to-Implement Differential Evolution Approach for Multi-objective Optimizations," *Engineering Computations: International Journal for Computer-Aided Engineering and Software,* vol. 23, pp. 124-138, 2006.

[42] T. Krink, B. Filipic, and G. B. Forgel, "Noisy Optimization Problems - A Particular Challenge for Differential Evolution?," in *Proceedings of IEEE Congress on Evolutionary Computation,* Portland, USA, 2004, pp. 332-339.

[43] J. H. Sickel, K. Y. Lee, and J. S. Heo, "Differential Evolution and its Applications to Power Plant Control," in *Proceedings of International Conference on Intelligent Systems Applications to Power Systems* Kaohsiung, Taiwan, 2007, pp. 1-6.

[44] X. Li, "Particle Swarm Optimization : An Introduction and Its Recent Development," in *Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation* London, United Kingdom: ACM, 2007.

[45] J. Kennedy, "Bare Bones Particle Swarms," in *Proceeding of The Swarm Intelligence Symposium*, 2003.

[46] K. E. Parsopoulos and M. N. Vrahatis, "Particle Swarm Optimization Method for Constrained Optimization Problem," *Intelligent Technologies--Theory and Application: New Trends in Intelligent Technologies, vol. 76 of Frontiers in Artificial Intelligence and Applications, pp. 214-220, IOS Press,* pp. 214-220, 2002.

[47] R. C. Eberhart and Y. Shi, "Comparison between Genetic Algorithms and Particle Swarm Optimization," in *Proceedings of the 7th International Conference on Evolutionary Programming VII*, 1998, pp. 611-618.

[48] R. Eberhart, P. Simpson, and R. Dobbins, *Computational Intelligence PC Tools*. San Diego: Academic Press Professional, 1996.

[49] Z. Michalewiza and M. Schoenauer, "Evolutionary Algorithms for Constrained Parameter Optimization Problems," *Evolutionary Computation,* vol. 4(1), pp. 1-32, 1996.

[50] G. Coath and S. K. Halgamuge, "A Comparison of Constraint-Handling Methods for the Application of Particle Swarm Optimization to Constrained Nonlinear Optimization Problems," in *Proceedings of IEEE Congress on Evolutionary Computation,* Newport Beach, CA, USA, 2003, pp. 2419-2425.

[51] A. Isaacs, T. Ray, and W. Smith, "Blessing of Maintaining Infeasible Solutions for Constrained Multi-objective Optimization Problems," in *Proceedings of IEEE Congress on Evolutionary Computation,* Hong Kong, China, 2008, pp. 2785-2792.

[52] Y. Yu and Z.-H. Zhou, "On the Usefulness of Infeasible Solutions in Evolutionary Search: A Theoretical Study," in *Proceedings of Congress on Evolutionary Computation,* Hong Kong, China, 2008, pp. 835-840.

[53] D. Powell and M. Skolnick, "Using Genetic Algorithms in Engineering Design Optimization with Non-linear Constraints," in *Proceedings of the Fifth ICGA,* Morgan Kaufmann, 1992, pp. 424-430.

[54] J. T. Rechardson, M. R. Palmer, G. E. Liepins, and M. R. Hilliard, "Some Guidelines for Genetic Algorithms with Penalty Functions," in *Proceeding of the 3rd International Conference on Genetic Algorithms*: Morgan Kaufmann, 1989, pp. 191-197.

[55] P. D. Surry and N. J. Radcliffe, "The COMOGA Method: Constrained Optimization by Multi-Objective Genetic Algorithms," *Control and Cybernetics,* vol. 26, 1997.

[56] Y. Zhou, Y. Li, J. He, and L. Kang, "Multi-objective and MGG Evolutionary Algorithm for Constrained Optimization," in *Proceedings of the Congress on Evolutionary Computation,* Canberra, Australia, 2003, pp. 1-5.

[57] Y. Wang and Z. Cai, "A Constrained Optimization Evolutionary Algorithm Based on Multiobjective Optimization Techniques," in *Proceedings of IEEE Congress on Evolutionary Computation,*. vol. 2 Edinburgh, Scotland, 2005, pp. 1081- 1087

[58] S. Venkatraman and G. G. Yen, "A Generic Framework for Constrained Optimization Using Genetic Algorithms," *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION,* vol. 9(4), pp. 424-435, 2005.

