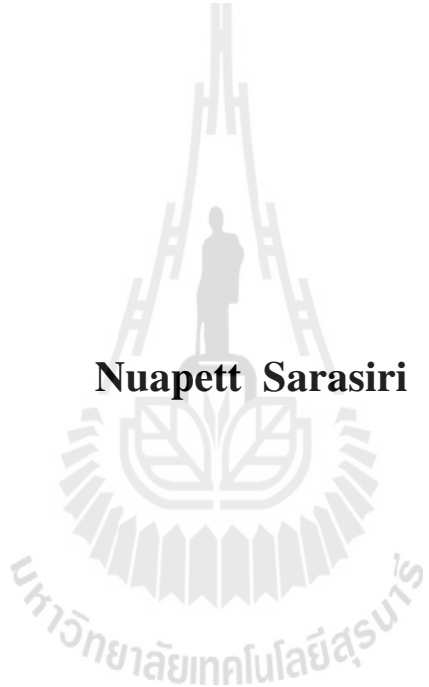
การดัดแปลงการค้นหาแบบตาบูเชิงปรับตัวบนพื้นฐานของการหาค่าเหมาะสมที่สุดแบบการเสาะหาอาหารของแบคทีเรีย: การพัฒนา การขนาน และการประยุกต์

นางสาวเนื้อเพชร สาระศิริ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรดุษฎีบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า
มหาวิทยาลัยเทคโนโลยีสุรนารี
ปีการศึกษา 2557

MODIFICATIONS TO ADAPTIVE TABU SEARCH BASED-ON BACTERIAL FORAGING OPTIMIZATION ALGORITHMS: DEVELOPMENT PARALLELIZATION AND APPLICATIONS

Nuapett Sarasiri



A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Electrical Engineering

Suranaree University of Technology

Academic Year 2014

MODIFICATIONS TO ADAPTIVE TABU SEARCH BASED-ON BACTERIAL FORAGING OPTIMIZATION ALGORITHMS: DEVELOPMENT PARALLELIZATION AND APPLICATIONS

Suranaree University of Technology has approved this thesis submitted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy.

Thesis Examining Committee

(Assoc. Prof. Dr. Kitti Attakitmongcol)
Chairperson

(Asst. Prof. Dr. Padej Pao-la-or)
Member (Thesis Advisor)

(Prof. Wng. Cmdr. Dr. Sarawut Sujitjorn)
Member

(Assoc. Prof. Dr. Arthit Srikaew)
Member

(Assoc. Prof. Dr. Deacha Puangdownreong)
Member

(Asst. Prof. Dr. Kongpan Areerak)
Member

(Prof. Dr. Sukit Limpijumnong)
Vice Rector for Academic Affairs

(Assoc. Prof. Flt. Lt. Dr. Kontorn Chamniprasart)
Dean of Institute of Engineering

เนื้อเพชร สาระศิริ : การดัดแปลงการค้นหาแบบตาบูเชิงปรับตัวบนพื้นฐานของการหาค่าเหมาะสมที่สุดแบบการเสาะหาอาหารของแบคทีเรีย: การพัฒนา การขนานและการประยุกต์ (MODIFICATIONS TO ADAPTIVE TABU SEARCH BASED-ON BACTERIAL FORAGING OPTIMIZATION ALGORITHMS: DEVELOPMENT PARALLELIZATION AND APPLICATIONS) อาจารย์ที่ปรึกษา : ผู้ช่วยศาสตราจารย์ ดร.เผด็จ เผ่าละออ, 408 หน้า.

งานวิจัยวิทยานิพนธ์นี้นำเสนอวิธีการดัดแปลงอัลกอริธึมการค้นหาแบบตาบูเชิงปรับตัว บนพื้นฐานแนวคิดของการหาค่าเหมาะสมที่สุดแบบการเสาะหาอาหารของแบคทีเรีย รวมทั้งพัฒนาการทำงานของอัลกอริธึมในเชิงขนาน และประยุกต์กับปัญหาจริงทางด้านวิศวกรรมศาสตร์ การดัดแปลงอัลกอริธึมกระทำใน 2 โครงสร้าง ได้แก่ การทำงานร่วมกันของอัลกอริธึมการเคลื่อนที่แบบแบคทีเรียกับการค้นหาแบบตาบูเชิงปรับตัว ทำให้เกิดอัลกอริธึมแบบผสมเรียกว่า cooperative bacteria foraging-tabu search หรือ BF-TS และการดัดแปลงอัลกอริธึมการค้นหาแบบตาบูเชิงปรับตัว ด้วยการใช้กลไกการเคลื่อนที่คล้ายกับแบคทีเรียผลิตคำตอบค่าใกล้เคียงเรียกว่า modified adaptive tabu search หรือ modified ATS อัลกอริธึม BF-TS มีการผสมผสานกลไกการเคลื่อนที่ของแบคทีเรียแบบการสุ่มกับกลไกการปรับรัศมีการค้นหา และการย้อนรอยของคำตอบ ส่วนอัลกอริธึม modified ATS ผสมผสานกลไกการเคลื่อนที่หาคำตอบแบบสุ่มด้วยช่วงก้าวที่เหมาะสมขึ้นอยู่กับค่าฟังก์ชันวัตถุประสงค์ โดยไม่ใช้กลไกการปรับรัศมีการค้นหา แต่ยังคงใช้กลไกการย้อนรอบของคำตอบไว้ อัลกอริธึม modified ATS ได้รับการวิเคราะห์คุณสมบัติการลู่เข้าตามแนวทางเสถียรภาพของไลอาพูนอฟ

วิทยานิพนธ์นี้ยังนำเสนอการประเมินสมรรถนะการค้นหาคำตอบของอัลกอริธึมต่างๆ ที่เกี่ยวข้อง โดยอาศัยฟังก์ชันทางคณิตศาสตร์อันเป็นนามธรรม และปัญหาจริงทางวิศวกรรมต่างๆ ที่ซับซ้อน ในองค์รวม รูปแบบของปัญหาจึงเป็นปัญหาการหาค่าเหมาะสมที่สุดแบบผสมผสาน (combinational optimization) ที่มีและไม่มีเงื่อนไขบังคับ โดยได้ใช้อัลกอริธึม BF-TS อัลกอริธึม modified ATS อัลกอริธึม ATS อัลกอริธึมเชิงพันธุกรรม (genetic algorithm, GA) อัลกอริธึมการค้นหาค่าเหมาะสมที่สุดแบบแบคทีเรียเชิงปรับตัว (adaptive bacterial foraging optimization, ABFO) และอัลกอริธึมการค้นหาค่าเหมาะสมที่สุดแบบการแพร่กระจายของวัชพืช (invasive weed optimization, IWO) เพื่อประเมินสมรรถนะการค้นหา ในภาพรวมพบว่า อัลกอริธึม modified ATS ให้สมรรถนะการทำงานที่เหนือกว่าอัลกอริธึมอื่นๆ ทั้งในด้าน การลู่เข้าหาคำตอบในจำนวนรอบการค้นหาที่น้อยกว่า และสามารถหลีกเลี่ยงคำตอบติดล็อกได้ดีกว่า แต่อัลกอริธึม modified ATS ใช้เวลา

การค้นหาต่อรอบที่นานกว่าเนื่องมาจากกลไกการเคลื่อนที่หาคำตอบแบบสุ่มที่ถูกเรียกใช้บ่อยๆ อัลกอริธึม modified ATS จึงได้รับการอนุวัตเชิงขนาน เพื่อทำงานบนเครื่องคอมพิวเตอร์ตั้งโต๊ะแบบ 4 แกนซีพียู โปรแกรมด้วย MATLAB เชิงขนาน ให้ผลการทดสอบกับปัญหาระบบขับเคลื่อนกำลังไฟฟ้าที่น่าพอใจมาก กล่าวคือ สามารถลดเวลาคำนวณได้ถึงร้อยละ 41.40

สาขาวิชา วิศวกรรมไฟฟ้า
ปีการศึกษา 2557

ลายมือชื่อนักศึกษา เสือเพชร สาระศรี
ลายมือชื่ออาจารย์ที่ปรึกษา
ลายมือชื่ออาจารย์ที่ปรึกษาร่วม

NUAPETT SARASIRI : MODIFICATIONS TO ADAPTIVE TABU SEARCH BASED-ON BACTERIAL FORAGING OPTIMIZATION ALGORITHMS: DEVELOPMENT PARALLELIZATION AND APPLICATIONS. THESIS ADVISOR : ASST. PROF. PADEJ PAO-LA-OR, Ph.D., 408 PP.

ADAPTIVE TABU SEARCH/MODIFICATION/PARALLELIZATION/MULTI-OBJECTIVE/SURROGATED CONSTRAINT OPTIMIZATION

This research thesis presents modifications to adaptive tabu search (ATS) based-on the concepts of bacterial foraging (BF) optimization algorithm. The works also cover a parallel implementation and applications of the proposed algorithms to some real-world engineering problems. The modifications consider two structural approaches, i.e. a straight-forward combination of the BF and the ATS resulting in a hybrid or cooperative BF-TS algorithms, and the ATS embedded with a bacterial movement alike for neighbour-solution generation so called a modified ATS. The BF-TS incorporates the bacterial random movement, the adaptive search radius (*AR*) and the back tracking (*BT*) mechanisms of the ATS. The modified ATS proposes a random-walk front-end simplified from the bacterial random movement accommodating suitable searching steps depending on cost values. The modified ATS abandons the *AR* mechanism while utilizing the *BT*. The convergence of the modified ATS has been analysed via the Lyapunov's stability method.

Search performance assessment of multiple algorithms is also an important issue addressed by the thesis. The assessment has utilized abstract mathematical

problems and complex real-world engineering ones. They are organized as combinatorial optimization problems with and without constraints. The algorithms being assessed include the BF-TS, the modified ATS, the ATS, the genetic algorithms (GA), the adaptive bacterial foraging optimization (ABFO) and the invasive weed optimization (IWO) algorithms, respectively. Our results indicate that the modified ATS is superior to the others in terms of tracking-down the solution by fewer numbers of search rounds, and its capability of local entrapment avoidance. However, the average time per search round of the modified ATS is longer than that of the others because it frequently invokes the random-walk front-end. In order to reduce the computing time, the modified ATS being proposed has been implemented in parallel form to be executed on a 4-core desktop computer using the MATLAB and its Parallel Computing Toolbox. As a result, computing time is reduced by 41.40%.

School of Electrical Engineering

Academic Year 2014

Student's Signature Nuapett Sarasiri

Advisor's Signature

Co-Advisor's Signature S. Sujitjorn

ACKNOWLEDGEMENTS

First of all, I would like to express my most sincere gratitude to Professor Wing Commander Dr. Sarawut Sujitjorn (formerly with School of Electrical Engineering, Suranaree University of Technology, Thailand, currently Director of Synchrotron Light Research Institute, Thailand) for the role he has played as my thesis supervisor. His precious advices and numerous discussions have enhanced my knowledge and scientific inspiration. He was guidance of this thesis from the start to success.

I very much appreciate Assistant Professor Dr. Padej Pao-la-or for his role in looking after my course of action on-behalf of Professor Sarawut. His strong support has been gratefully acknowledged.

With gratitude, I acknowledge the expert advice of all teachers in my childhood. I also thank the faculty and staff of Suranaree University of Technology, for providing support in a multitude of forms, and my colleagues at the EE laboratories for their company and encouragement.

This research has been supported financially by the Royal Golden Jubilee (RGJ) PhD Program of the Thailand Research Fund (TRF). The support is gratefully acknowledged. In addition, this scholarship has fully supported me to conduct research in the Department of Electrical and Electronic Engineering at the University of Nottingham, United Kingdom during 1st September 2011 to 29th February 2012.

For some guidances and discussions, I am grateful to Professor Dr. Pericle Zanchetta in the Department of Electrical and Electronic Engineering at the University

of Nottingham for his academic support for some parts of the works. I am also thankful for the support and useful advice from Professor Dr. Chris Gerada in the Department of Electrical and Electronic Engineering at the University of Nottingham. I also would like to thank the University of Nottingham for their support of my research activities as a Visiting Research Scholar in the Power Electronics, Machines and Control (PEMC) Research Group.

I would like to thank Dr. Butsakorn Yodkhumlue, Mr. Peter Bint and Mr. Eduardo Reyes Moraga, who have helped as my language reviewers.

Finally, I am most grateful to my dearly beloved parents and sister for their encouragements. They helped me to regain enthusiasm, strength and determination during the difficult times. Thanks for being there for me and believing in me, always. I dedicate this work to my grandmother, who now rests in serenity.

Nuapett Sarasiri

TABLE OF CONTENTS

Page

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

Page

LIST OF TABLES

Table **Page**

LIST OF TABLES (Continued)

Table **Page**

LIST OF TABLES (Continued)

Table **Page**

LIST OF TABLES (Continued)

Table **Page**

LIST OF TABLES (Continued)

Table **Page**

LIST OF FIGURES

Figure **Page**

LIST OF FIGURES (Continued)

Figure **Page**

LIST OF FIGURES (Continued)

Figure **Page**

LIST OF FIGURES (Continued)

Figure **Page**

LIST OF FIGURES (Continued)

Figure **Page**

LIST OF FIGURES (Continued)

Figure **Page**

LIST OF FIGURES (Continued)

Figure **Page**

LIST OF FIGURES (Continued)

Figure **Page**

LIST OF FIGURES (Continued)

Figure **Page**

SYMBOLS AND ABBREVIATIONS

A. Algorithms

$count_{max}$	=	maximum iteration
$d_{attract}$	=	coefficient representing the depth of attractant released
dim	=	dimension of parameters
$h_{repellant}$	=	coefficient representing the height of the repellant effect
it_{max}	=	maximum number of iterations
n	=	nonlinear modulation index
n_re_back	=	k^{th} backtracking solution retrieved from the *TL*
p	=	number of parameters to be optimized
p_{max}	=	maximum number of plant population
$rand$	=	random number on [0,1]
s_{max}	=	maximum number of seeds
s_{min}	=	minimum number of seeds
$w_{attract}$	=	coefficient representing the width of the attractant signal
$w_{repellant}$	=	coefficient representing the width of the repellant by the cell
AR	=	adaptive radius
BT	=	frequency of solution cycling
$C(i,j)$	=	step size taken in random direction specified by the tumble
$J(i, j)$	=	cost value of i^{th} bacterium
N	=	number of the neighbourhood

SYMBOLS AND ABBREVIATIONS (Continued)

N_c	=	number of iterations to be carried out in a chemotactic loop
N_{ed}	=	maximum number of elimination and dispersal events
N_0	=	number of initial populations
N_{re}	=	number of reproduction loop
N_s	=	swimming length after tumbling of bacteria in a chemotactic loop
P_{ed}	=	probability with which the elimination and dispersal continues
R	=	search radius
S	=	number of bacteria in the population
S_r	=	a half of number of bacteria ($S/2$)
TL	=	tabu list
$\Delta(i)$	=	random vector on [-1,1]
α	=	a positive constant
θ^i	=	position of i^{th} bacterium
σ_{final}	=	final value of standard deviation
$\sigma_{initial}$	=	initial value of standard deviation

B. Identification problem

a	=	gear ratio = 5.9
i	=	motor current *(A)*
k_{spring}	=	gravity constant (N/mm)

SYMBOLS AND ABBREVIATIONS (Continued)

l	=	ball screw lead = 5 *mm*
m	=	mass (*kg*)
n	=	number of data
v	=	velocity (*mm/s*)
vss	=	crossover velocity (*mm/s*)
x_i	=	displacement of spring (*mm*)
x_d	=	displacement of mass (*mm*)
D_m	=	viscous friction coefficient= 2.60×10^{-6} *Nm / rad* / sec
F_C	=	coulomb friction (*N*)
F_{ex}	=	external input force (*N*)
F_f	=	friction force (*N*)
F_{f_motor}	=	friction force of motor (*N*)
F_{in}	=	internal input force (*N*)
F_{m_motor}	=	force equivalent to the inertia of motor *(N)*
K_{kt}	=	intertia to force conversion factor = $K_n \times K_v = (2\pi a / l)^2 \eta_b \eta_c$
K_n	=	linear to angular velocity conversion factor = $2\pi a/l$ (*rad/m*)
K_t	=	torque constant of motor = 18.2×10^{-3} *N / A*
K_v	=	torque to ball screw force conversion factor = $2\pi\eta_b\eta_c a / l$
P	=	moment (*Nm*)

SYMBOLS AND ABBREVIATIONS (Continued)

F_S	=	static friction (*N*)
F_V	=	viscous friction (*Ns/mm*)
J_m	=	inertia of motor= $4.17 \times 10^{-6} kg.m^2$
X_{test}	=	displacement from measured (*mm*)
$\pm dv$	=	velocity band around zero velocity = $\pm 0.1\ mm/s$
$\pm dp$	=	momentum band ($\pm dv \times m$)
η_b	=	gear box efficiency = 0.81
η_c	=	ball screw efficiency = 0.925
$\hat{k}$	=	proportional controller gain

C. Inverter and PMSM

i_{as}, i_{bs}, i_{cs}	=	3-phase stator currents
i_d, i_q	=	*d*- and *q*-axis stator currents
f_d, f_q, f_0	=	$d-$, $q-$ and $0-$axis components
f_α, f_β, f_0	=	$\alpha-$, $\beta-$ and $0-$axis components
f_s	=	switching frequency
k_t	=	torque constant
v_{as}, v_{bs}, v_{cs}	=	3-phase stator voltages
v_d^*, v_q^*	=	input terminal voltages in the *dq*-frame

SYMBOLS AND ABBREVIATIONS (Continued)

v_d , v_q = *d*- and *q*-axis stator voltages

B_m = viscous coefficient (motor)

E_{DC} = DC-link voltage

J = moment of inertia (motor)

$L_{\Delta m}$ = half amplitude of the sinusoidal variation of the magnetizing inductance

L_{aa} , L_{bb} , L_{cc} = 3-phase self-inductances of stator windings

L_{ab} , L_{ac} , L_{ba} , L_{bc} , L_{ca} , L_{cb} = mutual inductances between the stator phases

L_d , L_q = *d*- and *q*-axis inductances

L_f = input filter inductance

L_{line} = input inductance of each phase

L_{ls} = stator leakage inductance

$\overline{L}_m$ = average value of the magnetizing inductance

L_s = stator inductance

N_m = motor speed

P = number of poles

R_f = internal resistance of input filter inductor

R_{line} = input resistance each phase

R_s = stator resistance

SYMBOLS AND ABBREVIATIONS (Continued)

T_e	=	electromagnetic torque
T_m	=	mechanical torque
T_L	=	external load torque
V_s	=	input voltage
$\psi_{as}, \psi_{bs}, \psi_{cs}$	=	3- phase flux linkages
ψ_d , ψ_q	=	*d*- and *q*-axis stator flux quantities
ψ_m	=	flux linkage of the permanent magnet mounted on the rotor shaft
ω	=	angular frequency of the rotor
ω_e	=	angular frequency of the rotating *dq* reference frame
ω_m	=	mechanical angular velocity of the rotor
θ	=	angle between the rotor *d*-axis and *q*-axis
θ_m	=	mechanical angular position of the rotor
θ_r	=	angular rotor position
ζ	=	damping factor

CHAPTER I

INTRODUCTION

1.1 Significance of the Problem

It is well-known that some complex problems like hard combinatorial optimization problems contain many local solutions and constraints. While conventional optimization methods often fail to solve such problems, metaheuristic techniques are strong alternative means for finding the optimal or sub-optimal solutions with reasonable computing time. For NP-hard problems, search tasks are still very tedious for metaheuristics. Therefore, this thesis aims to study and improve the search performance of metaheuristic algorithms namely the adaptive tabu search (ATS) and the adaptive bacterial foraging optimization (ABFO) algorithms. The ABFO is known for its attractive explorative characteristic. Using its chemotaxis strategy, the algorithm visits many high-quality solutions dispersed over the search space in a short duration. On the contrary, the ATS is known for its good exploitative behaviour. It is capable of tracking down an elite solution in a short duration due to its adaptive radius (*AR*) mechanism, but it lacks a capability of focusing an initial solution of high-quality. Regarding this, a simple random number generation has been used to coarsely generate initial solutions. Consequently, some parts of the search space containing high-quality solutions may not be found. As a result, the algorithm requires a considerable long time to achieve a final elite solution. In contrast, the ABFO via its adaptive chemotaxis step can exhaustively explore various parts of the entire search space. This provides a better

opportunity to obtain at least one elite solution at the very beginning of the search providing that the initial exploration is not repeated too many times. Search time will be significantly prolonged, if the initial exploration is many times repeated. Algorithm complexity and possibility of being locked by local solutions are additional drawbacks of ABFO. Some of these disadvantages can be overcome by the backtracking (*BT*) and the *AR* mechanisms of the ATS. Therefore, the two algorithms can be combined to form new metaheuristics called cooperative bacterial foraging-tabu search or BF-TS. Furthermore, generation of neighbour solutions of the ATS within a limited search radius can be replaced by the adaptive chemotaxis step of the ABFO to improve the explorative characteristic. This newly proposed version of the ATS is called modified ATS. To evaluate the search performances, the algorithms have been tested against several abstract and real-world problems. Parallelization has been made to the proposed algorithms based-on MATLAB and Parallel Computing Toolbox to significantly reduce search time.

1.2 Research Objectives

The objectives of this research are as follows:

- To combine the ABFO algorithms and the ATS such that new algorithms work in cooperative manner.
- To develop a modified version of the ATS.
- To investigate performances among the proposed algorithms, the ABFO, the ATS, the GA, and the IWO by using unconstrained surface optimization problems.
- To apply the proposed algorithms to abstract mathematical constrained problems, control problems, nonlinear stability analysis and identifications.

- To accomplish a parallel application of the proposed version of the ATS.
- To apply the proposed algorithms to a complex engineering problem under parallel computing environment.

1.3 Research Hypothesis

The proposed cooperative BF-TS algorithm and the new version of ATS achieve a superior performance when compared with the ATS. The proposed algorithms can be effectively applied to various engineering optimization problems.

1.4 Basic Assumptions

The approximate algorithms presented in this thesis can effectively handle combinatorial optimization problems. Relevant computing tasks are carried out by using MATLAB® and MATLAB Parallel Computing Toolbox™.

1.5 Scope and Limitations of the Study

- Studies of some existing metaheuristic algorithms covering the ABFO, the ATS, the genetic algorithm (GA) and the invasive weed optimization (IWO).
- Search performance comparisons of the above and our proposed algorithms can be conducted against some unconstrained optimization problems to determine average search time, average search rounds and quality of solutions. These unconstrained problems include the minimization of the Bohachevsky, Rastrigin, Shekel's fox-holes, Schwefel and Shubert functions.

- Application problems are limited to three abstract mathematical constraint problems, optimal control design of hard-disk heads (single-head and head-stacks), optimal control design of a truck braking system, stability analysis of nonlinear systems, identification problems of hard-disk head actuator and nonlinear Stribeck friction model, and optimal design of a power drive system.

- Parallelization of the modified ATS has been performed on Intel quad-core CPUs to access its performance.

- The paralleled version of the modified ATS has been applied to solve the multi-objective design optimization of power drive system with many constraints.

1.6 Expected Usefulnesses

- The proposed algorithms are effective and efficient for solving abstract optimization problems as well as real-world complex problems.

- The proposed algorithms with parallelization are effective, outperform and applicable to a complex engineering problem.

1.7 Research Procedures

- Studies of existing metaheuristic algorithms including the ABFO, the ATS, the IWO and the GA.

- Conduct search performance tests against some unconstraint surface optimization problems including BF, RF, SF, SchF and ShuF, which stand for Bohachevsky, Rastrigin, Shekel's fox-holes, Schwefel and Shubert functions, respectively.

- Combine the ABFO and ATS algorithms to work as cooperative bacterial foraging-tabu search (BF-TS), simplify the algorithms to obtain a modified ATS, and compare search performances among the ABFO, the ATS, the IWO and the GA algorithms by using the unconstraint surface optimization problems.

- Analyse the convergence of the modified ATS based-on Lyapunov's direct method.

- Use the proposed algorithms to solve compensator designs for a single R/W head and R/W head-stacks, controller designs for braking of a heavy-duty truck, nonlinear stability analysis, identification of an actuator model of a servo track writing system, identification of a nonlinear friction model, minimization of three abstract mathematical constraint problems and a multi-objective design optimization of a permanent magnet synchronous motor (PMSM) drive system.

- Apply the modified ATS to a multi-objective design optimization of the PMSM drive system under parallel computing environment.

1.8 Organization of the Thesis

This thesis is organized as follows. **Chapter I** defines the objectives, problems, rationalities, and the methodology of the research.

Chapter II presents literature reviews of metaheuristics including single-solution based and population-based metaheuristics, respectively. The main concepts of those metaheuristics are explained in this Chapter II.

Chapter III describes the important concept of combinatorial optimization problems, and gives a detailed explanation of metaheuristics covering single-solution based and population-based metaheuristics.

Chapter IV explains the following metaheuristics in details: the adaptive bacterial foraging optimization (ABFO), the adaptive tabu search (ATS), the invasive weed optimization (IWO) and the genetic algorithm (GA), respectively. Moreover, the proposed cooperative algorithms based-on the ABFO and the ATS denoted as bacterial foraging-tabu search (BF-TS), and modified ATS, respectively, are highlighted in details.

Chapter V briefly describes different types of random walks and mathematical approaches to convergence analysis of algorithms. The Chapter presents proof of the convergence of the random walk front-end employed by the proposed algorithms based-on Lyapunov's method. Convergence of the overall algorithms is then analysed due to the fact that the tabu search with *BT* mechanism has been known for its convergence property.

Chapter VI presents comparison studies of search performances among the proposed algorithms, the ABFO, the ATS, the BF-TS, the modified ATS, the IWO and the GA against benchmarking surface optimization problems. The performances have been considered in terms of number of local locks, number of search rounds, number of objective function evaluations, quality of initial solutions, quality of global solutions obtained, search time, and time consumed per search round.

Chapter VII reports the results of applying the proposed algorithms to solve various constrained abstract real-world problems. These are optimal control designs of hard-disk heads and a second-order system with delay, stability analysis of a nonlinear system, identification problems of hard-disk head actuator and a nonlinear friction model, as well as design of a power drive system. The computational results are compared between the proposed BF-TS and the modified ATS algorithms.

Chapter VIII describes relevant details of parallelization including parallel processing or computing, parallel programing along with parallel metaheuristics. In addition, the Parallel Computing Toolbox of MATLAB and its significant functions are explained in this Chapter. To reduce computing time consumed by the modified ATS for solving the drive problem, parallelization of the algorithm has been applied. Implementation, results and discussions appear in this Chapter.

Chapter IX summarizes all the findings and contributions of the thesis. The Chapter also brings recommendations for future works into attention.

CHAPTER II

LITERATURE SURVEY

2.1 Introduction

Engineering computational intelligence has been a fast growing research domain in recent years. Many algorithms have been successfully applied to give optimal solutions to complex and NP-hard problems in engineering. The most popular methods are population-based metaheuristics and single-solution based metaheuristics. The current study employs population-based metaheuristics to examine the attention on bacterial foraging optimization (BFO), whereas an adaptive version of tabu search known as the adaptive tabu search (ATS) has been proposed as single-solution based metaheuristics. The study eventually leads to a new version of the ATS in which population-based search mechanisms are missed at an initial phase of algorithmic execution.

Most of the population-based metaheuristics are natural-inspired algorithms, which include bacterial foraging optimization algorithm. The bacterial foraging optimization was introduced in 2002 for solving distributed optimization and control problems (Passino, 2002; Liu and Passino, 2002). The algorithm mimics the foraging behavior of *Escherichia coli* (*E.coli*) bacteria and the computer codes can be found in *http://*www2.ece.ohio-state.edu/~passino/ICbook/ic_code.html. This algorithm consists of four main steps namely chemotaxis, swarming, reproduction and elimination-

dispersal, respectively. The chemotaxis mechanism simulates the movement of an *E.coli* bacterium cell for swimming and tumbling via flagella. The swarming mechanism uses activation based on cell-to-cell signaling and foraging. This can be achieved through an objective function adjustment depending on the relative distance between each bacterium and the healthiest one. For the reproduction, the objective function is sorted in ascending order, and the unhealthy bacteria, i.e. ones with high objective values, will be eliminated. Afterward, the healthy group of bacteria is reproduced by splitting into two at the same location. Thus, this process maintains the population of the bacteria. For the elimination-dispersal process, the weak bacteria are eliminated, while the healthiest ones obtained from the reproduction process are randomly dispersed to new locations within the search space according to the probability P_{ed}.

2.2 Applications of the BFO algorithm

The BFO has been applied to several real-world optimization problems as follows:

- In 2006, Tripathy et. al. applied the BFO to solve, the optimum location and the amount of series injected voltage for the unified power flow controller (UPFC) and the best values of the taps present in the system (Tripathy et. al., 2006).

- In 2007, Mishra and Bhende used the BFO to optimize the coefficients of proportional plus integral (PI) controllers for active power filters. The BFO technique has been compared with the genetic algorithm (GA). The results were obtained to converge faster than GA to reach the global optimum solution (Mishra & Bhende, 2007).

- In 2007, Munoz et al. used the BFO for the dynamical resource allocation in a multiple input/output experimentation platform, which mimicked a temperature grid plant and was composed of multiple sensors and actuators organized in zones (Munoz et al, 2007).

- In 2007, Ulagammai et al. applied the BFO to train a wavelet-based neural network (WNN) and used the same for identifying the inherent non-linear characteristics of power system loads (Ulagammai et al., 2007).

- In 2010, Dasgupta et al. used the BFO for the automatic detection of circular shapes from complicated and noisy images without using the conventional Hough transform methods. The proposed algorithm was able to detect single or multiple circles from a digital image through one shot of optimization (Dasgupta et al, 2010).

2.3 Applications of the ABFO algorithm

However, the original BFO sometimes does not converge into a high-quality solution, particularly when applied to complicated problems. Furthermore, the original BFO algorithm involves many iterations and consequently needs more computational time. Since the introduction of BFO, various modifications have been attempted to improve its performances primarily through adaptive chemotaxis strategy and hybridization with other optimization algorithms as follows:

- In 2005, Mishra proposed a Takagi-Sugeno type fuzzy inference scheme for selecting the optimal chemotactic step-size in BFO. The resulting algorithm, referred to as fuzzy bacterial foraging (FBF), was shown to outperform both classical BFO and the GA when had applied to the harmonic estimation problem. However, the performance of the FBF crucially depended on the choice of the membership function and the fuzzy

rule parameters and there was no systematic method (other than trial and error) to determine these parameters for a given problem (Mishra, 2005).

- In 2007, Li et al. proposed a modified bacterial foraging algorithm with varying population (BFAVP) and applied to the optimal power flow (OPF) problems. The BFAVP also incorporated the mechanisms of bacterial proliferation and quorum sensing, which allowed a varying population in each generation of bacterial foraging process (Li et al., 2007).

- In 2007, Tripathy and Mishra proposed an improved BFO algorithm for simultaneous optimization of the real power losses and voltage stability limit (VSL) of a mesh power network. In their modified algorithm, instead of the average value, the minimum value of all chemotactic cost functions was retained for deciding the bacterium's health. Simulation results indicated the superiority of the proposed approach over classical BFO for the multi-objective optimization problem involving the unified power flow controller (UPFC) location, its series injected voltage, and the transformer tap positions as the variables (Tripathy and Mishra, 2007).

- In 2008, Tang et al. also presented a dynamic bacterial foraging algorithm (DBFA) for solving an OPF problem in a dynamic environment in which system loads were changing. DBFA was based on the recently proposed BFO which mimicked the basic foraging behavior of *E. coli* bacteria. A selection scheme for bacteria's reproduction was employed in DBFA, which explored the self-adaptability of each bacterium in the group searching activities (Tang et al., 2008).

- In recent year, Dasgupta et al. modified chemotactic movement of the BFO to become an adaptive mechanism, and hence the name adaptive bacterial foraging optimization, or ABFO. The significance of proposed mechanism was to avoid

oscillation, especially on flat fitness landscapes, when a bacterium cell was close to the optima and to accelerate the convergence speed of the group of bacteria near global optima. The results over several numerical benchmarks indicated that the ABFO obtained better convergence behavior, as compared to the classical BFO. (Dasgupta et al., 2009; Dasgupta et al., 2010; Majhi et al., 2009).

In this thesis, we focus on the recent modification of ABFO to enhance the performance of BFO. More details of the ABFO will be described in Chapter IV.

2.4 Applications of the hybrid BFO algorithm with the inspired metaheuristics

Hybridization of BFO with other optimization algorithms has been found in some researches which most of them are combined with naturally inspired metaheuristics as concisely described as follows:

- In 2007, Kim et al. proposed a hybrid approach involving GA and BFO algorithms for function optimization to improve the mutation and crossover operations in order to increase speed of convergence. The proposed algorithm outperformed both GA and BFO algorithms over a few numerical benchmarks and a practical PID controller design problem (Kim et al., 2007).

- In 2007, Biswas et al. proposed a synergism of the BFO algorithm with another very popular swarm intelligence algorithm well-known as the particle swarm optimization (PSO). The new algorithm, named by the authors as bacterial swarm optimization (BSO). This proposed approach used mutated of the PSO algorithm for each bacterium after undergoing the chemotactic step. The chemotactic movement operation of the BFO was performed like the local search whereas the PSO was

performed like the global search over the entire search space. The significant approach balances between exploration and exploitation. This proposed algorithm was shown to perform in a statistically significantly better way as compared to both of its classical counterparts over several numerical benchmarks (Biswas et al, 2007).

- In 2008, Saber and Venayagamoorthy presented a hybrid version of BFO and PSO. The main of this hybrid algorithm was to propose a modified version of bacterial foraging technique (BFT), which was suitable for economic load dispatch (ELD) problems in huge multi-dimensional space. Random velocity of BFT was improved by using PSO movement (evolution). Best cell (or particle) biased velocity (vector) was applied in the proposed method to reduce randomness in movement (evolution) and to increase swarming called BFT with PSO biased evolution (BFT-PSOBE). This study was shown that the BFT-PSOBE was enough scope to work on a real ELD application in unit commitment problems within practical execution time limit (Saber and Venayagamoorthy, 2008).

- In 2009, Shao and Chen proposed an alternative solution integrating bacterial foraging optimization algorithm and tabu search (TS) algorithm called TS-BFO to solve motif discovery problem. The proposed hybrid algorithms had used tabu list to contain elitist selection*s* for reproduction step of the BFO and also to conquer the trouble of local extremum which arises in the original BFO with higher probability (Shao and Chen, 2009).

2.5 Applications of the TS algorithm

The sample of single-solution based metaheuristics in this thesis is the adaptive tabu search. The original tabu search was generated by Glover (Glover, 1989; Glover,

1990) which has become one of the most efficient metaheuristic methods. It incorporates two major strategies: intensification and diversification. Successful applications of the TS have appeared in various fields, e.g. food processing (Zhang et al, 2003), optimal power flow (Kulworawanichpong and Sujitjorn, 2002), parameter estimation (Yao et al., 2001), flow shop problems (Nowicki and Smutnicki, 1996), etc.

2.6 Applications of modifications to the TS algorithm

For some complex systems containing many local optima, the simplistic TS is usually unable to release the search move from a local entrapment. This problem has been overcome by using different modifications made to the TS. These examples can be concisely described as follows:

- In 1994, Battiti and Tecchiolli published the reactive tabu search (RTS) which maintain the basic step of tabu search by varying the length of the tabu list in order to avoid repeated solutions. RTS had feedback-based tuning of the length of the tabu list and an automated balance of diversification and intensification. In this process, all searched states were stored. After the move was executed, the algorithm checks whether the current searching point had already been found. The length of the tabu list increases if a searching point was repeated. On the contrary, the tabu list decreases if no repetitions occurred during a sufficiently long period (Battiti and Tecchiolli, 1994).

- In 1998, Chiang and Chiang proposed the probabilistic tabu search (PTS). PTS was an extension of the tabu search mechanism based on random neighbour at each iteration with probability. The PTS had a candidate buffer with the length of neighbourhood. At each exchange process, the PTS selected the neighbourhood which was the top ranked candidate of the selected moves and stored them in the candidate

buffer in order of best to worst. The selected moves should not be in the tabu list or the avoidance list. After all the moves were selected, a “campaign” was conducted. First, the best move was selected according to probability. If the best move was rejected, the second best would be selected with probability. This process continues until a move was picked or the last move was tested. If all the moves were rejected, a randomly selected move or the best move could be selected as the next move. In order to improve the campaign process, a probability table was created (Chiang and Chiang, 1998).

- In 2006, Sujitjorn et al. proposed the adaptive tabu search (ATS) consisting of two major additional strategies made to the conventional TS. These were backtracking (*BT*) and adaptive search radius (*AR*) mechanisms, respectively. The former strategy assisted the TS to release itself from being locked by a local solution. It looked up the tabu list (*TL*), i.e. short-term memory, for a visited elite solution, and used this solution to start a new search move. The latter strategy enhanced the focusing characteristic of the TS. This strategy decreased the search radius gradually as the search came close to a solution of high-quality having the potential of being the optimal one. However, too short a search radius could result in a slow search. Recommendations for the selection of search parameters are in Sujitjorn et al. (Sujitjorn et al., 2006).

In this thesis, the ATS has been studied. This has been used for various successful applications such as signal and system identification (Puangdownreong and Sujitjorn, 2006; Puangdownreong et al., 2005; Kulworawanichpong et al., 2004), control engineering (Puangdownreong et al., 2006; Sujitjorn and Khawn-on, 2006; Sarasiri et al., 2010), and signal processing (Sriyingyong and Attakitmongcol, 2006).

2.7 Applications of the hybrid TS algorithm with the inspired metaheuristics

In the last decade, the hybrid versions of the tabu search have been published in many journals. Most applications have been combined with the population-based metaheuristics following instances.

- In 1999, Thangiah developed the hybrid algorithm that combined the genetic algorithms, the simulated annealing (SA) and the tabu search methods for solving the vehicle routing problem with time windows (VRPTW). In the beginning, the genetic algorithms had been used as a global search method to find an initial solution. The next process, the initial solution was improved using a customer interchange method guided by tabu search combined with non-monotonic simulated annealing with an evaluation function that allowed for acceptance of infeasible solutions with a penalty. This hybrid algorithm was structured in the exploitation manner to strengthen the metaheuristic search strategies (Thangiah, 1999).

- In 2001, Lin et al. integrated the evolutionary programing (EP), the tabu search and the quadratic programing (QP) algorithms to solve nonconvex economic dispatch (NED) problem. This proposed hybrid algorithm had been divided in two phases. The first phase was the hybrid EP and TS to solve the combinatorial optimization, and the QP was used to solve the nonlinear optimization, especially the NED problem. Therefore, the QP algorithm was used to calculate the main cost function and evaluate the fitness function, while as the new feasible population was created by the mutation of the EP. Moreover, this hybrid algorithm used the feature of

the TS which are a memory structure and the tabu list strategy to avoid the local solutions during searching (Lin et. al., 2001).

- In 2007, Xiang et al. combined the heuristic algorithms between the tabu search and particle swarm optimization as the hybrid optimization algorithm. This proposed hybrid algorithm was based on the excellence of both the tabu search and the particle swarm optimization. The new hybrid algorithm had been defined in two stages. In the first stage, it was based on the sacrifice and memory property of the particle swarm optimization to make a global exploration. In the second stage, the tabu search was applied in finding the better particle around the global best particle for its efficiency in searching. This proposed algorithm achieved the termination when the setting maximal iteration times were reached (Xiang et al., 2007).

- In 2011, Di and Ze proposed a hybrid algorithm which had been combined the advantage of global search ability of GA with the self-adaptive merit of the TS to solve scheduling problems in flexible production environment. The TS algorithm was embedded in genetic algorithm, which made those individuals with effective gene deletion, and to balance the exploration and exploitation abilities of simple GA (Di and Ze, 2011).

- In 2011, Jat and Yang also published the hybrid genetic algorithm and tabu search approach denoted as HGATS to solve the post enrolment course timetabling problem (PECTP). The proposed HGATS algorithm works in two phases. The first phase was used to involve a population of candidate solutions by a guided search genetic algorithm (GSGA). In addition, some new neighbourhood structures and relevant local search strategies were integrated into the GSGA. The hybrid approach had also employed a second phase, where feasible solutions were found during the first

phase. The second phase had been inspired from the TS heuristic to improve the optimality of the solution. The experimental results had shown that the proposed hybrid approach was better than or comparable to all other tested methods (Jat and Yang, 2011).

- In recent year, Xiuli and Yanchi developed the new hybrid heuristic procedure based on the greedy random adaptive search procedure (GRASP), and the tabu search algorithm. This proposed hybrid algorithm had been applied to the irregular flight recovery which is a kind of NP-hard problems. The GRASP phase was used to improve the solution and also allowed the procedure to go on along the cost reduced direction in order to achieve the optimal solution. If the current solution found from the GRASP was not in the local optimization, the TS algorithm would execute the following steps via updated tabu list until a fixed given number of non-improving consecutive iterations is achieved. According to the comparison of the original GRASP method, the proposed hybrid algorithm had been shown quite high global optimization capability (Xiuli and Yanchi, 2012).

- In 2012, Katsigiannis et al. investigated the hybridization of two metaheuristic algorithms, namely the simulated annealing and the tabu search (SA-TS) for solving the small autonomous power systems (SAPS) sizing problem. This proposed hybrid SA-TA method had been combined the advantages of individual optimization methods in order to find the optimal solution in a fast and effective manner. The hybrid SA-TA algorithm used the SA to provide the initial solution. According to the inferiority of the SA, it is a kind of a stochastic method, but its performance does not quickly converge to the optimum region. For this reason, the SA was combined with the TS to proceed iteratively from one solution to another until a

given termination criterion is satisfied. According to the adaptive memory of the TS ensures that the search can escape from local optima. As the results showed that the proposed hybrid method had improved the solution quality without increasing significantly the number of required simulations (Katsigiannis et al., 2012).

2.8 Conclusion

Most of the literatures of hybrid metaheuristic algorithms have been combined between the population-based metaheuristics and the local search to solve various optimization problems. If only pure population-based metaheuristic, like the adaptive bacterial foraging optimization, may be not well suited to fine-tuned search in highly combinatorial problems and also weak in the exploitation of the solution found. In contrast, single-solution based metaheuristic such as the adaptive tabu search has no outstanding of exploration to entire search space. Therefore, the ability of the ABFO in term of exploration into the search space can be combined with the local search like the ATS with ability of exploitation in order to enhance the convergence rate and improve the quality of the solutions. The modified algorithms will be further explained in Chapter IV.

CHAPTER III

METAHEURISTICS

3.1 Introduction

The first proposed heuristic tried to help computer for testing a huge amount of combinations in short time duration. However, heuristic algorithms may fail to solve some complex optimization problems like NP-hard problems. In 1986, Glover developed heuristics firstly called *modern heuristics* and finally became *metaheuristics.* Unlike exact methods, metaheuristics are used to solve large-size problem instances by delivering satisfactory solutions in a reasonable time. A disadvantage of metaheuristics is that there is no guarantee to find (near) global optimal solutions or even bounded solutions but is rather reach to good solutions easily. Metaheuristics have received more and more popularity over the past 20 years. Examples of theses approaches are simulated annealing (Kirkpatrick et al., 1983), tabu search (Glover, 1989), iterated local search (Martin et al., 1992), and various population-based models such as evolutionary algorithms (Forgel, 1962), genetic algorithms (Holland, 1992), scatter search (Glover et al., 2000) and ant colony optimization (Dorigo et al. 1996) etc. Most metaheuristics have two contradictory criteria that must be taken into account: exploration of the search space (diversification) and exploitation of the best solutions found (intensification). Morever, there are many classification criteria that may be applied to metaheuristics. For examples, various criteria for classification could be nature-inspired versus non-nature inspired, population-based search versus single-solution search, one

versus various neighbourhood structures, memory usage versus memoryless methods, deterministic versus stochastic and iterative versus greedy. In this thesis, we describe the most important metaheuristics according to the population-based search versus single-solution search, which are divided into single-solution based metaheuristics and population-based metaheuristics, respectively. Thus, the purpose of this chapter is to introduce the metaheuristic search methods for solving hard combinatorial optimization problems. We start with a discussion on combinatorial optimization problems, and explain the classical optimization algorithms in Section 3.2. In this thesis, we focus on the approximate algorithms for both heuristics and local search. Thus, we present a concise history of the general principles of heuristics in Section 3.3, and the development of local search algorithms in Section 3.4. It is well-known that solutions of some cases of hard combinatorial optimization problems may not be improved by general approximate algorithms. Therefore, we end this chapter with an introduction to metaheuristics and a classification of metaheuristics. Moreover, this last Section also describes the common concepts of single-solution based and population solution-based metaheuristics.

3.2 Combinatorial Optimization Problems

An optimization problem is either a maximization or a minimization problem sometimes regarded as a problem of extremum solution finding, in which there is an objective function f on a set S of feasible solutions. If the S contains a finite set of feasible solutions and a cost function over the solutions, the problem is known as a combinatorial optimization problem. Combinatorial optimization problems in general can be defined by the couple $P= (S, f)$, where S represents the set of feasible solutions

or objects and the objective function to optimize f: $S \rightarrow \mathbb{R}$. The goal is to find a global optimal solution $s^* \in S$, which has a better cost value than that of any other solution in S, that is $\forall s \in S, f(s^*) < f(s)$. Most combinatorial optimization problems appear in many real-world applications. The algorithmic approaches to combinatorial optimization problems can be classified as either an exact (complete) algorithm or an approximate (heuristic) algorithm. Exact algorithms are guaranteed to find a (an optimal) solution in finite time but they might need exponential computation times in worst case. Examples of these are branch and bound algorithms, also known as A^* (A^*, IDA^* – iterative deepening algorithm), constraint programming and dynamic programming, etc. However, when the size of the problem instance increases, many of them belong to the NP-hard problems (or non-deterministic polynomial time), the exact algorithms can take up so much computing time that it becomes unacceptable. Thus, algorithms need to be designed for solving these problems, which require a reasonable amount of time and effort to achieve an optimal solution. As such, approximate methods are often an alternative, while these methods generate high-quality solutions within short computational time. Approximate methods are often used to find approximate solutions for the NP-hard problems. Although, they cannot guarantee a globally optimal solution, they provide near optimal solutions to a wide range of optimization problem in significantly reduced amount of time. There are two basic types of approximate methods, heuristic and approximation algorithms (Alba, 2005; Talbi, 2009). The classical algorithms for solving combinatorial optimization problems can be viewed as a branch chart in Figure 3.1.