[59] E. Mezura-Montes and C. A. C. Coello, "A Survey of Constraint-Handling Techniques Based on Evolutionary Multiobjective Optimization," http://dbkgroup.org/knowles/MPSN3/mezura-workshop-ppsn06.pdf, Retrieved on 2008-10-14.

[60] E. Mezura-Montes and C. A. C. Coello, "Multiobjective-Based Concepts to Handle Constraints in Evolutionary Algorithms," in *Proceedings of the Fourth Mexican International Conference on Computer Science* Apizaco, MEXICO, 2003, pp. 192-199.

[61] E. Zitzler and L. Thiele, "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach," *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION,,* vol. 3 (4), pp. 257-271, 1999.

[62] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transaction onEvolutionary Computation,* vol. 6(2), pp. 197-215, 2002.

[63] C. A. C. Coello, "Treating Constraints as Objectives for Single-Objective Evolutionary Optimization," *Engineering Optimization,* vol. 32(3), 2000.

[64] J. J. Liang and P. N. Suganthan, "Dynamic Multi-Swarm Particle Swarm Optimizer with a Novel Constraint-Handling Mechanism," in *Proceedings of IEEE Congress on Evolutionary Computation,* Vancouver, Canada, 2006, pp. 9 -16.

[65] T. Ray, T. Kang, and S. K. Chye, "An Evolutionary Algorithm for Constrained Optimization,," in *Proceedings of the Genetic and Evolutionary Computation Conference* Las Vegas, USA: Morgan Kaufmann, 2000, pp. 771-777.

[66] S. A. KHTAR, K. Tai, and T. Ray, "A Socio-Behavioural Simulation Model for Engineering Design Optimizaiton," *Engineering Optimization,* vol. 34 (4), pp. 341-354, 2002.

[67] T. Ray and K. M. Liew, "Society and Civilization: An Optimization Algorithm Based on the Simulation of Social Behavior," *IEEE Transaction on Evolutionary Computation,* vol. 7(4), pp. 386- 396, 2003.

[68] C. A. C. Coello and E. Mezura-Montes, "Handling Constraints in Genetic Algorithms Using Dominance-Based Tournaments," in *Proceedings of the Fifth International Conference on Adaptive Computing Design and Manufacture,* Devon, UK, 2002, pp. 273-284.

[69] C. A. C. Coello and E. M. Montes, "Constraint-Handling in Genetic Algorithms Through the Use of Dominance-Based Tournament Selection," *Advanced Engineering Informatics,* vol. 16 pp. 193--203, 2002.

[70] G. T. Pulido and C. A. C. Coello, "A Constraint-Handling Mechanism for Particle Swarm Optimization," in *Proceeding of the 2004 Congress on Evolutionary Computation,* Portland, OR, USA, 2004, pp. 1396-1403.

[71] X. Hu and R. Eberhart, "Solving Constrained Nonlinear Optimization Problems with Particle Swarm Optimization," in *Proceedings of the Sixth World Multiconference on Systemics, Cybernetics and Informatics*, 2002.

[72] X. Hu, R. C. Eberhart, and Y. Shi, "Engineering Optimization with Particle Swarm," http://www.swarmintelligence.org/papers/SIS2003Engineering.pdf, Retrieved on 2008-10-14.

[73] K. E. Parsopoulos and M. N. Vrahatis, "Unified Particle Swarm Optimization for Solving Constrained Engineering Optimization Problems " in *Lecture Notes in Computer Science*. vol. 3612/2005: Springer, 2005, pp. 582-591.

[74] S. He, E. Prempain, and Q. Wu, "An Improved Particle Swarm Optimizer for Mechanical Design Optimization Problems," *Engineering Optimization,* vol. 36, pp. 585-605, 2004.