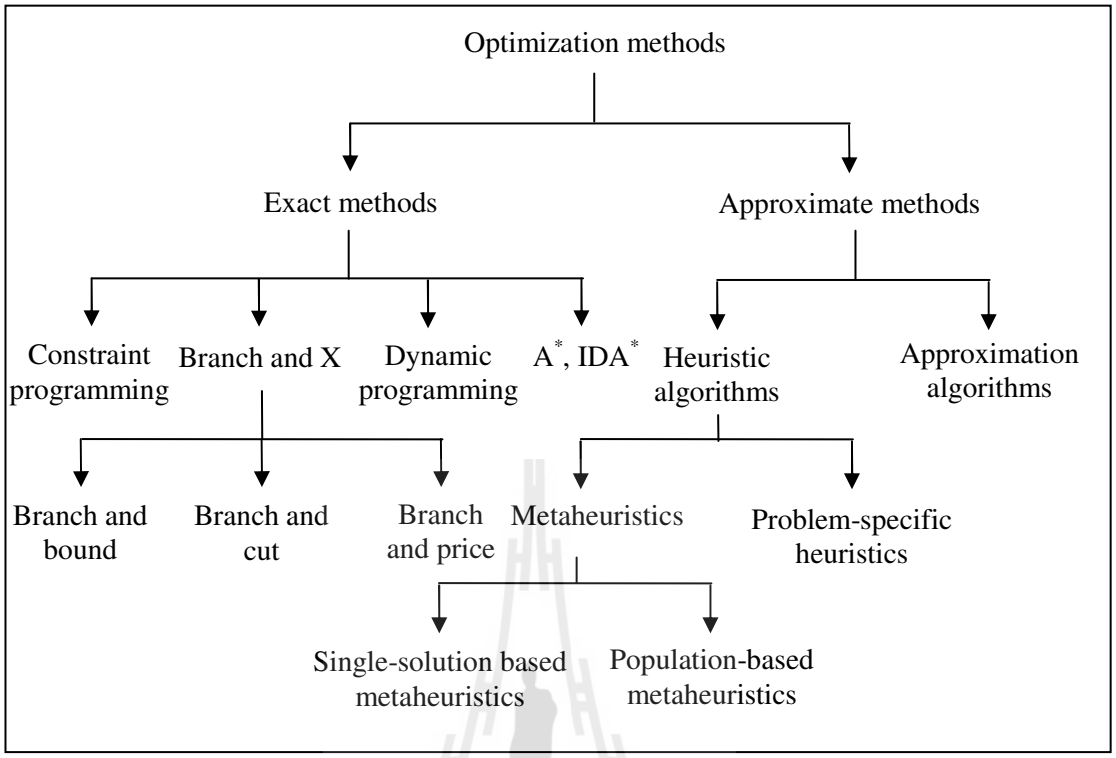


Figure 3.1 Classical optimization algorithms (Talbi, 2009).

3.2.1 NP-Complete Problems

The abbreviation of NP stands for to nondeterministic polynomial time. NP-complete problems are a set of each NP-problem (decision problems) which can be reduced in polynomial time, and whose solution may still be verified in polynomial time. This means that any NP problem can be transformed into any of the NP-complete problems. Basically, NP-complete problem is NP problem that is at least as hard as any other problem in NP. In the past, many decision problems were reduced their complexities which all these problems have a common property. This means that $P \subseteq NP$, where P is the set of problems which can be solved by polynomial time algorithms, and NP is the set of NP problems. If there are polynomial time algorithms

for any NP-complete problems then $P = NP$, and every NP problems can be solved in polynomial time. In some cases, NP may be equivalently defined as the set of decision problems that can be solved in polynomial time on a non-deterministic turing machine. The common belief nowadays is that $P \neq NP$ that can be solved with non-deterministic polynomial time algorithms or other algorithms running in super polynomial time. This is to say that NP-complete is a subset of NP. Thus, it is often said that the NP-complete problems are harder or more difficult than NP problems in general. For instances, some well-known NP-complete problems are travelling salesman problem, boolean satisfiability problem, n-puzzle, knapsack problem and vertex cover problem (Cook, 1971; Kann, 1992).

3.2.2 NP-Hard Problems

The NP-hard is an abbreviation of nondeterministic polynomial time hard. A problem which is harder than all NP problems or at least as hard as the hardest problems in NP. Such problems need not be in NP and indeed, they may not even be decision problems. It is called NP-hard problems, which are partly similar but more difficult problems than NP-complete problems. They do not belong to the class NP but all problems in the class can be reduced to them. Normally, the NP-hard problems require exponential computing time or even worse. Notice that NP-complete problems are a subset of NP-hard problems, and NP-complete problems are sometimes called NP-hard. The NP-complete and NP-hard are defined by the complexity theory studied within computer science. Both classes of NP-complete and NP-hard have been extensive efforts to find polynomial time algorithms. However, there remain a large number of problems in NP that defy such attempts, seeming to require super

polynomial time or exponential time. Whether these problems really are undecidable in polynomial time is one of the greatest open questions in computer science. It may happen that a problem nowadays known to be an NP-hard, will be proved to be an NP-complete in the future. This is due to new knowledge and advancement in computing innovation. Application problems of NP-hard are exampled as data mining, process monitoring and control, planning and tutoring systems as well as decision support (Hamalainen, 2006).

3.3 Heuristic Methods

3.3.1 A Brief History

The word *heuristic* originates from the old Greek word *heuriskein*, which means the art of discovering new strategies to solve problems. The first proposed heuristics tried to systematize decision-making processes and to help computers with testing a huge amount of combinations in a short amount of time. Heuristic strategies can also be designed to develop algorithms for optimization problems. The technical meaning of heuristic has undergone several changes in the history of artificial intelligence (AI). Originally, the term of heuristic was used to refer to the study of methods for discovering and inventing problem-solving techniques, especially those can be used to find mathematical proofs. In addition, heuristic was used as the opposite of algorithmic. It was defined as a process that may solve a problem, but offers no guarantee of solving it. Note that there is nothing random or nondeterministic about a heuristic search algorithm. Heuristic techniques dominated early applications of AI, which were viewed as rules of thumb that domain experts could use to generate good solutions without exhaustive searches (Russell and Norvig, 1995).

3.3.2 Constructive Heuristics

Constructive heuristics are typically the fastest approximate algorithms for solving combinatorial optimization problems. These algorithms generate solutions from scratch by adding opportunely defined solution components at each step starting from an empty initial solution. This algorithm is used until the solution is complete or the process is stopped by some criteria. The algorithm of a constructive heuristic is shown in Figure 3.2.

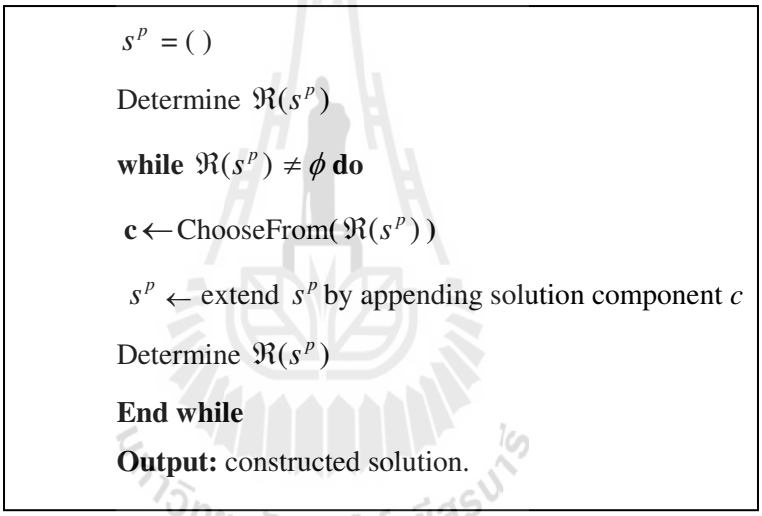


Figure 3.2 Generic algorithm of constructive heuristic.

Referring to Figure 3.2, first specifies the set of possible extensions for each feasible (partial) solution s^p. A set of solution components $\Re(s^p)$ can be derived for the extension of s^p. At each step, one of the possible extensions is chosen until $\Re(s^p) = \phi$, which means either that s^p is a solution or that s^p is a partial solution that cannot be extended to a feasible solution. A notable example of a constructive heuristic is a

greedy heuristic, which implements procedure *ChooseFrom(* $\Re(s^p)$ *)* by applying a weighting function. As a result a weighting function is a function that sometimes depends on a current (partial) solution and at each step assigns a heuristic value $\eta(c)$ to each solution component $c \in \Re(s^p)$ (Alba, 2005).

3.4 Local Search

Local search has become a widely accepted technique for the solution of hard combinatorial optimization problems. The strategy of local search algorithms starts from some solutions and iteratively tries to find a better solution in an appropriately defined *neighbourhood* of the current solution. A better solution is replaced the current solution, and the local search is continued from there until no better solution can be found in the neighbourhood of the current solution, meaning a local minimum is reached. The neighbourhood is formally defined as follows (Alba, 2005):

Definition 1 : ***A neighbourhood structure*** *is a function* $N : S \rightarrow 2^S$ *that assigns to each* $s \in S$ *at set of neighbouring solutions* $N(s) \subseteq S$*, where N(s) is the neighbourhood of s. Often, neighbourhood structures are implicitly defined by specifying the changes that must be applied to a solution s in order to generate all its neighbours. The application of such an operator that produces a neighbour* $s^* \in N(s)$ *of a solution s is commonly called a* ***move****.*

A neighbourhood with an instance of the problem defines a search space. A search space can be represented by a graph in which the nodes are solutions that are

labelled by the value of the objective function, and the arcs represent the neighbourhood relation between solutions. A solution $s^* \in S$ is called *globally minimal solution* (or global minimum) if for all $s \in S$ it holds that $f(s^*) \leq f(s)$.

Definition 2: ***A locally minimal solution*** *(or local minimum) with respect to a neighbourhood structure N is a solution* $\hat{s}$ *such that* $\forall s \in N(\hat{s}) : f(\hat{s}) \leq f(s)$. *And* $\hat{s}$ *is a strict locally minimal if* $\forall s \in N(\hat{s}) : f(\hat{s}) < f(s)$.

The template of a local search algorithm is shown in Figure 3.3. The algorithm generates an initial solution s_0 as a sequence $s_1, s_2, \ldots, s_n$.

```
s=s0      ; Generate an initial solution s0
While not Termination_Criterion Do
        Generate N(s); Generation of candidate neighbours
        If there is no better neighbour Then Stop
        s= s'     ; Select a better neighbour s' ∈N(s)
Endwhile
Output Final solution found (local optima).
```

Figure 3.3 Generic local search algorithm.

Local search can be seen as a descent walk on a directed graph representing the search space**,** which the movements connect each point with all its neighbours. The basic version of a local search is iterative improvement. This search starts with an initial solution, then selects a good neighbourhood for replacing the current solution,

and the search continues. If a local optimum is reached, the algorithm returns to the optimal solution.

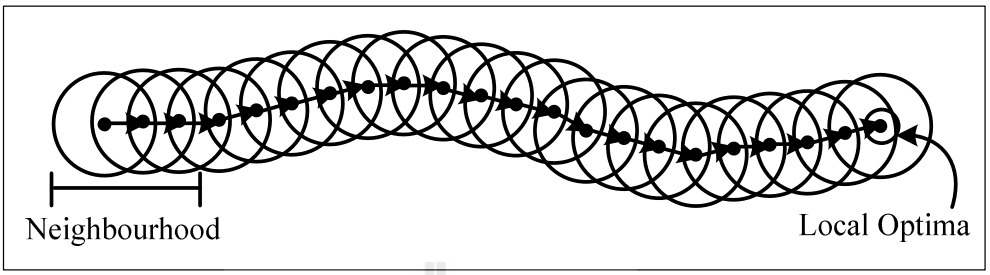


Figure 3.4 Basic iterative improvement scheme.

$s \leftarrow$ GenerateInitialSolution()
while $\exists s' \in N(s)$ **such that** $f(s') < f(s)$ **do**
    $s \leftarrow$ ChooseImprovingNeighbour($N(s)$)
end while
output: s

Figure 3.5 Framework of iterative improvement local search.

Figures 3.4-3.5 show a graphical depiction and framework of basic iterative improvement, respectively. Note that iterative improvement may not render a local optimal. To improve this, there are several variations made to the basic algorithm to define the initial solution and the neighbourhood, as explained below.

3.4.1 Selection of the Neighbour

There are several strategies to improve the selection function for a better neighbour *ChooseImprovingNeighbour(N(s)).* It is commonly found in the literatures, that researchers use first improvement, best improvement and random selection strategies.

- **First improvement:** In this strategy, the neighbourhood is generated incrementally and selected the first of better cost than the current solution *s*. In a cyclic exploration, the neighbourhood is evaluated in a deterministic way following a given order of generating the neighbourhood.

- **Best improvement** (steepest descent)**:** This strategy exhaustively explores the neighbourhood and selects the best solution with the lowest cost (minimization problem) within the neighbourhood. However, the exploration of a large neighbourhood may be very time-consuming.

- **Random selection:** In this strategy, a random selection is applied to choose the next neighbour. This approach is rather different from the others described so far in that it attempts to improve the current solution based on the corresponding cost.

To improve the quality of the solutions and the search time, one may use the first improvement strategy when the initial solution is randomly generated, and the best improvement strategy when the initial solution is generated using a greedy procedure. Based on many observed cases, the first improvement strategy leads to solutions of the same quality, whereas the best improving strategy uses a shorter computational time. Moreover, the probability of convergence speed to local optima is less important in the first improvement strategy. (Talbi, 2009)

3.4.2 Escaping from Local Optima

The local search is widely applied to hard combinatorial optimization problems because this method is easy to design and implement, and renders an elite solution in a considerably short time. Local search works well if there are not too many local optima in the search space or the qualities of the different local optima are more or less similar. Nevertheless, this kind of algorithm has a major drawback, i.e. it may converge to a local optimum, or it may stop and exit with a solution of poor quality. Moreover, this algorithm can be very sensitive to an initial solution. Thus in 1980s, there were many alternative algorithms proposed to improve search performance as summarized by the diagram in Figure 3.6.

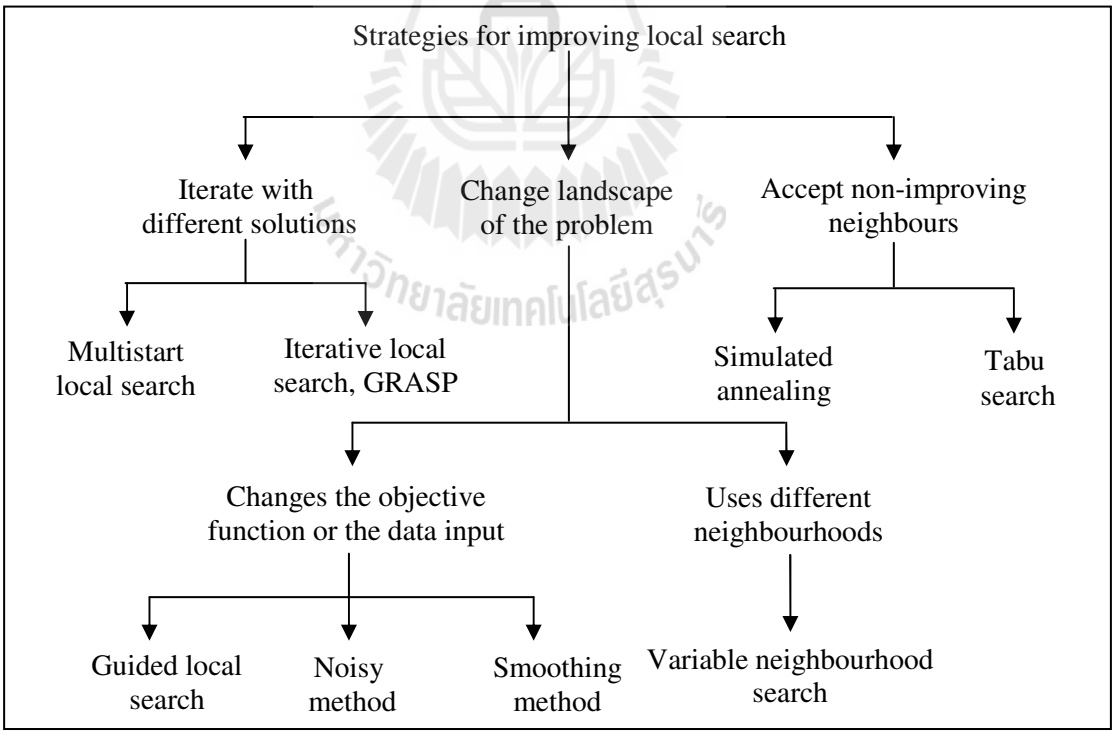


Figure 3.6 Family of strategies for escaping from local optima.

- **Iterating with different initial solutions:** This strategy is applied in multistart local search, iterated local search, GRASP etc.

- **Accepting non-improving neighbours:** These approaches provide a chance for the search to avoid being trapped by local minima. However, they may move out of the terrain containing a local optimum. Simulated annealing and tabu search are popular representatives of this class of algorithms.

- **Changing the neighbourhood:** The idea of these approaches consists of changing the neighbourhood during the search such as variable neighbourhood search strategies.

- **Changing the objective function or the input data of the problem:** In this strategy, the problem is transformed by perturbing the input data of the problem, and the objective function or the constraints to solve the original problem more efficiently. This approach has been implemented in the guided local search, the smoothing strategies, and the noisy method.

To avoid other possible shortcomings of the iterative improvement algorithms, one may employ one (or more) of the following tactics: (i) accept worse solutions in order to escape from a local entrapment, (ii) generate some good starting solutions to guide the search towards a better solution in subsequent iterations, (iii) accumulate search experiences (not possible with memoryless search) and use them to guide the search to achieve the goal efficiently. In general, these schemes are referred to as *metaheuristics*, which is explained next.

3.5. Metaheuristics

For over 20 years, a new kind of approximate algorithm has emerged which tries to combine basic heuristic methods in higher level frameworks aimed at efficiently and effectively exploring a search space. These methods are nowadays commonly known as metaheuristics (this word was firstly coined by Glover in 1986 (Glover, 1986)). The suffix *meta*, a Greek word, means upper level methodology. Before this term was adopted, metaheuristics were called *modern heuristics.* Metaheuristics may be viewed as upper level general methodologies that can be used as a guiding strategy for designing heuristics to solve combinatorial optimization problems by iterative attempts. Metaheuristics are powerful optimization techniques for solving a wide range of computationally complex optimizations and decision-making applications. Moreover, one can view metaheuristics as algorithms that perform directed random searches for possible solutions (near optimal) to a problem until a termination criterion is met. The goals of these algorithms are to escape from local minima efficiently in order to proceed with the explorations of the search space and to move on to find other better local minima. However, there is no guarantee of finding globally optimal solutions or even bounded solutions.

Successful metaheuristics rely on a good balance between *diversification* and *intensification* strategies. The term diversification is referred to as the exploration of the search space to identify regions with high quality or near optimal solutions, while the term intensification means the exploitation of the best solutions found. The balance between the two strategies is important because at first the search has to quickly identify regions in the search space with high quality solutions and, as the search goes on, it should avoid wasting too much time with already explored regions or with those

that fail to provide high quality solutions. Promising regions are determined by obtaining good solutions. A good balance between intensification and diversification should be found during the selection of the best solutions to improve the rate of algorithm convergence. The selection of the best ensures that solutions will converge to the optimum, while diversification via randomization allows the search to escape from local optima and, at the same time, increases the diversity of solutions. A good combination of these two major components will usually ensure that global optimality is achievable. In terms of intensification, the promising regions of search space are explored more thoroughly which are extremely local search. In diversification, all regions of the search space are evenly explored so that the search is not confined to only a reduced number of regions. Some of the well-known search methods explained later in this chapter can be classified along a spectrum line of diversification-intensification as shown in Figure 3.7. Random search leans toward diversification, whilst local search is rather an intensification-based method. Random search generates a random solution within search space without search memory. On the other hand, local search selects the best solution that improves the current solution. Population-based and single-solution based methods lie in the middle of the spectrum. In general, basic single-solution based searches are more intensification (or exploitation) oriented, whereas basic population-based searches are more diversification (or exploration) oriented (Alba, 2005; Talbi, 2009). The diagram in Figure 3.7 below is helpful for the researcher to select or design metaheuristics to suit applications.

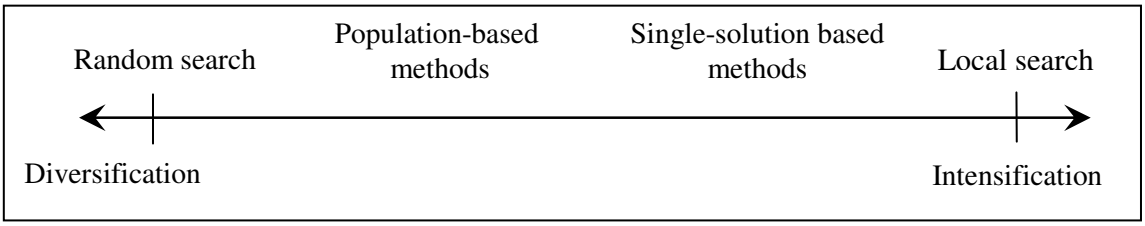


Figure 3.7 Diversification-intensification spectrum of metaheuristics.

3.5.1 Classification of Metaheuristics

There are different ways to classify and describe metaheuristic algorithms refer to (Alba, 2005; Blum and Roli, 2003; Talbi, 2009):

- **Nature-inspired versus non-nature inspired:** Many algorithms are actually inspired by naturally occurring phenomena. The algorithmic approaches try to imitate these phenomena to efficiently achieve elite solutions for combinatorial optimization problems. There are several nature-inspired algorithms, including evolutionary algorithms, evolution strategies, evolutionary programming, neural networks, genetic algorithms, ant colony optimization, simulated annealing, bacterial foraging optimization and invasive weed optimization, etc. Non-nature inspired algorithms include tabu search, iterated local search, guided local search, variable neighbourhood search and greedy randomized adaptive search procedures (GRASP), for instance.

- **Single-solution search versus population-based search:** The distinction between these methods is the number of solutions used at a time. Algorithms working on single solution are called trajectory-based methods, and encompass local search-based metaheuristics, like tabu search, iterated local search,

simulated annealing and variable neighbourhood search. They all share the property of describing a trajectory in the search space during the search process. On the contrary, population-based algorithms perform search processes which describe the evolution of a set of points in the search space.

- **One versus various neighbourhood structures:** Most metaheuristic algorithms are based on a single neighbourhood structure. In other words, the fitness landscape topology does not change in the course of the algorithm. This approach may limit diversification of the search. Other metaheuristics, such as variable neighbourhood search, use a set of neighbourhood structures which gives the possibility of diversifying the search by swapping between different fitness landscapes.

- **Memory usage versus memoryless methods:** One distinctive feature making the search methods different is whether they have memory or not. Some memoryless methods are local search, simulated annealing and GRASP, for instance. The search moves of these methods rely on the current information about the solution and the search trajectory. Search history cannot be stored or retrieved elsewhere even though it could be useful in accelerating the search. Some metaheuristics use memory to influence the future search direction. As a common case, the memory contains some useful information, such as visited elite solutions etc., that can be extracted online to accelerate the search. An example is tabu search that uses short-term and long-term memories.. Short-term memory is used to forbid revisiting recently found solutions and to avoid cycling, while long-term memory is used for diversification and intensification features.

- **Deterministic versus stochastic:** A deterministic metaheuristic is used to solve an optimization problem by making deterministic decisions like local

search and tabu search etc. Some random rules of a stochastic metaheuristic are applied during the search, like simulated annealing and evolutionary algorithms etc. In the case of deterministic metaheuristic, using the same initial solution will lead to the same final solution, whereas in stochastic metaheuristic, different final solutions may be obtained from the same initial solution.

- **Iterative versus greedy:** Iterative algorithms start with a complete solution or a population of solutions, and transform every iteration. Greedy algorithms start from an empty solution, and construct a solution by assigning values to one decision variable of a problem at each step until a complete solution is obtained. In general, most of the metaheuristics are iterative algorithms.

3.5.2 Single-Solution Based Metaheuristics

As mentioned before, the single-solution based approach is classified as a trajectory method because the search process can be characterized by search paths or search trajectories through the search space. Trajectories are performed by iterative procedures that move from the current solution to another. The common concepts of the single-solution based metaheuristics are represented by the procedural list in Figure 3.8. At the first start, an initial solution s_0 is randomly generated. Next, the algorithm produces current solutions s defined as a set of candidate solutions. This set $C(s)$ is generally obtained by local transformations of the solutions. In the replacement process,

Input: Initial solution s_0.
$t = 0$;
Repeat
Generate($C(s_t)$) ; Generate candidate solutions (partial or complete neighbourhood) from s_t
s_{t+1} = Select($C(s_t)$) ; Select a solution from $C(s)$ to replace the current solution s_t
$t=t+1$;
Until Stopping criteria satisfied
Output: Best solution found

Figure 3.8 Procedural list of single-solution based metaheuristics.

a solution $s' \in C(s)$ is selected to replace the current solution, and designated as new solution. The generation and the replacement processes maybe memoryless. In such a case, they are based only on the current solution. Otherwise, some searches employ memory for storing the historical generation of the candidate solution list and the historical selection of the new solution for further use. The most popular examples of the single-solution based metaheuristics are local search, simulated annealing and tabu search.

3.5.3 Population-Based Metaheuristics

Population-based metaheuristics can be viewed as an iterative improvement of a population of solutions. The search process starts from an initial population of solutions usually generated randomly. Then, a new population is generated and replaced by selection of the current population. The process is stopped when a given condition is satisfied. The generation and the replacement processes are based on the current population if they do not employ memory. Otherwise, historical information about visited elite solutions and search moves can be memorized and retrieved for search improvement. For those algorithms imitating biological breeding processes, they are referred to as evolutionary algorithms. Most of population-based metaheuristics are natural-inspired algorithms, for example evolutionary algorithms, ant colony optimization, scatter search, estimation of distribution algorithms, particle swarm optimization, bee colony, and artificial immune system. The procedural list shown in Figure 3.9 provides the framework for this family of algorithms.

$P = P_0$; Generation of the initial population
$t = 0$;
Repeat
  Generate(P'_t) ; Generation a new population
  P_{t+1} = select-Population($P_t \cup P'_t$) ; Select new population
  $t = t+1$;
Until Stopping criteria satisfied
Output: Best solution (s) found

Figure 3.9 Framework of population-based metaheuristics.

3.6 Conclusion

This chapter has presented the background to metaheuristics. It explains combinatorial optimization problems, and gives a summary of algorithms used to solve these problems. Metaheuristics play a major role in solving these problems usually NP-hard ones in an approximate manner. This chapter presents a brief historical background to heuristics, and elaborates on heuristic and metaheuristic algorithms, respectively. Generic algorithms of some important metaheuristics are also given. Noticeably, metaheuristic methods will gain more and more popularity in the future as optimization problems are increasing in size and in complexity.

CHAPTER IV

DESCRIPTIONS OF THE ALGORITHMS

4.1 Introduction

In recent years, researchers have used many metaheuristic algorithms to solve complex optimization and NP-hard problems. Some of those algorithms are inspired by nature and, among those, the bacterial foraging optimization (BFO) is well-known. Some published papers have reported a deficit of the original BFO in that under some situations it takes a very long time to render an elite solution, in other words, the BFO needs a very large number of iterative loops to track down the global solution. This problem has been resolved by introducing an adaptive jump in the chemotaxis step of the BFO. The modified version is known as the adaptive BFO, or ABFO, later presented in Section 4.2.

Tabu search (TS) is one of the single-solution based metaheuristics that has demonstrated many successful real-world applications as evidenced by a vast number of publications worldwide. The simplistic TS occasionally encounters an undefeatable local trap. Correspondingly, the algorithm moves around a local solution endlessly without any improvement, otherwise hits the iteration limits. The drawback has been overcome by introducing the backtracking (*BT*) and the adaptive search radius (*AR*) mechanisms into the TS. This modified version of the TS has been referred to as the adaptive tabu search, or ATS, later described in Section 4.3. Based on previous descriptions, the explorative characteristic of the ATS is rather limited due to the preset

length of the search radius. In contrast, the ABFO algorithms with the chemotaxis mechanism provide a rather thorough exploration on the search space. Therefore, both algorithms could complement each other to achieve an efficient search. Thus this thesis proposes cooperative algorithms based on the ABFO and the ATS, which are detailed in Section 4.4 and Section 4.5. Moreover, Sections 4.6 and Section 4.7 of this chapter also describe the algorithms used for comparison studies of the search performance that include invasive weed optimization algorithm (IWO) and genetic algorithm (GA), respectively.

4.2 Bacterial Foraging Optimization Algorithm

The BFO algorithm is regarded as a population-based method and imitates the foraging behaviour of bacteria. It consists of four main steps namely chemotaxis, swarming, reproduction and elimination-dispersal, respectively. The chemotactic step is regarded as the major step of the BFO that imitates the swift movement of bacteria by a fixed distance or height. It is a foraging strategy that implements a type of local optimization procedure, and resembles a biased random walk mode. Since the basic BFO is usually unable to provide a fine-quality local solution and oscillates when it approaches the global optimum point in case of a large step size, and thus has low accuracy. On the other hand, if a small step size is defined, the speed of convergence of the algorithm becomes slower. Moreover, under some complex circumstances, this basic BFO may take very long time to reach a satisfactory solution. To resolve this, some modifications have been made to the BFO to have an adaptive chemotatic step. Thus, a suitable strategy to cope with this problem is to apply a big step size when the cost function value is large so that the bacterium climbs down the hill faster and then

apply a very small step size when the bacterium is near the optimum point to ensure the bacterium is able to find the optimum point. There is an interesting modification from literature survey (Dasgupta et al., 2009; Dasgupta et al., 2010; Majhi et al., 2009) which has been known as the adaptive BFO or ABFO algorithm. This modified version is more efficient than the conventional BFO. The chemotactic step size varying as a function of current fitness value can provide a fine-quality solution with a better convergence rate than that achieved with a fixed step size, i.e., the value of chemotactic step size changes based on the cost function (nutrient value). If the cost function value is high then the step size is large and if the cost function value is low then the step size is small. By applying this mechanism, the ABFO will be faster in convergence and will also be able to reach the global optimum. Based on this concept, the adaptive step size, *C(i),* is defined by

$$C(i) = \frac{C_{\max} \cdot \left| J(\theta^i) \right|}{\left| J(\theta^i) \right| + \alpha} = \frac{C_{\max}}{1 + \frac{\alpha}{\left| J(\theta^i) \right|}} \tag{4.1}$$

where α is a positive constant. $C_{\max}$ is tune-able maximum chemotactic step size which is applied to a large search space. If the global optimum of the cost function is equal to zero, from equation (4.1). $J(\theta^i) \rightarrow 0$, then $C(i) \rightarrow 0$. Therefore, there would be no oscillation if the bacterium reaches an optimum point because the random search term vanishes as $C(i) \rightarrow 0$. The functional form given in equation (4.1) causes *C(i)* to vanish near the optimum. On the other hand, when $J(\theta^i)$ is large, $\alpha / |J(\theta^i)| \rightarrow 0$, and consequently $C(i) \rightarrow C_{\max}$ (> 0). The adaptive chemotatic step in equation (4.1) has an

important physical significance that this adaptation scheme helps to avoid the oscillation of the bacterium near optima and accelerates its convergence speed. If the magnitude of the objective function is large for an individual bacterium, it is in the vicinity of a noxious substance. It will then try to move to a place with better nutrient concentration by taking large steps. On the other hand, when the bacterium is in a nutrient-rich zone, i.e. with small magnitude of the objective value, it tries to retain its position. Naturally, its step size becomes small. Dasgupta et. al. have illustrated the effectiveness of the above mentioned adaptive scheme over several benchmark functions (Dasgupta et. al., 2009; Dasgupta et. al., 2010). The ABFO algorithm consists of 4 main mechanisms as follows:

- **Chemotaxis-**This mechanism imitates the swimming and tumbling movements of a bacterium via flagella. Biologically an *E.coli* bacterium can move in two different ways. Let j be the index for the chemotactic step, k be the index for the reproduction step, and l be the index of the elimination-dispersal event. Define

$$P(j,k,l) = \left\{ \theta^i(j,k,l) \middle| i = 1,2,...,S \right\} \tag{4.2}$$

to represent the position of each member in the population of the S bacteria at the j^{th} chemotactic step, k^{th} reproduction step, and l^{th} elimination-dispersal event. Here, let $J(i,j,k,l)$ denote the cost at the location of the i^{th} bacterium $\theta^i(j,k,l) \in \Re^p$. Let N_c be the length of the lifetime of the bacteria as measured by the number of chemotactic steps they take during their life. To represent a tumble, a unit length random direction, $\phi(j) = \Delta(i) \Big/ \sqrt{\Delta^T(i)\Delta(i)}$, is generated; this will be used to define the direction

of movement after a tumble. An *E.coli* bacterium can move in two different ways: it can swim (run) for a period of time in the same direction or it may tumble, and alternates between these two modes of operation for the entire lifetime. In the BFO, a unit walk with random direction represents a tumble, and a unit walk in the same direction indicates a run showing in Figure 4.1. In computational chemotaxis, the movement of the i^{th} bacterium after one step is represented as

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\phi(j) \tag{4.3}$$

so that *C(i)* is the size of the step taken in the random direction specified by the tumble (see equation (4.1)). If at $\theta^i(j+1,k,l)$ the cost $J(i, j+1,k,l)$ is better (lower) than that at $\theta^i(j,k,l)$, then another step of size *C(i)* in this same direction will be taken, and again, if that step results in a position with a better cost value than at the previous step, another step will be taken. This swim continues as long as the cost continuously decreases, but only up to the maximum number of steps, N_s. This represents that the bacterium will tend to keep moving if it is headed in the direction of increasingly favorable environments (Passino, 2002; Liu and Passino, 2002).

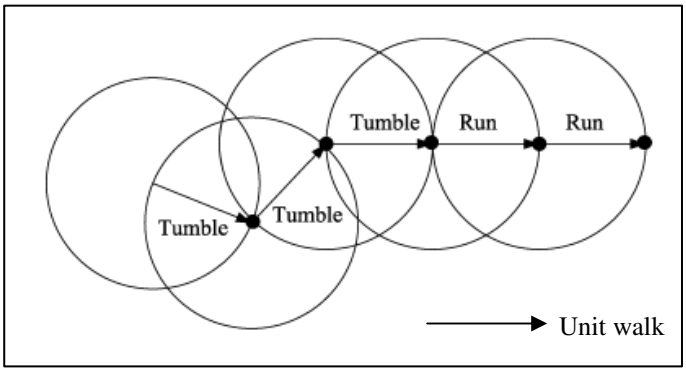


Figure 4.1 Swimming and tumbling movements.

- **Swarming-**When one bacterium presents itself in an elite position, i.e. a local hill or valley, it attracts the other bacteria to that location. Regarding this, the *E. coli* cells send attraction signals to each other so that they swarm together. Two stimuli, i.e. cell-to-cell signaling and foraging, affect the swarming pattern. In terms of algorithmic approach, this swarming to an elite location is achieved through an objective function adjustment depending on the relative distance between each bacterium and the healthiest one. Swarming helps bacteria congregate into groups and move as concentric patterns. Simultaneously, each bacterium also releases repellant to signal the others nearby. Thus, all of them will have a cell-to-cell attraction via attractant, and cell to cell repulsion via repellant. The attractive and the repellent effects are modeled as weighted summation of exponential terms representing the objective function, J_{cc}. The cell-to-cell signaling in *E. coli* swarm can be mathematically represented by

$$J_{CC}\left(\theta, P(j,k,l)\right) = \sum_{i=1}^{S} J_{CC}^{i}\left(\theta, \theta^{i}(j,k,l)\right)$$
$$= \sum_{i=1}^{S}\left[-d_{attract}\exp\left(-w_{attract}\sum_{m=1}^{p}\left(\theta_m - \theta_m^i\right)^2\right)\right] \quad (4.4)$$
$$+ \sum_{i=1}^{S}\left[h_{repellent}\exp\left(-w_{repellent}\sum_{m=1}^{p}\left(\theta_m - \theta_m^i\right)^2\right)\right]$$

where $J_{CC}\left(\theta, P(j,k,l)\right)$ can be seen as an updating term to the actual objective function that would be minimized to represent a time varying objective function. The terms $d_{attract}$, $w_{attract}$, $h_{repellant}$ and $w_{repellant}$ control the strength of the cell-to-cell signaling. More specifically $d_{attract}$ is the depth of the attractant released by the cell, $w_{attract}$ is a measure of the width of the attractant signal (a quantification of the diffusion rate of the chemical), $h_{repellant} = d_{attract}$ is the height of the repellant effect (a bacterium cell also repels a nearby cell in the sense that it consumes nearby nutrients, and it is not physically possible to have two cells at the same location), and $w_{repellant}$ is a measure of the width of the repellant. The weighting factors are $d_{attract}$, $w_{attract}$, $h_{repellant}$ and $w_{repellant}$, and can be chosen arbitrarily. As Passino's suggestion, the swarming parameters are constantly considered such that $d_{attract} = h_{repellant} = 0.1$, $w_{attract} = 0.2$ and $w_{repellant} = 10$.

- **Reproduction-**After all N_c chemotactic steps have been covered, a reproduction step is taken. Let N_{re} be the number of reproduction step to be taken. The bacteria are classified during the computing process as healthy and unhealthy due to their cost values. For the sake of convenience, we assume that the number of bacteria in the population, S, is a positive even integer. Define $S_r = S/2$ be the number of population members with sufficient nutrients so that they will reproduce (split into two) with no mutations. During the process of reproduction, the bacterial population are

sorted in order to ascend accumulated cost values, then the worse half of the population, S_r, containing the least healthy bacteria, dies while the other of the better half S_r split into two at the same location. This keeps the population of the bacteria constant.

- **Elimination and dispersal-**Since bacteria may stick around the initial or local optimum positions, it is required to diversify the bacteria either gradually or suddenly so that the possibility of being trapped into local minima is eliminated or reduced. Therefore, after the reproduction process, the healthy bacteria are dispersed randomly within the search space with the probability P_{ed}, whereas the unhealthy ones are discarded. With this approach, it is expected that the search could escape from a local entrapment.

The procedural list below provides the ABFO algorithm (see the flowchart in Figure 4.2).

Step0**:** Initialization of parameters: *p, search space, S,* N_c, N_s, N_{re}, N_{ed}, α ,$d_{attract}$, $w_{attract}$, $h_{repellant}$ and $w_{repellant}$.

Step1**:** Iterative algorithm for optimization.

Elimination-dispersal loop: $l = l + 1$.

Reproduction loop: $k = k + 1$.

Chemotaxis loop: $j = j + 1$.

(a) For $i = 1,2,\ldots,S$, take a chemotactic step for bacterium i as follows:

(a.1) Compute and update objective functions according to (4.5).

$$J(i,j,k,l) = J(i,j,k,l) + J_{CC}\left(\theta^{i}(j,k,l), P(j,k,l)\right) \tag{4.5}$$

$$J_{last} = J(i,j,k,l) \tag{4.6}$$

(a.2) Tumble: generate randomly ([-1,1]) the elements of the random vector $\Delta_m(i) \in \Re^p$, $m=1,2,\ldots,p$.

(a.3) Move: compute the adaptive step size, *C(i)*, according to (4.1) and update the location, θ^i, of a bacterium.

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^{\mathrm{T}}(i)\Delta(i)}} \tag{4.7}$$

(a.4) Compute the objective function

$$J(i,j+1,k,l) = J(i,j+1,k,l) + J_{CC}\left(\theta^{i}(j+1,k,l), P(j+1,k,l)\right) \tag{4.8}$$

(a.5) Swim: reset swim length counter, $m=0$; while $m < N_S$ update m, if $J(i,j+1,k,l) < J_{last}$, assign $J_{last} = J(i,j+1,k,l)$ and compute

$$\theta^i(j+1,k,l) = \theta^i(j+1,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^{\mathrm{T}}(i)\Delta(i)}} \tag{4.9}$$

Use the $\theta^i(j+1,k,l)$ and compute the new $J(i,j+1,k,l)$ according to (4.5). Go to step (a.1), if m = N_S.

Step2: If $j < N_c$, repeat the chemotaxis loop in step 1. (Continue chemotaxis since the lives of the bacteria are not over).

Step3: Reproduction:

For i = 1,2,…,S.

- For the given k and l, and for each i = 1,2,…,S, compute

$$J_{health}^{i} = \sum_{j=1}^{N_c+1} J(i,j,k,l) \tag{4.10}$$

- Do max-min sorting for J_{health}.
- Classify half of the S_r bacteria as healthy (with low values of J_{health}) and the other half as weak; split the healthy ones into two at their current locations.

Step4: If $k < N_{re}$, repeat the reproduction loop in step1.

Step5: Elimination-dispersal:

- Dispose the weak bacteria resulted from the classification in step3 (ones with high values of J_{health}).
- Assign P_{ed}= 0.25. For i=1,2,...,S: randomly generate *rand*; if P_{ed}>*rand* disperse the bacteria to random locations θ^i, else do nothing.

Step6: If $l < N_{ed}$ repeat the elimination-dispersal loop in step1; otherwise end.

The flowchart of ABFO is illustrated in Figure 4.2.

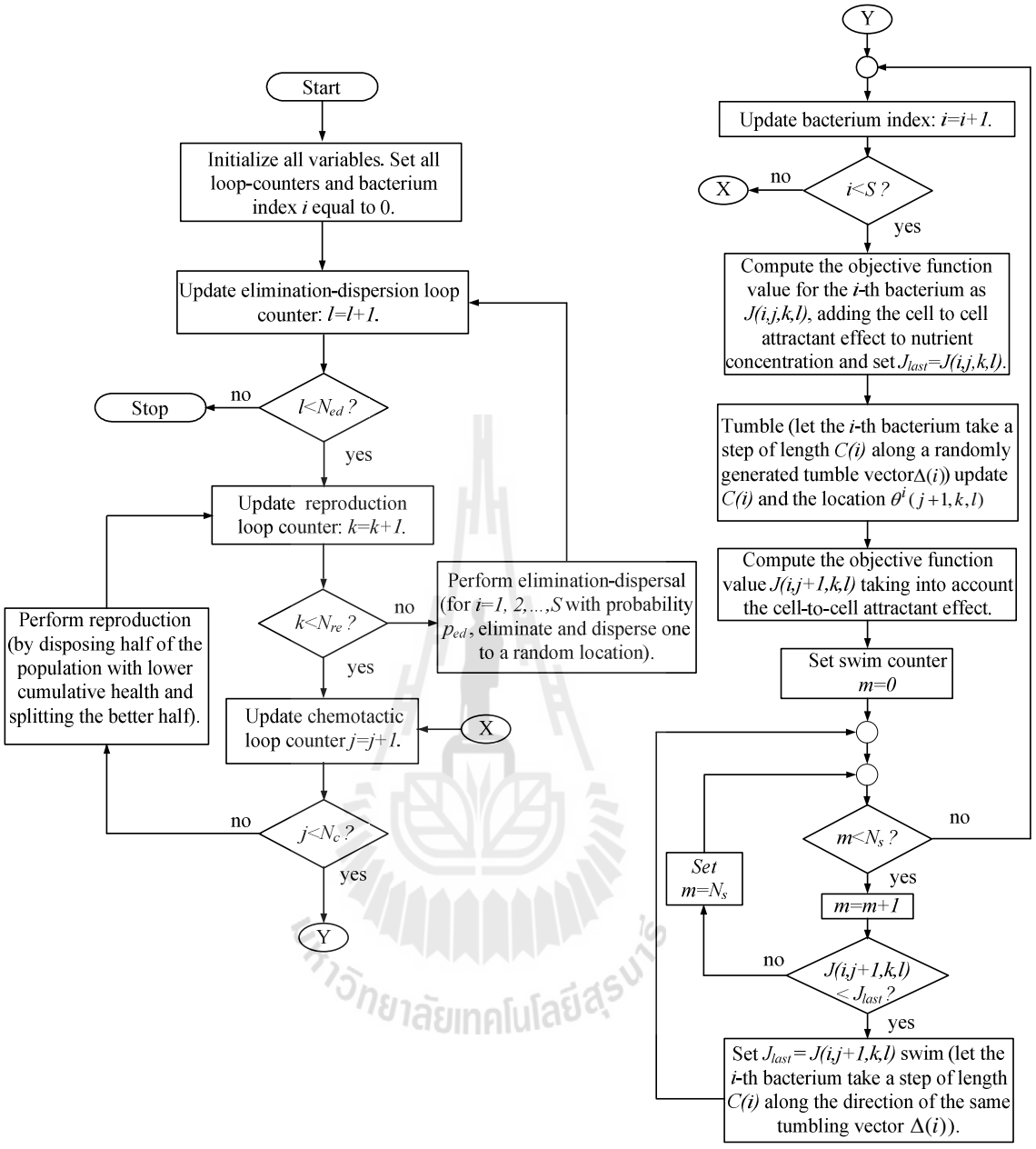


Figure 4.2 Flowchart of the ABFO algorithm.

4.3 Tabu Search Algorithm

The TS is suitable for combinatorial optimization problems containing many local solutions. Through an iterative process of neighbourhood search, the TS moves

toward better and better local solutions. The search stops according to some termination criteria that could be maximum iteration, cost threshold, etc. As a memory-based method, the TS employs a memory block called tabu list (*TL*) storing the characteristics of elite solutions previously visited. Judicious design of the *TL* and how to utilize it leads to a fast and efficient search. In general, two main strategies recommended for efficient searches are intensification and diversification strategies, respectively (Glover, 1989; Glover, 1990). The conventional TS is usually unable to escape from a local trap, or may succeed with a considerably long time taken. One efficient modification to resolve the problem is the approach used by the ATS, in which the backtracking (*BT*) and the adaptive search radius (*AR*) mechanisms have been added. The *BT* mechanism permits the search to use an elite solution sorted in the *TL* as a new initial solution. The search can immediately escape from a solution deadlock, and thus increase the search efficacy. However, the new solution which is selected as the current solution is not necessary to be the best solution in the current search space. The *AR* mechanism decreases the search radius during the process running until the search comes close to the global optimum. A long radius is suitable for a sparse visit to neighbouring solutions in order to find some solution candidates while the quality is not a prime interest. To find a high-quality solution, a short radius is more appropriate. The search radius should be adjusted to suit the problems, and increase the effectiveness of the ATS. The following procedural lists represent the ATS algorithms.

Step0: Initialize search parameters: *R*, *search space*, *N*, *TL*, $count_{max}$, *BT*, *n_re_back*, *best_neighbour, best_error,* R_i and ε_i. Define S_0 = *best_neighbour*.

Step1: Randomly or heuristically select an initial solution S_0 from the search space. Set S_0 as a current solution or *best_neighbour* with its cost J_0.

Step2: Generate a neighbourhood.

For *count* = 1,2,…, $count_{max}$.

Generate a neighbourhood with an initial search radius *R*. Set *N* solutions as the members of the set $S_1(r)$.

Step3: Evaluate the objective function (*J*) of each member in $S_1(r)$. Define S_1 = *best_neighbour1* as a neighbour solution with the minimum cost, J_1.

Step4: If $J_1 < J_0$, store S_0 in the *TL*, assign $S_0 = S_1$, otherwise, store S_1 in the *TL*.

Step5: Invoke the *BT* when a solution deadlock occurs. If the current solution has been repeated a number of times as defined by $n = 1,2,\ldots, BT$.

BT: **if** $n \geq BT$

$n=n+1$

best_error = *RANK(TL)*

look back in the *TL*, then retrieve the $n\_re\_back^{th}$ solution from the *TL*.

else

$n=0$

define S_0 = *best_neighbour*

best_error = *best_error*

end if

(A better solution is selected as a new initial solution for the next search to start with.)

Step6: If the termination criterion is met, exit with the global solution.

Step7: Invoke the *AR* when the current solution S_0 is relatively close to a local minimum.

AR: **if** $best\_error < \varepsilon_1$

$R=R_1$ where $R<R_1$.

end

if $best\_error < \varepsilon_2$

$R=R_2$ where $R_2<R_1$ and $\varepsilon_2 < \varepsilon_1$.

end

…

if $best\_error < \varepsilon_n$

$R=R_n$ where $R_n<R_{n-1}$ and $\varepsilon_n < \varepsilon_{n-1}$.

end

Step8: Update iteration counter (count), and go to Step2.

The flowchart of ATS is shown in Figure 4.3.

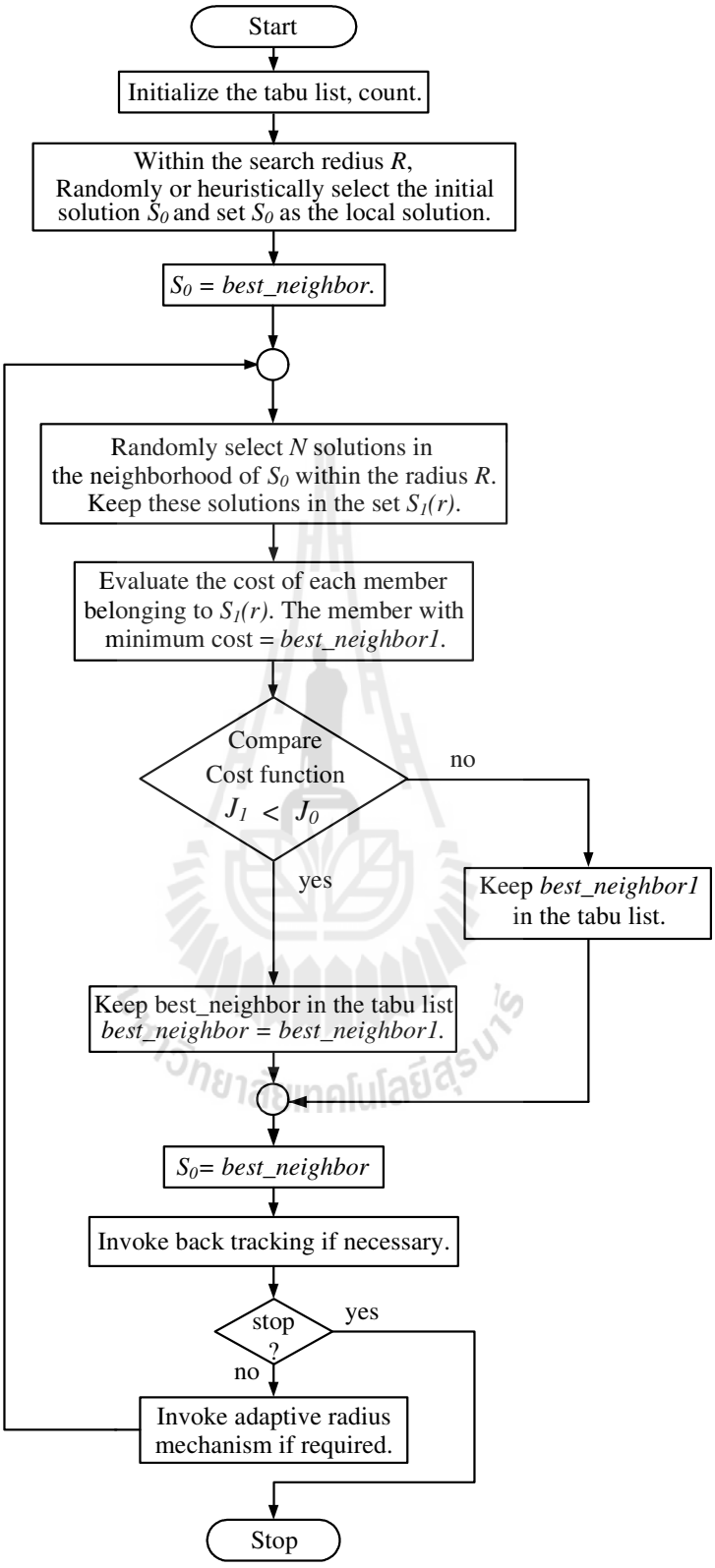


Figure 4.3 Flowchart of the ATS algorithm.

4.4. Cooperative Bacterial Foraging-Tabu Search Algorithm

A lot of metaheuristic approaches have been explored in the last two decades in order to tackle large size optimization problems. These areas include hybrid metaheuristic and cooperative search algorithms. As previously mentioned, the ATS has a dominant focusing or exploitative characteristic, while the ABFO is strong in explorative operation. Such properties can complement each other. Since the ATS has straightforward procedures, and moves rapidly towards a local solution, this method is used to perform the 'hunting' steps and search for a satisfactory solution to the problem. The two algorithms are combined to form new metaheuristics working in a cooperative manner. In this new algorithmic form, ranking the available solutions to single out one with the minimum cost is an important step, it is therefore unnecessary to employ the reproduction and elimination-dispersal mechanisms of the ABFO. This specific solution is transferred to the ATS part as an initial solution. It means that the ABFO is applied first to collect promising initial solutions, and the ATS is applied afterward to track down a global solution. Furthermore, the *BT* mechanism of the ATS still remains a handy tool to escape a local solution lock. The two algorithms are combined, which the ABFO algorithm operates once to provide a high-quality initial solution for the ATS. Such algorithm is referred to shortly as BF-TS. The procedural list of the BF-TS algorithm is as follows:

Step0: Initialize search parameters: *p, search space, S,* N_c, N_s, α ,$d_{attract}$, $w_{attract}$, $h_{repellant}$, $w_{repellant}$, *R*, *N*, *TL*, $count_{max}$, *BT*, *n_re_back, best_neighbour1, best_error,* R_i and ε_i .

Step1: Randomly or heuristically select an initial solution θ^i from the search space. Set θ^i as the current solution.

Step2: Compute objective functions $J(i,j)$ according to equation (4.11)-(4.12) (i=1,2,…,S). Set $J_{last} = J(i,j)$; j=1,2,… N_C.

$$J(i,j) = J(i,j) + J_{CC}\left(\theta^i(j), P(i,j)\right) \tag{4.11}$$

$$\begin{aligned} J_{CC}\left(\theta^i, P(i,j)\right) &= \sum_{i=1}^{S} J_{CC}^{i}\left(\theta, \theta^i(j)\right) \\ &= \sum_{i=1}^{S}\left[-d_{attract}\exp\left(-w_{attract}\sum_{m=1}^{p}\left(\theta_m - \theta_m^i\right)^2\right)\right] \\ &+ \sum_{i=1}^{S}\left[h_{repellent}\exp\left(-w_{repellent}\sum_{m=1}^{p}\left(\theta_m - \theta_m^i\right)^2\right)\right] \end{aligned} \tag{4.12}$$

Step3: Generate randomly ([-1,1]) the elements of the random vector $\Delta_m(i) \in \Re^p$, m=1,2,…,p, then compute the adaptive step size, $C(i)$ using equation (4.13), and update the solution $\theta^i(j+1)$ according to equation (4.14). Compute the objective function for $j = j+1$ according to equation (4.11)-(4.12). Set m=0.

$$C(i) = \frac{C_{\max}\cdot\left|J(\theta^i)\right|}{\left|J(\theta^i)\right| + \alpha} = \frac{C_{\max}}{1 + \dfrac{\alpha}{\left|J(\theta^i)\right|}} \tag{4.13}$$

$$\theta^i(j+1) = \theta^i(j) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^{\mathrm{T}}(i)\Delta(i)}} \tag{4.14}$$

Step4: If $J(i, j+1) < J_{last}$ then $J_{last} = J(i, j+1)$; use the direction of the same random vector $\Delta(i)$ to compute $\theta^i(j+1)$ and $J(i, j+1)$. Update *m* and repeat Step 4 until $m > N_s$.

Step5: If $j \leq N_c$, go to Step 2.

Step6: Do minimum sorting of the objective functions *J*. Define *best_ neighbour* (elite solution from sorting) as the solution with the minimum *J*. Set S_0=*best_neighbour*.

Step7: Generate a neighbourhood around S_0 within an initial search radius *R*. Set *N* solutions as the members of the set $S_1(r)$.

Step8: Evaluate the objective function of each member belonging to $S_1(r)$. Define S_1 = *best_neighbour1* as a solution with the minimum cost, J_1.

Step9: If $J_1 < J_0$, store S_0 in the *TL*, assign $S_0 = S_1$, otherwise, store S_1 in the *TL*.

Step10: Invoke the *BT* when a solution deadlock occurs (the current solution has been repeated many times as defined by *n*= 1,2,…, *BT*).

BT: **if** $n \geq BT$

$n=n+1$

best_error = RANK(TL)

look back in the *TL,* then retrieve the *n_re_back*th solution from the *TL.*

else

$n=0$

define S_0 = *best_neighbour*

best_error = best_error

end if

Step11: If the termination criterion based on the *J* values is met or *count* > $count_{max}$ (*count* = 1,2,…, $count_{max}$) exit with the global solution.

Step12: Invoke the *AR* when the current solution is relatively close to a local minimum.

AR: **if** $best\_error < \varepsilon_1$

$R=R_1$ where $R<R_1$.

end

if $best\_error < \varepsilon_2$

$R=R_2$ where $R_2<R_1$ and $\varepsilon_2 < \varepsilon_1$.

end

…

if $best\_error < \varepsilon_n$

$R=R_n$ where $R_n<R_{n-1}$ and $\varepsilon_n < \varepsilon_{n-1}$.

end

Step13: Updated count. If $count \leq count_{max}$ then go to Step 7.

The flowchart of BF-TS is illustrated in Figure 4.4.

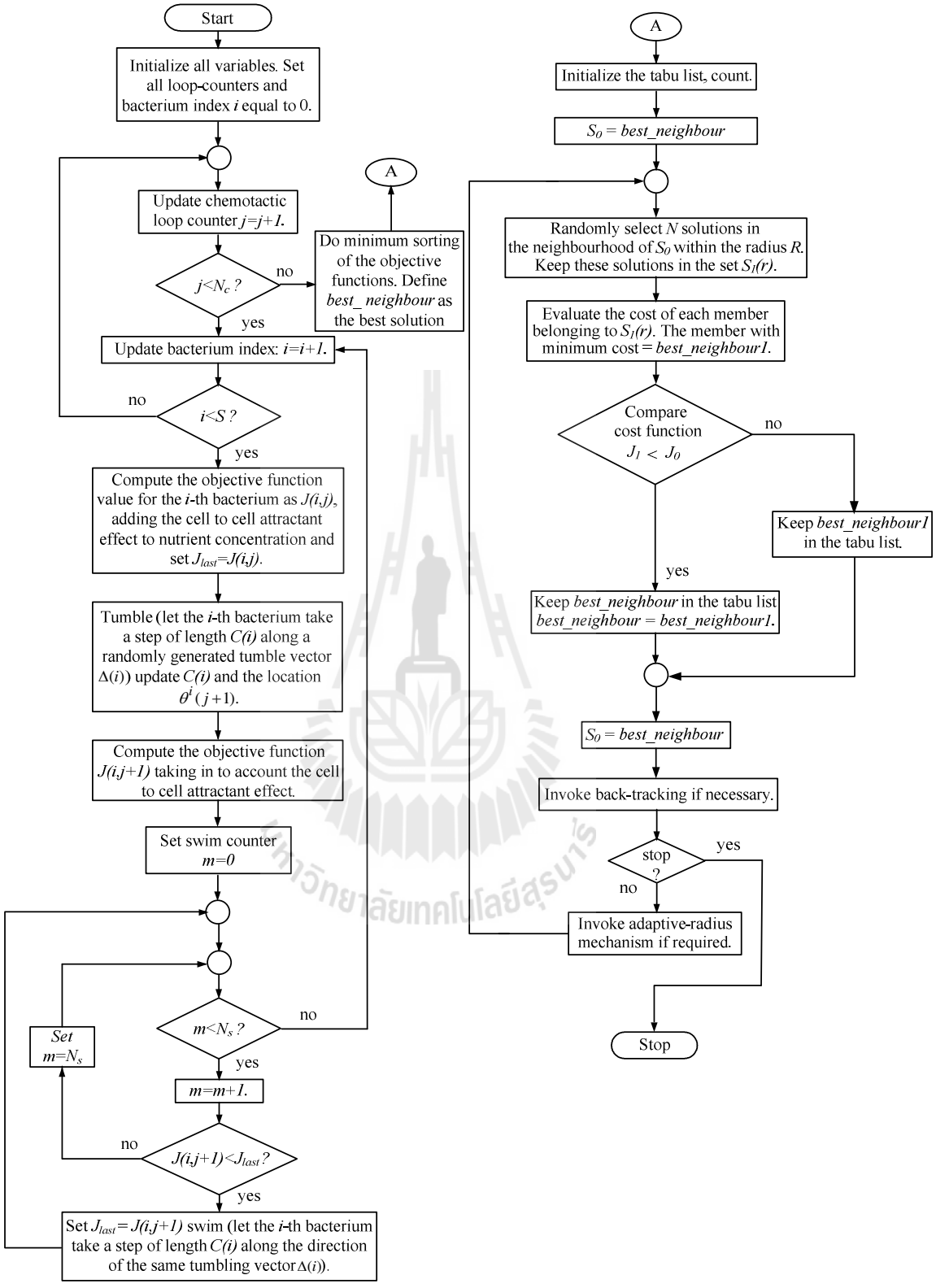


Figure 4.4 Flowchart of the BF-TS algorithm.

4.5 Modifications to Adaptive Tabu Search based-on Adaptive Random Movement of Bacterial foraging Optimization Approach

Search techniques have been widely used to solve various optimization problems which can be divided into complete (exact) and approximate (heuristic) algorithms as mentioned in Chapter III. According to many issues, the complete search algorithms such as branch and bound or dynamic programming guarantee to find an optimal solution for a finite sized problem in bounded time. Nevertheless, as the size of the problem gets larger, the time-consuming by the complete algorithms may increase exponentially. On the other hand, approximate search algorithms find a good (non-optimal) solution in less amount of time. According to the previous literature researches in Chapter II, the ABFO with ability of exploration and the ATS with ability of exploitation are applied to various engineering problems. In some cases, those single algorithms may not cope with the complex problems like combinatorial optimization problems or cannot obtain good enough results. In order to overcome these problems, one approach is to combine different metaheuristic algorithms referred to as hybrid optimization algorithms having received considerable interest in recent years. The wide variety of hybrid approaches has been proposed in many literatures as mentioned in Chapter II. Most of the population-based metaheuristics have been cooperated with the single-solution based metaheuristics utilizing their capabilities namely exploration and exploitation, respectively. Therefore, the ABFO and the ATS algorithms have been loosely combined as described by the previous section. From the experience learned, distribution of good quality solutions is dominantly achieved by the random movement

of bacteria in the ABFO process. This sub-process of random movement can enhance or speed-up the ATS to hit a global solution. Thus, this thesis proposes a modified ATS based-on both high-level relay hybrid (HRH) in the first process to randomly generate initial solution, and low-level teamwork hybrid (LTH) in the second process to randomly select neighbour solutions around the current best solutions with small step size. The proposed algorithms even though use a similar title to that of (Kluabwang and Thomthong, 2012), ones of ours are rather different and far more complicated. Those appeared previously in the 2012 literature are referred to the original ATS with an additional adaptive neighbourhood mechanism. Below is an elaborative description of our modified ATS.

The HRH algorithms are self-contained metaheuristics and run in a sequential manner. It contains hybrid in which one of the metaheuristic is used either to generate initial solution for another metaheuristic or to improve its final solution. For example in some problems, such a kind of evolutionary metaheuristics is used to generate a good quality initial solution and this solution is used as a starting solution in the metaheuristic algorithms. The other instance is to use one metaheuristic to globally optimize the problem and use a different metaheuristic to optimize locally around the final best found solution in global optimization process. Noticeable combining the population-based metaheuristics with the single-solution based metaheuristics in the HRH approach is largely applied. As well known that the population-based metaheuristics are not well suited for fine-tuning structures, when are very close to optimal solutions. Indeed, the strength of these metaheuristics is in quickly locating the high-performance regions of wide search space. Once those regions are located, it may be useful to apply to the single-solution based metaheuristics. This class of LTH

consists of two competing goals namely exploration and exploitation. To balance these properties for instances, the LTH algorithms use the single-solution based metaheuristics embedded into the population-based metaheuristics or use the population-based metaheuristics integrated into the single-solution based metaheuristics. The combination of these goals of metaheuristics achieves advantage from the points of strength of each method. This class of hybrid algorithms is very popular and has been applied successfully to many optimization problems (Talbi, 2002).

Therefore, in this contained hybrid, firstly, the adaptive random movement of ABFO is used to generate initial solution for the second method of ATS based on the HRH method. Secondly, the LTH method is applied. This proposed hybrid uses the adaptive random movement inspired from bacteria which is embedded in the ATS to randomly selected $N_{c2} \times S_2$ solutions in the neighborhood of S_0 around the global solution with small step size and also to improve its final solution without the limit of radius. The combination of these two classes of metaheuristics is powerful in exploration of feasible regions and in exploitation of good quality solutions. The structure of this hybrid algorithm is shown in Figure 4.5.

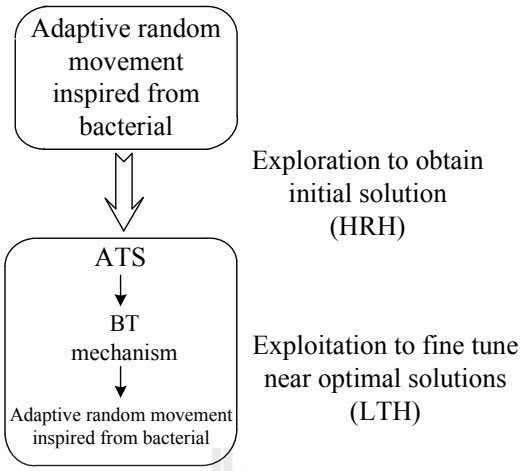


Figure 4.5 Structure of the modified ATS.

As the main procedural list of the modified ATS, the same list for the BF-TS algorithm shown in section 4.4 is used except the Step7 is replaced by the adaptive random movement inspired from bacteria. That Step12 is not used in this proposed algorithm. The following list declares the new scheme of Step7.

Step7: Generate neighbour solutions around S_0 via the adaptive random movement of bacteria. Set $N_{c2} \times S_2$ solutions as the members of the set S_1:

(a) Set the solution S_0 as the current solution $\theta_2^{i_2}$.

(b) Compute objective functions $J(i_2, j_2)$ according to equation (4.15)-(4.16) (i_2=1,2,…,S_2). Set $J_{last} = J(i_2, j_2)$; j_2=1,2,… N_{C2}.

$$J(i_2, j_2) = J(i_2, j_2) + J_{CC}\left(\theta^{i_2}(j_2), P(i_2, j_2)\right) \tag{4.15}$$

$$J_{CC}\left(\theta_2^{i_2}, P(i_2, j_2)\right) = \sum_{i_2=1}^{S_2} J_{CC}^{i_2}\left(\theta_2, \theta_2^{i_2}(j_2)\right)$$
$$= \sum_{i_2=1}^{S_2}\left[-d_{attract}\exp\left(-w_{attract}\sum_{m_2=1}^{p}\left(\theta_{m2}-\theta_{m2}^{i_2}\right)^2\right)\right] \quad (4.16)$$
$$+\sum_{i_2=1}^{S_2}\left[h_{repellent}\exp\left(-w_{repellent}\sum_{m_2=1}^{p}\left(\theta_{m2}-\theta_{m2}^{i_2}\right)^2\right)\right]$$

(c) Generate randomly ([-1, 1]) the elements of the random vector $\Delta_{m2}(i_2) \in \Re^p$, $m_2=1,2,\ldots,p$, then compute the adaptive step size, $C_2(i_2)$ using equation (4.17), and update the solution $\theta_2^{i_2}(j_2+1)$ according to equation (4.18). Compute the objective function for $j_2 = j_2+1$ according to equation (4.15)-(4.16). Set $m_2=0$.

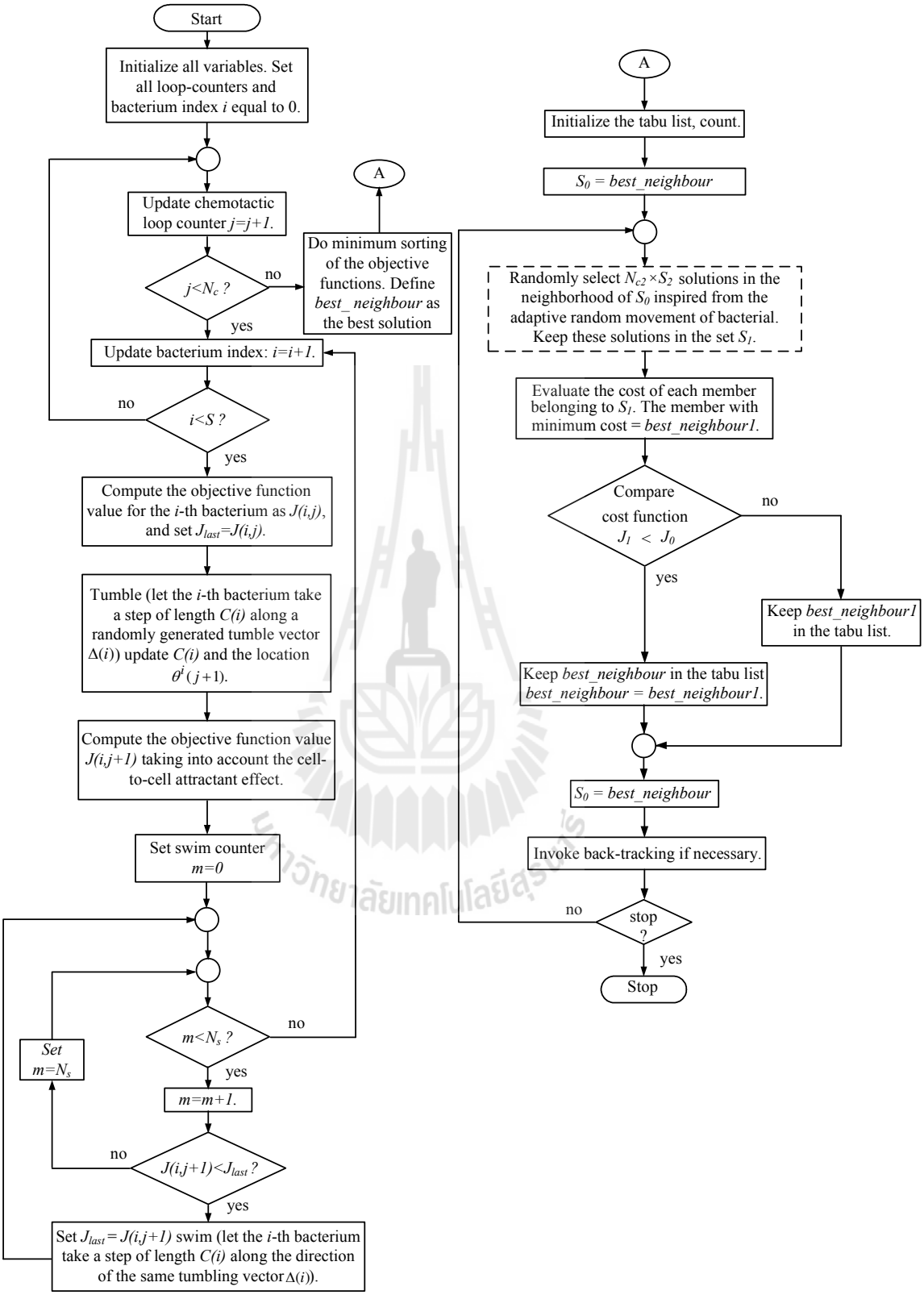
$$C(i_2) = \frac{C_{\max 2}\cdot\left|J(\theta^{i_2})\right|}{\left|J(\theta^{i_2})\right|+\alpha_2} = \frac{C_{\max 2}}{1+\dfrac{\alpha_2}{\left|J(\theta^{i_2})\right|}} \quad (4.17)$$

$$\theta_2^{i_2}(j_2+1) = \theta_2^{i_2}(j_2) + C(i_2)\frac{\Delta(i_2)}{\sqrt{\Delta^{\mathrm{T}}(i_2)\Delta(i_2)}} \quad (4.18)$$

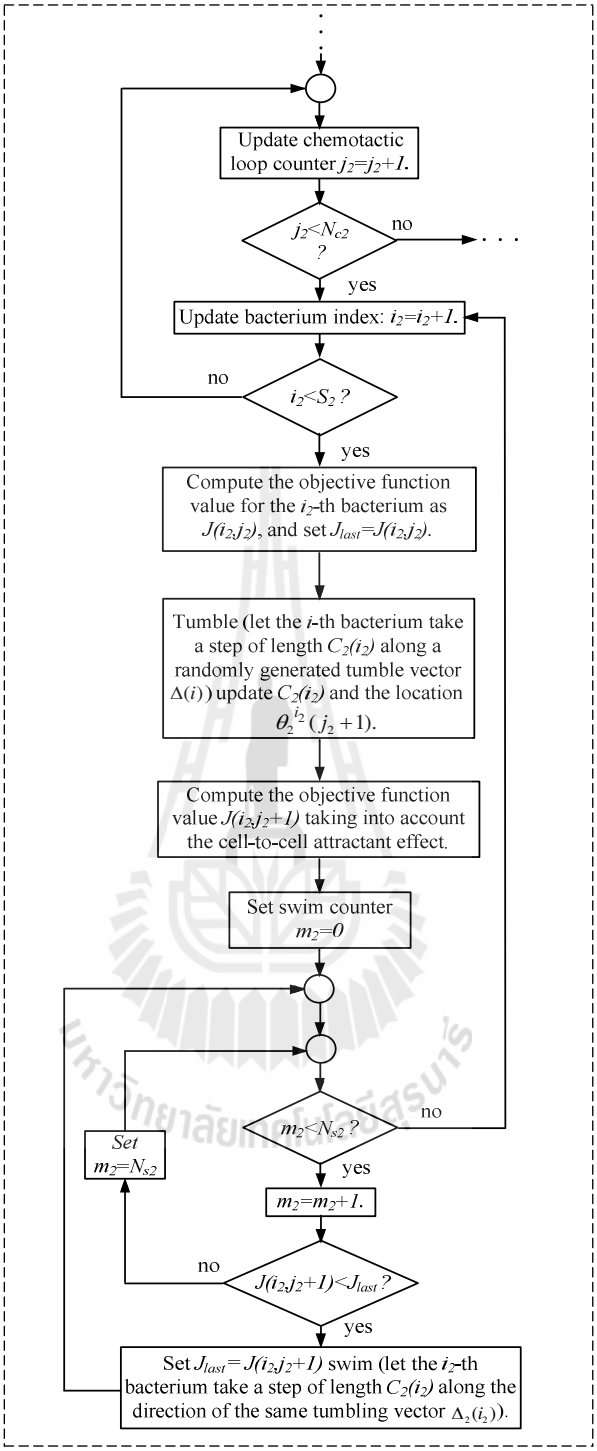
(d) If $J(i_2, j_2+1) < J_{last}$ then $J_{last} = J(i_2, j_2+1)$; use the direction of the same random vector $\Delta(i_2)$ to compute $\theta_2^{i_2}(j_2+1)$ and $J(i_2, j_2+1)$. Update m_2 and repeat Step7(d) until $m_2 > N_{S2}$.

(e) If $j_2 \le N_{C2}$, go to Step7(b).

The flowchart of the modified ATS is illustrated in Figure 4.6 which is embedded with the adaptive random movement of bacteria. This random movement is shown in a dotted-line inset in Figure 4.6(a). Figure 4.6(b) gives detailed list representation of this sub-algorithm.


Start
Initialize all variables. Set all loop-counters and bacterium index i equal to 0.
Update chemotactic loop counter j=j+1.
j<Nc ?
no
yes
Do minimum sorting of the objective functions. Define best_ neighbour as the best solution
A
Update bacterium index: i=i+1.
i<S ?
no
yes
Compute the objective function value for the i-th bacterium as J(i,j), and set Jlast=J(i,j).
Tumble (let the i-th bacterium take a step of length C(i) along a randomly generated tumble vector Δ(i)) update C(i) and the location θi(j+1).
Compute the objective function value J(i,j+1) taking into account the cell-to-cell attractant effect.
Set swim counter m=0
m<Ns ?
no
yes
Set m=Ns
m=m+1.
J(i,j+1)<Jlast ?
no
yes
Set Jlast = J(i,j+1) swim (let the i-th bacterium take a step of length C(i) along the direction of the same tumbling vector Δ(i)).
A
Initialize the tabu list, count.
S0 = best_neighbour
Randomly select Nc2×S2 solutions in the neighborhood of S0 inspired from the adaptive random movement of bacterial. Keep these solutions in the set S1.
Evaluate the cost of each member belonging to S1. The member with minimum cost = best_neighbour1.
Compare cost function J1 < J0
no
yes
Keep best_neighbour1 in the tabu list.
Keep best_neighbour in the tabu list best_neighbour = best_neighbour1.
S0 = best_neighbour
Invoke back-tracking if necessary.
stop ?
no
yes
Stop

(a)



(b)

Figure 4.6 Flowchart of the modified ATS algorithm (a) the whole algorithm; (b) the random selection of neighbour solutions.

4.6 Invasive Weed Optimization Algorithm

Invasive weed optimization (IWO) algorithm was first introduced by Mehrabian and Lucas (Mehrabian and Lucas, 2006). This algorithm is a new natural heuristic optimization algorithm inspired by the colonizing behavior of weeds in nature. In weed colonization, weeds invade a cropping field by means of dispersal and occupy opportunity spaces between the crops. Each invading weed takes the unused resources in the field, grows to a flowering weed, and produces new weeds, independently. The number of new weeds produced by each flowering weed depends on the fitness of that flowering weed in the colony. Those weeds that have better adoption to the environment and take more unused resources grow faster, and produce more seeds. The new produced weeds are randomly spread over the field, and grow to become flowering weeds. This process continues until the maximum number of weeds is reached on the field due to the limited resources. Now, only those weeds with better fitness can survive and produce new weeds. This competitive contest between the weeds makes them become well adapted and improved over the time. From those behaviors of weed colonization, Mehrabian and Lucas have divided the IWO algorithm into 4 processes namely, initial a population, reproduction, spatial dispersal and competitive exclusion processes (Mehrabian and Lucas, 2006; Karimkashi and Kishk, 2010). The process is addressed in details as follows:

- **Initialize a population-**A population of initial solutions N_0 is randomly explored over the search space.

- **Reproduction-**A member of the population of plants is allowed to produce seeds depending on its own and the colony's lowest and highest fitness. The number of seeds each plant produces increases linearly from the minimum possible seed

production to its maximum. In other words, a plant will produce seeds based on its fitness, the colony's lowest fitness and highest fitness to make sure the increase is linear. Figure 4.7 illustrates the procedure.

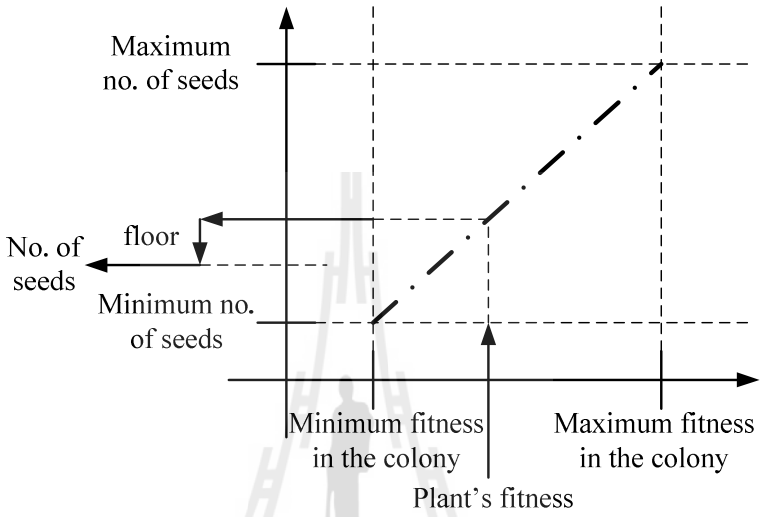


Figure 4.7 Seed production procedure in a colony of weeds.

Denote the best fitness (minimum cost) of the colony as F_C, the worst fitness (maximum cost) of the colony as F_D and the fitness of the *i*th weed as F_X respectively. Accordingly, the numbers of seeds *i*th produces can be calculated by equation (4.19), where the number of seed must be integer number under the command of *floor*.

$$No.\ of\ \ seeds(i) = floor\left[\left(\frac{F_X(i) - F_D}{F_C - F_D}\right)\left(s_{\max} - s_{\min}\right) + s_{\min}\right] \tag{4.19}$$

- **Spatial dispersal-**The generated seeds are being randomly distributed over the search space based on the use of normally distributed random numbers with zero

mean, but varying variance. This means that seeds will be randomly distributed such that they lie close to the parent plant. However, standard deviation (SD), σ_{iter} of the random function will be reduced from a previously defined initial value, $\sigma_{initial}$ to a final value, σ_{final} in every step (generation). In simulations, a nonlinear alteration has shown satisfactory performance, which is given in equation (4.20).

$$\sigma_{iter} = \frac{\left(iter_{\max} - iter\right)^{n}}{\left(iter_{\max}\right)^{n}}\left(\sigma_{initial} - \sigma_{final}\right) + \sigma_{final} \tag{4.20}$$

According to equation (4.20), the positions of new seeds are given by:

$$\text{position of seeds} = \text{parent's position} + \sigma_{iter} \times randn(\text{no.of parent } i\text{th}, dim.) \tag{4.21}$$

- **Competitive exclusion-**If a plant leaves no offspring then it would go extinct, otherwise they would take over the world. Thus, there is a need of competition between plants for limiting maximum number of plants in a colony. When the maximum number of weeds in a colony is reached its maximum by fast reproduction, the produced seeds are then allowed to spread over the search space according to spatial dispersal mechanism. When all seeds have found their positions in the search space, they are ranked together with their parents (as a colony of weeds). Next, weeds with lower fitness are eliminated to reach the maximum allowable population in a colony. In this way, plants and offspring are ranked together and the ones with better fitness survive, and are allowed to replicate. As mentioned in the reproduction mechanism, this approach provides the plants with lower fitness a chance to reproduce, and if their

offspring have good fitness in the colony then they can survive, otherwise they will be disposed. The population control mechanism is applied to their offspring by the end of a given run, realizing competitive exclusion. This process continues until the maximum number of plants is reached; now only the plants with lower fitness can survive and produce seeds, others are eliminated.

The IWO process continues until the iterations hit the maximum or a solution satisfies the termination criterion with a hope for the plant with best fitness being the closest one to the optimum solution. The procedural list of the IWO algorithm is as follows (see Figure 4.8):

Step0: Initialize search parameters: N_0, *search space*, $iter_{max}$, *dim*, p_{max}, s_{max}, s_{mim}, n, σ_{final} and $\sigma_{initial}$.

Step1: Randomly generate a population of weeds within the search space.

Step2: The seeds are produced depending on the lowest and highest fitness of their parents that relative to equation (4.19).

Step3: After reproduction, seeds are randomly distributed over the search space with zero mean and varying standard deviation (σ_{iter}) according to equation (4.20).

Step4: Compute the positions of new seeds according to equation (4.21).

Step5: If the total number of plants is not over p_{max}, the new seeds and their parents are combined as weeds within the search space. Otherwise, rank seeds and their parents, eliminate weeds with lower fitness. Otherwise, the better fitness survives and it is allowed to duplicate within a limited number of p_{max}.

Step6: If the iteration > it_{max} or the termination criterion is met. Stop and exit with solution. Otherwise go to Step2.

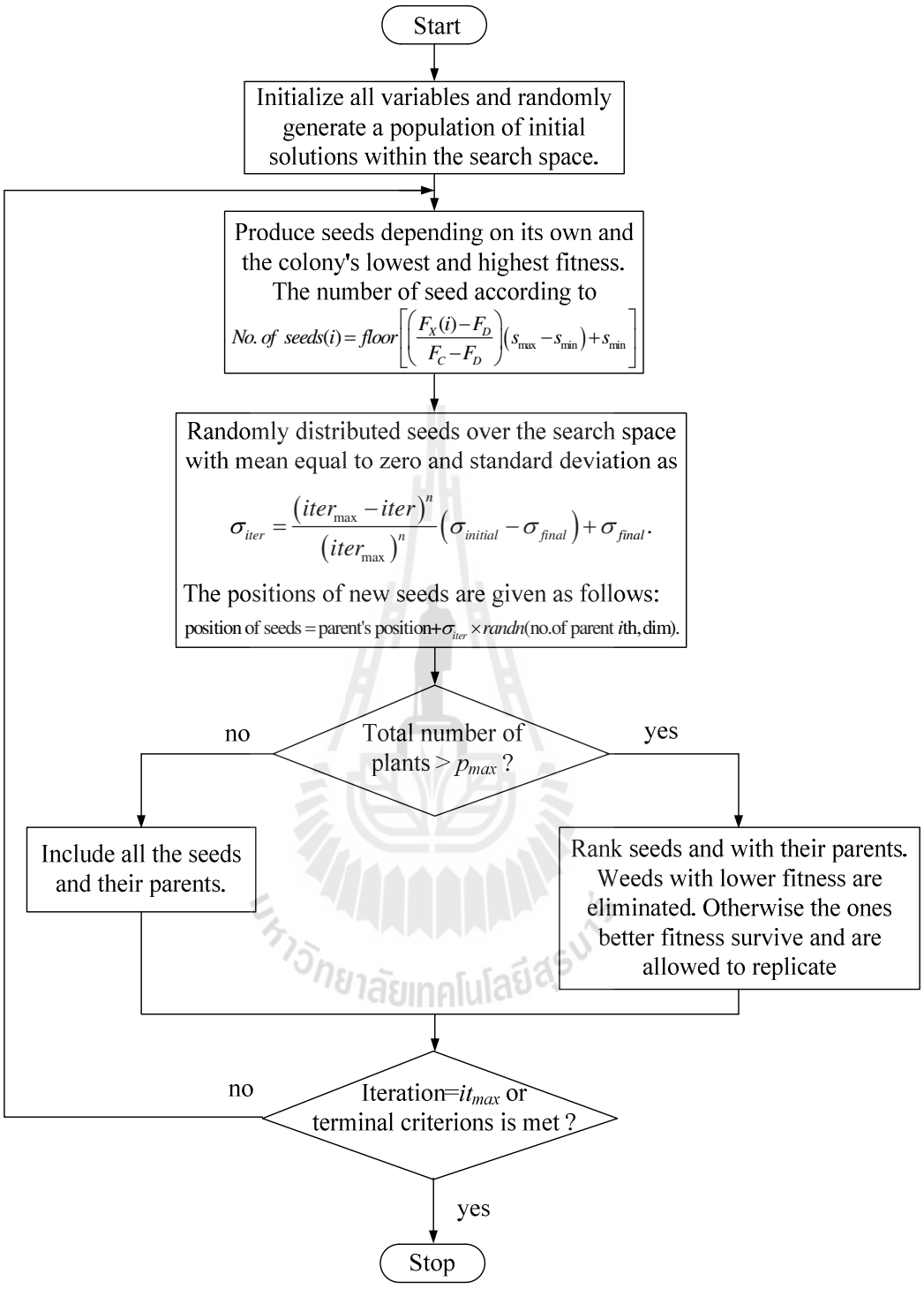


Figure 4.8 Flowchart of the IWO algorithm.

This algorithm has been applied to many engineering problems such as dynamic and control systems (Mehrabian and Lucas, 2006), optimization technique for antenna (Dadalipour et al., 2008; Mallahzadeh et al. 2009) and optimizing the locations of piezoelectric actuators for vibration suppression of flexible structures (Mehrabian and Yousefi-Koma, 2007; Mehrabian and Yousefi-Koma, 2009) etc.

4.7 Genetic Algorithm

Genetic algorithm (GA) was firstly introduced by Holland in 1975. This algorithm is based on the principle of evolution via natural selection (Holland, 1975; Holland 1992). The most common type of genetic algorithm consists of three strategies namely parent selection, genetic operation and replacement of parents. GA starts with a population of individuals generated randomly. The population comprises a group of chromosomes, where a particular group of chromosomes (parents) is selected to generate the offspring by the defined genetic operations. Individuals in the current population are evaluated using a measure of their objective function values, called fitness functions. A fitness function measures the fitness of an individual to survive in a population of individuals. The chromosomes in the current population are then replaced by their offspring, based on a certain replacement strategy (Tang et al., 1996). In literature, a wide variety of GA applications have been found including control design in power electronics and drives (Cupertino et al., 2004), system identification of an induction motor and power networks (Abdelhadi et al., 2005; Bagis, 2006; Dong, et al., 2008), and design of resonant compensators for active filters (Lenwari et al., 2009).

In this thesis, the GA toolbox available from MATLAB^{TM} (MathWorks, 2005) is used for search performance comparison. The problem of interest is organized in an

objective function which represents the fitness of a candidate solution. A typical implementation of the GA may follow the steps below:

Step0: Initialize a population of candidate solutions subject to existing constraints and search space. Encode the solutions in the forms of binary represented chromosomes.

Step1: Decode the solutions into real number representation, and evaluate their objectives and fitness values.

Step2: Assign a probability of reproduction to each chromosome proportional to its fitness relative to the others. Use the universal sampling technique, or others, to select chromosomes of high fitness values.

Step3: Apply crossover and mutation to chromosomes of parents to create offspring (reproduction process).

Step4: Decode the offspring chromosomes into real number representations. Evaluate their objectives and fitness values. Replace the existing solutions with the offspring of high fitness.

Step5: Terminate the process if a suitable solution is found or if the computing expires. Otherwise, go to Step1 and iterate the process.

The flowchart of GA is shown in Figure 4.9.

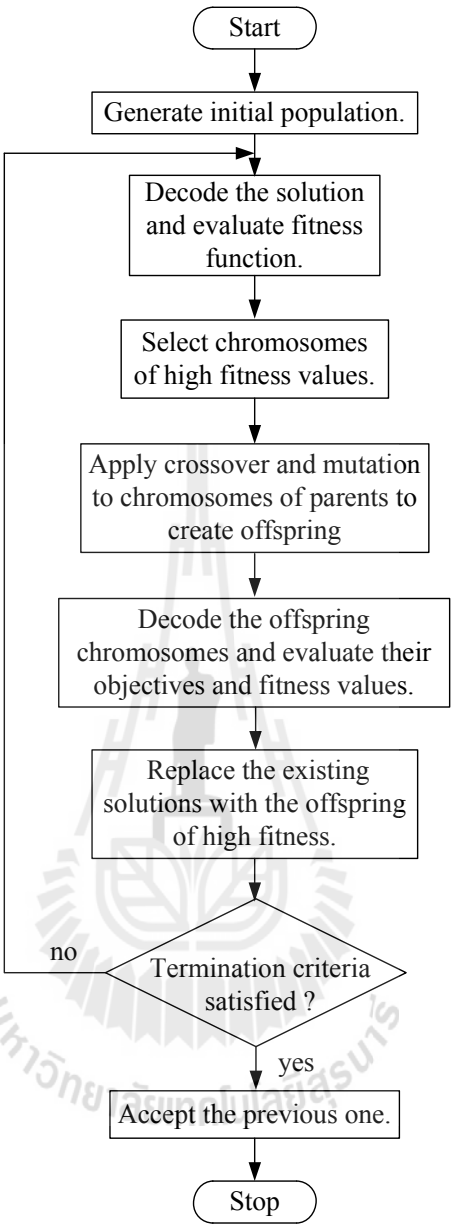


Figure 4.9 Flowchart of the GA algorithm.

4.8 Conclusion

In this chapter has presented the descriptions of the metaheuristic algorithms consisting of the adaptive bacterial foraging (ABFO), the adaptive tabu search (ATS), the invasive weed optimization (IWO) and the genetic algorithm (GA). Moreover, the

modified versions of ATS are also presented. The first one mentioned is the cooperative algorithm based on the ABFO and the ATS under the name of bacterial foraging-tabu search (BF-TS) which uses the adaptive random movement of bacteria to improve the quality of the initial solution for the ATS. In addition, this Chapter presents the new cooperative algorithm viewed as the hybrid metaheuristic by combination of the outstanding capability of the ABFO in terms of exploration and the ability of the ATS in terms of exploitation. This proposed algorithm uses the adaptive random movement of bacteria to generate the neighbour solutions around the global solution without the limit of radius. The step movement of this proposed algorithm can be adaptable finely when is very close to optimum solution depending on the current cost function. This new version is called the modified ATS. The details in this Chapter cover the basic concepts of each algorithm, which are presented by the given procedural lists and the flowcharts. The convergence analysis and search performance comparisons of those algorithms will be further presented in Chapter V and Chapter VI, respectively.