[75] A. E. M. Zavala, A. Hernandez, and E. R. V. Diharce, "Constrained Optimization via Particle Evolutionary Swarm Optmization Algorithm (PESO)," in *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation,* Washington DC, USA 2005, pp. 209 - 216

[76] J. Wei and Y. Wang, "A Novel Multi-objective PSO Algorithm for Constrained Optimization Problems," in *Proceeding of 6th International Conference on Simulated Evolution And Learning, LNCS 4247*, 2006, pp. 174-180.

[77] K. Zielinski and R. Laur, "Constrained Single-Objective Optimization Using Particle Swarm Optimization," in *Proceedings of IEEE Congress on Evolutionary Computation,* Vancouver, Canada, 2006, pp. 443-450.

[78] Q. He and L. Wang, "An Effective Co-Evolutionary Particle Swarm Optimization for Constrained Engineering Design Problems," *Engineering Application of Artificial Intelligence,* vol. 20, pp. 89-99, 2007.

[79] X. Hu and R. Eberhart, "Multiobjective Optimization Using Dynamic Neighborhood Particle Swarm Optimization," in *Proceeding of IEEE Congress on Evolutionary Computation,* Honolulu, Hawaii, 2002.

[80] C. A. C. Coello and M. S. Lechuga, "MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization," in *Proceedings of IEEE Congress on Evolutionary Computation*. vol. 2 Honolulu, HI, USA, 2002, pp. 1051-1056.

[81] K. E. Parsopoulos and M. N. Vrahatis, "Particle Swarm Optimization Method in Multiobjective Problems," in *Proceeding of ACM Symposium on Applied Computing* Madrid, Spain, 2002, pp. 603-607.

[82] K. E. Parsopoulos and M. N. Vrahatis, "Multiobjective Optimization using Parallel Vector Evaluated Particle Swarm Optimization," in *In Proceedings of the IASTED International Conference on Artificial Intelligence and Applications,*: ACTA Press, 2004, pp. 823--828.

[83] J. E. Fieldsend and S. Singh, "A Multi-objective Algorithm Based upon Particle Swarm Optimization, An Efficient Data Structure and Turbulence," in *Proceeding of UK Workshop on Computational Intelligence (UKCI'02),* Birminghan, UK, 2002, pp. 37-44.

[84] D. E. Goldberg and J. Richardson, "Genetic Algorithms with Sharing for Multimodal Function Optimization," in *Proceedings of the 2nd International Conference on Genetic Algorithms*, J. Grefenstette, Ed., 1987, pp. 41-49.

[85] M. Salazar-Lechuga and J. E. Rowe, "Particle Swarm Optimization and Fitness Sharing to solve Multi-Objective Optimization Problems," in *Proceedings of IEEE Congress on Evolutionary Computation,* Edinburgh, UK, 2005, pp. 1204-1211.

[86] X. Huo, L. Shen, and H. Zhu, "A Smart Particle Swarm Optimization Algorithm for Multi-Objective Problems," in *Lecture Notes in Computer Science*. vol. 4115/2006, 2006.

[87] M. Reyes-Sierra and C. A. C. Coello, "Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art," *International Journal of Computational Intelligence Research,* vol. 2, pp. 287-308, 2006.

[88] C. Ji, "A Revised Particle Swarm Optimization Approach for Multi-objective and Multi-constraint Optimization," in *Proceeding of 2004 Genetic and Evolutionary Computation Conference (GECCO)* Seattle, Washington, USA, 2004.

[89] M. J. Reddy and D. N. Kumar, "Multi-objective Particle Swarm Optimization for Generating Optimal Trade-offs in Reservoir Operation," *Hydrological Process,* vol. 21, pp. 2897-2909, 2007.

[90] C. A. C. Coello and E. Mezura-Montes, "Handling Constraints in Genetic Algorithms Using Dominance-Based Tournaments," in *Proceedings of the Fifth International Conference on Adaptive Computing Design and Manufacture* Devon, UK, 2002, pp. 273-284.

[91] C. A. C. Coello, "The use of a multiobjective optimization technique to handle constraints," citeseer.ist.psu.edu/64436.html Retreived on 2008-10-14.

[92] F. Jimenez, A. F. Gomez-Skarmeta, and G. Sanchez, "How Evolutionary Multiobjective Optimization can be used for Goals and Priorities based Optimization," in *Proceeding of AEB'02,* Merida Espana., 2002.