CHAPTER V

CONVERGENCE OF MODIFICATIONS TO ADAPTIVE TABU SEARCH BASED-ON BACTERIAL FORAGING OPTIMIZATION APPROACH

5.1 Introduction

The previous Chapter described the proposed cooperative algorithms formed from the existing adaptive tabu search (ATS) and the random movement inspired by bacterial behaviour. The adaptive random movement of bacteria sometimes called foraging is an outstanding characteristic to explore an entire search space at the beginning of search. It is also useful to generate exhaustively neighbour solutions with small step sizes. The main propose of this mechanism is to improve the search performance in order to obtain good solutions, and avoid local traps.

Despite, the metaheuristic algorithms have been claimed as the most practical approach to solve complex combinatorial optimization problems, the design of metaheuristics always focuses on computational efficiencies rather than rigorous analyses of their convergence property. We recognize the importance of convergence analysis of algorithms. Therefore, this Chapter presents our convergence analysis of the proposed algorithms. Due to the fact that, the proposed algorithms contain two parts namely a random movement or random walk front-end, and the ATS without search radius adjustment procedures, respectively. The Chapter starts with a brief explanation

on random walk in section 5.2. Sections 5.3 and 5.4 describe our convergence analysis and conclusion, respectively.

5.2 Importance of Random Walks

The important characteristics of metaheuristic algorithms are intensification and diversification. Intensification is also called exploitation as it typically searches around the current best solution, and selects the best candidate or solution. Along the search sequence, there can be many current best solutions, and solution candidates. In the contrary, diversification also called exploration explores the search space more efficiently in a large-scale randomization. In fact, different algorithms use different ways of searching the balance between exploration and exploitation to achieve efficient searches.

There are many ways of carrying out intensification and diversification. The randomization combined with a deterministic technique is one of many methods to achieve exploration or diversification. This process ensures that new solutions are generated and distributed as diversely as possible in a feasible search space. One of the simplest randomization techniques is represented by (5.1).

$$x_{new} = L + (U - L) * \varepsilon_u \tag{5.1}$$

where L and U are the lower and the upper bounds, respectively. ε_u is uniformly distributed random number lying between 0 and 1. This form is often used by many algorithms such as harmony search (Yang, 2009), particle swarm optimization (Kennedy and Eberhart, 1995) and firefly algorithms (Yang, 2009), etc. To generate

new solutions around a promising or better solution locally and more intensively, an exploitation technique is needed. This can be easily achieved by a local random walk described by (5.2).

$$x_{new} = x_{old} + s.w \tag{5.2}$$

where w is a Gaussian distribution with zero mean as typically being used, and s is the step size of the random walk. Notice that, if s is too large, a region found can be too far away from an interested region, which will increase diversification significantly but reduce intensification considerably. On the other hand, if s is too small, the region found can be trapped easily by a local minimum, which dominates intensification but degrades diversification. Therefore, the step size should be appropriately selected for a specific problem, or at least, much smaller than the scale of the search space for each problem. There are alternative ways to increase the efficiency of the random walk, which also increase the efficiency of exploration by using other forms of random walks such as Lévy flights (Viswanathan et al., 2000), Cuckoo search (Yang and Deb, 2009), Bat search (Yang, 2010), Krill herd algorithm (Gandomi and Alavi, 2012) etc. The step size in (5.2), s, is drawn from a Lévy disribution with large step sizes. This helps increase the step size and the distance of such random walks. Apart from the standard random walk in (5.2), there is a more selective or controlled walk around the current best, x_{best} , as described by (5.3).

$$x_{new} = x_{best} + s.w \tag{5.3}$$

Randomization provides a good way to move away from a local search to another search on a global area. Therefore, most metaheuristics are intended for global optimization.

Randomness reduction is also another important issue for metaheuristics. Specifically, the randomization is usually used to diversely explore the search space on the global scale, and also intensively on some local scales. To obtain better results and accelerate convergence, the degree of randomness should be reduced or adapted, otherwise, the convergence will be slow down. In practice, the random walk of metaheuristics should be adequately small to carry out iterative search around a current best solution, which is to exploit the current best more effectively. This approach can be found in the *AR* mechanism of the ATS, and also in the adaptive step size of bacterial movement techniques, which are usually smaller and smaller to specifically limit the randomization when the search comes close to the global area. The adaptive random used in the ATS can be represented by (5.4).

$$S_1 = S_{0,best} + radius * rand1(-1,1) * (U - L) \tag{5.4}$$

where *rand*1 is from (5.1). Note that the radius can be smaller depending on each conditional requirement of cost function. This random form is used to generate a group of neighbour solutions within the reasonable search radius which intensively converges to better solutions, locally as shown in Figure 5.1. However, in some worse cases against hard problems containing many local minima, it is possible that the search is trapped in a local area limited by search radius. For the case of the adaptive random movement inspired by bacteria, it uses a uniform distribution to obtain randomization,

and an adaptive random walk to generate better solutions around a current best solution.

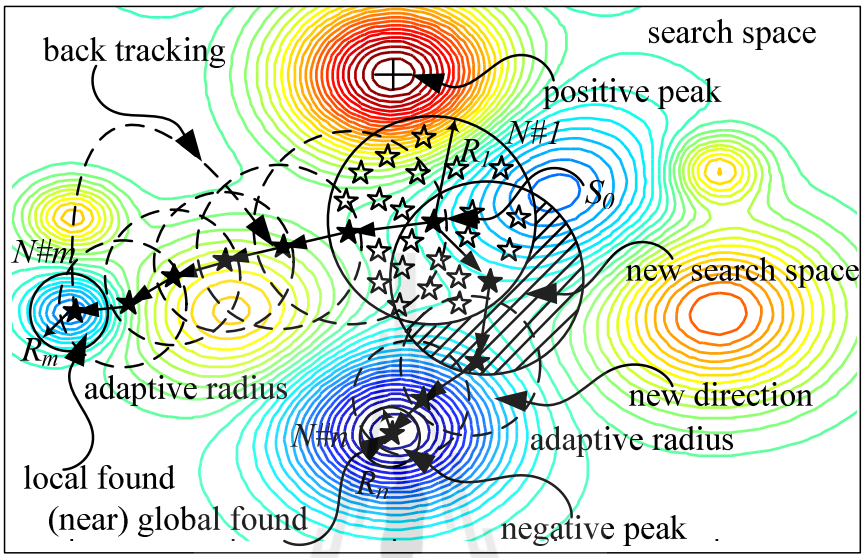


Figure 5.1 Random movements of the ATS.

This randomization form is rewritten in equation (5.5).

$$\theta^i(j+1) = \theta^i(j) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^{\mathrm{T}}(i)\Delta(i)}} \tag{5.5}$$

where $C(i) = C_{\max} \Big/ 1 + \frac{\alpha}{\left|J(\theta^i)\right|}$ refers to the adaptive size of the step taken in the random direction. $\Delta(i) \Big/ \sqrt{\Delta^{\mathrm{T}}(i)\Delta(i)}$ represents a unit direction, and $\Delta(i)$ is a random vector on [-1, 1]. Depending on the cost function, $J(\theta^i)$, adaptive approach can automatically adjust itself to avoid oscillations of the state of solutions when they are close to the optimal value as previously explained in Chapter 4. The step size, *C(i),* is also specified

by tumble and run behaviours, so that the current solution is better than the previous one, while another step size in the same direction will be taken as a run behaviour. On the other hand, the adaptive random generates a new element of the random vector with [-1, 1] as a tumble behaviour. This randomization generates the population solution, thus it can flow through the global best. The approach provides more opportunities to find reasonably good solutions with the *BT* mechanism to avoid dead lock occurrence. The adaptive random movement inspired by bacteria can be represented by the diagram shown in figure 5.2.

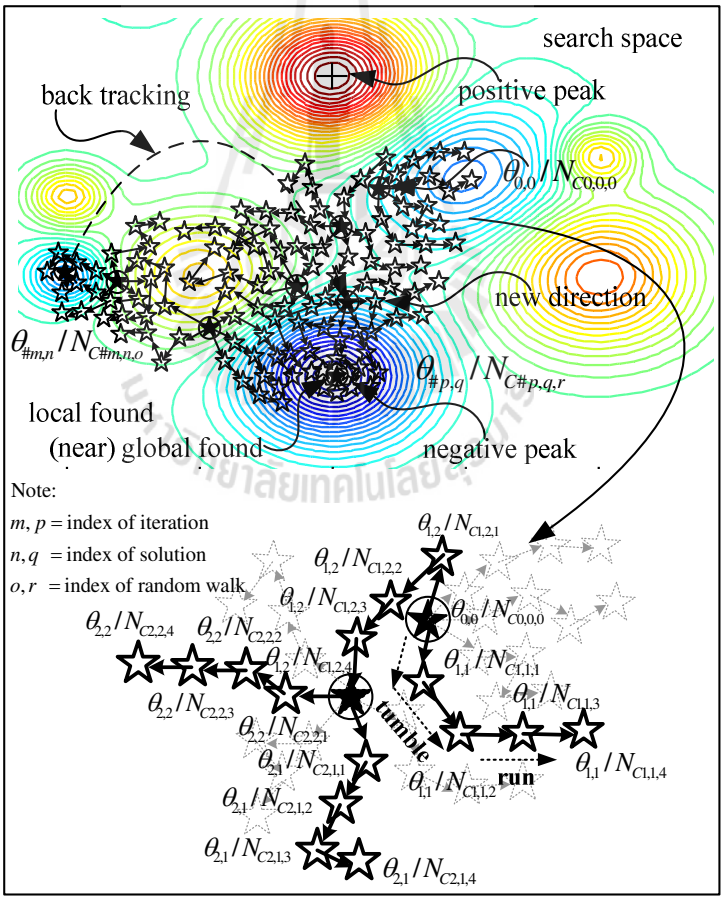


Figure 5.2 Random movements of the modified ATS.

So far, we have explained that our proposed algorithms consist of two parts, i.e. a random walk front-end adapted from bacterial foraging, and the existing ATS with the *AR* mechanism. The random walk front-end helps increase diversification of solutions, and effectively removes the need for the *AR*. To track down an optimum solution effectively, the ATS (with *BT* mechanism) plays an important role.

To ensure a convergent property of the proposed algorithms, a rigorous analysis is necessary. Due to the fact that the convergence of the ATS has been analysed in modular manner (Puangdownreong et al., 2004; Puangdownreong et al., 2004; Sujitjorn et al., 2006), our attempt is to show that the proposed front-end has a convergence property so that it will not destroy the convergence property of the ATS. To clarify this, the *BT* mechanism helps the TS to escape from an entrapment by a local solution, and to identify the most potential search terrain expected to contain a global optimum. The *AR* mechanism helps accelerate the search, however, it is not applied to the modified ATS. The next section gives details of our convergence analysis.

5.3 Convergence Analysis of the Modification to Adaptive Tabu Search Based-on Adaptive Random Movement of Bacteria

This thesis presents two modified algorithms namely the BF-TS and the modified ATS algorithms. In a similar manner, they have applied the adaptive random movement of bacteria to heuristically generate an initial (elite) solution for the ATS. Moreover, a similar approach is applied to select the neighbour solutions without the implementation of the *AR* mechanism. According to the adaptive version of BFO algorithms, the foraging (chemotaxis) mechanism influences the convergence rate of the algorithms (Dasgupta et al., 2009; Dasgupta et al., 2010; Majhi et al., 2009). The

run-length unit or step size parameter is of prime importance to achieve a fast and efficient search. Recent research results have shown that the random movement with large step size *C(i)* can explore the whole search space, and also escape from an entrapment by some local optima. On the other hand, the random movement with small step size *C(i)* is rather attached to some local optima close to its start point, and exploits them for its whole life cycle. It means that the random movement with small step size *C(i)* is not able to escape from the local minima. Obviously, the random movement with a large step size is explorative, while that with small step size is exploitative (Chen et. al., 2009; Passino, 2002; Liu and Passino, 2002). Adaptive ability of random movement is significantly useful as illustrated by (Das et. al. 2009; Dasgupta et. al., 2009).

As mentioned earlier, our proposed random movement front-end explores almost entire search space before the ATS begins its search. This section provides a mathematical analysis of the convergence property of the front-end. Then, reviews of the ATS' convergence follow.

5.3.1 Convergence Analysis of Bacterial Random Movement

Our approach to analyse the convergence of bacterial random movement utilizes Lyapunov stability theorem pioneered by (Das et. al., 2009; Dasgupta et. al., 2009). In order to proceed with this, an approximation function of the equation (5.5) is needed. The section, therefore, starts with obtaining the approximation function.

Referring to (5.5), θ^i is a time function representing a position of an individual bacterium (or solution); *C(i)* represents an adaptive step size taken in a

random direction; $\Delta(i)$ is a random vector, [-1, 1] and $\Delta(i)/\sqrt{\Delta^T(i).\Delta(i)}$ represents a unit direction. Let J be an objective function expressed shortly by $J(\theta(t))$ in which $\theta(t)=\theta^i$. Our analysis considers a single bacterium that undergoes chemotactic steps according to (5.5).

Assumptions:

i. The objective function $J(\theta)$ is continuous and differentiable at all points in search space, and unimodal in the region of interest. It means that each region in entire search space contains only one local minimum and the local minimum must not be located at the region boundary. Therefore, the unimodal properties can be defined as $\bigcup_{i=1}^{k} \boldsymbol{s_i} = \boldsymbol{S}$ and $\bigcap_{i=1}^{k} \boldsymbol{s_i} = \varnothing$, where $\boldsymbol{S}$ is defined as the finite search space having k strictly local minima and divided into k regions and $\boldsymbol{s_i} \subseteq \boldsymbol{S}$ denoted by $\boldsymbol{s_i}$ (i = 1, 2,…,k)

ii. Only one optimum (minimum) exists at the location $\theta=\theta_0$ such that $J(\theta_0)\approx 0$ and $J(\theta)|_{\theta\neq\theta_0} \neq 0$.

iii. The step size C is smaller than 1.

iv. The analysis is applied to the regions of fitness landscape, where gradients of $J(\theta)$ are small, i.e. near optimum.

Lemma 1: The random position $\theta(t)$ of a single bacterium expressed by

$$\theta^i(j+1)=\theta^i(j)+C(i)\frac{\Delta(i)}{\sqrt{\Delta^{\mathrm{T}}(i)\Delta(i)}}$$

can be approximated by $\theta(t)=\int\left[\frac{C\Delta}{2}-\frac{qC^2}{4}G\right]dt$, where $\Delta^2=1$ or $|\Delta|=1$, and $G=\frac{dJ(\theta)}{d\theta}$.

Proof: Consider that the solution moves in the direction of decreasing cost value, i.e.

$$J(\theta) - J(\theta + \Delta\theta) > 0 \tag{5.6}$$

For the sake of simplicity, we drop the indices *i* and *j*. So, the solution moves by an amount of $C\Delta$.

In a very small finite time, Δt seconds, the bacterium's position (or solution location) is changed by $(C\Delta)\Delta t$.

Therefore, $\Delta\theta = (C\Delta)\Delta t$ if $J(\theta) > J(\theta + \Delta\theta)$, otherwise $\Delta\theta = 0$. If Δt divides sign of the quantity $J(\theta) - J(\theta + \Delta\theta)$, the result remains unchanged. The solution will change its position if and only if the term of $\frac{J(\theta) - J(\theta + \Delta\theta)}{\Delta t}$ is positive.

Let the crucial decision making activity of the random movement be modeled by a unit-step function

$$u(x) = \begin{cases} 1, & if \quad x > 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.7}$$

$\Delta\theta$ can be expressed by

$$\Delta\theta = u\left(\frac{J(\theta) - J(\theta + \Delta\theta)}{\Delta t}\right)(C\Delta)(\Delta t) \tag{5.8}$$

in which $u = 1$. Hence,

$$\upsilon = \underset{\Delta t \to 0}{Lim} \frac{\Delta\theta}{\Delta t} = \underset{\Delta t \to 0}{Lim} \left[u \left\{ -\frac{J(\theta + \Delta\theta) - J(\theta)}{\Delta t} \right\} (C\Delta) \right] \tag{5.9}$$

$$\begin{aligned} \upsilon &= \underset{\Delta t \to 0}{Lim} \left[u \left\{ -\frac{J(\theta + \Delta\theta) - J(\theta)}{\Delta\theta} \frac{\Delta\theta}{\Delta t} \right\} (C\Delta) \right] \\ &= u \left[-\left\{ \underset{\Delta\theta \to 0}{Lim} \frac{J(\theta + \Delta\theta) - J(\theta)}{\Delta\theta} \right\} \left\{ \underset{\Delta t \to 0}{Lim} \frac{\Delta\theta}{\Delta t} \right\} \right] (C\Delta) \end{aligned} \tag{5.10}$$

as $\Delta t \to 0$ and $\Delta\theta \to 0$. The velocity, υ, in equation (5.10) can be rewritten as

$$\upsilon = u(-G\upsilon)C\Delta \tag{5.11}$$

where $G = \frac{dJ(\theta)}{d\theta} = \underset{\Delta\theta \to 0}{Lim} \frac{J(\theta + \Delta\theta) - J(\theta)}{\Delta\theta}$ is gradient of the objective function.

In equation (5.11), $(-G\upsilon)$ is the argument of the unit-step function, *u*. If *G* and υ are different signs, *u* = 1, and the velocity, $\upsilon = C\Delta$, otherwise the solution (or bacterium) is motionless. Since $u(x)$ has a jump discontinuity at *x* = 0, to simplify our analysis further, *u(x)* is replaced by a smooth approximation to the step response (Davies, 2002).

$$u(x) = \underset{q \to \infty}{Lim} \varphi(x) = \underset{q \to \infty}{Lim} \frac{1}{1 + e^{-qx}} \tag{5.12}$$

From equation (5.11) and (5.12), the equation (5.11) can be rewritten as

$$\upsilon = \frac{C\Delta}{1 + e^{qG\upsilon}} \tag{5.13}$$

According to our assumptions, C and G are very small, and $q \approx 10$. Then, we can say that $|qG\upsilon| << 1$. The exponential function is now approximated by neglecting high order terms, i.e. $e^{qG\upsilon} \approx 1 + qG\upsilon$. Hence, equation (5.13) becomes

$$\upsilon = \frac{C\Delta}{2}\left(\frac{1}{1+\frac{qG\upsilon}{2}}\right) \tag{5.14}$$

Since $\frac{qG\upsilon}{2} << 1$, $\left(1+\frac{qG\upsilon}{2}\right)^{-1} \approx 1-\frac{qG\upsilon}{2}$. The equation (5.14) can be rewritten as

$$\upsilon = \frac{C\Delta}{2}\left(1-\frac{qG\upsilon}{2}\right) = \frac{C\Delta}{2} - \frac{qC^2G}{4} = \frac{d\theta}{dt} \tag{5.15}$$

Therefore,

$$\theta(t) = \int\left[\frac{C\Delta}{2} - \frac{qC^2}{4}G\right]dt \tag{5.16}$$

This completes the proof. ❑

It is stated that the random walk front-end can be seen as a dynamical system modeled by (5.15), and its solution is given by (5.16). From the expression (5.15), a corresponding search can be performed in a similar way to the classical gradient descent search algorithm in single dimension. Equation (5.15) can be written as

$$\upsilon = \frac{d\theta}{dt} = -\frac{qC^2}{4}G + \frac{C\Delta}{2} = -\alpha' G + \beta' \tag{5.17}$$

where α' is $qC^2/4$ and β' is $C\Delta/2$. The general form of the classical gradient descent search (Das et. al. 2009; Dasgupta et. al., 2009) is given by

$$\frac{d\theta}{dt} = -\alpha G + \beta \tag{5.18}$$

where α is the learning rate, and β is the momentum. Similarity between (5.17) and (5.18) can be considered as a modified gradient descent search, where α' is a function of step size and can be identified as the learning rate parameter. Note that the random search or momentum term $(C\Delta)/2$ appearing in (5.15) and (5.16) provides an additional feature to the classical gradient descent search. If the gradient, *G*, becomes very small, the random term plays an important role over gradient descent term, and the solution changes its position. However, the random search term may lead to a change in position in the direction of increasing cost value, then the magnitude of the gradient increases and dominates the random search term. If the magnitude of the gradient in equation (5.17) decreases consistently near the optimum until $|G| \to 0$, then gradually β becomes dominant. Thus, the velocity can be approximated by $\left|\frac{d\theta}{dt}\right| \approx |\beta| = \left|\frac{C\Delta}{2}\right| = \frac{C}{2}$ (as $|\Delta| = 1$). As previously mentioned, the adaptive random movement is an important physical significance to avoid oscillation of the solution around the optimum, and to accelerate convergence. In this case, the random search has been assumed to reach

close to the optimum. Since $\left|\frac{d\theta}{dt}\right| \approx \frac{C}{2}$, it does not stop taking steps, and also oscillates close to the optimum. This crisis can be improved by an adaptation according to the relation, $C(i) = \frac{C_{\max}.\left|J(\theta^i)\right|}{\left|J(\theta^i)\right| + \alpha} = \frac{C_{\max}}{1 + \frac{\alpha}{\left|J(\theta^i)\right|}}$.

From the fundamental dynamics of the computational random search movement inspired by bacteria, the model (5.15) is considered for a convergence analysis based-on Lyapunov stability theorem. The main concept of Lyapunov stability is that the total energy in a system continually decreases, then the system will asymptotically reach a zero energy state associated with an equilibrium point of the system. A system is said to be asymptotically stable if all the states approach the equilibrium state in finite time (Das et. al., 2009).

In general analysis definition, a vector variable is denoted by $\vec{x}$ instead of θ, and its objective function of the vector variable as $f(\vec{x})$ instead of $J(\theta)$. The following materials review some basic concepts and interpretations of the theorems from a standard textbook of nonlinear control theory:

Definition 1: A point $\vec{x} = \vec{x}_e$ is called an *equilibrium state*, if the dynamics of the system is given by $\frac{d\vec{x}}{dt} = f(\vec{x}(t))$ becomes zero at $\vec{x} = \vec{x}_e$ for any time t, i.e. $f(\vec{x}(t)) = 0$. The equilibrium state is also called equilibrium (stable) point in D-dimensional hyperspace, when the state $\vec{x}_e$ has D components.

Definition 2: A scalar function $V(\vec{x})$ is said to be *positive definite* with respect to the point $\vec{x}_e$ in the region $\|\vec{x}-\vec{x}_e\| \le K$, if $V(\vec{x}) > 0$ at all points of the region except at $\vec{x}_e$ where it is zero.

Definition 3: A scalar function $V(\vec{x})$ is said to be *negative definite* if $-V(\vec{x})$ is positive definite.

Definition 4: A dynamics $(d\vec{x}/dt) = f(\vec{x}(t))$ is *asymptotically stable* at the equilibrium point $\vec{x}_e$, if it is stable in the sense of Lyapunov, i.e., for any neighbourhood $S(\varepsilon)$ surrounding $\vec{x}_e$ ($S(\varepsilon)$ contains points $\vec{x}$ for $\|\vec{x}-\vec{x}_e\| \le \varepsilon$) where there is a region $S(\delta)$ ($S(\delta)$ contains points $\vec{x}$ for which $\|\vec{x}-\vec{x}_e\| \le \delta$), $\delta < \varepsilon$, such that trajectories of the dynamics starting within $S(\delta)$ do not leave $S(\varepsilon)$ as time $t \to \infty$; the trajectory starting within $S(\delta)$ converges to the origin as time t approaches infinity.

The condition for stability of a dynamics from the Lyapunov's theorem can be presented as follows.

Lyapunov's Stability Theorem: Given a scalar function $V(\vec{x})$ and some real number $\varepsilon > 0$, such that, for all $\vec{x}$ in the region $\|\vec{x}-\vec{x}_e\| \le \varepsilon$, the following conditions hold.

- $V(\vec{x}_e) = 0$.
- $V(\vec{x}) > 0$ for $\vec{x} \ne \vec{x}_e$, i.e. $V(\vec{x})$ is positive definite.
- $V(\vec{x})$ has continuous first partial derivatives with respect to all components of $\vec{x}$.

Then, the equilibrium state $\vec{x}_e$ of the system $(d\vec{x}/dt) = f(\vec{x}(t))$ is as follows.

- *Asymptotically stable* if $(dV/dt) < 0$, i.e. dV/dt is negative definite.
- *Asymptotically stable* in the large if $(dV/dt) < 0$ for $\vec{x} \neq \vec{x}_e$, and in addition, $V(\vec{x}) \to \infty$ as $\|\vec{x} - \vec{x}_e\| \to \infty$.

Theorem 1 *(Main Result)***:** Let the random movement dynamics be represented by (5.17), and $\theta=\theta_0$ is the single optimum (minimum case) in the region of search space. Then, this optimum is asymptotically stable if

$$C \begin{cases} > \frac{2}{q}\left|\frac{\theta-\theta_0}{J(\theta)}\right|, & \text{if } \theta \neq \theta_0 \\ = 0, & \text{if } \theta = \theta_0 \end{cases} \tag{5.19}$$

Proof: In order to determine the equilibrium point for the system (by Definition 1). $\frac{d\theta}{dt} = 0$ is considered, then obtaining,

$$\frac{C\Delta}{2} - \frac{qC^2}{4}G = 0 \tag{5.20}$$

If the search movement converges to the optimum point, then the equilibrium point is $\theta_e = \theta_0$, and also the gradient G is equal to zero at this point. Substituting $G = 0$ in equation (5.20), $C = 0$ is obtained as well. Thus, the step-height C should become zero at $\theta = \theta_0$ for the equilibrium point to be located at the desired optimum, i.e.

$$C = 0, \qquad \text{if } \theta = \theta_0 \tag{5.21}$$

To test the stability, consider a scalar function,

$$V(\theta) = \frac{qC^2}{4} J(\theta) - \frac{C\Delta}{2}(\theta - \theta_0) \tag{5.22}$$

In order to qualify as a Lyapunov function, $V(\theta)$ must be positive definite with respect to the equilibrium point θ_0. From definition 2, $V(\theta)$ must satisfy the relation $V(\theta_0) = 0$, and $V(\theta) > 0$ if $\theta \neq \theta_0$. At the point θ_0,

$$V(\theta_0) = \frac{qC^2}{4} J(\theta_0) - \frac{C\Delta}{2}(\theta - \theta_0) \tag{5.23}$$

According to (5.21), $C = 0$ at $\theta = \theta_0$, then

$$V(\theta_0) = \frac{qC^2}{4} J(\theta_0) = 0 \tag{5.24}$$

For definition 2 to be satisfied, the following inequality must hold

$$\frac{qC^2}{4} J(\theta) - \frac{C\Delta}{2}(\theta - \theta_0) > 0, \qquad \forall \theta \neq \theta_0 \tag{5.25}$$

or

$$\frac{qC}{2} J(\theta) > (\theta - \theta_0)\Delta, \qquad \forall \theta \neq \theta_0 \tag{5.26}$$

According to the first assumption, $J(\theta) \neq 0$ for all $\theta \neq \theta_0$, and consider that the constant value $q > 0$, dividing both sides of equation (5.26) by $qJ(\theta)/2$, we obtain

$$C > \frac{2(\theta - \theta_0)\Delta}{qJ(\theta)}, \qquad \forall \theta \neq \theta_0 \tag{5.27}$$

If the right-hand side of (5.27) is negative, it will lead to a trivial condition as step-height C is always positive.

$$\left| \frac{2(\theta - \theta_0)\Delta}{qJ(\theta)} \right| \geq \frac{2(\theta - \theta_0)\Delta}{qJ(\theta)} \tag{5.28}$$

where $|\Delta| = 1$,

$$\frac{2}{q}\left| \frac{\theta - \theta_0}{J(\theta)} \right| \geq \frac{2(\theta - \theta_0)\Delta}{qJ(\theta)} \tag{5.29}$$

Thus, if C satisfies the relational form $C > \frac{2}{q}\left| \frac{\theta - \theta_0}{J(\theta)} \right|$ for all $\theta \neq \theta_0$, then $C > \frac{2}{q}\left| \frac{\theta - \theta_0}{J(\theta)} \right| \geq \frac{2(\theta - \theta_0)\Delta}{qJ(\theta)}$ for all $\theta \neq 0$, $V(\theta)$ is then positive definite. Thus, C satisfies conditions (5.21) and (5.27). Now, consider the first derivative of V,

$$\frac{dV}{dt} = \frac{dV}{d\theta} \cdot \frac{d\theta}{dt} \tag{5.30}$$

From (5.22), we could obtain

$$\frac{dV}{d\theta} = \frac{qC^2}{4} \cdot \frac{dJ(\theta)}{d\theta} - \frac{C\Delta}{2} = -\left(-\frac{qC^2}{4} G + \frac{C\Delta}{2} \right) \tag{5.31}$$

Substituting $\frac{d\theta}{dt}$ and $\frac{dV}{d\theta}$ in (5.30) by (5.15) and (5.31), respectively, we obtain that

$$\frac{dV}{dt} = -\left(-\frac{qC^2}{4} G + \frac{C\Delta}{2} \right)^2 < 0, \quad \text{if } \forall \theta \neq \theta_0 \tag{5.32}$$

and also, $\frac{dV}{dt} = 0$, if $\theta = \theta_0$ (as $C = 0$ and $G = 0$ at $\theta = \theta_0$). Referring to definition 3, $\frac{dV}{dt}$ is negative definite. Therefore, the random search movement inspired by bacterial dynamics or the random walk front-end as represented by (5.17) is asymptotically stable with respect to the optimum $\theta = \theta_0$ if the step size satisfies the conditions (5.21) and (5.27) simultaneously. This proof is to ensure that the proposed random walk front-end will not impair the convergence of the proposed algorithms.

This completes the proof. ❑

The benefit of the above proof is to guarantee that the search movement of the random walk front-end has convergent property. In other words, this part of algorithms always concentrates its searches in some sub-spaces having good-quality solutions. Therefore, it provides some solutions of good quality to be used further by the ATS as its initial solutions. Once the front-end finishes its task, it has no connection to the ATS, and the ATS performs solely until an optimum solution is found.

5.3.2 Convergence of the Tabu Search with Backtracking Mechanism

Researchers at the School of Electrical Engineering, Suranaree University of Technology, have incorporated the backtracking (*BT*) mechanism to the original Tabu Search (TS) for almost ten years. They also conducted rigorous analysis of the algorithms namely adaptive tabu search (ATS). This section reviews their main theorems without proofs.

Theorem 2: If a total number of member, *m*, in a sub search space is large enough to provide good representatives of a neighborhood, a local minimum nearby can be found by generating a sequence of a few successive sub search spaces.

For proof see (Puangdownreong et al., 2004; Sujitjorn et al., 2006).

Theorem 3: The *BT* mechanism leads the search process to obtain multiple local minima within search space. Among them, one is the global minimum.

For proof see (Puangdownreong et al., 2004; Sujitjorn et al., 2006).

By intuition, considering the theorems 1, 2 and 3, one can conclude that the proposed algorithms consisting of the random walk front-end and the ATS without AR mechanism possesses a convergence property because the ATS and the front-end are convergent.

5.4 Conclusion

This chapter has presented the convergence analysis of the modified ATS algorithm (included an initial part of BF-TS algorithm) by means of mathematical

proofs. Firstly, the convergence of the random walk front-end of the proposed algorithms is proved via Lyapunov's stability concepts. Secondly, the TS with *BT* mechanism has been known for its convergence property (Puangdownreong et al., 2004; Sujitjorn et al., 2006). By intuition, the convergence of the proposed algorithms resulted from combination of the two can be concluded.

CHAPTER VI

SEARCH PERFORMANCE COMPARISONS

6.1 Introduction

As previously mentioned, one advantage of the ABFO algorithm is its dominant explorative characteristic. Using the adaptive random movement strategy, the algorithm can increase the opportunity to visit many high-quality solutions dispersed over the search space in a short duration. In contrast, the ATS is capable of tracking down an elite solution in a short duration due to its *AR* mechanism. Furthermore, the *BT* mechanism of the ATS can also help the search to escape from a deadlock, but it lacks a capability of focusing on a high-quality initial solution and also, in some cases, the neighbour solutions of the ATS are randomly generated within the limit of search radius until it cannot escape by the *BT* mechanism. Therefore, the adaptive random movement has been also applied to heuristically generate neighbour solutions with small step sizes depending on the current cost function. Thus, the two algorithms complement each other. So far, the thesis has proposed hybrid or cooperative algorithms, namely, the BF-TS and the modified ATS algorithm.

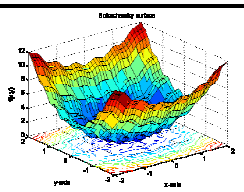
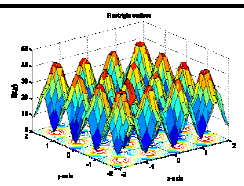
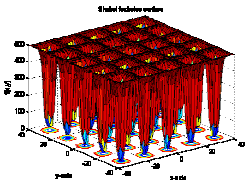
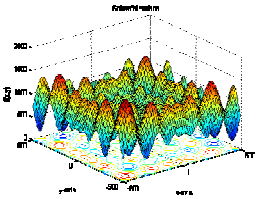
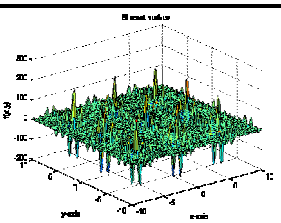
Regarding the previous Chapter, it has presented the convergence proof of the proposed algorithm. The results show that the convergence of the random walk front-end of the proposed algorithms via Lyapunov's stability concepts will not impair the convergence of the proposed algorithm. Consequently, this chapter presents search

comparison analysis among the proposed algorithms, the original ones and other existing algorithms, namely, the ABFO, the ATS, the BF-TS, the modified ATS, the IWO and the GA. These algorithms described previously are performed by MATLAB codes on Pentium IV, 2.4 GHz, 640 MB SD-RAM with the same environment in which search performances are assessed by using five surface optimization problems, details are given in Section 6.2. Subsequently, Section 6.3 shows tuning search factors of the proposed algorithms to obtain the global best results and suggestions of search parameters for each algorithm. Section 6.4 and 6.5 presents the search performance comparison analysis and conclusion, respectively.

6.2 Surface Optimization Functions

In the field of algorithm comparison, it is common to compare the performance of each search algorithm by using various surface optimizations known as benchmark functions which are also useful to evaluate characteristic of optimization algorithms (Chen et al., 2009; Dasgupta et al., 2010; Gandomi and Alavi, 2012; Kluabwang et al., 2009; Mehrabian and Lucas, 2006). Therefore, this thesis uses five surface optimization problems. The search problems include the minimizations of the Bohachevsky, Rastrigin, Shekel's fox-holes, Schwefel and Shubert functions. Table 6.1 summarizes the functions with their abbreviations of BF, RF, SF, SchF and ShuF, respectively, with minimum cost, J_{min}, as a stop criterion.

Table 6.1 Summary of the functions used for performance test.

Test functions	Equations	Surfaces plots
BF	$f(x,y)=x^2+2y^2-0.3\cos(3\pi x)-0.4\cos(4\pi y)+0.7$ global minimum of 0 at (0,0), search space: [2 2;-2 -2] and $J_{min} \le 1\times10^{-9}$.	
RF	$f(x,y)=x^2+y^2-10\cos(2\pi x)-10\cos(2\pi y)+20$ global minimum of 0 at (0,0), search space: [2 2;-2 -2] and $J_{min} \le 1\times10^{-8}$.	
SF	$f(x_1,x_2)=\left[\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right]^{-1}$ when $a_{ij}=\begin{pmatrix}-32 & -16 & 0 & 16 & 32 & -32 & \dots & 0 & 16 & 32\\ -32 & -32 & -32 & -32 & -32 & -16 & \dots & 32 & 32 & 32\end{pmatrix}$. global minimum of 0.9980 at (-32,-32), search space: [40 40;-40 -40] and $J_{min} \le$ 0.9990.	
SchF	$f(x)=418.9829n-\sum_{i=1}^{n}(x_i\sin\sqrt{\lvert x_i\rvert})\quad ;n=2$ global minimum of 0 at (420.9687, 420.9687), search space: [500 500;-500 -500] and $J_{min} \le 1\times10^{-4}$.	
ShuF	$f(x_1,x_2)=\sum_{i=1}^{5} i\cos((i+1)x_1+i)*\sum_{i=1}^{5} i\cos((i+1)x_2+i)$ when $-10\le x_1,x_2\le10$. global minima of equal values -186.7309 at 18 different locations, search space: [10 10;-10 -10] and $J_{min} \le$ -186.73.	

6.3 Tuning Search Factors of the Proposed Algorithms

In order to compare the search performance of each optimization algorithm, search parameters are carefully considered to achieve the best results. Many published researches (Karimkashi and Kishk, 2010; Mehrabian and Lucas, 2006; Passino, 2002; Puangdownreong et al., 2004; Sujitjorn et al., 2006) presented the tuning for appropriate search parameters, especially for new algorithm approaches, before testing the specified problems. Therefore, this section presents the tuning search parameters of the proposed algorithms and also presents the suggestions of the tuning search parameters which were employed by the previous researchers, such as the search parameters of the ATS follow the recommendation by Sujitjorn et al. (Sujitjorn et al., 2006). As the results of the ATS showed that the radius is 7.5-15 % of search space, the member of neighbour solutions, *N*, should be set between 30-40 for not too long search time. The numbers of backtracking and maximum backtracking are set nearly 5-15. The reduction of the radius should be set 20-25 % of the previous radius. The results are summarized in Table 6.2, which was exhaustively studied considering the effects of tuning parameters in order to find global optima on surface functions.

Table 6.2 ATS parameters.

Test functions	N	$Count_{max}$	R	BT, n_re_back	AR		
					Stage I	Stage II	Stage III
BF	30	10,000	0.2	5	$J<1\times10^{-1}$, $R=2\times10^{-3}$	$J<1\times10^{-3}$, $R=2\times10^{-4}$	-
RF	30	10,000	0.2	5			-
SF	30	10,000	0.8	5	$J<5$, $R=0.5$	$J<2$, $R=0.1$	-
SchF	30	10,000	100	5	$J<100$, $R=50$	$J<10$, $R=0.01$	$J<1$, $R=0.001$
ShuF	30	10,000	1.0	5	$J<1$, $R=1\times10^{-2}$	$J<-1$, $R=1\times10^{-3}$	-

Among the parameters that affect the convergence of the IWO algorithm, three parameters, the a initial value of standard deviation, $\sigma_{initial}$, the final value of standard deviation, σ_{final}, and the nonlinear modulation index, n, should be tuned carefully. From the experimental study of search parameters with benchmark multi-dimensional functions of Mehrabian and Locus (Mehrabian and Locus, 2006; Karimkashi and Kishk, 2010), these can be summarized as follows:

- The initial standard deviation, $\sigma_{initial}$, should be chosen to allow the algorithm to explore the whole search space, aggressively. It seems that the IWO works well if the $\sigma_{initial}$ value is set around 1-5 % of the range of each variable.

- The final standard deviation, σ_{final}, should be selected carefully to allow the optimizer to find the optimal solution as accurate as possible. A finer local optimum solution can be achieved by decreasing this parameter. However, it should be noticed that tuning the σ_{final} value much smaller than the precision criteria of the optimization variables does not improve the cost values and could deteriorate the convergence rate of the optimization. Therefore, the σ_{final} in each problem should be selected based on the resolution requested for the final answer.

- It was shown that the value of nonlinear modulation index has a considerable effect on the performance of IWO. It was suggested that the best choice for this is 3.

- Maximum and minimum numbers of seeds are the two other important parameters needed to be selected. Based on different examples, it can be concluded that selecting the maximum number of seed, $s_{\max}$, between 2 and 5 leads to a good performance of the optimizer. Moreover, the minimum number of seed, $s_{\min}$ is set to zero for all examples.

- The maximum number of plants is another parameter that should be chosen in the IWO. Parametric studies show that increasing this parameter not necessarily the performance of the algorithm increases. It was found that the best performance can be achieved for many problems when the maximum number of plant, p_{max}, is set between 10-30.

The search parameters of the IWO according to the experimental study from the above description are summarized in Table 6.3.

Table 6.3 IWO parameters.

Test functions	N_0	p_{max}	s_{max}	s_{min}	n	$\sigma_{initial}$	σ_{final}	$iter_{max}$
BF	10	30	2	0	3	0.2	1×10^{-9}	1000
RF	10	30	2	0	3	0.5	1×10^{-8}	1000
SF	10	30	2	0	3	15	0.9990	1000
SchF	10	30	2	0	3	50	1×10^{-1}	10000
ShuF	10	30	2	0	3	1	1×10^{-10}	1000

Search parameters for search performance comparison of the GA follow default values set by MATLAB-GA Toolbox. The problem of interest is organized in an objective function which represents the fitness of a candidate solution. The maximum number of population is set to 30 for this experimental study.

Passino (Passino, 2002; Liu and Passino, 2002) has given some suggestions for search parameters of the BFO as follows:

- If the *C(i)* values are too large and the optimum value lies in a valley with steep edges, it will tend to jump out of the valley, or it may simply miss possible local

minima by swimming through them without stopping. On the other hand, if the *C(i)* values are too small, the convergent rate will be slow or the search will not be able to move out of the valley.

- If setting of large values for N_c, it means that there are many chemotactic steps, hopefully, more optimization progress, but more computational complexity. If the size of N_c is chosen to be too small, the algorithm will generally rely more on reproduction, and in some cases, it could more easily get trapped by a local minimum (premature convergence). While N_s is creating a bias in the random walk (which would not occur if $N_s = 0$), with large values tend to bias the walk more in the direction of climbing down the hill but it should not be very large to prevent the solution being trapped in local minima.

- If N_c is large enough, the value of N_{re} affects how the algorithm ignores bad regions and focuses on good ones, since bacteria in relatively nutrient-poor regions die (this models, with a fixed population size, the characteristic where bacteria will tend to reproduce at higher rates in favorable environments). If N_{re} is too small, the algorithm may converge prematurely; however, larger values of N_{re} clearly increase computational complexity

- For a low value of N_{ed}, the algorithm will not rely on random elimination-dispersal events to try to find favorable regions. A high value increases computational complexity but allows the bacteria to look in more regions to find good nutrient concentrations. Clearly, if p_{ed} is also large, the algorithm can degrade to random exhaustive search. If, however, it is chosen appropriately, it can help the algorithm jump out of local optima and into a global optimum. So it is set equal to 0.25.

- The frequency of chemotaxis steps is greater than the frequency of reproduction steps, which is in-turn greater in frequency than elimination-dispersal events, $N_C > N_{re} > N_{ed}$.

- The magnitude of $d_{attract}$ and $h_{repellant}$ should be the same so that when the bacteria population converges there is no penalty added to the cost function, i.e. J_{cc} will be 0. On the other hand, their numerical values could be decided based on the required variation in magnitude of the actual cost function J to obtain a satisfactory result.

- The value of $w_{attract}$ and $w_{repellant}$ should be selected appropriately, when the distance between bacteria is large the penalty J_{cc} should be more so that the bacteria will try to move together.

Due to the ABFO algorithm is a new set of bio-inspired algorithms, the tuning search factors for specific problems were not carefully studied yet, there were only some useful suggestions from the inventors. For a fair comparison, in this thesis, the ABFO has been studied to reveal the effects of search parameters in order to obtain appropriate search factors. From previous recommendations, the *C(i)* values have to be varied correspondingly to the cost functions and a positive constant, α. Thus, α will be tested against five surface optimization problems by fixing $N_C = 200$ and $S = 10$ for all problems. The example ranges of α are 0.001, 0.01, 0.1, 1, 10, 100 and 1000. The N_C will be selected appropriately after testing the positive constant, α as well. The various values of the N_C are 10, 30, 50, 100, 150, 200 and 250. While the number of bacteria, S, is tuned after obtaining the appropriate search parameter, N_C, for each problem. The values of S are varied as 10, 20, 30, 40 and 50. These tuning of

parameters will be tested over 50 trial runs for all five problems. Setting the cell-to-cell attractant parameters following Passino; $d_{attract} = h_{repellant} = 0.1$, $w_{attract} = 0.2$ and $w_{repellant} = 10$ (equal to 1 for BF and RF). The values for these parameters are simply chosen to illustrate general bacterial behaviors, not to represent a particular bacterial chemical signaling scheme. Note that N_S, N_{re} and N_{ed} are used as Passino's suggestions, these are 4, 4 and 2, respectively. Although they do not seriously influent convergent rate, it has been considered that their large values could result in a long search time. The maximum numbers of iteration is 10,000. The stop criterions, J_{min} are referred to Table 6.1. The results of tuning search factors of the ABFO algorithm for five surface optimization problems are shown in Table 6.4 to 6.6.

Table 6.4 Results of a positive constant α related to step size (inner parameters of step size C).

	Average search time (seconds)					Average search rounds					Number of times to obtain J_{min}				
α	BF	RF	SF	SchF	ShuF	BF	RF	SF	SchF	ShuF	BF	RF	SF	SchF	ShuF
0.001	-	-	110.9239	7903.4551	901.2376	-	-	14.84	290.16	312.08	-	-	50	50	50
0.01	-	-	123.4290	5758.2321	1531.3264	-	-	13.82	214.36	626.04	-	-	50	50	50
0.1	-	-	**100.1083**	1734.4656	1313.1886	-	-	**11.29**	64.58	509.08	-	-	**50**	50	50
1	1953.4733	-	106.9219	**935.0031**	1077.3161	505.56	-	13.66	**33.54**	458.28	50	-	50	**50**	50
10	**94.8669**	1834.1274	121.9938	1171.1556	763.9401	**25.06**	1306.02	14.22	42.00	287.72	**50**	50	50	50	50
100	-	**141.6667**	103.1503	2727.5101	1306.8985	-	**100.12**	12.08	106.62	506.72	-	**50**	50	50	50
1000	-	-	129.9436	-	**127.8304**	-	-	15.70	-	**49.12**	-	-	50	-	**50**

Table 6.4 shows the tuning search parameters of a positive constant, α for each surface optimization problem. This search parameter is inversely related to the step size *C* but it is directly related to the cost function, *J* at that location. As the results, the BF and RF are 2-dimensional values having a short search space within [2 2; -2 -2], therefore, the random step walk should not be larger than its search space but should be small enough to exploit an area around the global solution, nevertheless, depending on the current cost function. The appropriate ones for BF and RF are obtained from experiments to achieve better average search time and search round, and satisfy the stop criterion over 50 trial runs that are around 10 and 100, respectively. Note that the positive constants, α for BF are approximately 1 and 10, can only provide the global solution and similarly, the positive constants, α for RF are around 10 and 100, can provide the global solution over trial runs. The SF and ShcF are kinds of surface optimization problems containing many local traps as shown the 2-dimentional graphics in Table 6.1. Therefore, the step size that is embedded by a positive constant, α should be large enough to escape the local deadlocks and turn to be small when the cost values decrease nearly to the global optimum. As the results, the constant α for SF and SchF are around 0.1 and 1, respectively. Note that the α for SchF of 1,000 cannot provide a global solution on all trial runs. While the ShuF consists of 18 global optima and contains many local traps as well, as the result in Table 6.4 shows that all among interested values of α can achieve the trial runs to obtain J_{min} but the appropriate positive constant, α to incur the best average search time and search round is around 1,000. Therefore, the appropriate α are achieved in Table 6.4, they will be used to obtain further numbers of chemotactic steps, N_C as follows.

Table 6.5 Results of number of iterations to be carried out in chemotactic events.

N_C	Average search time (seconds)					Average search rounds					Number of times to obtain J_{min}				
	BF	RF	SF	SchF	ShuF	BF	RF	SF	SchF	ShuF	BF	RF	SF	SchF	ShuF
10	-	-	819.6080	**155.4058**	813.8530	-	-	331.52	**59.24**	5517	-	-	50	**50**	32
30	-	-	36.3287	464.6628	614.5856	-	-	10.04	44.64	975.30	-	-	50	50	50
50	5541.1005	834.1899	**35.4932**	706.7253	480.6572	4935.90	860.72	**8.60**	38.52	597.30	37	50	**50**	50	50
100	2034.4599	6115.8947	63.1532	851.8879	851.8879	755.14	4873.48	9.26	39.46	346.90	50	50	50	50	50
150	943.2302	354.6018	77.1374	958.5308	186.9483	295.00	267.80	10.06	33.68	87.24	50	50	50	50	50
200	94.8669	141.6667	100.1083	935.0031	127.8304	25.06	100.12	11.29	33.54	49.12	50	50	50	50	50
250	**78.1629**	**152.9535**	115.4228	1100.9582	**119.5011**	**12.52**	**32.86**	9.74	38.62	**39.66**	**50**	**50**	50	50	**50**

Note that blanks shown in Tables 6.4 and 6.5 indicate that the J_{min} criteria are not satisfied on all trial runs.

Table 6.6 Results of a number of bacterial population.

S	Average search time (seconds)					Average search rounds					Number of times to obtain J_{min}				
	BF	RF	SF	SchF	ShuF	BF	RF	SF	SchF	ShuF	BF	RF	SF	SchF	ShuF
10	**78.1629**	152.9535	**35.4932**	155.4058	119.5011	**12.52**	32.86	**8.60**	59.24	39.66	**50**	50	**50**	50	50
20	87.2269	128.1587	114.8042	142.5779	293.0735	9.92	16.28	24.06	35.32	44.84	50	50	50	50	50
30	100.5877	**118.6347**	53.3919	**162.5890**	**32.8721**	7.48	**9.82**	10.78	**28.44**	**2.28**	50	**50**	50	**50**	**50**
40	124.3222	130.4746	43.9693	220.3722	45.3216	5.88	6.36	8.58	27.72	2.34	50	50	50	50	50
50	140.2700	188.7482	45.9329	220.4426	59.1667	5.76	5.32	8.46	23.28	2.84	50	50	50	50	50

Regarding Table 6.5, it shows the conclusions of tuning search parameters N_C for each surface optimization problem. Nevertheless, the number of iterations to be carried out in chemotactic step, according to Passino's suggestions should be selected not too small because it could get trapped by a local minimum and not too high in order to avoid heavily loaded computation. These N_C values are obtained from the experiment taking into account the reasonable average search time and search round, and also the trial runs to satisfy J_{min}. Referring to the figures shown in Table 6.5, one can see that all values of N_C work-out well for the ABFO to find the global solutions for SF, SchF and ShuF. This is not the case for BF and RF because the N_C of 10 and 30 are too small for enabling the searches to escape from some local entrapments. Therefore, the appropriate values of N_C are 250, 250, 50, 10 and 250 for BF, RF, SF, SchF and ShuF, respectively. These N_C values are further used for studying the effects of the parameter S in Table 6.6.

Table 6.6 shows experimental results of tuning the number of bacterial population, S. Notice that all sample values of S can achieve the global solutions over 50 trial runs for each surface optimization problem. If the S is increased, the search algorithm gradually takes a short search round to meet the stop criterion, J_{min}, meanwhile it spends too much search time. Therefore, the appropriate results of S from the tuning tests are 10, 30, 10, 30 and 30 for BF, RF, SF, SchF and ShuF, respectively.

Based-on our tuning experiments so far, the following recommendations for the search parameters, α, N_C and S of the ABFO have been obtained as a guideline when one applies the algorithm: α should be 0.1-1,000, N_C should be 50-250, and S should be 10-40. The proper search parameters of the ABFO are summarized in Table 6.7. The

best results obtained from experiments using such parameters will be taken to compare the performances against other mentioned algorithms.

Table 6.7 ABFO parameters.

Test functions	S	N_c	N_s	N_{re}	N_{ed}	P_{ed}	α	$d_{attract}$	$h_{repellant}$	$w_{attract}$	$w_{repellant}$
BF	10	250	4	4	2	0.25	10	0.1	0.1	0.2	1
RF	30	250	4	4	2	0.25	100	0.1	0.1	0.2	1
SF	10	50	4	4	2	0.25	0.1	0.1	0.1	0.2	10
SchF	30	10	4	4	2	0.25	1	0.1	0.1	0.2	10
ShuF	30	250	4	4	2	0.25	1,000	0.1	0.1	0.2	10

According to the previous studies to assess the tuning factors affecting the performance of the ABFO, the search parameters including S, N_C and α are applied to the BF-TS and modified ATS algorithms as a part of initial search parameters. The cell-to-cell attractant effect is neglected for these proposed algorithms. The other search parameters related to the ATS follow the materials shown in Table 6.2. The new modified version of ATS should be extensively studied to obtain appropriate search parameters in order to compare its search performance against other optimization techniques on a fair basis.

The search parameters of adaptive random movement inspired by bacteria including, S_2, N_{C2} and α_2 should be considered taking into account reasonable average search time, search round and trial runs to satisfy J_{min}. The search parameter α_2 will be

firstly tuned to obtain the stop criterion J_{min} following Table 6.1. Other variables, N_{C2} for BF, RF, SF, SchF and ShuF are set as 100, 300, 100, 100 and 250, respectively. The maximum count and S_2 are defined as 10,000 and 1, respectively, for all surface optimization problems. After obtaining the appropriate parameters α_2, the search parameters S_2 and N_{C2} will be further tested, respectively. Trial runs of 50 are used among these tests. The results of tuning search parameters of the modified ATS based-on adaptive random movement inspired by bacteria are shown in Tables 6.8-6.10.

Table 6.8 summarizes the results of positive constants, α_2, which are inner parameters of the step size, C_2. Appearing in the modified ATS, the factor α_2 influences the step size for generating neighbour solutions. The step size C_2 should be small enough to avoid an oscillation near an optimum point. Moreover, it also depends on a current cost value. Therefore, the main purpose of tuning this parameter is to obtain an appropriate α_2 value for each problem without using computing resources exhaustively. From previous studies of tuning the factor α of the ABFO, it can be assumed that if α is large, the step size should be small. This idea can be applied for generating neighbour solutions, and selecting an initial set of parameter ranges. The interesting parameter ranges should be at least over the appropriate parameters, α of the ABFO, nevertheless, the smaller values also have been considered.

Table 6.8 Results of a positive constant α_2 related to step size (inner parameters of step size, C_2).

α_2	Average search time (seconds)		Average search rounds		Average deadlocks		Number of times to obtain J_{min}	
	BF	RF	BF	RF	BF	RF	BF	RF
1	**33.59**	**-**	**60.08**	**-**	**0**	-	**50**	**-**
5	154.17	6434.11	275.76	4049.60	0	0	50	30
10	322.61	102.32	589.42	72.28	0	**0**	50	50
15	491.90	**87.12**	898.62	**61.76**	0	0	50	**50**
20	611.10	111.26	1248.78	79.12	0	0	50	50
25	3502.25	144.99	6360.44	101.74	0	0	50	50
50	2245.17	379.54	10000	237.56	0	0	0	50
100	3874.44	860.23	9996.78	571.50	0	0	1	50

α_2	Average search time (seconds)		Average search rounds		Average deadlocks		Number of times to obtain J_{min}	
	SF	SchF	SF	SchF	SF	SchF	SF	SchF
0.001	12.60	127.41	26.02	320.08	0	0	50	50
0.01	11.66	**11.08**	23.80	**30.22**	0	**0**	50	**50**
0.10	9.86	86.99	20.56	185.48	0	0	50	50
1	**6.27**	86.71	**13.46**	185.38	**0**	0	**50**	50
10	69.35	83.75	141.72	179.04	0	0	50	50
100	-	80.83	-	172.46	-	0	-	50

α_2	Average search time (seconds)	Average search rounds	Average deadlocks	Number of times to obtain J_{min}
	ShuF	ShuF	ShuF	ShuF
5000	697.39	661.84	0	50
10000	487.91	477.08	0	50
15000	291.51	236.98	0	50
20000	125.38	123.26	0	50
25000	77.41	78.28	0	50
30000	**35.69**	**33.22**	**0**	**50**

Referring to the figures shown in Table 6.8, if α_2 for BF becomes larger, the average search time and search round increase. When α_2 is 50, the search failed for all 50 trails. α_2 = 1 provides the shortest average search time of 33.59 seconds, and the minimum search round of 60.08. For the case of RF, if α_2 values are lower than 5, the search cannot provide an optimum solution on all trial runs. As a result, the suitable value of α_2 is approximately 15, which provides the shortest average search time of 87.12 seconds, and the minimum search round of 61.76. For the case of SF, our computing results indicate that an appropriate value of α_2 for SF should be around 1. It renders the shortest average search time of 6.27 seconds, and the minimum search round of 13.46. With the SF problem, all α_2 values lead to the global solutions over 50 trial runs. Based on the results shown in Table 6.8 for SchF, the following points can be drawn: (i) the search can achieve the global solution over 50 trial runs, (ii) for α_2 less than 0.01, it takes longer average search time and more search rounds when compare to the other values, and (iii) for α_2 more than 0.01 up to100, the average values of search time and search round are quite similar to the other cases. Thus, the best result of α_2 value for SchF is about 0.01 that renders average search time and search round of 11.08 seconds and 30.22, respectively. Considering the ShuF, this particular function has many local optima near to each other, and contains several global solutions (see Table 6.1). One way to avoid local traps or oscillations around terrains containing a global solution is to employ a fairly short step size. Among those parameters, 5,000-30,000 are considered, which are stated over α, 5-30 times. According to the results, such parameters provide the global solutions over 50 trials. When we consider α_2 for

the ShuF, it is found that an increase in α_2 leads to some gradual increases in the average values of search times and search round. While the α_2 is approximately 30,000, it provides the shortest search time and search round of 35.69 seconds and 33.22, respectively. Note that the deadlock situations are not found in this test. Such parameters obtained for these compact experiments will be further used to aid an adjustment of the adaptive random movement, N_{C2}.

Now we consider the effects of the adaptive random movement number, N_{C2}, on search performance in order to find out the most suitable value for further uses. The number N_{C2} generates randomly neighbour solutions in the range of 10-200 for each surface optimization problem. Regarding the results shown in Table 6.9, the stop criterion J_{min} can be met for all problems over 50 trial runs. For the case of BF, the average values of search time and search round gradually decrease when the N_{C2} increases. N_{C2} = 200 provides the shortest average search time and the minimum search round of 27.22 seconds and 23.96, respectively. The results of RF show similar trends. Furthermore, the shortest average search time (79.64 seconds) and the minimum search round (66.42 rounds) are obtained when N_{C2} = 200. For the case of SF, the searches hit the global solutions with smaller numbers of search round when N_{C2} increases. Notice that the best result is obtained when N_{C2} is 100. The average search time and search round are 6.27 seconds and 13.46, respectively. Similarly, an appropriate value of N_{C2} = 100 for SchF is obtained, which provides reasonable average search time and search round of 11.08 seconds and 30.22, respectively. The

best result of N_{C2} equal to 50 for ShuF is obtained providing reasonable average search time and search round of 8.75 seconds and 39.46, respectively.

Table 6.9 Numbers of adaptive random movement N_{C2} for generating neighbour solution.

N_{C2}	Average search time (seconds)					Average search rounds					Average deadlocks					Number of times to obtain J_{min}				
	BF	RF	SF	SchF	ShuF	BF	RF	SF	SchF	ShuF	BF	RF	SF	SchF	ShuF	BF	RF	SF	SchF	ShuF
10	54.29	237.07	8.53	44.59	419.51	1001.4	2904.04	78.66	375.36	8374.92	0	0	0	0	0	50	50	50	50	9
30	39.22	127.68	9.73	12.74	75.81	252.72	822.94	48.48	95.82	535.64	0	0	0	0	0	50	50	50	50	50
50	36.12	242.23	9.70	7.39	**8.75**	138.30	437.70	37.30	48.12	**39.46**	0	0	0	0	**0**	50	50	50	50	**50**
100	33.59	82.96	**6.27**	**11.08**	18.58	60.08	88.86	**13.46**	**30.22**	48.30	0	0	**0**	**0**	0	50	50	**50**	**50**	50
150	31.33	92.82	9.58	21.67	14.66	29.04	81.02	11.58	22.38	24.08	0	0	0	0	0	50	50	50	50	50
200	**27.22**	**79.64**	11.56	47.56	26.49	**23.96**	**66.42**	8.88	21.26	32.86	**0**	**0**	0	0	0	**50**	**50**	50	50	50

From what described above, it can be concluded that if the number of the adaptive random movement, N_{C2}, increases, the average search round would decrease. When we consider the ratios between search time and search rounds, it is found that the resulted figures become greater. This means that the search consumes more time when N_{C2} increases. Interestingly, the deadlock does not occur at all. The N_{C2} results are further used to vary the numbers of bacteria S_2 for generating neighbour solutions.

Table 6.10 Numbers of bacteria S_2 for generating neighbour solutions.

S_2	Average search time (seconds)					Average search rounds					Average deadlocks					Number of times to obtain J_{min}				
	BF	RF	SF	SchF	ShuF	BF	RF	SF	SchF	ShuF	BF	RF	SF	SchF	ShuF	BF	RF	SF	SchF	ShuF
1	**27.22**	**79.64**	6.27	**11.08**	**8.75**	**23.96**	**66.42**	13.46	**30.22**	**39.46**	**0**	**0**	0	**0**	**0**	**50**	**50**	50	**50**	**50**
2	60.68	198.06	**7.75**	132.87	37.36	51.78	151.06	**9.54**	256.42	31.14	0	0	**0.24**	24.88	0	50	50	**50**	50	50
5	774.37	1688.31	36.74	748.54	44.02	62.22	121.86	5.20	403.42	15.60	1	0	0.12	46.42	0	50	50	50	50	50
10	712.46	1664.64	28.12	963.65	47.06	56.84	117.96	4.14	443.10	8.12	0.74	0	0.08	53.20	0.02	50	50	50	50	50

Table 6.10 presents results obtained from tuning the numbers of bacteria, S_2, for generating neighbour solutions. In order not to spend too much search time, we consider S_2 in the range of 1-10. Fifty trial runs based on this parameter help us obtain the stop criterion, J_{min}, for all test problems. For the BF problem, it is found that the averages of search time and search round increase when S_2 increases. The shortest average search time and the minimum search round of 27.22 seconds and 23.96, respectively, are obtained with S_2=1 without a deadlock. Similarly for the RF problem, S_2=1 provides reasonable averages of search time and search round at 79.64 seconds and 66.42, respectively. For the SF problem, in contrary, the average search time increases but the average search round decreases when S_2 increases. S_2=2 provides satisfactory averages search time and search round of 7.75 seconds and 9.54, respectively, with a few numbers of deadlocks. Overview results of the average deadlocks collected from tuning S_2 for the SF problem indicate that if S_2 is greater than 1, there will be more chances for the deadlock to occur. Similar situations are found with the SchF and the ShuF problems, where S_2=1 seems to provide the best results. Furthermore, for the ShuF problem, $S_2 > 5$ introduces more deadlocks and longer search time consumed as a consequence although the average search round shows smaller numbers. This argument is evident by the detailed figures in Table 6.10.

At this stage, some useful recommendations can be drawn to aid users of the modified ATS algorithms. These include (i) α_2= 0.1-20 % (related to the step size C_2) of the α values, (ii) N_{C2} = 50-200, and (iii) S_2 = 1-5. Using the search parameters being recommended helps the searches avoid deadlocks, and subsequently consume short

search time. Nonetheless, users of the algorithms are recommended to carefully run some trail searches in order to find out the most suitable set of the search parameters. For our performance comparison purposes, the suitable search parameters of the modified ATS are summarized in Table 6.11.

Table 6.11 Modified ATS parameters.

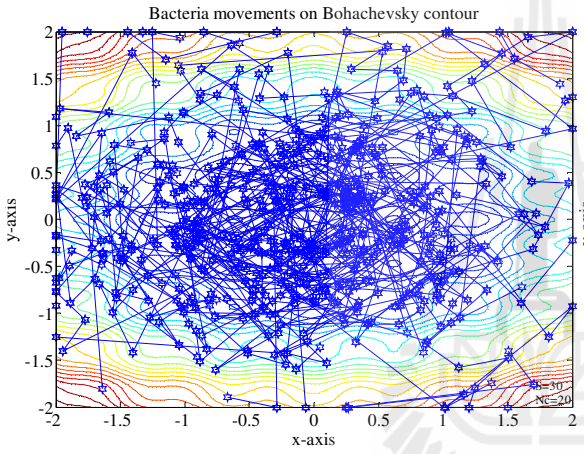
Test functions	S	N_C	S_2	N_{C2}	N_S	α	α_2	BT, n_re_back	$Count_{max}$
BF	10	250	1	200	4	10	1	5	10,000
RF	30	250	1	200	4	100	15	5	10,000
SF	10	50	2	100	4	0.1	1	5	10,000
SchF	30	10	1	100	4	1	0.01	5	10,000
ShuF	30	250	1	50	4	1000	30,000	5	10,000

The best results which are obtained from the tuning significant search parameters of modified ATS algorithm will be used to compare the performances with other algorithms in the next section.

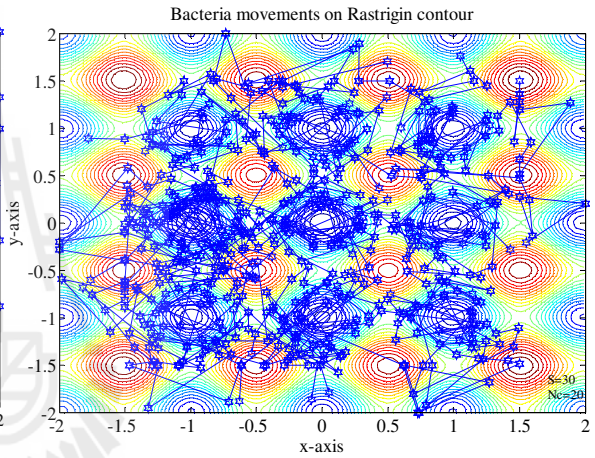
6.4 Search Performance Comparison Analysis

This section presents the details of search performance comparison among the following algorithms: ABFO, ATS, BF-TS, modified ATS, IWO and GA. According to our literature survey, many search algorithms lack a focusing capability on a high-quality initial solution. This can be considered as a disadvantage because a high-quality initial solution is likely to increase convergence rate of a particular search algorithm.

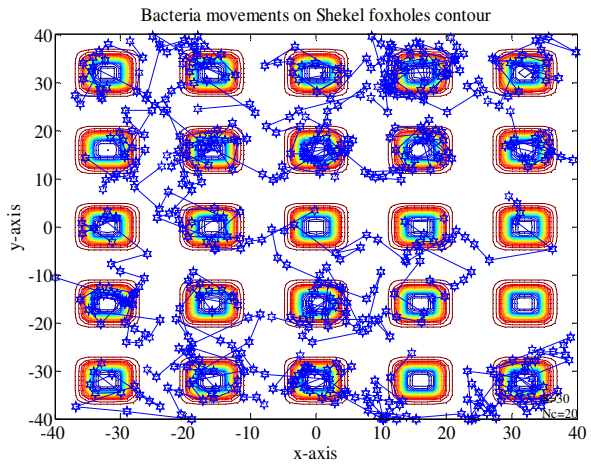
Regarding this, algorithms possessing strong explorative characteristics are very likely to spot an initial solution of high-quality. Therefore, some algorithms like the ABFO, for instance, are able to locate this kind of initial solutions. In contrast, algorithms possessing strong exploitative characteristics, such as the TS, are not. It means that the random movement mechanisms of the explorative type algorithms are very useful for generating high-quality initial solutions on real terrains being searched.



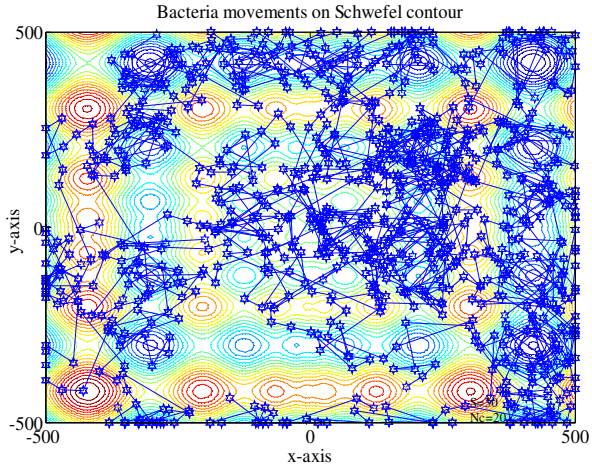
(a)



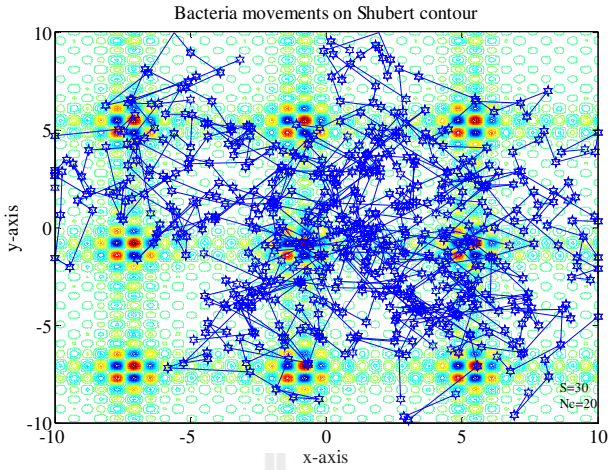
(b)



(c)



(d)



(e)

Figure 6.1 Random movement solutions inspired by bacteria on: (a) BF, (b) RF, (c) SF, (d) SchF and (e) ShuF surfaces.

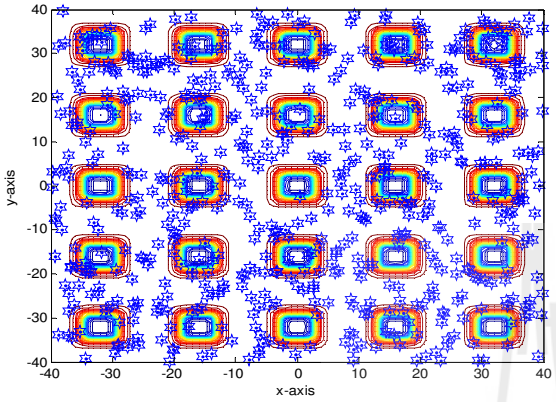
The idea has been applied for a modification made to the ATS such that it possesses a random movement front-end. Figure 6.1 illustrates an example of random movement front-end inspired by bacteria foraging during an initial solution generating phase of the proposed algorithms referred to as modified ATS. Noticeably, large areas of search spaces are explored by the bacteria. The results obtained from the ATS, BF-TS and modified ATS are shown in Table 6.12 for comparison purposes. The numeric figures shown in the Table are averaged over 50 trial runs. The modified ATS and the BF-TS algorithms provide superior elite solutions for each problem compared to what obtained from the ATS. In terms of an average rank as a summary, the BF-TS, modified ATS and ATS are sequentially ranked as 1, 2 and 3, respectively.

Table 6.12 Comparison of generated initial solutions among the ATS, BF-TS and modified ATS on surface optimization problems (averaged over 50 trials).

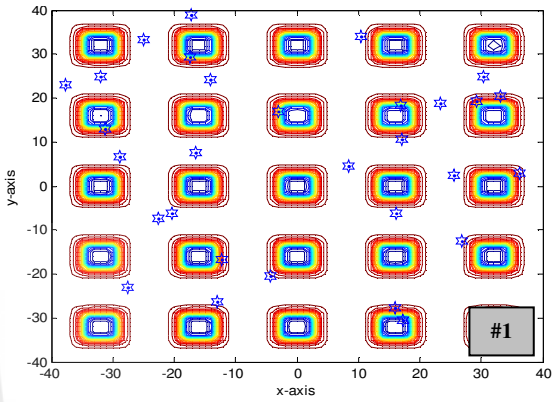
Test functions	Objective values	ATS	BF-TS	Modified ATS
BF	Average	0.5419	5.3586e-04	5.9541e-04
	Min	0.0162	1.8332e-05	9.7305e-07
	Max	1.8786	1.0936e-03	6.7372e-03
	Std.	0.3716	2.9912e-04	1.1546e-03
	Rank	3	1	2
RF	Average	5.1074	3.3731e-03	1.1870e-03
	Min	0.0334	2.3920e-04	1.6322e-05
	Max	16.6020	7.2237e-03	3.4842e-03
	Std.	3.7022	1.9420e-03	9.1732e-04
	Rank	3	2	1
SF	Average	14.0659	5.1032	1.7539
	Min	1.0018	1.0020	0.9982
	Max	143.5442	14.6019	4.5593
	Std.	19.6528	2.6921	0.8669
	Rank	3	2	1
SchF	Average	99.7347	15.3099	32.6897
	Min	0.0371	5.4606e-04	0.4058
	Max	355.4702	92.7943	161.2067
	Std.	95.5511	22.3200	38.6206
	Rank	3	1	2
ShuF	Average	-81.6787	-155.3778	-147.9609
	Min	-167.563	-186.6572	-186.1420
	Max	-25.7296	-109.8247	-48.6279
	Std.	36.2067	23.5441	41.7047
	Rank	3	1	2
Average rank		3	1.4	1.6
Final rank		3	1	2

Furthermore, the modified ATS proposed by this thesis also possesses neighbor-solutions generating sub-algorithm without search radius adjustment procedure. The sub-algorithm is designed based-on the adaptive random movement inspired from bacteria. As illustrated by Figures 6.2-6.4, physical movements of solutions can be

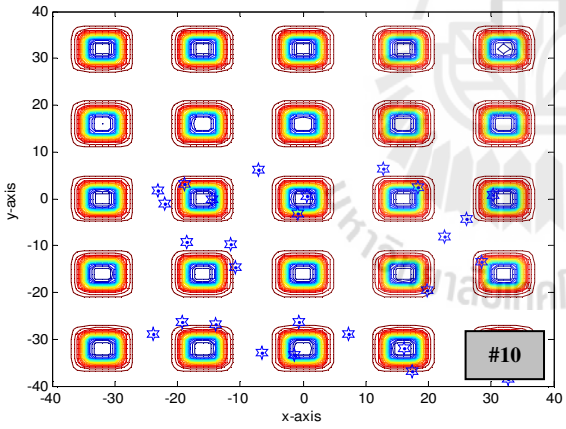
observed from the beginning of the search till obtaining global solutions for the original ATS and both of the proposed algorithms on the SF surface.



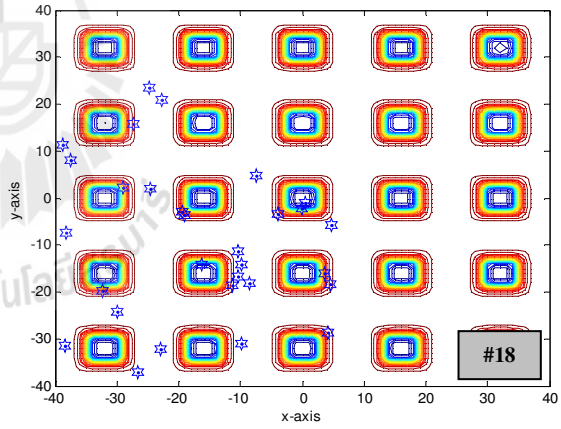
(a)



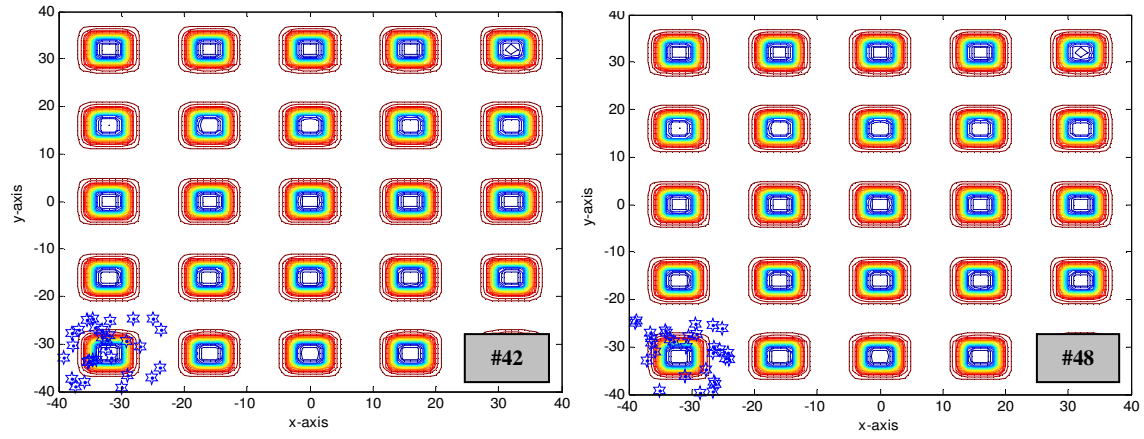
(b)



(c)

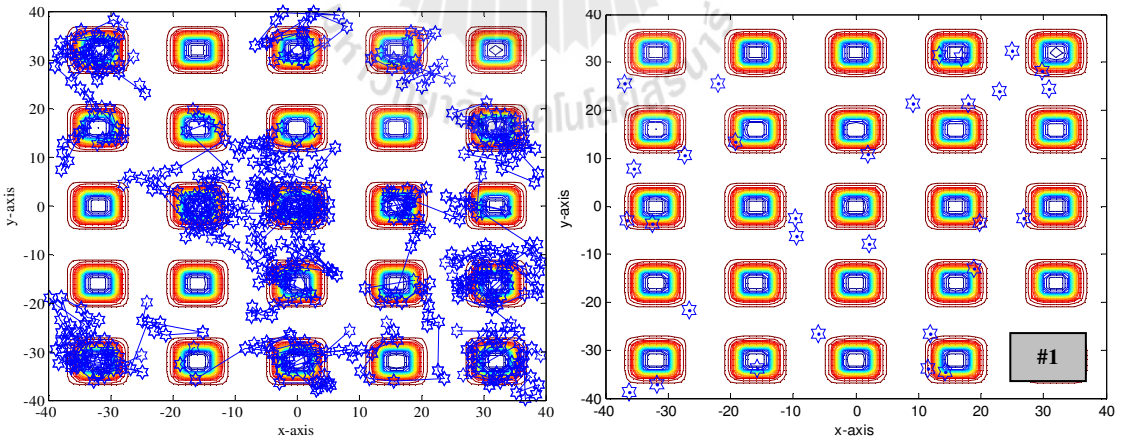


(d)

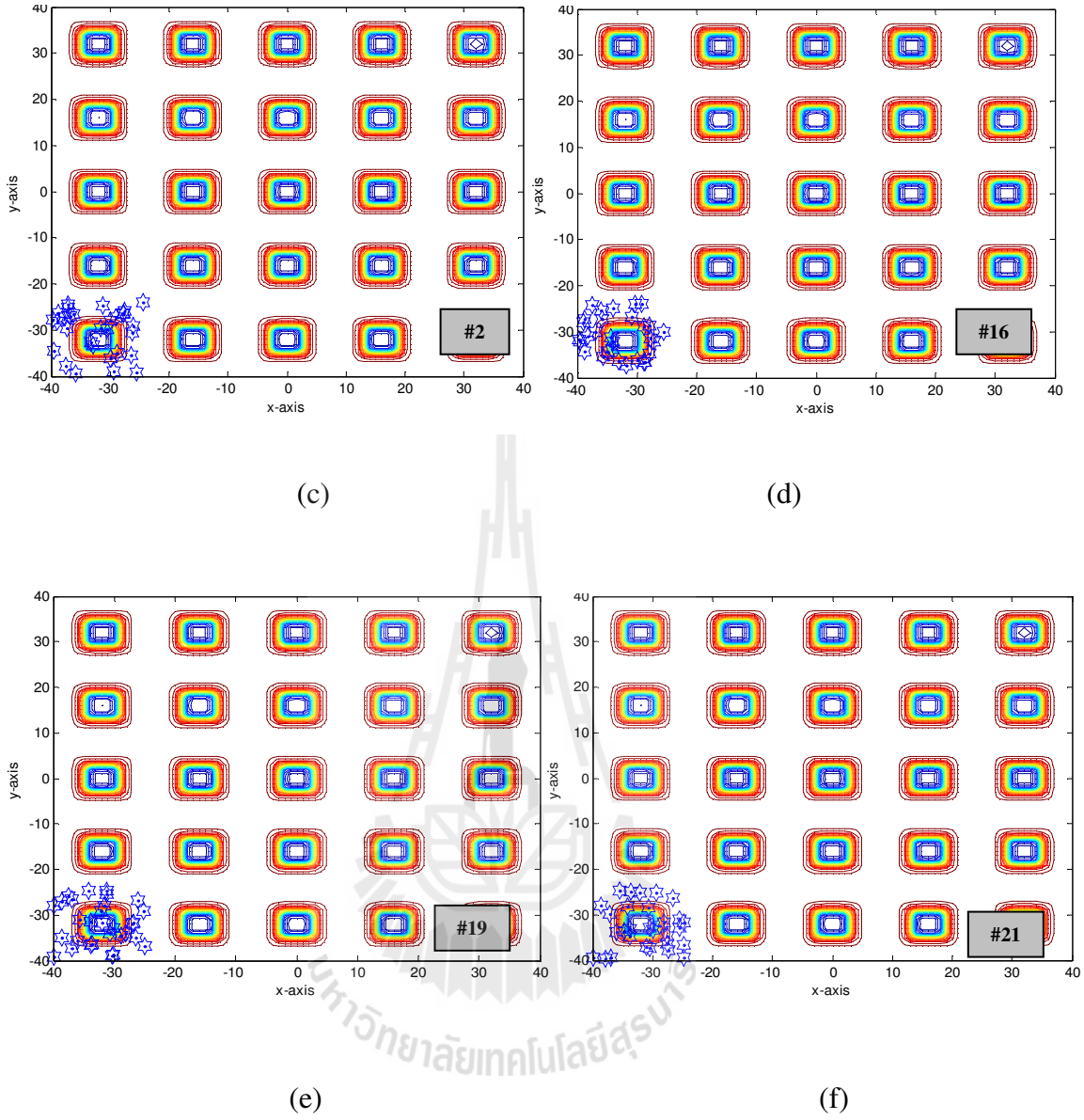


(e) (f)

Figure 6.2 Neighbour solution movements of ATS; (a) initial solutions randomly generated 600 solutions, (b) 1st iteration, (c) 10th iteration, (d) 18th iteration, (e) 42th iteration and (f) the final iterations (number of neighbour solution, N = 30).



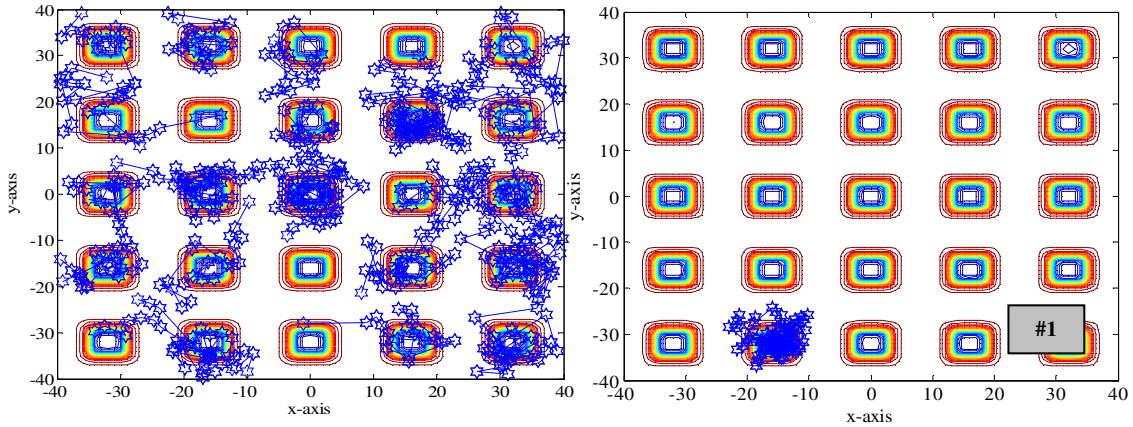
(a) (b)



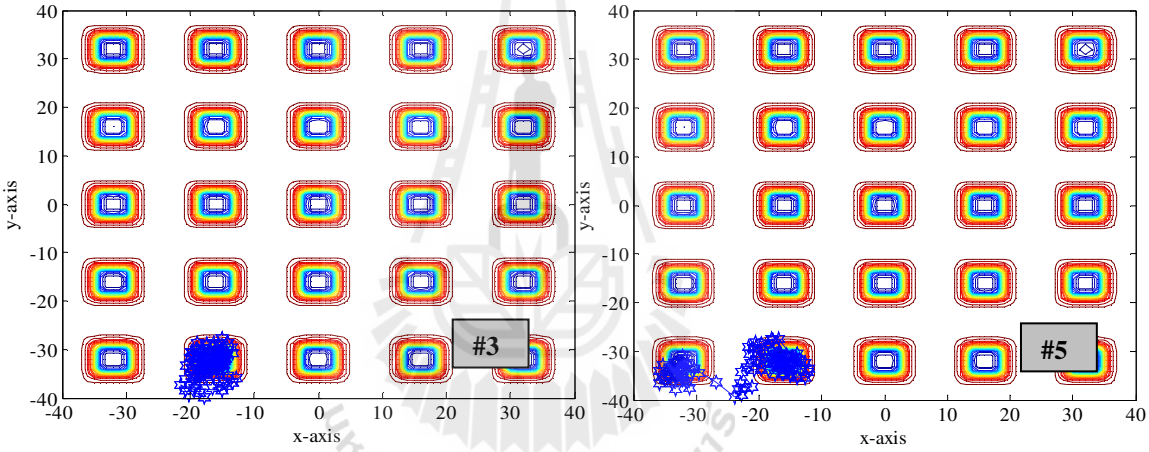
(c) (d)

(e) (f)

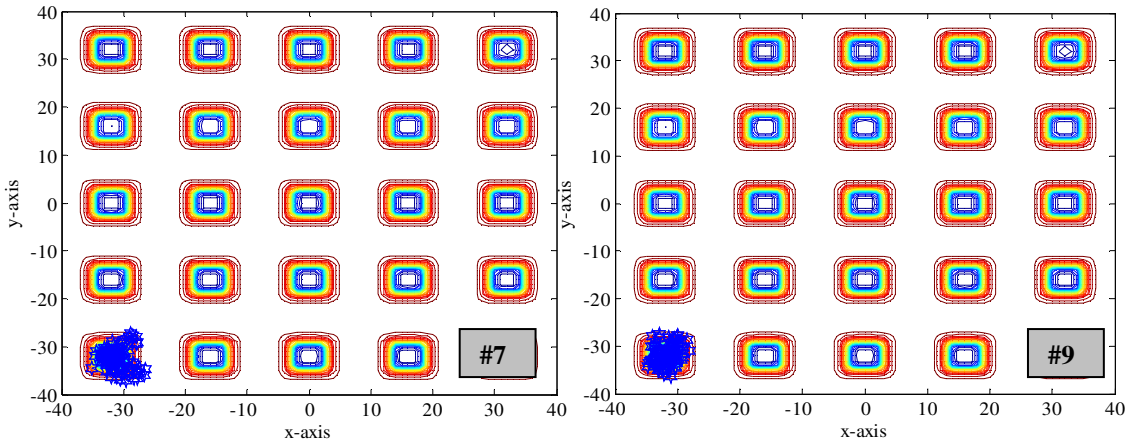
Figure 6.3 Neighbour solution movements of BF-TS; (a) initial solutions generated by the proposed random movement front-end, N_C = 20 and S = 30, (b) 1st iteration, (c) 2nd iteration, (d) 16th iteration, (e) 19th iteration and (f) the final iterations (number of neighbour solution, N = 30).



(a) (b)



(c) (d)



(e) (f)

Figure 6.4 Neighbour solution movements of modified ATS; (a) initial solutions generated by the proposed random movement front-end, N_C = 20 and S = 30, (b) 1st iteration, (c) 3rd iteration, (d) 5th iteration, (e) 7th iteration and (f) the final iteration (N_{C2} =100 and S_2 =2).

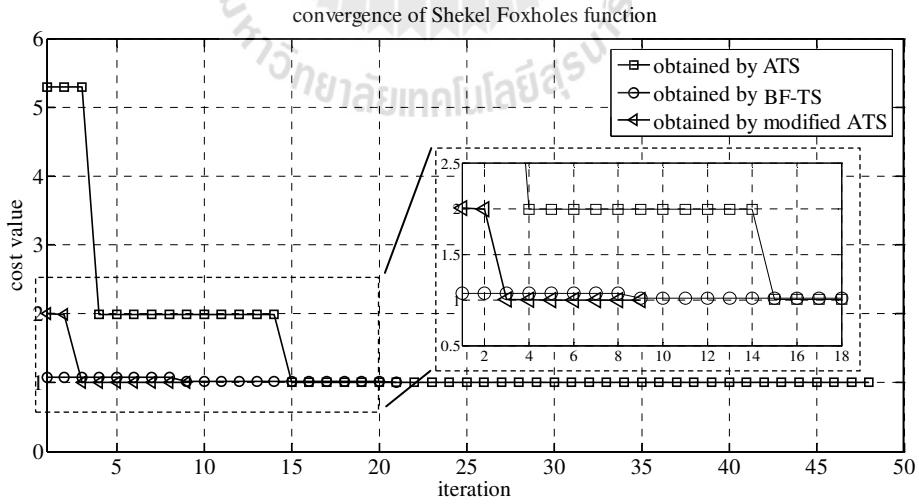


Figure 6.5 Convergence curves of the proposed algorithms.

Referring to Figure 6.2(a), the star symbols indicate the 600 locations of solutions randomly generated by the ATS. One can see that the locations spread over the entire search space without any concentration. On the contrary as being illustrated by the star locations in Figures 6.3(a) and 6.4(a), there are some certain areas having the star concentrations that are generated by the adaptive random walk front-ends of the BF-TS and the modified ATS algorithms, respectively. The results depict the capability of the front-ends to explore over the entire search space, and avoid oscillation around local areas. This leads to an increase in an opportunity to obtain at least a better elite solution. The mechanism occurs consequently until a satisfied high-quality solution is transferred to the main part of the algorithms. Afterward, searching for a global solution is performed. According to the mechanism to generate neighbour solutions of the ATS, Figures 6.2(b-f) and 6.3(b-f) illustrate the characteristics of generating neighbour solutions within a predefined search radius. Neighbourhoods are generated in a subsequent manner until the ATS tracks down an optimal solution. As explained previously, the BF-TS can obtain a better initial solution, thus it tends to converge to a global solution faster than the ATS does. This is evident by the convergent curves shown in Figure 6.5. The random walk front-end of the modified ATS generates neighbourhoods in a similar manner to that of the BF-TS. The algorithms operate without the *AR* mechanism, and reach to a searched terrain containing a global solution with a small numbers of step sizes, i.e. a small numbers of search rounds. Oscillations around the global terrain being searched rarely occur as the *BT* mechanism is still in operation together with the random walk front-end. The curves in Figure 6.5 represent numerically the convergence of each algorithm in terms of iterations. It is clearly seen that the modified ATS consumes minimum iterations.

Table 6.13 Occurrence of deadlocks in average (averaged over 50 trials).

Test functions	Average deadlocks		
	ATS	BF-TS	Modified ATS
BF	56.28	14.28	0
RF	343.18	32.20	0
SF	9.68	2.20	0.24
SchF	18.00	8.20	0
ShuF	7.60	4.44	0

Since the ATS, BF-TS and modified ATS have the same algorithmic approaches, comparisons of their local deadlocks are meaningful. Referring to Table 6.13 for the number of local entrapments, the BF-TS and modified ATS encounter local locks of 67.71% and 99.50% less than the ATS does, respectively. The modified ATS can avoid local dead locks of 97.82% than the BF-TS does as well. The modified ATS being proposed achieves this good performance because it can avoid local solution entrapments and oscillations around the targeted terrain containing a global solution. The mechanism supporting this is the newly developed random walk front-end as an enhancement to the original ATS. Table 6.14 summarizes the results of search time, search rounds and number of objective function evaluations averaged over 50 trial runs. In average, the BF-TS spends 33.67% shorter search time than the ATS does. On the other hand, the modified ATS spends 61.85% (except the SchF) longer search time than the ATS does. This is due to computing time demanded for objective function evaluation every moment the neighbourhood generating mechanisms being invoked. The modified ATS consumes the search times of 42.13% more than the BF-TS does. The modified ATS spends 68.55% shorter search time than the ABFO, 60.95% (except the RF) shorter than the IWO, and 50.48% (for the BF, SF and SchF problems) shorter than the GA do.