[93] J. Horn, "Evolutionary Computation Applications F1.9 Multicriterion decision making," in *Handbook of Evolutionary Computation*: IOP Publishing LTD and Oxford University Press, 1997.

[94] M. Clerc and J. Kennedy, "The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space," *IEEE Transactions on Evolutionary Computation,* vol. VOL. 6, NO.1, pp. 58-73, 2002.

[95] J. Kennedy, "Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance," in *Proceedings of IEEE Congress on Evolutionary Computation,* Washington D.C, USA, 1999, pp. 1931-1938.

[96] J. Kennedy and R. Mendes, "Population Structure and Particle Swarm Performance," in *Proceedings of IEEE Congress on Evolutionary Computation,* Honolulu, HI, USA, 2002, pp. 1671-1676.

[97] Wikipedia, "Sociometry," http://en.wikipedia.org/wiki/Sociometry, Retrieved on 2008-10-14.

[98] J. Flores-Mendoza and E. Mezura-Montes, "Dynamic Adaptation and Multiobjective Concepts in a Particle Swarm Optimizer for Constrained Optimization," in *Proceedings of IEEE Congress on Evolutionary Computation,* Hong Kong, China, 2008, pp. 3426-3433.

[99] D. H. Wolpert and W. G. Macready, "No Free Lunch Theorems for Optimization," *IEEE Transactions on Evolutionary Computation* vol. 1, pp. 67-, 1997.

[100] X. Li, "Better Spread and Convergence: Particle Swarm Multiobjective Optimization Using the Maximin Fitness Function," in *Proceedings of the Genetic and Evolutioanry Computation Conference (GECCO'2004)* Seattle, Washington,

USA: Springer-Verlag, Lecture Notes in Computer Science Vol.3102, 2004, pp. 117-128.

[101] T. Back, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*: Oxford University Press US, 1996.

[102] C. A. C. Coello, "Use of A Self-Adaptive Penalty Approach for Engineering Optimization Problems," *Computers in Industry,* vol. 41, pp. 113-127, 2000.

[103] T. Blackwell and J. Branke, "Multi-swarm Optimization in Dynamic Environment," in *Lecture Notes in Computer Science*. vol. 3005, 2004, pp. 489-500.

[104] C. Horstmann, *Big java*: John Wiley & Sons, 2007.

[105] M. Goodrich and R. Tamassia, *Data Structures and Algorithms in Java*: John Wiley & Sons, 1998.

[106] OMG, "OMG Unified Modeling Language (OMG UML), Infrastructure, V2.1.2," http://www.omg.org/docs/formal/07-11-04.pdf, Retrieved on 2008-10-14.

[107] Sun Microsystems Inc., "Java Platform, Standard Edition 6 API Specification," http://java.sun.com/javase/6/docs/api/, Retrieved on 2008-11-3.

[108] Wikipedia, "Fortran," http://en.wikipedia.org/wiki/Fortran, Retrieved on 2008-11-4.

[109] Wikipedia, "C (programming language)," http://en.wikipedia.org/wiki/C_(programming_language), Retrieved on 2008-11-4.

[110] J. L. Volakis, *IEEE Antennas and Propagation Magazine,* vol. 40, No. 5, 1998.

[111] Commonwealth of Australia, "Carbon Pollution Reduction Scheme - Green Paper," The Department of Climate Change, 2008.

[112] A. Farag, S. Al-baiyat, and T. C. Cheng, "Economic Load Dispatch Multiobjective Optimization Procedures using Linear Programming Techniques," *IEEE Transactions on Power System,* vol. 10, No. 2, 1995.

[113] I. N. d. Silva and L. Nepomuceno, "An Efficient Neural Approach to Economic Load Dispatch in Power Systems," in *Proceedings of the IEEE PES Summer Meeting, CD-Rom*, 2001.

[114] I. N. d. Silva, L. Nepomuceno, and T. M. Bastos, "An Efficient Hopfield Network to Solve Economic Dispatch Problems with Transmission System Representation," *International Journal of Electrical Power & Energy Systems,* vol. 26, pp. 733-738, 2004.