Table 6.14 Summary of the results of average search time, search rounds and number of objective function evaluations (averaged over 50 trials).

Test functions	Average search time (seconds)						Average search rounds						Number of objective function evaluations					
	ABFO	ATS	BF-TS	Modified ATS	IWO	GA	ABFO	ATS	BF-TS	Modified ATS	IWO	GA	ABFO	ATS	BF-TS	Modified ATS	IWO	GA
BF	78.16	11.66	6.83	27.22	37.20	49.12	12.52	616.48	151.20	23.96	950.42	1177.18	62600	19094.4	7036	21668	57035.2	36315.4
RF	118.63	14.60	5.81	79.64	35.98	11.78	9.82	868.28	323.30	66.42	949.84	225.54	147300	26648.4	17199	60636	57000.4	7766.2
SF	35.49	4.18	3.69	7.75	15.90	8.53	8.68	139.36	25.70	9.54	290.56	141.28	8600	4780.8	1271	8132	17443.6	5238.4
SchF	162.59	32.93	19.65	11.08	(17) 1219.92	488.27	28.44	186.64	87.08	30.22	(17) 6831.18	201.44	17064	6199.2	2912.4	12388	409880.8	7043.2
ShuF	32.87	3.28	2.80	8.75	26.20	3.39	2.28	68.06	55.28	39.46	740.22	34.24	34200	2641.8	9158.4	15392	44423.2	2027.2

Although, the modified ATS algorithm spends average search time more than other single-solution based metaheuristics do, these results have been considered that the proposed modified ATS consume shorter search time than the population-based metaheuristics do. Generally, the search mechanisms of the population-based metaheuristics invoke a group of generating cost functions as a population and they also disperse like a swarming. Moreover, those algorithms use the specific evaluation techniques to eliminate low quality of solutions. It can be observed that the search time could be spent longer as shown in the performance testing results.

For the average search rounds, the BF-TS and the modified ATS consume 58.38% and 80.91%, respectively less than the ATS does. It can be noticed that the proposed algorithms with a high quality solution provides a short search round than the original one. The modified ATS consumes the search round of 64.08% less than the BF-TS does. In addition, the modified ATS spends 96.29% fewer search rounds than the IWO, 86.69% (except the ShuF) fewer than the GA do, respectively, but 48.42% more than the ABFO does.

According to the number of objective function evaluations in Table 6.14, the results show that the ATS spends number of objective function evaluations of 56.26% (except ShuF) more than the BF-TS does, while the modified ATS consumes 48.39%, 68.10% and 63.19% (except BF) more than the ATS, BF-TS and GA do, respectively. In contrast, the modified ATS provides numbers of objective function evaluation of 42.41% and 54.27% fewer than ABFO and IWO do, respectively. This can be noticed that the BF-TS consumes the minimum number of objective function evaluation because its bacterial movement frontend selects high-quality initial solutions at the beginning of search. Regarding this, the ATS and the modified ATS are inferior. The

processes of the algorithms that generate population solutions demand high numbers of objective function evaluation, for examples, ABFO and IWO.

Table 6.15 Solutions obtained from different algorithms (averaged over 50 trials).

Test functions	Objective values	ATS	ABFO	BF-TS	Modified ATS	IWO	GA
BF	Average	4.5090e-10	5.3476e-10	5.3011e-10	6.4297e-10	4.4796e-10	3.6587e-10
	Min	7.3184e-11	2.2100e-11	3.5690e-12	6.3320e-12	1.0217e-11	1.1830e-11
	Max	9.4151e-10	9.5500e-10	9.9723e-10	9.4536e-10	9.8633e-10	9.7641e-10
	Std.	2.7476e-10	2.6887e-10	3.1352e-10	3.7297e-10	3.2129e-10	2.8916e-10
	Rank	3	5	4	6	2	1
RF	Average	5.3478e-09	4.9407e-09	5.2460e-09	8.0079e-10	4.0062e-09	4.9642e-09
	Min	2.9051e-10	2.7300e-10	5.9587e-10	5.1800e-12	9.2719e-11	1.1486e-10
	Max	9.3771e-09	9.9300e-09	9.6620e-09	9.9900e-10	9.7248e-09	9.5235e-09
	Std.	2.4328e-09	2.6000e-09	2.6721e-09	5.1800e-12	2.6838e-09	2.6740e-09
	Rank	6	3	5	1	2	4
SF	Average	0.9982	0.9981	0.9983	0.9982	0.9983	0.9983
	Min	0.9980	0.9980	0.9980	0.9980	0.9980	0.9980
	Max	0.9989	0.9989	0.9990	0.9987	0.9990	0.9990
	Std.	0.0002	0.0002	0.0003	0.0002	0.0003	0.0003
	Rank	2	1	3	2	3	3
SchF	Average	6.0600e-05	2.5506e-05	6.2286e-05	3.1106e-05	1.9266e+02	6.3063e-05
	Min	2.5700e-05	2.5500e-05	2.8430e-05	2.5700e-05	2.5900e-05	2.6886e-05
	Max	9.7791e-05	2.5700e-05	9.7833e-05	8.9600e-05	5.1325e+02	9.9456e-05
	Std.	2.3286e-05	3.1364e-08	2.0853e-05	1.0538e-05	1.7757e+02	2.3125e-05
	Rank	3	1	4	2	6	5
ShuF	Average	-186.7305	-186.7306	-186.7305	-186.7305	-186.7305	-186.7304
	Min	-186.7309	-186.7309	-186.7309	-186.7309	-186.7309	-186.7309
	Max	-186.7300	-186.7300	-186.7301	-186.7300	-186.7300	-186.7300
	Std.	0.0003	0.0003	0.0003	0.0003	0.0003	0.0002
	Rank	2	1	2	2	2	3
Average rank		3.2	2.2	3.6	2.6	3.0	3.2
Final rank		4	1	5	2	3	4

The results obtained so far are from running the algorithms at least 50 times such that our statistical analysis is meaningful. Table 6.15 summarizes ranking of the 6 algorithms based-on the averages of costs and variances. Referring to the numeric figures in the Tables 6.14 and 6.15, some conclusions can be drawn as follows: (i) the ABFO provides solutions of best quality, despite it needs a considerably longer search time, (ii) the modified ATS provides solutions with second to the best, (iii) the modified ATS consumes the minimum numbers of search rounds and spends shorter search time than the ABFO, IWO and GA. Referring again to the numeric figures in the Table 6.13, it is clear that the modified ATS is excel in deadlock avoidance compared with the other algorithms. Unfortunately, the IWO performs poorly, and often fails to reach an optimum solution within a given iterations. Therefore, the IWO is unsuitable for solving complex combinatorial optimization problems which usually contain many local solutions. Nonetheless, the IWO provides solutions of third to the best quality as average; the ATS comes in fourth. The GA is an alternative in terms of its capability of deadlock escapement although it provides solutions with fourth as well the ATS to the best quality. From these results, it can be seen that the proposed BF-TS and modified ATS algorithms achieve significantly better performances in terms of the averages of search time, search round, deadlock occurrence and solution quality than the other algorithms do. Note that all solutions found by the BT-TS and modified ATS algorithms meet the stop criterion of minimum cost. These outstanding performances are achieved due to the explorative characteristic of the random movement front-end, the exploitative characteristic of ATS, and the deadlock releasing property of the ATS. Moreover, the random movement with small step size is employed to generate the neighbour solutions for the modified ATS. It is also the significant mechanism to avoid

oscillations around the targeted search terrain, and to escape the local deadlocks in a short iteration.

For comparison purposes, the numbers of neighbour solutions are chosen from $S_2 \times N_{C2}$ to be employed by the original and the modified ATS, i.e. 200, 200, 200, 100 and 50 for the BF, RF, SF, SchF and ShuF problems, respectively. In addition, the results obtained from running the ATS when *N*=30 according to the recommendations by Puangdownreong (Puangdownreong et al., 2006) are also shown. Figures 6.6-6.9 summarize the results in terms of the averages 50 trials of deadlock, search time, search round and search time per iteration, respectively.

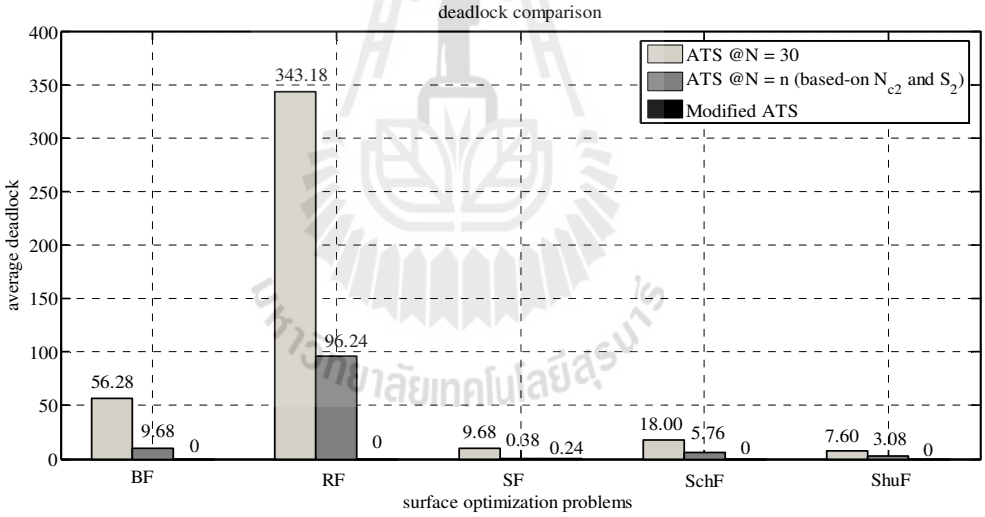


Figure 6.6 Comparisons of average deadlocks between the ATS and the modified ATS algorithms with the same neighbour solution numbers.

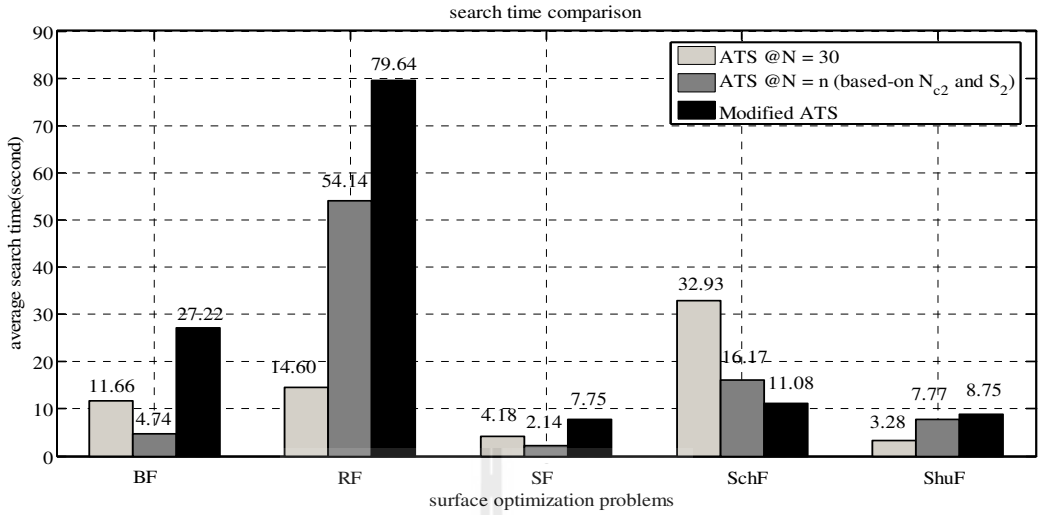


Figure 6.7 Average search time comparisons between the ATS and the modified ATS algorithms with the same neighbour solution numbers.

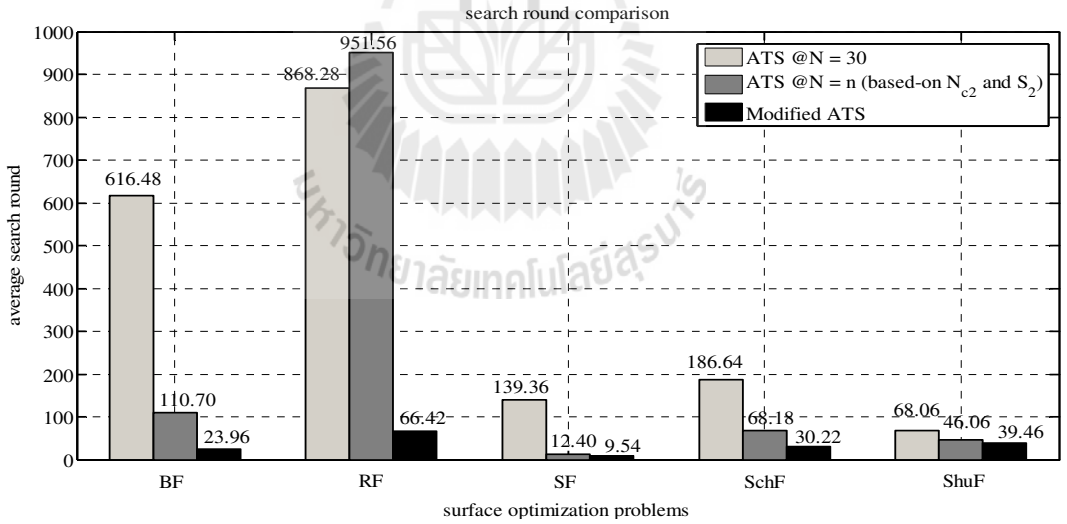


Figure 6.8 Average search round comparisons between the ATS and the modified ATS algorithms with the same neighbour solution numbers.

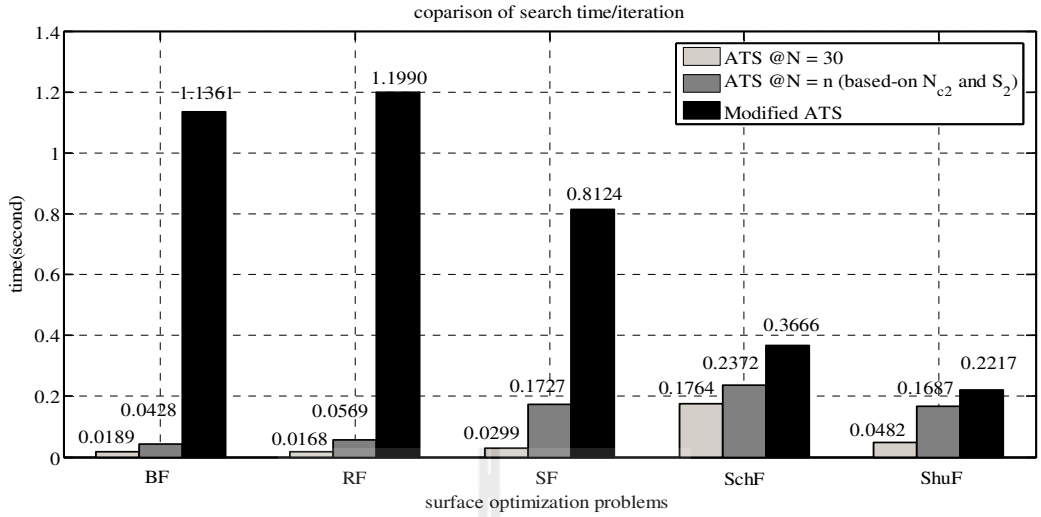


Figure 6.9 Comparisons of the search time per iteration of the ATS and the modified ATS algorithms with the same neighbour solution numbers.

Despite, the numbers of neighbour solutions of the ATS and the modified ATS are equal, Figure 6.6 indicates that the ATS (ATS@ N=n) encounters deadlocks of 87.37% more than the modified ATS does, while the ATS (ATS@ N=n) spends the average search time of 30.45% less than the modified ATS does (see Figure 6.7). Furthermore, the modified ATS consumes fewer search rounds of 52.89% compared with the ATS (ATS@ N=n) under the same numbers of neighbour solutions (see Figure 6.8). If we compare the results of the ATS with different numbers of neighbour solutions, it can be observed that the ATS with higher neighbour solution numbers (ATS@ N=n) provide higher performance to avoid local deadlocks, 75.67% more than the ATS@ N=30, and it can reach the optimum solution in shorter search time and fewer search rounds of 53.02% (except the RF and ShuF problems) and 67.23% (except RF), respectively. We may conclude from such results that if the neighbour solution

numbers of the ATS become increased, it leads to increase in an opportunity to find the optimum solution. Figure 6.9 presents the comparison results of the search time consumed per iteration. From this, it can be summarized that the modified ATS spends the longest search time compared with the ATS@ N=n and ATS@ N=30. The ATS@ N=n and ATS@ N=30 are the second and the third in rank of average search time consumed per iteration, respectively.

6.5 Conclusion

This Chapter has presented the performances of the adaptive bacterial foraging optimization (ABFO), the adaptive tabu search (ATS), the cooperative manner of bacterial foraging-tabu search approach (BF-TS), the modified ATS, the invasive weed optimization (IWO) and the genetic algorithms (GA). The performance assessment was carried out on some unconstrained optimization problems also referred to as benchmark functions. The tasks were conducted on a Pentium IV, 2.4 GHz, 640 Mbytes SD-RAM. For a fair comparison, each algorithm was subjected to 50 trial tests in order to find out its search parameters rendering best performance. Those parameters were applied for extensive computing tasks for collecting the results.

Comparison studies employed five well-known benchmark functions, i.e. Bohachevsky, Rastrigin, Shekel's fox-holes, Schwefel and Shubert functions, respectively. The results were averaged over fifty trial runs to assess the following issues: number of local locks, search time, number of search rounds, quality of initial solutions, quality of global solutions obtained, time consumed per search round and numbers of objective function evaluation. These issues reflect the performance of the search algorithms. From the results, it has been found that the modified ATS proposed

by this thesis has the best performance among the tested algorithms in terms of the quality of initial solutions, the quality of global solution obtained, the minimum number of local locks, and the minimum number of search rounds. It has been observed that good quality initial solutions provided by the random-walk frontend lead to high-quality global solutions obtained in a considerably low number of search rounds. The mechanisms imitating tumble and swim of bacteria effectively prevent oscillations in a searched region. Nonetheless, the proposed algorithms consumed longer search time of 2.53 and 4.70 times than the ATS and the BF-TS did, respectively, since the algorithms invoked the random-walk frontend many times during the search process. In other words, the random-walk frontend introduces an overhead cost in terms of search time due to objective function calculations. The drawback can be overcome by parallelization, sometimes referred to as parallel computing, of the algorithms as explained further in Chapter 8. Furthermore, the population-based metaheuristics under tests showed moderate search performances compared with the single-solution bases ones. Next chapter of the thesis demonstrates the usefulness of the modified ATS and the BF-TS algorithms for various complex engineering problems.

CHAPTER VII

APPLICATIONS OF THE PROPOSED ALGORITHMS TO ENGINEERING PROBLEMS

7.1 Introduction

The previous Chapter presented how preferable search parameters could be obtained as well as primary performance assessments. The assessments utilized some well-known unconstrained optimization problems. The performance indices include number of local locks, search time, number of search rounds, quality of initial solutions, quality of global solutions, and time consumed per search round, respectively. The modified ATS algorithm being proposed has shown superior performance among the others in terms of the quality of initial solutions, the quality of global solution, the minimum number of local locks, and the minimum number of search rounds.

As the next step, the proposed algorithms are utilized to solve various constrained real-world problems and some abstract mathematical ones as described in this Chapter. Such problems include optimal control designs of hard-disk heads and a second-order system with delay, stability analysis of a nonlinear system, identification problems of hard-disk head actuator and nonlinear Stribeck friction model, and a complex engineering problem, i.e. power drive system. Sections 7.2 to 7.8 give detailed

presentations of the cases, where the proposed algorithms are compared with the BF-TS. Section 7.9 presents the conclusion.

7.2 Abstract Mathematical Constraint Problems

The form of constrained optimization problem depends on the particular type of problem and function to be solved. The constraints can be combined in equation forms, such as equality ($f(x) = 0$), weak inequality ($g(x) \geq 0$), or strict inequality ($g(x) > 0$), etc. in which a set of given constraints must be satisfied. A constrained optimization problem can be defined as a regular constraint satisfaction problem. The aim of constrained optimization is to find a solution that can be evaluated as the sum of the cost functions $f_i(x)$. A general constrained optimization problem may be written as follows:

$$\begin{aligned} &\text{Min (or Max)} && f_i(x) && \text{for } i = 1,\ 2,...,\ I \\ &\text{Subject to} && h_j(x) = 0 && \text{for } j = 1,\ 2,...,\ J \\ & && g_k(x) \leq 0 && \text{for } k = 1,\ 2,...,\ K \end{aligned}$$

where $x = [x_1, x_2, ..., x_n] \in \mathbb{R}^n$ here the components of x are called design or decision variables. $f_i(x)$ is the objective function or simply cost function, and in the case of $I = 1$, only single objective is considered. $h_j(x)$ is an equality constraint, and $g_k(x)$ is an inequality constraint. On another hand, the inequality can be written in form of "≥ 0", and it can be formulated as a maximization problem.

Three abstract mathematical constraint problems are given in this thesis as follows (Oftadeh et al., 2009):

1. Constrained function 1: Fcon1

The optimum solution is at x^* = (0.82288, 0.91144) with an objective function value equal to $f(x^*)$=1.3933.

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

Subject to $g_1(x) = x_1 - 2x_2 + 1 = 0$

$g_2(x) = -x_1^2/4 - x_2^2 + 1 \geq 0$

where $-10 \leq x_1 \leq 10$ and $-10 \leq x_2 \leq 10$.

2. Constrained function 2: Fcon2

The optimum solution is at x^* = (2.330875, 1.951370, -0.474593, 4.365553, -0.624525, 1.037936, 1.594065) with the objective function value equal to $f(x^*)$=680.6300771.

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

Subject to $g_1(x) = 127 - 2x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 \geq 0$

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

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

$g_4(x) = -4x_1^2 - x_2^2 + 3x_1x_2 - 2x_3^2 - 5x_6 + 11x_7 \geq 0$

where $-10 \leq x_i \leq 10 \quad (i = 1, 2, ..., 7)$.

3. Constrained function 3: Fcon3

The optimal solution is at x^* = (522.806838, 1380.644472, 5147.870997, 177.101116, 294.085207, 222.898280, 283.015889, 394.085207) with corresponding objective function value equal to $f(x^*)$ = 7051.322306.

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

Subject to $g_1(x) = 1 - 0.0025(x_4 + x_6) \geq 0$

$g_2(x) = 1 - 0.0025(x_5 + x_7 - x_4) \geq 0$

$g_3(x) = 1 - 0.01(x_8 - x_5) \geq 0$

$g_4(x) = x_1 x_6 - 833.332522 x_4 - 100 x_1 + 83333.333 \geq 0$

$g_5(x) = x_2 x_7 - 1250 x_5 - x_2 x_4 + 1250 x_4 \geq 0$

$g_6(x) = x_3 x_8 - x_3 x_5 + 2500 x_5 - 1250000 \geq 0$

where $100 \leq x_1 \leq 10000$, $x_2 \geq 1000$, $x_3 \geq 10000$, $10 \leq x_i \leq 1000$ $(i = 4,5,...,8)$.

The search parameters of the BF-TS and the modified ATS have been used following Table 7.1 and Table 7.2.

Table 7.1 Search parameters of the BF-TS for constrained optimization problems.

Constrained problems	S	N_C	N_S	α	R	N	J_{min} <	BT, n_re_back	$Count_{max}$
Fcon1	30	100	4	1×10^3	1.5	1×10^5	1.3935	5	50000
Fcon2	30	100	4	1×10^4	1.5	1×10^5	680.6400	5	50000
Fcon3	30	100	4	1×10^4	0.1	1×10^5	7051.3223	5	50000

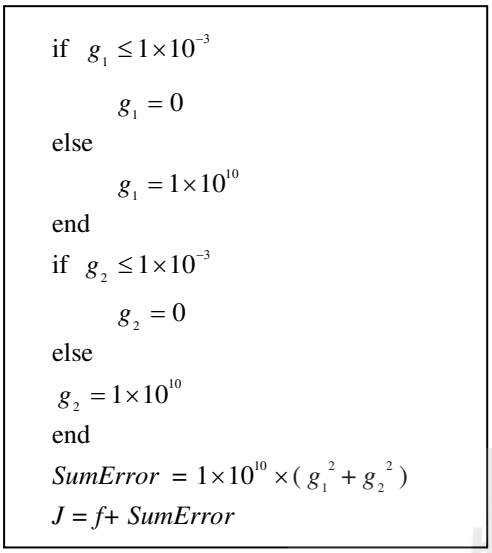
The adaptive search radius mechanism of the BF-TS has been used for each problem as follows:

- Fcon1: if $J < 20$ then R=0.3, if $J < 5$ then R=0.06, if $J < 1.5$ then R=0.012, and if $J < 1.396$ then $R = 0.0024$.
- Fcon2: if $J < 900$ then R=0.0075, if $J < 750$ then R=0.0015, and if $J < 682$ then R=0.000375.
- Fcon3: if $J < 7500$ then R=0.01, if $J < 7100$ then R=0.005, if $J < 7060$ then R=0.0001, and if $J < 7055$ then R=0.000005.

Table 7.2 Search parameters of the modified ATS for constrained optimization problems.

Constrained problems	S	N_C	S_2	N_{C2}	N_S	α	α_2	BT, n_re_back	$Count_{max}$
Fcon1	30	100	100	1000	4	1×10^3	1×10^4	5	50000
Fcon2	30	100	100	1000	4	1×10^4	1×10^6	5	50000
Fcon3	30	100	100	1000	4	1×10^4	1×10^6	5	50000

The program lists of objective function, J_{min}, for each constrained optimization problem have been used following Table 7.1. The objective functions and the constraint conditions have been used as Figure 7.1.



if $g_1 \le 1\times10^{-3}$

$g_1 = 0$

else

$g_1 = 1\times10^{10}$

end

if $g_2 \le 1\times10^{-3}$

$g_2 = 0$

else

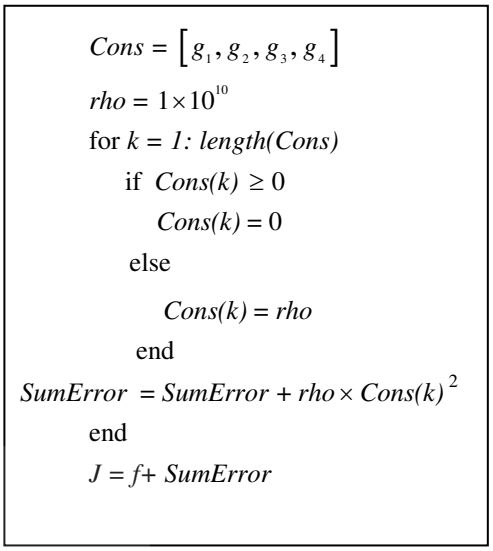
$g_2 = 1\times10^{10}$

end

$SumError = 1\times10^{10}\times(g_1^{\,2} + g_2^{\,2})$

$J = f + SumError$

(a)



$Cons = [g_1, g_2, g_3, g_4]$

$rho = 1\times10^{10}$

for $k = 1: length(Cons)$

if $Cons(k) \ge 0$

$Cons(k) = 0$

else

$Cons(k) = rho$

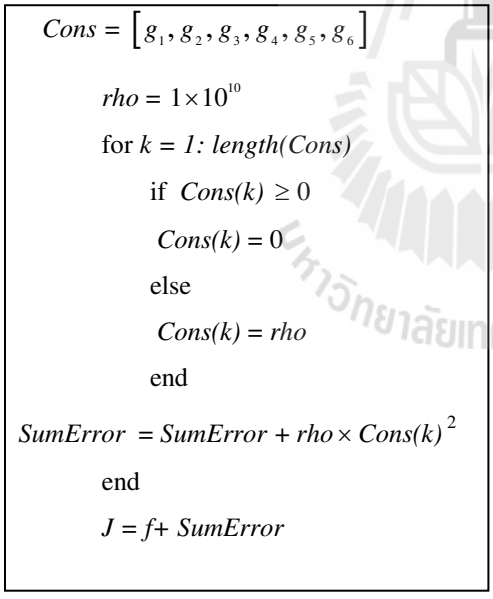
end

$SumError = SumError + rho \times Cons(k)^2$

end

$J = f + SumError$

(b)



$Cons = [g_1, g_2, g_3, g_4, g_5, g_6]$

$rho = 1\times10^{10}$

for $k = 1: length(Cons)$

if $Cons(k) \ge 0$

$Cons(k) = 0$

else

$Cons(k) = rho$

end

$SumError = SumError + rho \times Cons(k)^2$

end

$J = f + SumError$

(c)

Figure 7.1 Objective functions: (a) Fcon1, (b) Fcon2 and (c) Fcon3.

Table 7.3 summarizes the solutions obtained as average over 50 trial runs of the BF-TS and the modified ATS, respectively. Considering the results of Fcon1, the average cost functions obtained from the BF-TS and the modified ATS are 1.3929 and 1.3933, respectively. While both algorithms effectively solved Fcon1, the modified ATS is superior in terms of achieving low values of cost functions, i.e. g_1 and g_2, whose average values are 9.48×10^{-5} and 3.92×10^{-6}, respectively. Similarly, Fcon2 can be successfully solved by the BF-TS and modified ATS. The average results of cost values are reported as 680.6379 and 680.6300 for the BF-TS and the modified ATS, respectively. Average results show that the modified ATS has given better qualities of cost value and conditions, which are 680.6300, 1.76×10^{-4}, 252.59, 144.87 and 4.29×10^{-5}, respectively. For Fcon3, although the maximum count of the BF-TS is set to 50,000, the searches fail to obtain an optimal solution. Among those search trials, only 12 of them managed to reach an optimum solution. Further results show that the modified ATS provides a better cost function of 7051.3175 with reasonable average constraints compared to the BF-TS.

Table 7.3 Summary of the results obtained from the BF-TS and the modified ATS for constrained optimization problems (averaged over 50 trials).

Constrained problems	Objective values	BF-TS							Modified ATS						
		g_1	g_2	g_3	g_4	g_5	g_6	$f(x)$	g_1	g_2	g_3	g_4	g_5	g_6	$f(x)$
Fcon1	Average	6.45e-4	2.66e-4	-	-	-	-	1.3929	9.48e-5	3.92e-6	-	-	-	-	1.3933
	Min	7.10e-6	7.68e-7	-	-	-	-	1.3922	8.41e-5	2.28e-7	-	-	-	-	1.3933
	Max	9.94e-4	8.35e-4	-	-	-	-	1.3935	9.98e-5	2.09e-5	-	-	-	-	1.3933
	Std.	2.56e-4	2.02e-4	-	-	-	-	3.36e-4	4.35e-6	3.34e-6	-	-	-	-	1.01e-5
	Rank	2	2	-	-	-	-	2	1	1	-	-	-	-	1
Fcon2	Average	1.27e-3	252.72	144.81	3.10e-3	-	-	680.6379	1.76e-4	252.59	144.87	4.29e-5	-	-	680.6300
	Min	3.66e-6	252.16	144.64	2.53e-5	-	-	680.6338	1.36e-4	252.47	144.83	2.94e-8	-	-	680.6299
	Max	4.47e-3	253.02	145.07	9.08e-3	-	-	680.6390	1.88e-4	252.65	144.92	2.22e-4	-	-	680.6301
	Std.	1.17e-3	0.26	0.11	2.40e-3	-	-	9.78e-4	1.30e-5	0.06	0.03	4.69e-5	-	-	2.98e-5
	Rank	2	2	1	2	-	-	2	1	1	2	1	-	-	1
Fcon3	Average	3.77e-5	5.47e-6	3.72e-6	2.60	2.18	2.48	7054.5866	3.44e-4	1.15e-5	1.28e-5	12.46	10.11	2.95	7051.3175
	Min	1.01e-7	2.31e-8	5.20e-8	6.59e-3	6.59e-3	2.09e-2	7051.2954	1.30e-7	3.78e-10	1.94e-10	1.39e-4	4.01e-4	3.25e-4	7051.2350
	Max	3.54e-4	3.19e-5	1.43e-5	20.39	22.10	24.76	7064.8423	9.23e-4	1.12e-4	9.85e-5	168.21	81.90	53.28	7051.3222
	Std.	7.02e-5	7.27e-6	4.49e-6	4.07	3.44	4.27	3.4373	2.71e-4	2.13e-5	2.47e-5	34.44	22.01	8.32	0.0151
	Rank	1	1	1	1	1	1	2	2	2	2	2	2	2	1
Average rank		1.67	1.67	1.00	1.50	1.00	1.00	2	1.33	1.33	2.00	1.50	2.00	2.00	1
Final rank		2	2	1	1	1	1	2	1	1	2	1	2	2	1

Table 7.4 Summary of the average search time, search round and local deadlocks of the BF-TS and modified ATS approaches for constrained optimization problems (averaged over 50 trials).

Constrained problems	Average search time (seconds)		Average search rounds		Average deadlocks	
	BF-TS	Modified ATS	BF-TS	Modified ATS	BF-TS	Modified ATS
Fcon1	11.5655	214.8662	72.28	108.76	5.70	0
Fcon2	6.9359	283.2332	344.32	59.68	25.38	0
Fcon3	(12) 1394.2844	234.3396	(12) 40866.06	359.06	(12) 2948.30	0

Referring to Table 7.4, the average deadlocks of the modified ATS have not been found. These results ensure that the modified ATS provides the superior performance to escape deadlocks. For the average search times and search rounds, the results show that the modified ATS consumes the average search time of 96.08% (except Fcon3) more than the BF-TS does, while it spends the average search rounds less than the BF-TS by 90.89% (except Fcon1). If we consider in terms of average search time consumed per iteration for each constrained function, the results show that the modified ATS spends longer search time than the BF-TS as shown in Figure 7.2.

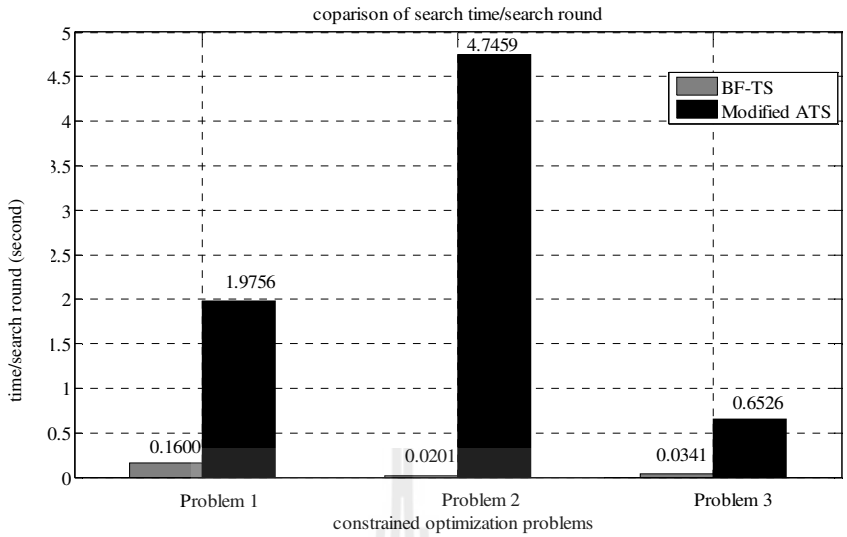
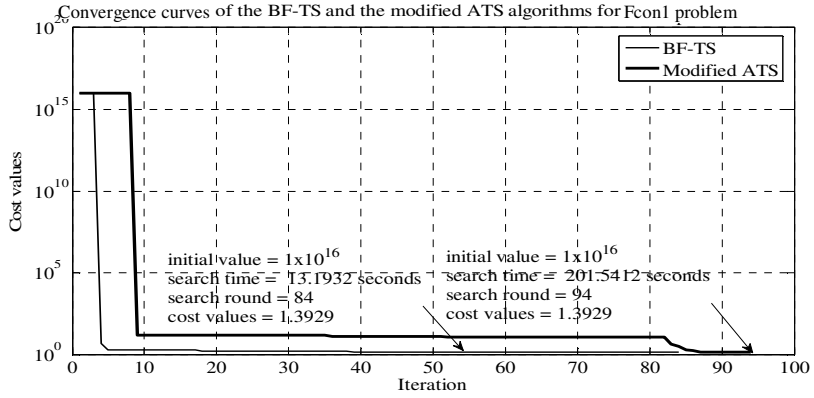
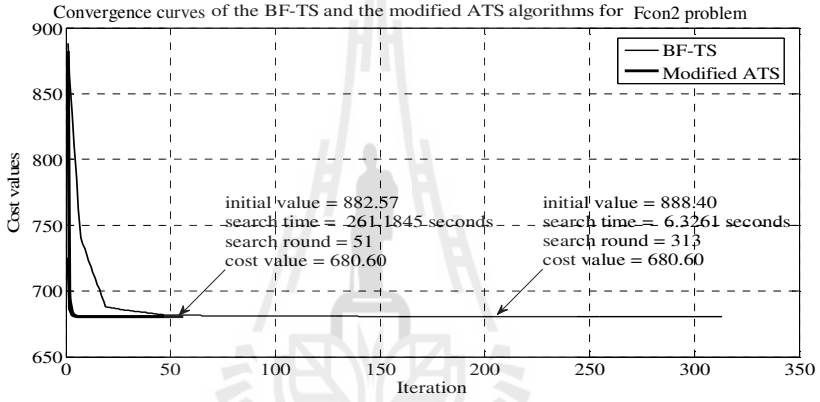


Figure 7.2 Comparisons of the search time per iteration of the BF-TS and modified ATS algorithms for abstract mathematical constraint problems.

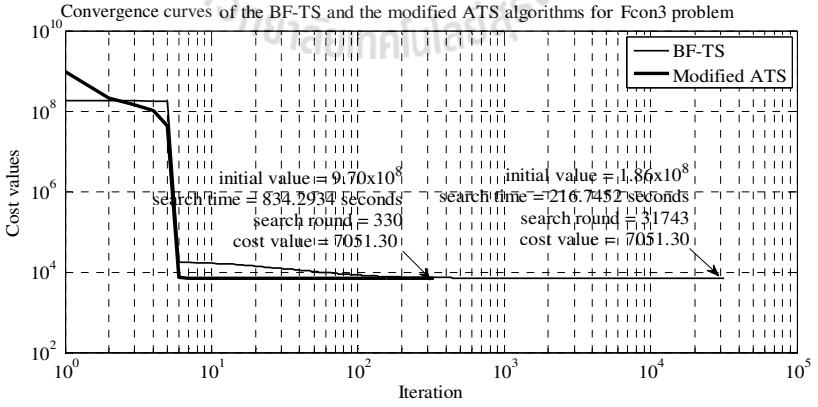
Figure 7.3 illustrates some convergent curves of these constraint problems as examples. Notice that, for an uncomplicated problem like Fcon1, the BF-TS reached to the global solution faster than the modified ATS. For Fcon2 that contains many constraint conditions, the modified ATS successfully solved it in a fewer numbers of iteration with significantly short time. Figure 7.3(b) clearly shows this. For more complex problem like Fcon3, although the initial solutions obtained from the BF-TS in some trial runs are better than those given by the modified ATS, it converges to the solutions much slower than the modified ATS. Moreover, it fails in some trial runs. This is evident by the convergent curves shown in Figure 7.3(c).



(a)



(b)



(c)

Figure 7.3 Comparison of convergences between the BF-TS and the modified ATS algorithms: (a) Fcon1, (b) Fcon2 and (c) Fcon3.

7.3 Hard Disk Drive Control Design Applications

Read/write (R/W) technology of hard disk drives (HDDs) employs both single-head and head-stack types, respectively. Both types of the heads exhibit highly underdamped dynamic responses that severely need proper compensations. In recent years, there have been many attempts to compensate for undesirable and unstable responses of these heads. Robust control approach has been proposed (Goh et al., 2001) to handle this problem. Nonlinearity, friction, and resonance have been compensated for by the composite nonlinear feedback control (Chen et al., 2003; Peng et al., 2005), linear compensation and fuzzy control (Ngernbaht et al., 2009). Nonlinear PID, and adaptive robust control approaches for the problems can be found in (Isayed and Hawwa, 2007), and (Taghirad and Jamei, 2008), respectively. The review is not exhaustive, but serves to show some good examples in the field.

This section intends to show that using linear control technique is possible to solve the dynamic compensation problems of the HDD-heads. Since the dynamic of the head is quite complicated, manual design is usually not possible. Computational efforts using optimization algorithms are particularly useful to achieve the design criteria. The following materials are presented the comparison studies of various control system structures, conventional control designs, single R/W head and R/W head-stacks dynamic compensations, and resonance countermeasure via notch filtering.

The design phase is aimed for hard disk drive to produce high servo performance due to unit-step input. In practice of the hard-disk industry, the design criteria are given by GM $\geq$ 5 dB and PM $\geq$ 25 deg. The response overshoot can be allowed up to 25%. However, engineers tend to design toward the time response having as small overshoot and settling time as possible. The compensators used by this thesis

are the third order with real poles and zeros. Figure 7.4 shows three block diagrams representing different control structures under consideration (Sarasiri et al., 2010).

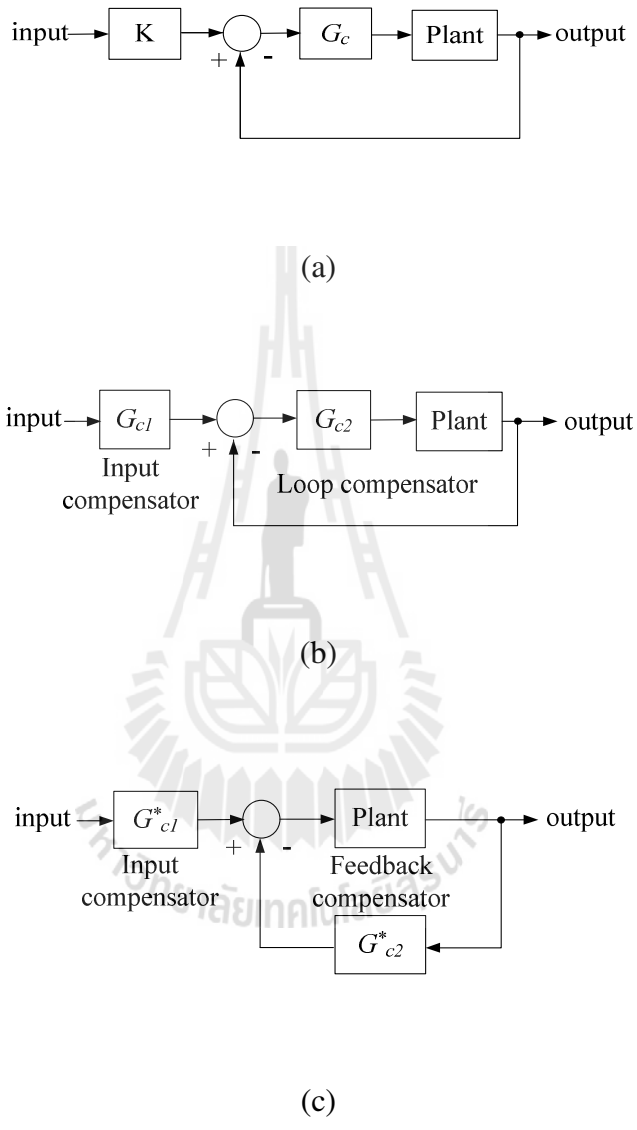


Figure 7.4 Control structures: (a) 1-DOF, (b) 2-DOF type 1and (c) 2-DOF type 2.

According to the conventional feedback control system, the control signal is generated by processing the difference between the reference inputs and the actual outputs. It is well known that in such a case can be dealt with a controller which has

one-degree-of-freedom (1-DOF) as the standard structure in Figure 7.4(a). This structure is also called a series compensator. Its performance is not particularly outstanding. On the other hand, two-degrees of freedom (2-DOF) type 1 and type 2 compensators (see Figure 7.4(b)-(c)) allows one to process the reference and measurements independently. The control system structure as shown in Figure 7.4(c) has some well-known advantages. The compensator G_{c1}^{*} dominantly shapes the output to follow the input (tracking mode). The compensator G_{c2}^{*} plays an important role in regulation and stabilization modes. In addition, the feedback property of this 2-DOF type 2 provides reasonable pole placements and good system stabilisation (Sujitjorn, 2003; Vilanova and Serra, 1997).

7.3.1 Single R/W Head

The main components of the single-head of a hard drive are a voice-coil-motor (VCM), a pivot, an actuator arm, and suspension. The small tip of the suspension holds the actual R/W head. The structure is presented in Figure 7.5 (Peng et al., 2005).

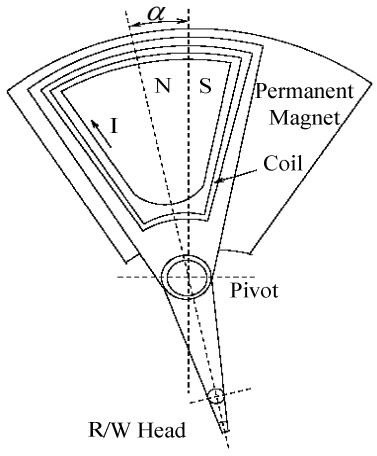
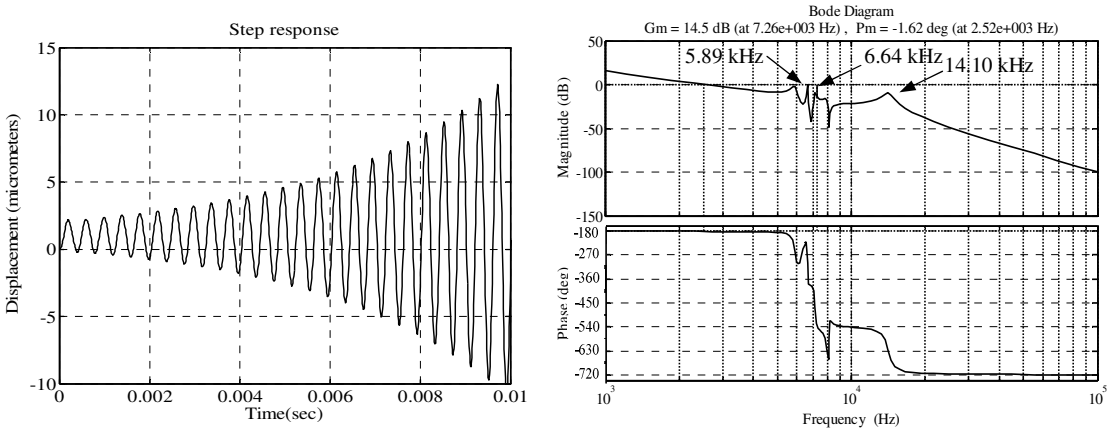


Figure 7.5 Structure of single R/W head.



(a) (b)

Figure 7.6 Open-loop responses: (a) time-domain and (b) frequency-domain.

Figure 7.6 illustrates the unstable open-loop responses of the single-head. It is reported that the dynamic of the single-head contains 5 resonance modes (Chen et al., 2003; Chen et al., 2006) which can be described by the following transfer function

$$G_{s.h}(s) = \frac{2.35\times10^{8}}{s^{2}} G_{s.h.1}(s)G_{s.h.2}(s)G_{s.h.3}(s)G_{s.h.4}(s)G_{s.h.5}(s) \tag{7.1}$$

where

$$G_{s.h.1}(s) = \frac{0.8709s^{2} + 1726s + 1.369\times10^{9}}{s^{2} + 1480s + 1.369\times10^{9}} \tag{7.2}$$

$$G_{s.h.2}(s) = \frac{0.9332s^{2} - 805.8s + 1.739\times10^{9}}{s^{2} + 125.1s + 1.739\times10^{9}} \tag{7.3}$$

$$G_{s.h.3}(s) = \frac{1.072s^2 + 925.1s + 1.997\times10^9}{s^2 + 536.2s + 1.997\times10^9} \tag{7.4}$$

$$G_{s.h.4}(s) = \frac{0.9594s^2 + 98.22s + 2.514\times10^9}{s^2 + 1805s + 2.514\times10^9} \tag{7.5}$$

$$G_{s.h.5}(s) = \frac{7.877\times10^9}{s^2 + 6212s + 7.877\times10^9} \tag{7.6}$$

The anti-resonance filters or notch filters have been used for this single R/W head to suppress the severe resonant frequencies occurred and increase the robusted stability. The anti-resonance filters have been detailed in Appendix A.

Conventional control designs are firstly considered for the single-head HDD, which has a phase margin of -78.3 deg. Three notch filters are used to compensate for resonance modes, and one phase-lead compensator is to provide a desirable phase characteristic. Appendix A summarizes the design details. The conventional transfer function is prescribed in equation (7.7).

$$G_{PL1} = 1.80\times10^3\left(\frac{s + 6.67\times10^3}{s + 4.23\times10^4}\right)\left(\frac{s + 2.377\times10^4}{s + 3.311\times10^5}\right)\left(\frac{s + 8.557\times10^3}{s + 1.118\times10^6}\right) \tag{7.7}$$

After applying anti-resonance filters and a third-order phase-lead compensators to the single R/W head system, the results of time and frequency domains can be satisfied with gain and phase margins of 26.10 dB and 48.70 deg, (see Figure A.5(b)), respectively. However, its time responses after compensation still has a high overshoot of 32% (see Figure A.5(a)).

Further designs are attempted for the control structures as shown in Figures 7.4 (a)-7.4(c) using the BF-TS and the modified ATS algorithms. Each compensator is the third-order containing 3 real poles, 3 real zeros, and a gain. The search problem is to find the optimal values of 7 compensator parameters for the 1-DOF structure, and 14 compensator parameters for the 2-DOF structures of types 1 and 2. The present problem becomes an optimal synthesis of compensators of the form shown in equation (7.8).

$$G_C(s) = K_1 \frac{(s+z_1)(s+z_2)(s+z_3)}{(s+p_1)(s+p_2)(s+p_3)} \tag{7.8}$$

The objective is to minimize the sum of absolute errors between the input shape and the actual response in time-domain. Simultaneously, the closed-loop control must meet the requirements of stability margins. Due to the high order of the plant models and the filters, design the compensators manually is not possible. Although with an aid of a computer, some trials-and-errors are necessary, and the design process would be very tedious and time consuming. Thus, the design process is formulated as a search and optimization problem of multi parameter control synthesis. The optimization problem can be expressed as a constrained optimization problem as follows

$$\begin{aligned} &\text{minimize } J: && J = \Sigma |e(t)| \\ &\text{subject to} && P.O < 5\% \\ & && GM > 5dB \\ & && PM > 25deg \end{aligned}$$

where $e(t)=r(t)-c(t)$, $r(t)$ is the unit-step input, $c(t)$ is the response and $e(t)$ is the response error.

The search parameters for the BF-TS and the modified ATS applied to the single R/W head are shown in Tables 7.5 and 7.6, respectively.

Table 7.5 Boundaries and search parameters of the BF-TS for the single R/W head.

Compensators	Search parameters	S	N_C	N_S	α	R	N	$J_{min} \leq$	BT, n_re_back
1-DOF	$z_1, z_2\ z_3, p_1, p_2, p_3, K_1 = [100\text{-}10^{11}]$	30	100	4	100	20	50	50	5
2-DOF type 1	$z_1, z_2\ z_3, z_4, z_5, z_6, p_1, p_2, p_3, p_4, p_5, p_6, K_1, K_2 = [100\text{-}10^{11}]$								
2-DOF type 2									

The adaptive radiuses are utilized, if $J < 70$ then $R = 10$, if $J < 65$ then $R = 7$, if $J < 55$ then $R = 4$.

Table 7.6 Boundaries and search parameters of the modified ATS for the single R/W head.

Compensators	Search parameters	S	N_C	S_2	N_{C2}	N_S	α	α_2	$J_{min} \leq$
1-DOF	$z_1, z_2\ z_3, p_1, p_2, p_3, K_1 = [100\text{-}10^{11}]$	30	100	1	50	4	100	1×10^3	50
2-DOF type I	$z_1, z_2\ z_3, z_4, z_5, z_6, p_1, p_2, p_3, p_4, p_5, p_6, K_1, K_2 = [100\text{-}10^{11}]$								
2-DOF type 2									

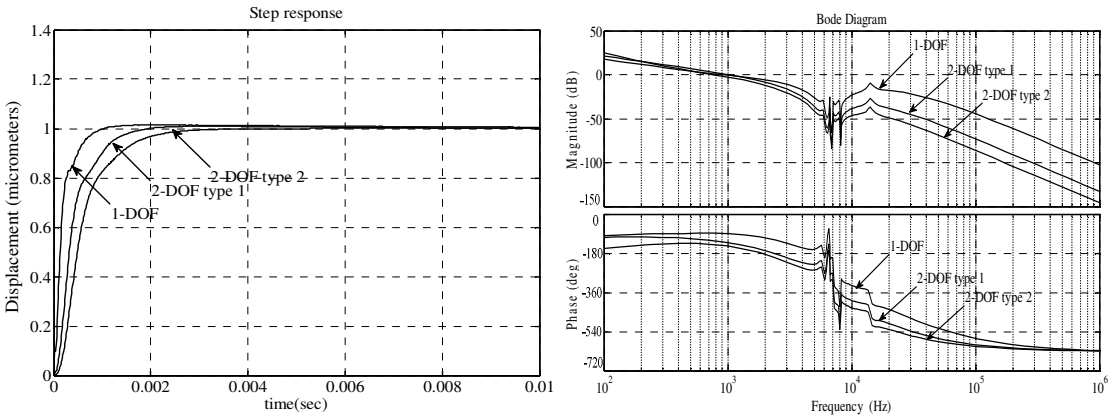
The maximum count ($count_{max}$) is 5,000 for these approaches. The objective function, J list of the single R/W head has been presented in Figure 7.7.

```
SAE = sum(abs(1-Close_system))
if overshoot < 5
        overshoot = 0
else
        overshoot= 1×10^3
end
if Gm > 5
        Gm = 0
else
        Gm =1×10^3
end
if Pm > 25
        Pm = 0
else
        Pm = 1×10^3
end
J = SAE + (overshoot^2 + Gm^2 + Pm^2 )× 1×10^3
```

Figure 7.7 Objective function of the single R/W head.

Table 7.7 Summary of the compensators and the corresponding responses of the single-head using the BF-TS algorithm.

Compensators	Mp (%)	t_r (ms)	t_s (ms) $e_{ss}=5\%$	GM (dB)	PM (deg.)	SAE
1-DOF; $Gc(s)=4.117\times10^{5}\frac{(s+6.868\times10^{3})(s+1.451\times10^{8})(s+100.80)}{(s+1.005\times10^{9})(s+4.0\times10^{5})(s+5.051\times10^{7})}$	1.83	0.48	0.79	29.30	91.10	31.37
2-DOF type 1; $Gc1(s)=7.812\times10^{3}\frac{(s+6.001\times10^{4})(s+4.014\times10^{9})(s+9.103\times10^{3})}{(s+1.265\times10^{9})(s+2.201\times10^{3})(s+6.210\times10^{9})}$ $Gc2(s)=8.110\times10^{3}\frac{(s+8.132\times10^{4})(s+8.010\times10^{9})(s+1.503\times10^{2})}{(s+2.985\times10^{8})(s+9.895\times10^{4})(s+5.911\times10^{9})}$	1.35	0.85	1.47	11.30	48.20	47.26
2-DOF type 2; $G^{*}c1(s)=7.822\times10^{3}\frac{(s+6.961\times10^{4})(s+6.452\times10^{9})(s+6.758\times10^{3})}{(s+4.584\times10^{9})(s+2.683\times10^{4})(s+7.291\times10^{9})}$ $G^{*}c2(s)=8.904\times10^{3}\frac{(s+9.968\times10^{4})(s+1.082\times10^{10})(s+1.260\times10^{3})}{(s+2.726\times10^{9})(s+3.852\times10^{4})(s+4.337\times10^{9})}$	0.32	1.17	3.13	10.80	39.90	49.29



(a) (b)

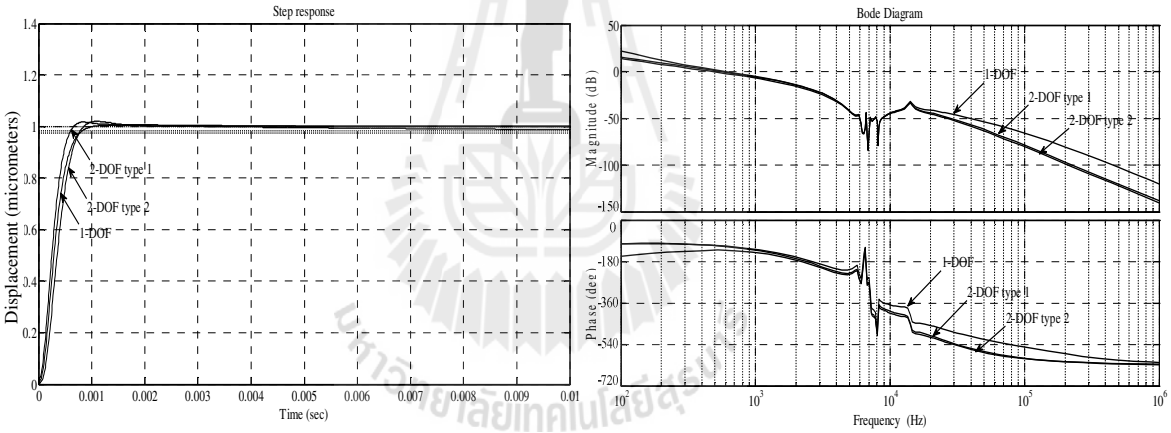
Figure 7.8 Compensated responses of single R/W head using the BF-TS:

(a) time-domains and (b) frequency-domains.

Table 7.7 summarizes the optimal compensators by the BF-TS, and their corresponding performance indices. Figures 7.8(a) and 7.8(b) show the step and the frequency responses of the compensated head. It is noticed that all compensation structures render very satisfactory stability margins and overshoots. In the case of strictly demanded low overshoot, the 2-DOF type 2 is the optimum choice. If the overshoot is lenient, the 1-DOF can be an alternative.

Table 7.8 Summary of the compensators and the responses of the single-head using the modified ATS algorithm.

Compensators	Mp (%)	t_r (ms)	t_s (ms) $e_{ss}=5\%$	GM (dB)	PM (deg.)	SAE
1-DOF; $Gc(s)=8.397\times10^{5}\dfrac{(s+100)(s+1.934\times10^{9})(s+8.407\times10^{4})}{(s+2.825\times10^{10})(s+9.899\times10^{5})(s+3.668\times10^{8})}$	0.48	0.44	0.98	20.30	70.20	37.83
2-DOF type 1; $Gc1(s)=9.803\times10^{3}\dfrac{(s+2.482\times10^{4})(s+7.145\times10^{9})(s+1.076\times10^{4})}{(s+1.434\times10^{9})(s+8.067\times10^{3})(s+1.646\times10^{9})}$ $Gc2(s)=9.932\times10^{3}\dfrac{(s+9.555\times10^{4})(s+9.350\times10^{9})(s+1.0\times10^{2})}{(s+6.446\times10^{8})(s+1.161\times10^{5})(s+7.595\times10^{9})}$	1.67	0.37	1.21	17.02	65.67	38.79
2-DOF type 2; $G^{*}c1(s)=7.289\times10^{3}\dfrac{(s+7.212\times10^{4})(s+8.246\times10^{9})(s+2.060\times10^{3})}{(s+6.658\times10^{9})(s+3.343\times10^{4})(s+2.086\times10^{9})}$ $G^{*}c2(s)=9.817\times10^{3}\dfrac{(s+8.128\times10^{4})(s+9.632\times10^{10})(s+1.211\times10^{3})}{(s+6.877\times10^{9})(s+7.821\times10^{4})(s+8.997\times10^{9})}$	1.97	0.48	1.10	15.38	47.59	48.53

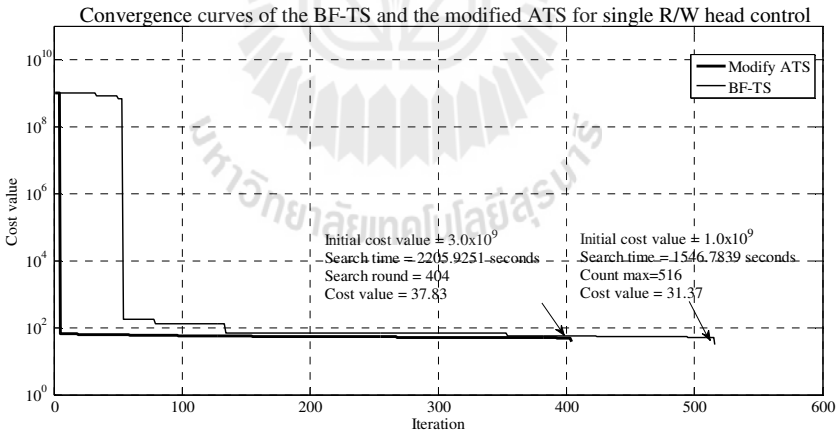


(a) (b)

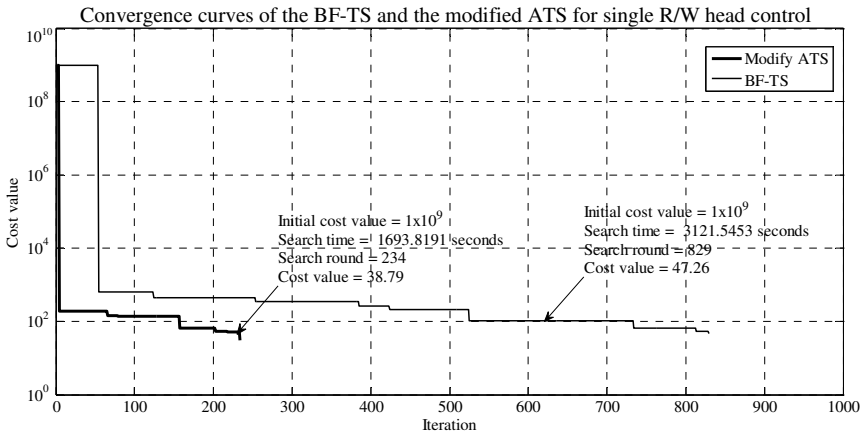
Figure 7.9 Compensated responses of single R/W head using the modified ATS: (a) time-domains and (b) frequency-domains.

Table 7.8 summarizes the optimal compensators for the single-head using the modified ATS algorithm. The time and frequency responses are depicted in Figures 7.9(a) and 7.9(b), respectively. The results show that all compensation structures can

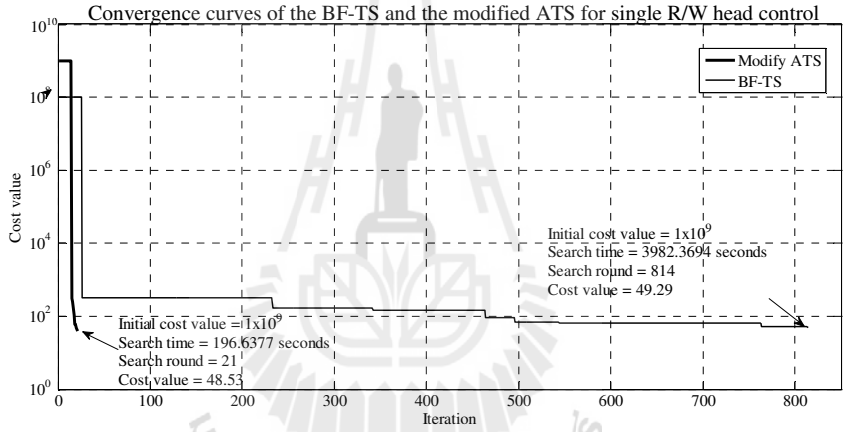
satisfy stability margins. All compensations obtained from the proposed modified ATS render the performance responses with low overshoots, rise times and settling times. Consequently, all compensation structures can make the single R/W head have better compensated performances and higher stability margins. A simple structure as the 1-DOF can be the first option to be used to minimize overshoot. To decrease rise time, the 1-DOF or the 2-DOF type 1 structures can be a good option. The 2-DOF type 2 is the best choice to obtain the shortest settling time and to increase the stability margins. Note that, the control systems considered are theoretic. More practical approaches can be considered by imposing technological limitations into the search problems, for examples, saturation characteristics of electronic controllers, quantization effects of digital controllers, etc.



(a)



(b)



(c)

Figure 7.10 Comparisons of convergence curves between the results of the BF-TS and the modified ATS on the single R/W head: (a) 1-DOF structure, (b) 2-DOF type 1 structure and (c) 2-DOF type 2 structure.

Figure 7.10 shows the convergent curves for comparison purposes between the results of the BF-TS and the modified ATS algorithms. These results show that the proposed approaches can satisfy termination criteria and all other conditions. Based-on these results, it can be observed that the modified ATS spends fewer numbers of

iteration than the BF-TS does for all compensation structures. Furthermore, the modified ATS as being proposed consumes fewer numbers of search rounds. If we compare the search time consumed per iteration between the approaches, the modified ATS spends time twice as much as that of the BF-TS for all control structures. Note that, a strong point of the modified ATS is that the algorithm can effectively avoid local deadlocks. This can be observed from the results obtained by using the BF-TS encounter many local deadlocks, which are 34, 58 and 45 times more for the 1-DOF, 2-DOF type 1 and 2-DOF type 2, respectively.

7.3.2 R/W Head-Stacks

Figure 7.11 provides a sketch of a hard-disk R/W head-stack that consists of a voice coil, pivot, actuator arm, and suspension. The tips of the suspensions are the read/write heads. A R/W head-stack has been modeled as a transfer-function comprising of 6 resonance modes with 12 poles, and 10 zeros (Mamun et al., 2002).

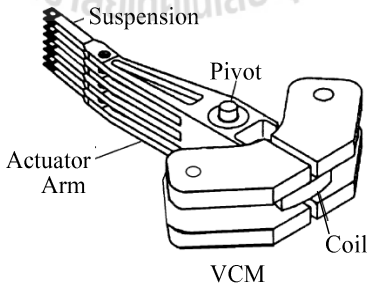


Figure 7.11 Mechanical structure of a hard-disk head-stack.

This model is expressed by

$$G_{hs}(s) = G_{hs1}(s)G_{hs2}(s)G_{hs3}(s)G_{hs4}(s)G_{hs5}(s)G_{hs6}(s) \tag{7.9}$$

where

$$G_{hs1}(s) = \frac{1}{s^2 + 4656s + 9.65\times10^8} \tag{7.10}$$

$$G_{hs2}(s) = \frac{s^2 - 2410s + 2.676\times10^8}{s^2 + 53.76s + 2.903\times10^4} \tag{7.11}$$

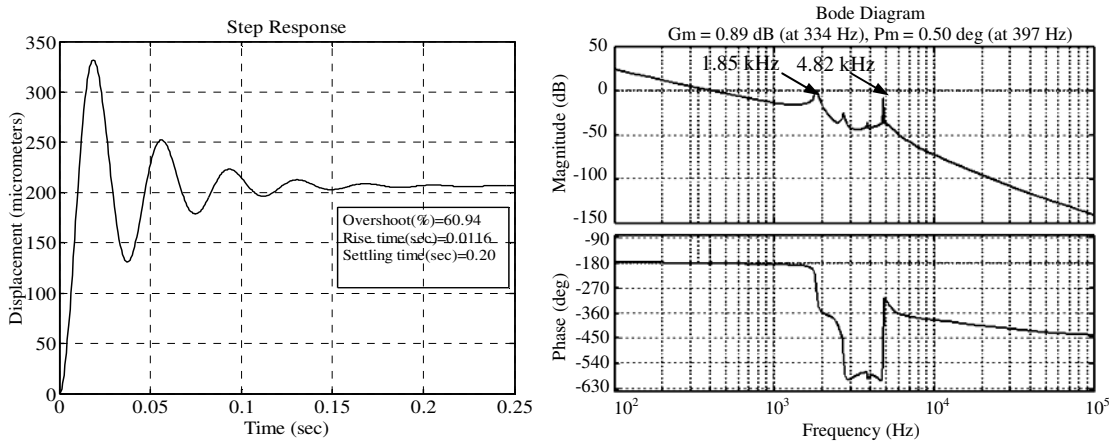
$$G_{hs3}(s) = \frac{s^2 + 4645s + 3.368\times10^8}{s^2 + 342.3s + 1.352\times10^8} \tag{7.12}$$

$$G_{hs4}(s) = \frac{s^2 + 375.6s + 5.82\times10^8}{s^2 + 400.4s + 2.913\times10^8} \tag{7.13}$$

$$G_{hs5}(s) = \frac{s^2 + 413.7s + 9.176\times10^8}{s^2 + 213.6s + 5.854\times10^8} \tag{7.14}$$

$$G_{hs6}(s) = \frac{s^2 - 2.162\times10^{10}s + 2.522\times10^{15}}{s^2 + 342.4s + 9.1\times10^8} \tag{7.15}$$

Due to severity of the resonance modes, the head-stack has low stability margins (GM=0.89 dB and PM=0.5deg.), and its step response contains an overshoot as high as 60.94% as illustrated by the step and the frequency-response plots in Figures 7.12(a) and 7.12(b), respectively. Referring to Figure 7.12(b), there are two resonance modes, at 1.85 and 4.82 kHz, having very low damping ratios that need proper pre-compensations. The pre-compensators utilize the common notch filters described in Appendix A.



(a) (b)

Figure 7.12 Open-loop responses: (a) time-domain and (b) frequency-domain.

After the pre-compensation, the head-stack still encounters an instability problem, i.e. its GM and PM are -28.60 dB and -31.30 deg., respectively. Bode plot in Figure A.6 reveals this fact (see Appendix A). A further dynamic compensation is therefore unavoidable. Based-on the 1-DOF structure of Figure 7.4(a), a 3-stage phase-lead compensator is designed first using the conventional design method found in textbooks. The design details can be found in Appendix A. As a result, a compensator $G_{PL2}(s)$ is obtained and expressed by

$$G_{PL2} = 50\left(\frac{s + 2.82\times10^{2}}{s + 2.78\times10^{5}}\right)\left(\frac{s + 1.570\times10^{3}}{s + 7.225\times10^{3}}\right)\left(\frac{s + 1.780\times10^{3}}{s + 5.341\times10^{3}}\right) \tag{7.16}$$

By using the anti-resonance filters and a third-order lead compensator, we can achieve the compensated system having GM = 9.33 dB and PM = 40.90 deg. (see Figure A.9(b)). Unfortunately, the system still contains a high overshoot of 41.28% (see

Figure A.9(a)). This requires a more stringent compensation for an overshoot suppression.

Therefore, the BF-TS and modified ATS algorithms have been applied in order to satisfy better compensated performances, and stability margins simultaneously according to the control structures shown in Figures 7.4(a)-7.4(c). The optimal multi-parameter control synthesis of compensators is formulated as in equation (7.8) for each control structure. The objective to be minimized and the constraint conditions for control performances are similar to those of the single R/W head. The search parameters for the BF-TS and the modified ATS have been utilized to the R/W head stacks shown in Tables 7.9 and 7.10, respectively.

Table 7.9 Boundaries and search parameters of the BF-TS for the R/W head stacks.

Compensators	Search parameters	S	N_C	N_S	α	R	N	$J_{min} \leq$	BT, n_re_back
1-DOF	$z_1, z_2\ z_3, p_1, p_2, p_3, K_1 = [100\text{-}10^{11}]$	30	100	4	1×10^{-6}	20	50	25	5
2-DOF type I	$z_1, z_2\ z_3, z_4, z_5, z_6, p_1, p_2, p_3, p_4, p_5, p_6, K_1, K_2 = [100\text{-}10^{11}]$								
2-DOF type 2									

The adaptive radius mechanisms are utilized as follows: if $J < 70$ then $R = 10$, if $J < 65$ then $R = 7$, if $J < 55$ then $R = 4$.

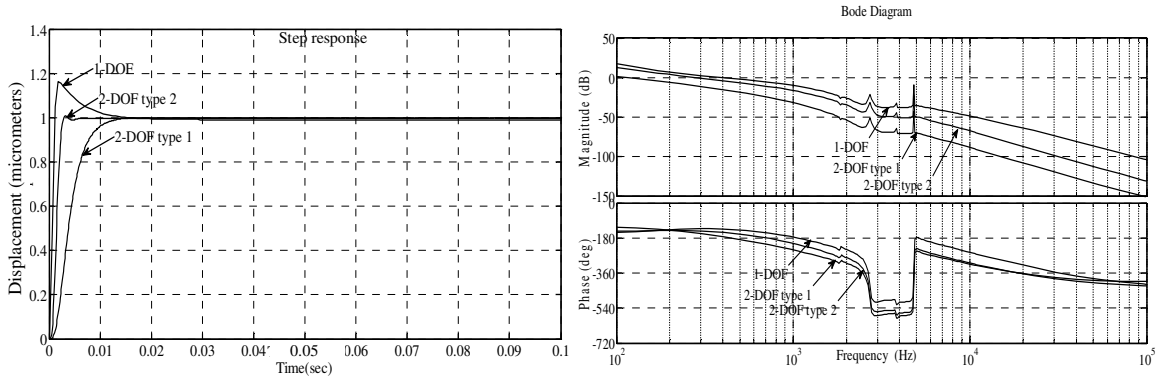
Table 7.10 Boundaries and search parameters of the modified ATS for the R/W head stacks.

Compensators	Search parameters	S	N_C	S_2	N_{C2}	N_S	α	α_2	J_{min} ≤
1-DOF	$z_1, z_2\ z_3,\ p_1,\ p_2,$ $p_3,\ K_1 = [100\text{-}10^{11}]$	30	100	1	50	4	1×10^{-6}	1×10^{6}	25
2-DOF type I	$z_1, z_2\ z_3,\ z_4, z_5,\ z_6,$ $p_1, p_2,\ p_3, p_4,\ p_5, p_6,$ $K_1, K_2 = [100\text{-}10^{11}]$							1×10^{-3}	
2-DOF type 2								1×10^{-3}	

The maximum count ($count_{max}$) is 5,000 for these approaches. The objective function (J) as listed in Figure 7.7 is also used.

Table 7.11 Summary of the compensators for the head-stacks using the BF-TS algorithm.

Compensators		(%) Mp	t_r (ms)	t_s (ms) $e_{ss}=5\%$	GM (dB)	PM (deg)	Σerror
1-DOF	$Gc(s) = 8.946\dfrac{(s+4.869\times10^4)(s+179.455)(s+4.376\times10^3)}{(s+1.344\times10^5)(s+8.148\times10^4)(s+7.312\times10^3)}$	16.397	1.02	15.38	14.16	63.21	1×10^9
2-DOF Type 1;	$Gc1(s) = 1.752\times10^3\dfrac{(s+4.293\times10^6)(s+2.932\times10^6)(s+3.083\times10^7)}{(s+3.434\times10^9)(s+209.242)(s+3.776\times10^9)}$	0.023	5.38	14.08	16.62	52.95	23.61
	$Gc2(s) = 1.038\times10^3\dfrac{(s+1.113\times10^6)(s+8.250\times10^8)(s+196.892)}{(s+2.132\times10^9)(s+2.439\times10^3)(s+1.606\times10^9)}$						
2-DOF Type 2;	$G^*c1(s) = 5.460\times10^3\dfrac{(s+5.258\times10^6)(s+3.093\times10^7)(s+8.385\times10^7)}{(s+5.466\times10^{10})(s+4.425\times10^3)(s+6.946\times10^9)}$	0.787	2.08	7.21	10.80	35.60	24.74
	$G^*c2(s) = 8.875\times10^3\dfrac{(s+5.681\times10^6)(s+3.704\times10^9)(s+744.0)}{(s+6.153\times10^9)(s+1.120\times10^4)(s+9.568\times10^9)}$						



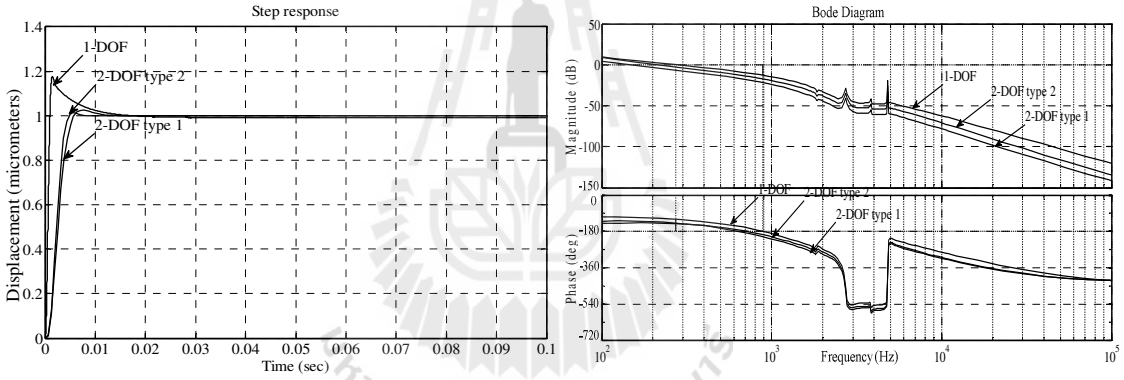
(a) (b)

Figure 7.13 Compensated responses of R/W head-stacks using BF-TS: (a) time-domains and (b) frequency-domains.

As a result, Table 7.11 summarizes the optimal compensators for the R/W head-stack found by the BF-TS with the performance indices. The corresponding step and frequency responses of the compensated head-stack are shown in Figures 7.13(a) and 7.13(b), respectively. It is observed that the compensation structures of the 2-DOF type 1 and the 2-DOF type 2 have given satisfactory stability margins and overshoots, while the 1-DOF structure results in high overshoots. Hence, while the R/W head-stacks need low overshoot responses, the 2-DOF type 1 structure is the optimum choice. The 2-DOF type 2 is another option when the system demands a rapid response with short rise time and settling time.

Table 7.12 Summary of the compensators for the head-stacks using the modified ATS algorithm.

Compensators	Mp (%)	t_r (ms)	t_s (ms) $e_{ss}=5\%$	GM (dB)	PM (deg.)	$\lvert\Sigma \text{error}\rvert$
1-DOF; $Gc(s)=10.816\dfrac{(s+4.866\times10^4)(s+156.601)(s+4.327\times10^3)}{(s+1.344\times10^5)(s+8.062\times10^4)(s+7.396\times10^3)}$	18.68	1.01	15.70	12.66	60.82	1×10^9
2-DOF type 1; $Gc1(s)=3.859\times10^3\dfrac{(s+3.085\times10^6)(s+7.084\times10^6)(s+7.352\times10^7)}{(s+9.340\times10^9)(s+3.798\times10^2)(s+7.709\times10^9)}$ $Gc2(s)=4.007\times10^3\dfrac{(s+4.695\times10^6)(s+1.279\times10^8)(s+3.546\times10^2)}{(s+1.807\times10^9)(s+6.688\times10^3)(s+1.433\times10^9)}$	2.47	4.43	11.40	16.67	51.89	23.63
2-DOF type 2; $G^*c1(s)=5.023\times10^3\dfrac{(s+3.347\times10^6)(s+6.999\times10^7)(s+5.460\times10^5)}{(s+5.712\times10^9)(s+1.391\times10^3)(s+3.204\times10^9)}$ $G^*c2(s)=4.598\times10^3\dfrac{(s+5.017\times10^6)(s+2.005\times10^9)(s+5.820\times10^2)}{(s+4.556\times10^9)(s+1.082\times10^4)(s+4.652\times10^9)}$	1.26	3.70	5.45	14.10	43.45	24.68

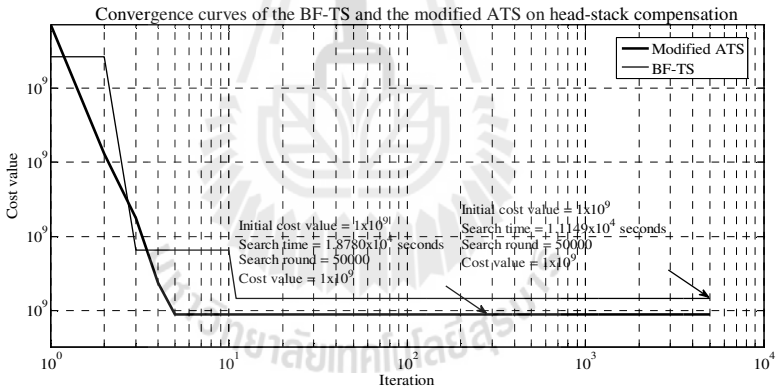


(a) (b)

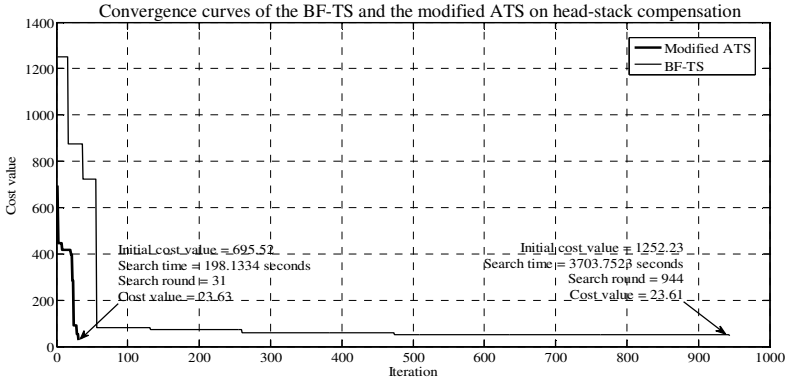
Figure 7.14 Compensated responses of R/W head-stacks using the modified ATS: (a) time-domains and (b) frequency-domains.

Table 7.12 presents the summary of the optimal compensators for the head-stacks using the modified ATS algorithm. The time and frequency responses are depicted in Figures 7.14(a) and 7.14(b), respectively. These results show that the proposed algorithms provide better compensated performances and stability margins

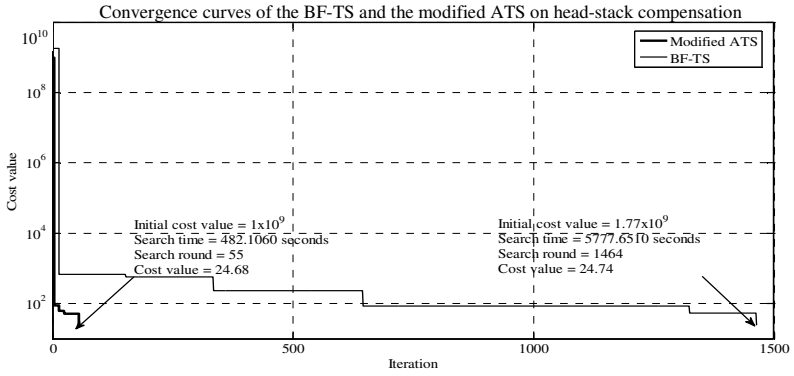
than the conventional control designs do. All compensation structures that are obtained from the proposed algorithm can satisfy stability margins. Notice that the response of 1-DOF compensator still provides a high overshoot over the criterion, while the 2-DOF type 2 structure provides a superior time response in terms of low overshoot, short rise time and settling time. As a recommendation, in case to minimize the overshoot errors and to increase the stability margins, the 2-DOF type 1 would be a good choice. To decrease the rise time and settling time, the 2-DOF type 2 structure is the best option. Unfortunately, the 1-DOF structure is not suitable for dynamic compensation problem of the R/W head-stacks. Similar to the previous control problems, practical constraints concerning technological limitations can be implemented into the search problems.



(a)



(b)



(c)

Figure 7.15 Comparisons of convergence curves between the results of the BF-TS and the modified ATS for the R/W head-stacks: (a) 1-DOF structure, (b) 2-DOF type 1 structure and (c) 2-DOF type 2 structure.

Figure 7.15 presents the convergent curves for comparison purposes between the results of the BF-TS and the modified ATS for the R/W head-stacks based-on the proposed compensation structures. The 2-DOF type 1 and type 2 structures satisfy termination criteria of the sum of absolute errors and all other conditions. The 1-DOF controller unsatisfactorily hit the maximum numbers of iterations without rendering solutions that satisfy the performance and the stability criteria (see Figure 7.15(a)). All the convergent curves shown in Figure 7.15 indicate that the modified ATS converges to the global region faster than the BF-TS does. As a result, the modified ATS being proposed spends a fewer numbers of search round than the BF-TS does by 30.45 and 26.62 times for the 2-DOF type 1 and type 2, respectively. If we compare the search time consumed per iteration between these approaches, the modified ATS spends approximately double times more than the BF-TS does for all control structures. Note

that the modified ATS can avoid the local deadlocks effectively, the BF-TS is usually trapped. The following figures indicate the times the BF-TS being trapped by the local locks during the search: 734, 39 and 61 times for 1-DOF, 2-DOF type 1 and 2-DOF type 2, respectively.

7.4 Brake Control of Heavy-Duty Truck

7.4.1 Conventional Brake Control of a Heavy-Duty Truck

Reliable and effective braking system is an essential part of a heavy-duty dump trucks. Although conventional braking systems are generally adequate, in the cases of uphill and downhill running the truck encounters severe braking problems. These problems arise because the truck braking system is subject to variations in dead time, changes in the input function and variations in its other parameters. If these changes are considerable and the stability region is limited, then critical tolerances can be exceeded, and even instability may be resulted if conventional control design methods are used. This means that strict regulation must be maintained for all operating conditions in order to avoid catastrophic situations caused by inadequacy or failure of the brake system. Taking into account safety and financial considerations, the need exists for a robust braking control system to ensure strict regulation of truck speed over a wide range of operating conditions (Bada, 1987; Zakian, 2005; Sarasiri and Sujitjorn, 2011).

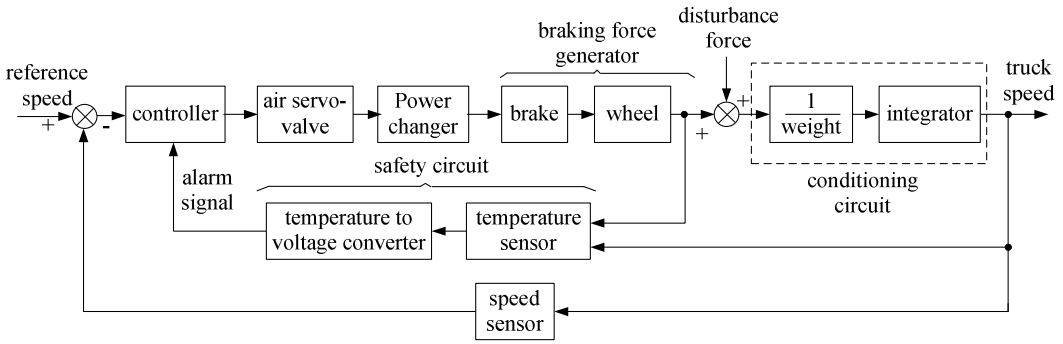
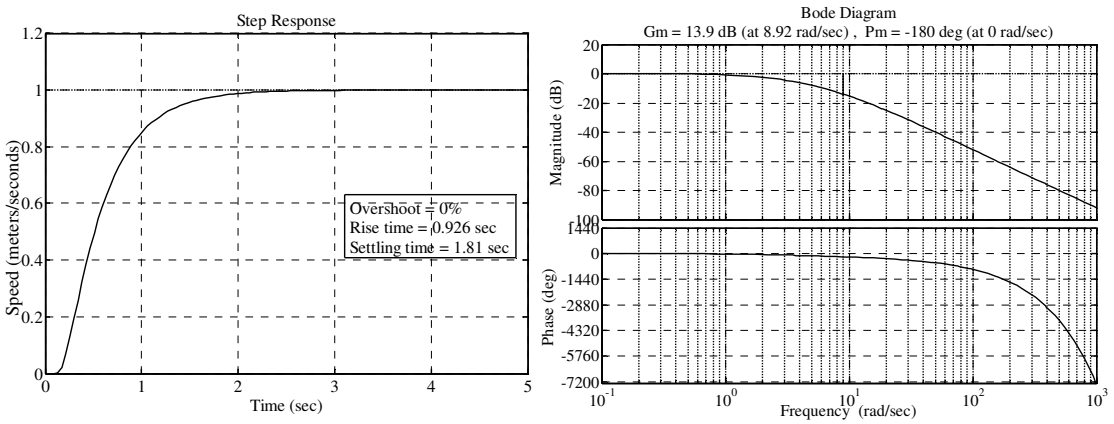


Figure 7.16 Truck braking control loop.

A closed-loop system of a truck braking control can be represented by the block diagram in Figure 7.16, where the control loop consists of 5 major components. These components are a microprocessor as a controller, an air-servo valve converting the electrical signal from the controller to air pressure, a power changer converting the air pressure to hydraulic pressure, a hydraulic brake generating the required braking force, a speed sensor in the feedback loop, and a safety circuit issuing warnings against excessive hydraulic brake temperature. For conventional control design, an open-loop braking system is commonly approximated by a plant transfer function as in Equation (7.17).

$$G_p(s) = \frac{e^{-0.125s}}{s(1+0.4s)(1+0.1s)} \tag{7.17}$$



(a) (b)

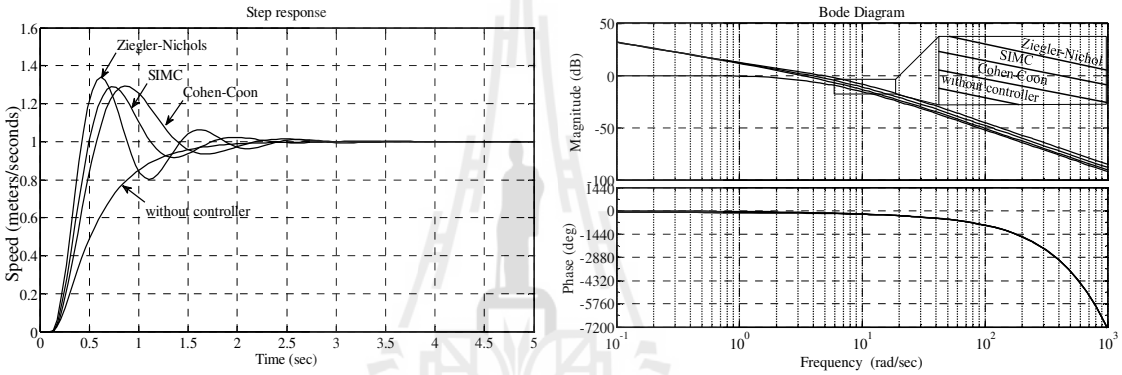
Figure 7.17 Open-loop responses of truck braking system: (a) time-domain and (b) frequency-domain.

Figure 7.17 shows the open-loop responses of the truck braking system. Even though the time-domain response settles in 1.81 seconds, the PM of -180 deg. of the uncontrolled plant is unacceptable. Stability margins of the system can be improved using closed-loop control aiming for GM $\geq$ 5 dB and PM $\geq$ 50 deg.

In order to design the controllers, conditioning circuits and disturbance forces are omitted. For comparison purposes, PI-controllers are designed using some conventional methods namely the simple internal model control (SIMC) method (Normey-Rico and Camacho, 2007), Ziegler-Nichol method (Ziegler and Nichols, 1942), Cohen-Coon method (Cohen and Coon, 1953), respectively. The design details are described in Appendix B. Table 7.13 summarizes closed-loop performance indices and stability margins, while the corresponding response plots are shown in Figure 7.18.

Table 7.13 Summary of performance indices and stability margins based-on conventional PI-controllers for a truck braking system.

Controller methods	P.O. (%)	t_s (sec.) $e_{ss} < 0.5\%$	t_r (sec.)	GM (dB)	PM (deg.)
without controller	0	1.81	0.93	13.90	-180
SIMC	29.0	2.10	0.26	7.31	42.60
Ziegler-Nichol	34.0	2.29	0.21	5.37	38.40
Cohen-Coon	29.40	2.04	0.31	8.73	41.30



(a) (b)

Figure 7.18 Responses of truck braking system with conventional PI-controllers: (a) time-domains and (b) frequency-domain.

7.4.2 Braking Control of Truck Based-on Search

Despite the system with conventional PI-controllers is more robustly stable than the system without a controller as can be noticed from the high values of GM and PM summarized in Table 7.13, the time-domain responses still possess high overshoots as shown in Figure 7.18(a). To achieve overshoot minimization under some stability constraints, the controllers could be obtained by using search methods.