[115] R. Danraj and F. Gajendran, "An Efficient Algorithm to Find Optimal Economic Load Dispatch for Plants having Discontinuous Fuel Cost Functions," *IE(I) Journal - EL,* vol. 85, 2004.

[116] M. Basu, P. K. Chattopadhyay, R. N. Chakrabarti, and T. K. Ghoshal, "Economic Emission Load Dispatch with Non-smooth Fuel Cost and Emission Level Functions through an Interactive Fuzzy Satisfying Method and Evolutionary Programming Technique," *IE(I) Journal - EL,* vol. 87, 2006.

[117] 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.

[118] M. Basu, "Dynamic Economic Emission Dispatch Using Evolutionary Programming and Fuzzy Satisfying Methods," *International Journal of Emerging Electric Power Systems,* vol. 8 (2007), Issue 4, 2007.

[119] Queensland Government: http://www.epa.qld.gov.au/, Retrieved on 2009-3-25.

[120] G. V. Reklaitis, A. Ravindran, and K. M. Ragsdell, *Engineering Optimization Methods and Applications*. New York: Wiley, 1983.

PUBLICATION LIST

[1]. **Lily D Li,** Xinghuo Yu, Xiaodong Li and William Guo. "A Modified PSO Algorithm for Constrained Multi-Objective Optimization", *in Proceeding of IEEE International Conference on Networks & System Security (IEEE NSS 2009)*, Gold Coast, Australia.

[2]. **Lily D Li**, Xiaodong Li and Xinghuo Yu. "Power Generation Loading Optimization using a Multi-Objective Constraint-Handling Method via PSO Algorithm", *in Proceeding of IEEE International Conference on Industrial Informatics (IEEE INDIN 2008)*, Daejeon, Korea, 2008, p1632-1637.

[3]. **Lily D Li**, Xiaodong Li and Xinghuo Yu. "A Multi-Objective Constraint-Handling Method with PSO Algorithm for Constrained Engineering Optimization Problems", *in Proceeding of IEEE Congress on Evolutionary Computation (IEEE CEC 2008)*, Hong Kong, China, 2008, pp1528-1535.

[4]. **Lily D Li**, Jiping Zhou, Xinghuo Yu, Xiaodong Li. "Constrained Power Plants Unit Loading Optimization using Particle Swarm Optimization Algorithm". *WSEAS Transactions on Information Science and Applications,* Issue 2, Volume 4, February 2007, pp296-302, ISSN 1790-0832.

[5]. **Lily D Li**, Jiping Zhou, Xinghuo Yu. "Performance Based Unit Loading Optimization using Particle Swarm Optimization Approach", *in Proceeding of CIMMACS'06 ,* Venice, Italy 2006, pp 351-356.

[6]. **Lily D Li**, Wei Li, Russel Stonier and Xinghuo Yu. "Breaking the Constraints to a High Performance Metacomputing System: A System Approach", *in Proceeding of*

ISCA 20th International Conference on Computers and Their Applications, New Orleans, USA, 2005, pp379-384.

[7]. **Lily D Li**, ASM Sajeev, F. Han. "Time Scheduling using Agent Technology", *in Proceedings of 2002 International Symposium on Nonlinear Theory and its Applications,* Xian, China, 2002, pp555-558.

[8]. D.H Wood and **Lily D Li**. "Assessment of the Accuracy of Representing a Helical Vortex by Straight Segments". *AIAA Journal*. Vol. 40, No. 4, p647-p651, 2002. ISSN 0001-1452.

[9]. Fengling Han, Xinghuo Yu, **Lily D Li**, and Russel Stonier. "Complex Dynamic Behaviour in a Third-Order Continuous System with Hysteresis Series", *in Proceedings of 2002 International Symposium on Nonlinear Theory and its Applications,* 2002, Xian, China, pp933-936.

[10]. William Guo, **Lily D Li** and Greg Whymark. "Statistics and Neural Networks for Approaching Nonlinear Relations between Wheat Plantation and Production in Queensland of Australia*", in Proceeding of IEEE International Conference on Industrial Technology (IEEE ICIT 2009),* Churchill, Australia, 2009.