In this, we employ the BF-TS and the modified ATS metaheuristics, and the problem can be formulated as follows: the controller representation is

$$\hat{G}_c(s) = \frac{k_1\left(1+k_2 s\right)}{s} \tag{7.18}$$

, in which k_1 and k_2 are positive real. The problem is to

$$\begin{aligned} &\text{minimize } J\text{:} && J = \Sigma\left|e(t)\right| \\ &\text{subject to} && P.O. < 0.01\% \\ & && GM > 5dB \\ & && PM > 25deg \end{aligned}$$

, in which $e(t)=r(t)-c(t)$ or step-response errors in the form of sum absolute error (*SAE*). The search parameters are declared in Table 7.14 and Table 7.15 for the BF-TS and modified ATS algorithms, respectively.

Table 7.14 Boundaries and search parameters of the BF-TS for the truck braking system.

Search parameters	S	N_c	N_s	α	R	N	J_{min} ≤	BT, n_re_back	AR	$count_{max}$
k_1=[0 5] and k_1=[0 1]	30	20	4	100	1	50	480	5	if J < 650 then R = 0.1, if J < 500 then R = 0.05, if J < 485 then R = 0.001	1,000

Table 7.15 Boundaries and search parameters of the modified ATS for the truck braking system.

Search parameters	S	N_c	S_2	N_{c2}	N_s	α	α_2	$J_{min} \leq$	$count_{max}$
k_1=[0 5] and k_1=[0 1]	30	20	1	50	4	100	1×10^6	480	1,000

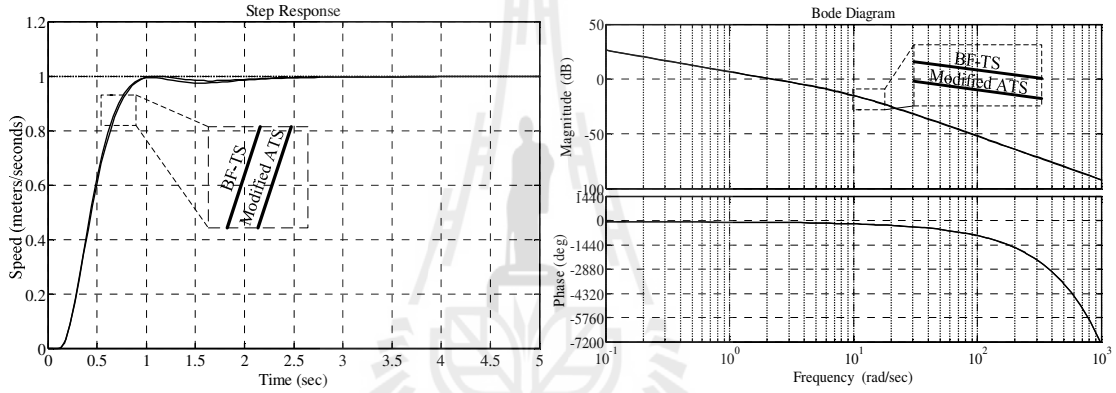
Figure 7.19 gives the code list of the objective function (J).

```
SAE = sum(abs(1-Close_system))
if overshoot < 0.01
        overshoot = 0
else
        overshoot= 1×10^3
end
if Gm > 5
        Gm = 0
else
        Gm =1×10^3
end
if Pm > 25
        Pm = 0
else
        Pm = 1×10^3
end
J = SAE + (overshoot^2 + Gm^2 + Pm^2 )× 1×10^3
```

Figure 7.19 Objective function of the truck braking system.

Table 7.16 Summary of the PI-controllers, performance indices and stability margins obtained from the BF-TS and the modified ATS algorithms.

Metaheuristic method and controller	P.O. (%)	t_s (sec.) $e_{ss} < 5\%$	t_r (sec.)	GM (dB)	PM (deg.)	SAE
BF-TS: $K_{BF-TS}(s) = \frac{2.090(1+0.490s)}{s}$	0	1.79	0.488	11.80	66.90	478.9279
Modified ATS: $K_{moATS}(s) = \frac{2.086(1+0.468s)}{s}$	0	0.90	0.506	12.10	66.50	479.9856



(a) (b)

Figure 7.20 Response plots of the truck braking system with PI-controllers via search: (a) time-domain and (b) frequency-domain.

Table 7.16 gives a summary of the results obtained from search. The curves shown in Figure 7.20 and the numeric values in Table 7.16 confirm that the closed-loop system is robustly stable, and possesses very smooth step responses. Noticeably, both search methods render similarly good controllers under no practical constraints being considered. Interested readers could implement control limits into the search processes, if more practical solutions are to be obtained. Moreover, the closed-loop performances

of the systems with the searched controllers are much better than those with conventional control design.

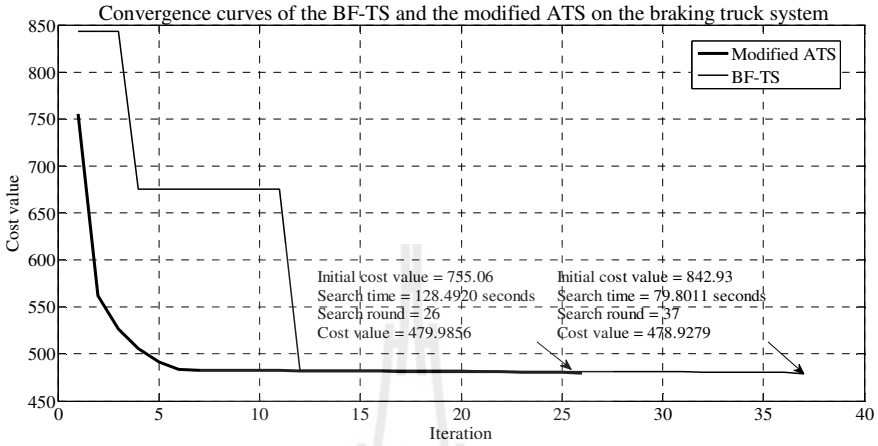


Figure 7.21 Comparisons of convergence curves between the BF-TS and the modified ATS for the truck braking system.

The modified ATS converges to the optimal solution without being trapped by local locks, and faster than the BF-TS does as indicated by the curves shown in Figure 7.21, while the BF-TS encounters 5 entrapments by local locks. The time consumed per iteration by the BF-TS and the modified ATS are 2.16 and 4.94 seconds, respectively, which means that the modified ATS spends search time more than the BF-TS by 2.30 times.

7.5 Stability Analysis Problem

Lyapunov's stability methods have been successfully applied for many years by engineers and scientists (Khalil, 1992; Slotine and Li, 1991) to analyse stability of nonlinear dynamical systems. The direct method of Lyapunov's is regarded as clean

and concise, and its principal idea is based-on the fact to that the total energy of a stable system decreases along its trajectories. This implies that the states of the stable system are bounded on the phase plane. The method requires the knowledge of a positive scalar function *V(x)*, which is referred to as the Lyapunov function. This energy-like function is constructed in terms of the system states. The theorem is commonly stated as follows:

A given system is globally asymptotically stable providing that *V(x)* satisfies the following properties:

- $V(0) = 0;$
- $V(x) > 0, \forall x \neq 0;$
- $\dot{V}(x) < 0, \forall x \neq 0;$
- $V(x) \to \infty \ as \ \|x\| \to \infty.$

The theorem is very clear and concise, but the construction of *V(x)* is rather intuitive and based-on skills.

For some systems, finding and constructing the Lyapunov function is not straightforward. Doing this manually in some cases is very time consuming, perhaps not possible. Once the Lyapunov function is obtained for the system of interest, the next practical issue becomes seeking for the region of attraction. In order to find this, some computational approaches, e.g. geometrical, numerical methods, etc., have to be applied. For instance, various previous works have proposed the construction of Lypaunov functions based-on conventional methods (Golub et al., 1979; Wang and Dayawansa, 1999), numerical methods (Sorensen and Zhou 2003; Zhaolu and

Chuanqing, 2008) and artificial intelligent methods (Grosman and Lewin, 2008; Carl, 2002).

The search method is particularly useful for finding the coefficients of *V(x)* and the construction of the stability region or region of attraction. Let us consider a nonlinear autonomous system described by

$$\dot{\mathbf{x}}_1 = -x_1 + 2x_1^2 x_2 \tag{7.19}$$

$$\dot{\mathbf{x}}_2 = -x_2 \tag{7.20}$$

having equilibrium at the origin. Genesio and Vicino (Genesio and Vicino, 1984) proposed *V(x)* of the form

$$V(x,p) = p_{21}x_1^2 + p_{22}x_1x_2 + p_{23}x_2^2 + p_{41}x_1^4 + p_{42}x_1^3x_2 + p_{43}x_1^2x_2^2 + p_{44}x_1x_2^3 + p_{45}x_2^4 \tag{7.21}$$

and assumed p_{41}, p_{44} and p_{45} to be zero. The proposed BF-TS and modified ATS algorithms are used to search for five coefficients of *V(x)* as a constrained optimization problem:

$$\text{minimize} \quad J:\ J = 1/\rho$$

$$\text{subject to} \quad V(x) > 0 \text{ and } \dot{V}(x) < 0$$

in which ρ is the area of the region (Panikhom et. al, 2010) bounded by *V(x)* on the phase plane.

Table 7.17 Boundaries and search parameters of the BF-TS for the stability analysis.

Search parameters	S	N_C	N_S	α	R	N	J_{min} ≤	BT and n_re_back $count_{max}$	AR
$p_{21}, p_{22}, p_{23}, p_{42}, p_{43}$ = [0 1]	30	20	4	1000	0.1	50	0.0605	5, 1,000	if J<0.08 then R=0.001, if J<0.065 then R=0.0005, if J<0.061 then R=0.00001

Table 7.18 Boundaries and search parameters of the modified ATS for the stability analysis.

Search parameters	S	N_C	S_2	N_{C2}	N_S	α	α_2	$J_{min} \leq$	$count_{max}$
$p_{21}, p_{22}, p_{23}, p_{42}, p_{43}$ = [0 1]	30	20	1	50	4	1000	1×10^6	0.0605	1,000

The objective function (J) is listed in Figure 7.22.

if $V(x) < 0 \,||\, \dot{V}(x) > 0$

$J = 1\times10^3$

else

$J = 1/\rho$

end

Figure 7.22 Objective function of the stability analysis.

Hence, the BF-TS returns the $V(x, p)$ function in (7.22), whilst the modified ATS returns (7.23).

$$V(x_1, x_2) = 0.222x_1^{\,2} + 0.545x_1x_2 + 0.5292x_2^{\,2} + 0.0125x_1^{\,3}x_2 + 0.0175x_1^{\,2}x_2^{\,2} \quad (7.22)$$

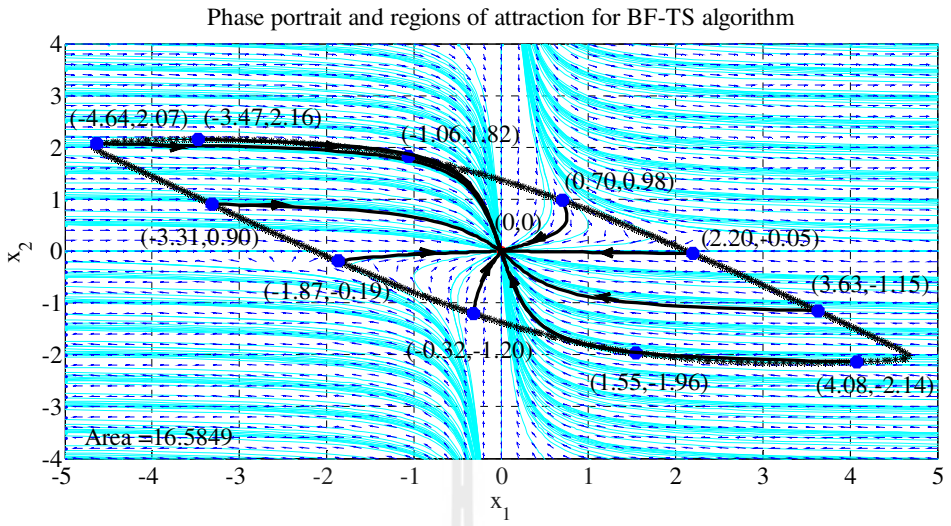
$$V(x_1, x_2) = 0.2097x_1^{\,2} + 0.5484x_1x_2 + 0.5459x_2^{\,2} + 0.0109x_1^{\,3}x_2 + 0.0186x_1^{\,2}x_2^{\,2} \quad (7.23)$$

Both $V(x, p)$ functions are positive definite, and their corresponding $\dot{V}(x_1, x_2)$ are negative definite as expressed in (7.24) and (7.25), respectively.

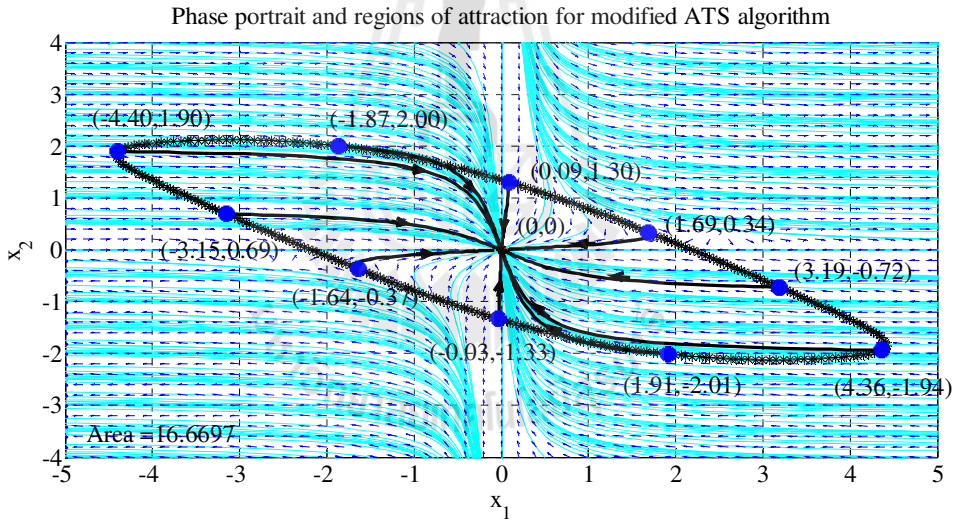
$$\begin{aligned}\dot{V}(x_1, x_2) = {} & 0.444x_1\dot{x}_1 + 0.545(\dot{x}_1x_2 + x_1\dot{x}_2) + 1.058x_2\dot{x}_2 + 0.0125(3x_1^{\,2}\dot{x}_1x_2 + x_1^{\,3}\dot{x}_2) \\ & + 0.035(x_1\dot{x}_1x_2^{\,2} + x_1^{\,2}x_2\dot{x}_2)\end{aligned} \quad (7.24)$$

$$\begin{aligned}\dot{V}(x_1, x_2) = {} & 0.4194x_1\dot{x}_1 + 0.5484(\dot{x}_1x_2 + x_1\dot{x}_2) + 1.0918x_2\dot{x}_2 + 0.0109(3x_1^{\,2}\dot{x}_1x_2 + x_1^{\,3}\dot{x}_2) \\ & + 0.0372(x_1\dot{x}_1x_2^{\,2} + x_1^{\,2}x_2\dot{x}_2)\end{aligned} \quad (7.25)$$

According to the coefficients of *V(x)* obtained by the BF-TS and modified ATS in (7.22) and (7.23) present almost similar numbers and stability areas. This can be noticed that the results obtained are the critical areas of the available regions bounded by *V(x)*. The phase portraits and regions of attractions are computed by the proposed algorithms given in Figure 7.23.



(a)



(b)

Figure 7.23 Phase portraits and regions of attraction: (a) obtained from BF-TS algorithm and (b) obtained from modified ATS algorithm.

Figures 7.23(a) and (b) illustrate the plots of Lyapunov's functions obtained by the BF-TS and modified ATS algorithms and their associative regions of attractions,

respectively. The results obtained by the proposed algorithms are less conservative, and provide larger stability regions than that of the previous work, 12.9799 (unit-less in area) (Genesio and Vicino, 1984). The modified ATS provides the largest region of 16.6697 (unit-less in area), and the BF-TS provides the second large-area size of 16.5849 (unit-less in area). It can be noticed that all initial trajectories on boundaries obtained from the BF-TS and the modified ATS converge to equilibrium points. Therefore, this nonlinear autonomous system is globally asymptotically stable, and the corresponding stability regions are contained inside the elliptical curves surrounding the equilibrium points. Figure 7.24 depicts the convergence curves indicating that the BF-TS and the modified ATS stop when the termination criterions are satisfied, which are $J = 0.0603$ and $J = 0.0600$, respectively.

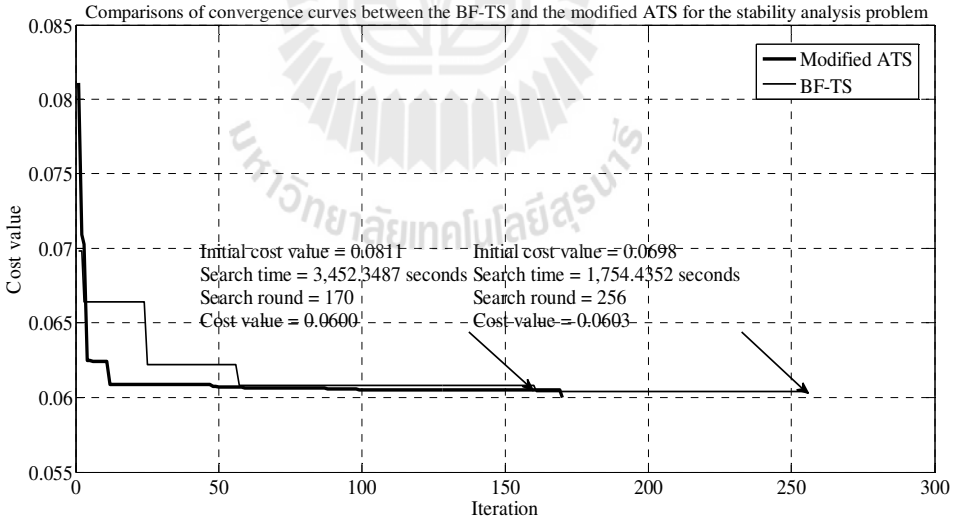


Figure 7.24 Comparisons of convergence curves between the BF-TS and the modified ATS for the stability analysis.

Referring to Figure 7.24, the modified ATS spends fewer numbers of iteration to reach for the *V(x)* function than the BF-TS does for 1.50 times with high-quality initial solutions. However, the modified ATS spends 3 times as much of the search time in comparison with the BF-TS considering in terms of search time consumed per iteration. Searches performed by the modified ATS do not encounter any local deadlocks, while the BF-TS hits some entrapments 8 times. As a result, both algorithms effectively search for satisfactory coefficients of the Lyapunov function; the modified ATS does provide the result with somewhat larger region of attraction.

7.6 Identification Problems

An identification task can be formulated as an optimization problem solvable by available optimization algorithms. Artificial intelligence (AI) based methods are efficient candidates of the present technology. For some instances, the tabu search and simulated annealing algorithms were applied to identify the optimal parameter structure for groundwater models (Zheng and Wang, 1996); the adaptive tabu search was applied to harmonic identification for an active power filter (Kulworawanichpong et al., 2004); the particle swarm optimization was applied to solve the modelling identification of thermal power plant (Liu and He, 2005), and recently the bacterial foraging algorithm was applied for the radio frequency identification (RFID) communication system (Chen et al., 2010).

In this thesis, we consider two identification problems of the nonlinear Stribeck friction model and the actuator model in a servo track writing system which are identified by the proposed BF-TS algorithm. Furthermore, the modified ATS is applied to identify the parameters of actuator model in a servo track writing system.

7.6.1 Identification of Nonlinear Friction Model

The nonlinear Stribeck friction model and experimental setup are detailed in Appendix C. For more details refer to (Suthamno, 2004; Sarasiri et al., 2012). During the search process to identify the friction model parameters, an objective function (J) has to be evaluated repeatedly. To calculate the objective function, it is assumed that the translational dynamic can be represented by the mass-spring model as follows

$$m\ddot{x} = k_{spring}(x_i - x_d) - F_f(v, F_{in}) + F_{ex} \tag{7.26}$$

in which the parameters F_S (static friction), F_C (coulomb friction), F_V (viscous friction), *vss* (crossover velocity) and k_{spring} (gravity constant) are to be identified. In this identification problem, the mass (m) is known, m=10.90 kg. Objective function implementation is shown below as the procedural list.

Step1: Calculate an average displacement $x_i = \left(\sum_{j=1}^{50} x_{test}(j)\right) \Big/ 50$.

For ramping-up motion, calculate an approximated force $F_{motor} = \hat{k}(x_i - x_d) \approx 3.5(x_i - x_d)$.

For ramping-down motion, calculate an approximated force $F_{motor} = \hat{k}(x_i - x_d) \approx 2.8(x_i - x_d)$.

Calculate $F_{fm} = T_m K_v = (2.40 \times 10^{-3})(2\pi \eta_b \eta_c a)/l = 13.33\ N$

Step2: if $\left(F_{motor} \geq F_{fm}\right)$ then $\left(F_{f\_motor} = F_{fm}\right)$.

if $\left(F_{motor} \leq -F_{fm}\right)$ then $\left(F_{f\_motor} = -F_{fm}\right)$.

if $\left(-F_{fm} < F_{motor} < F_{fm}\right)$ then $\left(F_{f\_motor} = F_{motor}\right)$.

Step3: Calculate $F_{m\_motor}(v) = K_{kt}\left(J_m \frac{dv}{dt} + D_m v \right)$.

if $\left(F_{m\_motor} > 30 \text{ N}\right)$ then $\left(F_{m\_motor} = 30 \text{ N}\right)$.

if $\left(F_{m\_motor} < -30 \text{ N}\right)$ then $\left(F_{m\_motor} = -30 \text{ N}\right)$.

Step4: Calculate the following forces:

externally applied force: $F_{ex} = F_{motor} - F_{f\_motor}(F_{motor}) - F_{m\_motor}(v)$,

spring force: $F_{sp} = k_{spring}(x_i - x_d)$,

internally applied force: $F_{in} = F_{ex} + F_{sp}$,

stick friction force: if $\left(F_{in} > F_S\right)$ then $\left(F_{stick} = F_S\right)$,

if $\left(F_{in} < -F_S\right)$ then $\left(F_{stick} = -F_S\right)$,

if $\left(-F_S \le F_{in} \le F_S\right)$ then $\left(F_{stick} = F_{in}\right)$, and

slip friction force: $F_{slip}(v) = \left(F_C + \left(F_S - F_C \right).e^{-\left(v/vss\right)} \right).\text{sgn}(v) + F_v.v$.

Step5: Calculate velocity and displacement of the mass:

$P = \int \left[F_{in} - F_f(v, F_{in}) \right] dt.$

if $(-dp < P < dp)$ then $(v = 0)$ otherwise $(v = \frac{P}{m})$.

$x_d = \int v \, dt.$

Step6: Calculate the objective function:

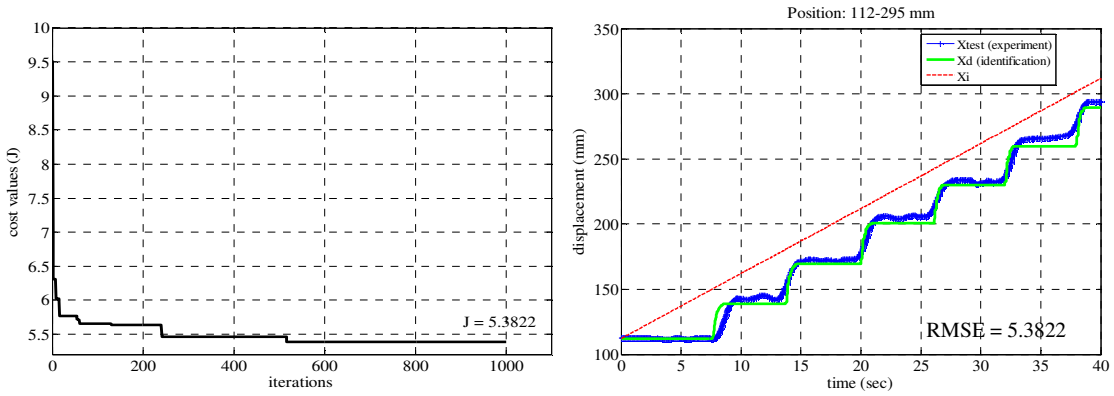
$$J = \sqrt{\frac{\sum_{i=1}^{n} (x_{test} - x_d)^2}{n}}.$$

Step7: Return to main search.

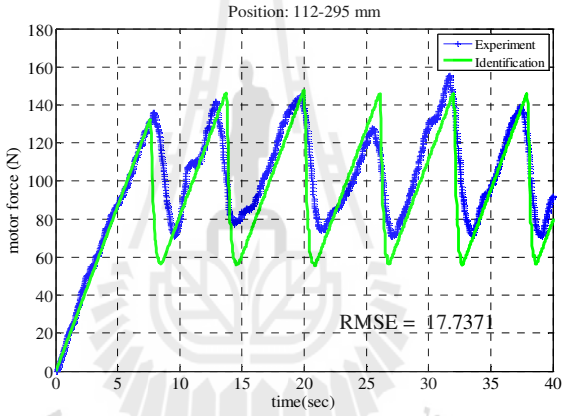
Table 7.19 Boundaries and search parameters of the BF-TS for identification of the nonlinear Stribeck friction model.

Search parameters	S	N_c	N_s	α	R	N	J_{min} ≤	BT and n_re_back	$count_{max}$	AR
F_s=[100-180] F_c=[40-80], F_v=[0.4-1.0], vss=[1-7] k_{spring}=[0.1-1.5]	30	20	4	1000	0.15	50	4.5	5	1000	if J<15 then$R$=0.0375, if J<8 then R=0.0095, if J<5 then R=0.0025

Due to the strong nonlinearity in friction force, it is necessary to identify two sets of model parameters corresponding to rightward and leftward motions. Referring to Figure 7.25, the illustrated graphs correspond to the rightward motion (positive direction, ramp-up command), i.e. ramp-up command of 5 mm/s, that uses the data in the range of 112-295 mm for identification. The convergence curve in Figure 7.25(a) indicates that the search terminated by $count_{max}$=1000. The cost value returned J = 5.3822. The obtained parameters are as follows: { F_s =144.5463 N, F_c = 47.8514 N, F_v = 0.9583 Ns/mm, vss =1.0964 mm/s, k_{spring} = 0.96798 N/mm}. The experimental data and the model plots for displacement, and force exerted by motor are illustrated in Figure 7.25(b)-(c), respectively. The results show a good agreement between the experiment and the model indicated as root mean square errors (RMSE) of 5.3822 and 17.7371 for the displacement, and force exerted by motor, respectively.



(a) (b)



(c)

Figure 7.25 Identification results of ramp-up command at 5 mm/s: (a) convergence curve, (b) displacement and (c) force exerted by motor. (Note: positions in the range of 112-295 mm).

For the leftward motion (negative direction, ramp-down command), i.e. ramp-down command of -5 mm/s, the graphical displays of identification results are shown in Figure 7.26. The data used for identification are in the range of 325-127 mm.

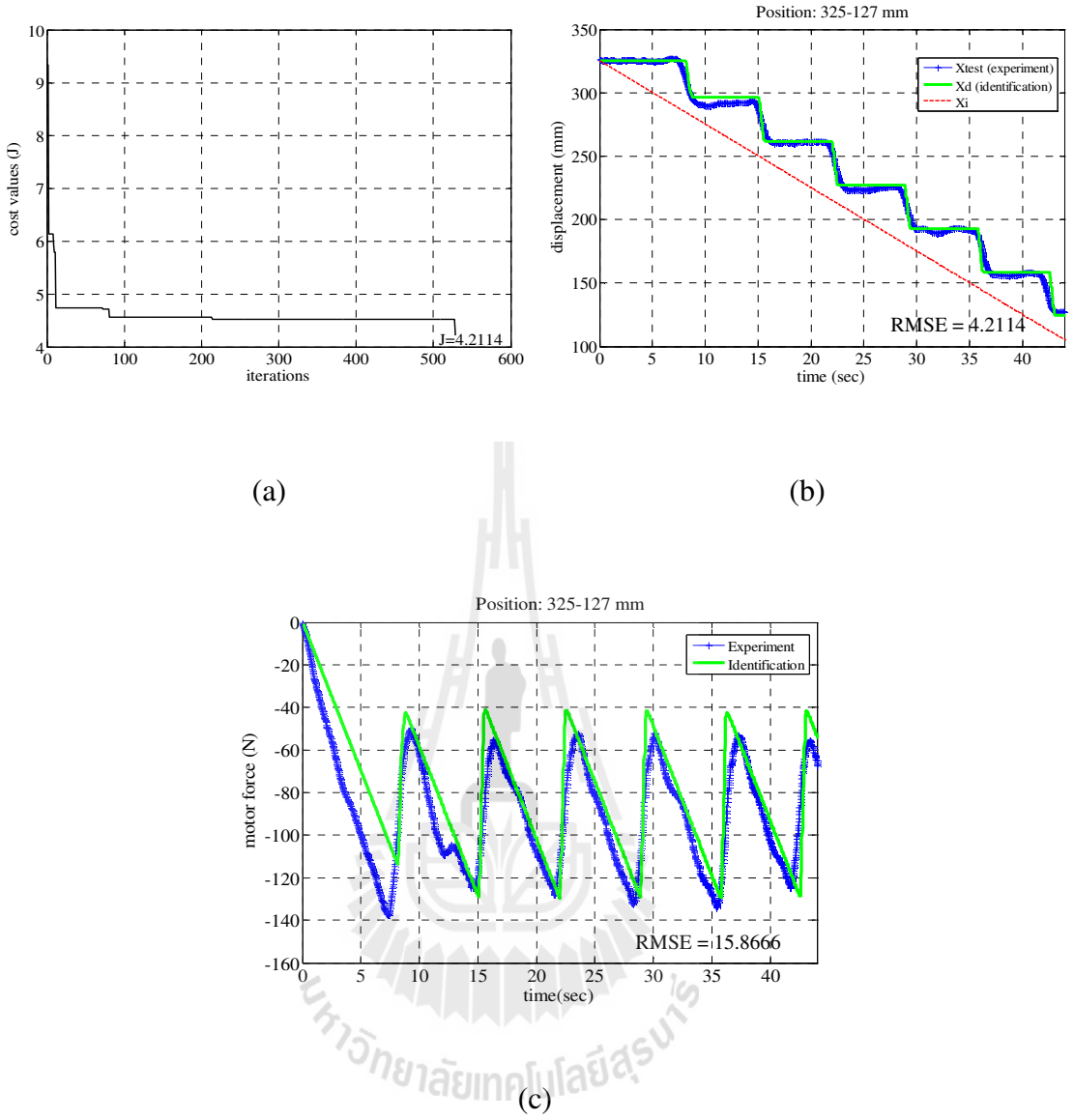
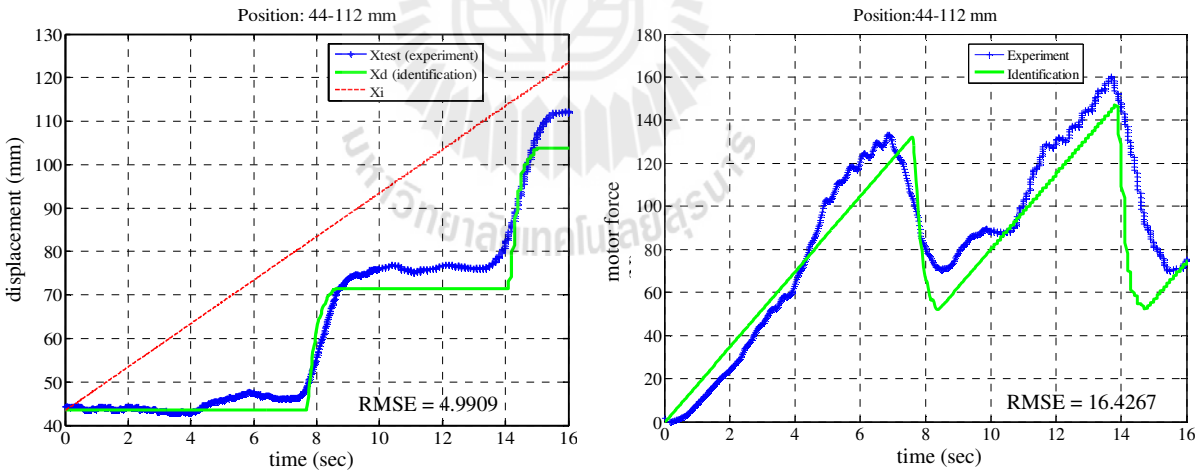


Figure 7.26 Identification results of ramp-down command at -5 mm/s: (a) convergence curve, (b) displacement and (c) force exerted by motor. (Note: positions in the range of 325-127 mm).

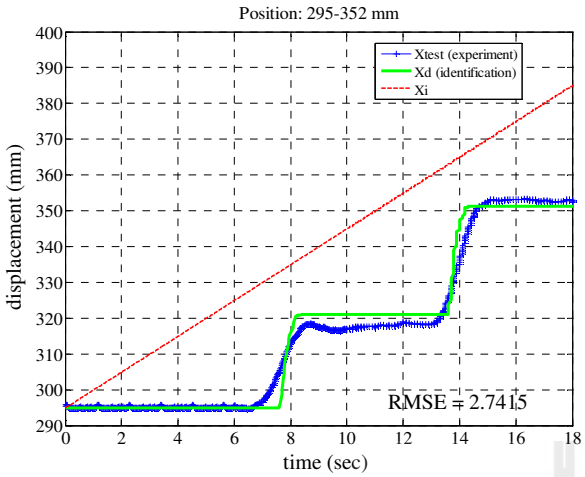
As indicated by the convergence curve in Figure 7.26(a), the search terminated at the 528th iteration, and returned the solutions with the cost J = 4.2114. The obtained parameters are as follows: { F_S = -152.2804 N, F_C = -40.4153 N, F_V = -0.9757 Ns/mm,

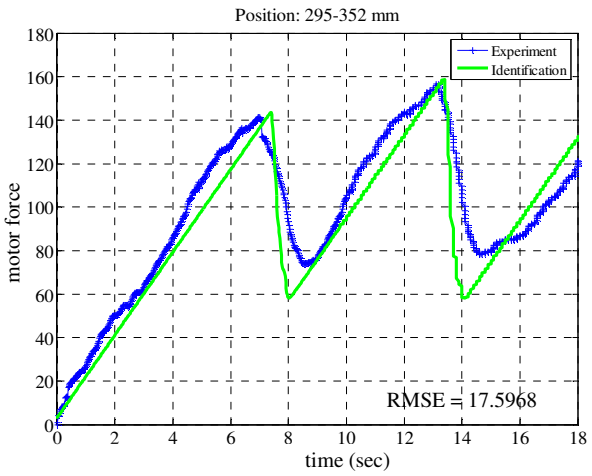
vss = -3.21475 mm/s, k_{spring} = 0.55384 N/mm}. The experimental data and the model plots in Figures 7.26(b)-(c) show a good agreement indicated as RMSEs, 4.2144 and 15.8666 for the displacement, and force exerted by motor, respectively.

Model validation was conducted for both directions of motion. Figure 7.27 illustrates the experimental data and the model plots for the rightward direction covering two ranges, i.e. 44-112 mm and 295-352 mm. Figure 7.27(a)-(b) display the plots of the displacement and the force exerted by motor for 44-112 mm range. Similarly, the results for 295-352 mm range are shown in Figure 7.27(c)-(d). For the leftward direction covering 352-325 mm and 127-68 mm ranges, similar graphical displays are illustrated in Figure 7.28(a)-(d). Very good agreement among the practical and the theoretical results can be observed.



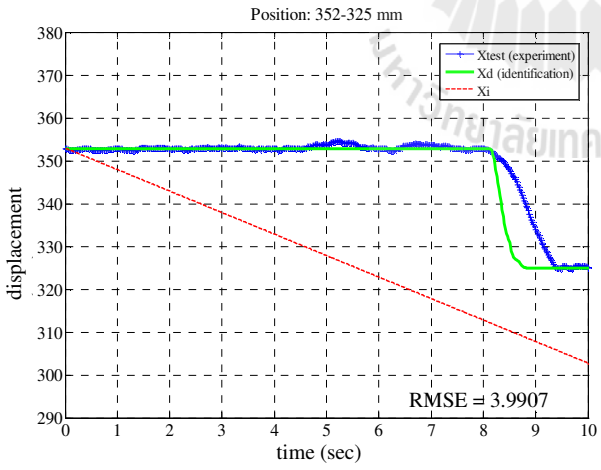
(a) (b)

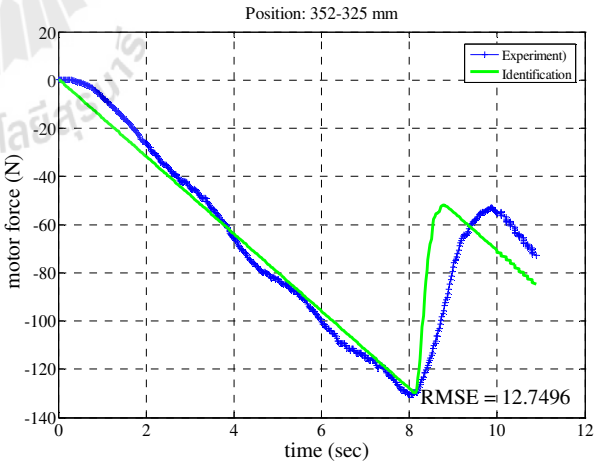




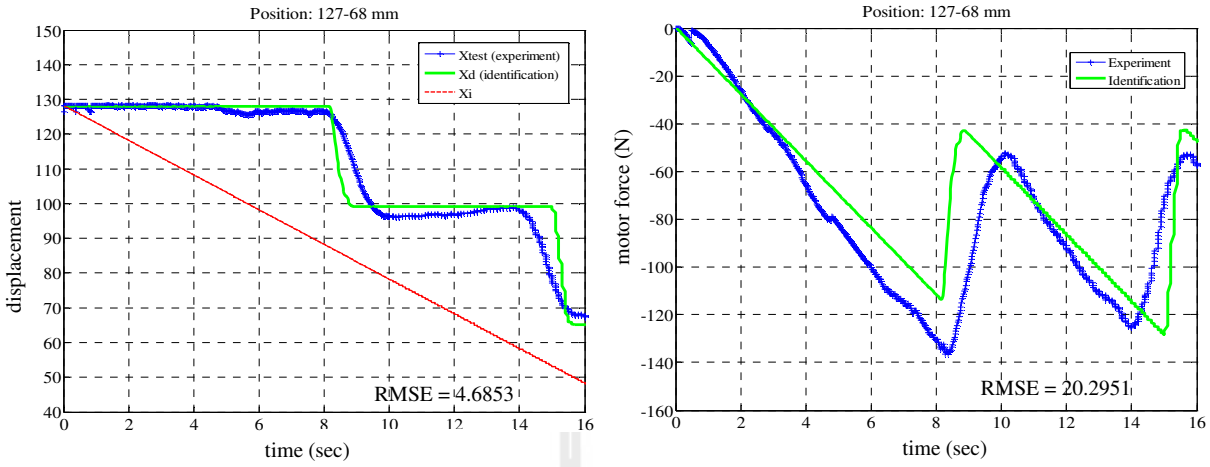
(c) (d)

Figure 7.27 Validation results of ramp-up command: (a) displacement (44-112 mm), (b) force exerted by motor (44-112 mm), (c) displacement (295-352 mm) and (d) force exerted by motor (295-352 mm).





(a) (b)



(c) (d)

Figure 7.28 Validation results of ramp-down command: (a) displacement (352-325-mm), (b) force exerted by motor (352-325 mm), (c) displacement (127-68 mm) and (d) force exerted by motor (127-68 mm).

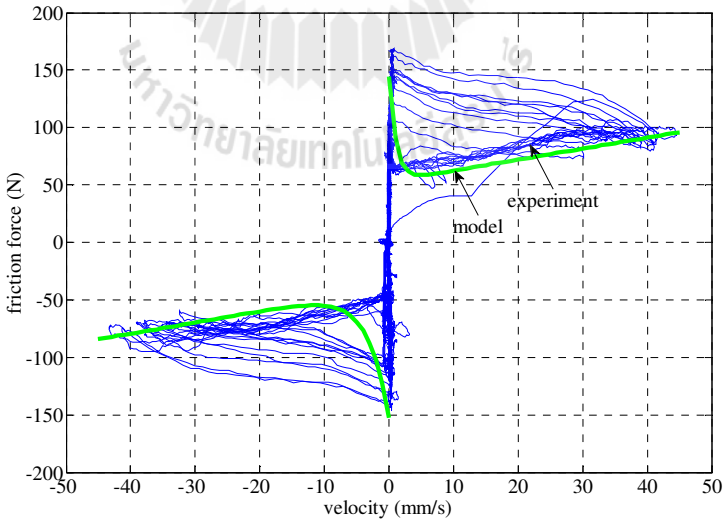


Figure 7.29 Plots of friction force curves (ramp command of ± 5 mm/s).

Furthermore, the friction curves based on model plots are shown against the experimental data in Figure 7.29. Very good agreement between the two can be observed. From the validation results, it can be said that the identified models provide a very good representation of the nonlinear friction forces.

7.6.2 Identification of Actuator Model in a Servo Track Writing System

Hard disk drives (HDDs) have been widely used to store information for today's computer systems. A key process of HDD manufacturing to access data is the servo track writing (STW) process, which provides servo patterns as references for the addresses of stored data. There are two critical control problems in STW process (Mamun et al., 2002). First, repeatable and non-repeatable run outs (RRO and NRRO) disturb the formation of the servo patterns. Second, the misalignment of servo sectors occurs to the servo sectors of adjacent tracks. To cure the problems, the self-servo writing (SSW) (Brown et. al, 2000 and Dong et. al., 2006) has been introduced instead of the conventional servo writing (Uematsu et al., 2001). The SSW process can be represented by the diagram in Figure 7.30. The read and write heads, and the SSW system are utilized to write servo patterns without using any external equipment. The heads with an arm and a voice-coil motor are regarded as a hard-disk head actuator set, or hard-disk actuator in brief. In normal servo writing process, the next track will be written the servo pattern based-on the read back signal from the read head. The subsequent track is built until the corresponding servo patterns are written on the whole disk. With the current SSW technology, some problems related to vibrations and disturbances still exist. In order to achieve better performances, engineers need to utilize more advanced concepts of control technology. The tasks need an appropriate

control model. One may use an accurately nonlinear model for the design. However, a simplified model can possibly be used in accordance with more advanced control concept such as learning control, robust control, intelligent control, and so forth.

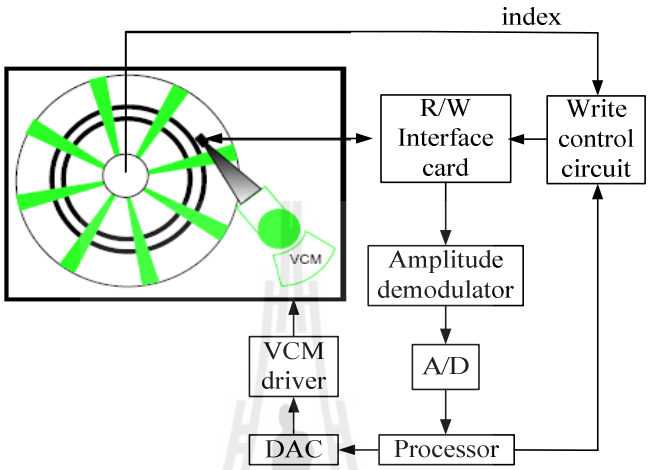


Figure 7.30 Block diagram of self-servo writing process.

The simplified block diagram in Figure 7.31 represents the closed-loop system of the self-servo writing technology (Brown et al., 2000; Dong et al., 2006), in which PES stands for position error signals. The repeatable and non-repeatable run outs are represented by the disturbances D_A and D_M, respectively.

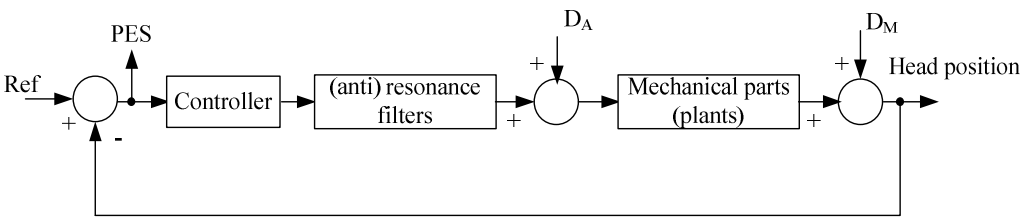


Figure 7.31 Block Diagram of hard-disk actuator.

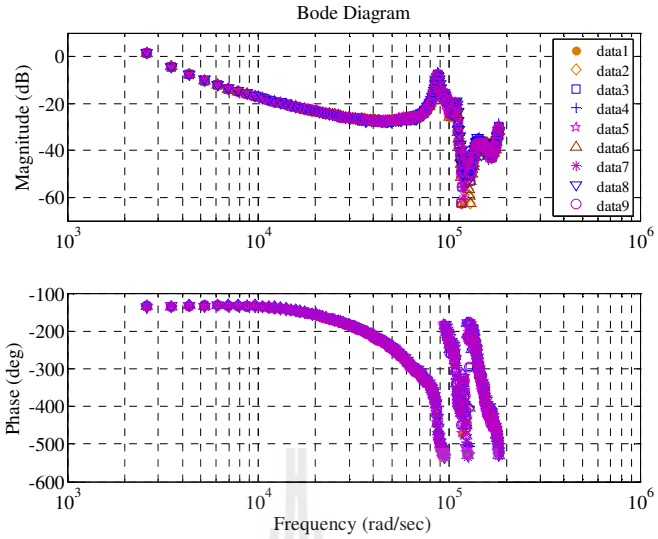


Figure 7.32 Frequency responses of the hard-disk head actuator (courtesy of Hitachi Global Storage, Prachinburi, Thailand).

Experiments were conducted from measurement at the plant situated in Prachinburi, Thailand, to collect data for identification purposes. The collected data appear as open-loop frequency responses shown in Figure 7.32, when the (anti) resonant filters were disconnected. It is notice that the actuator and arm dynamic in low frequency is dominated by the inertia that can be approximated by a double integrator. A torsional resonance mode clearly pronounces alongside with butterfly and sway modes of resonance (Li et al., 2003; Liu et al., 2003). Correspondingly, the *s*-domain model representing the dynamic can be expressed as

$$G_p(s) = \frac{K_a}{J_m s^2}\left(\frac{\omega_{n1}^2}{s^2 + 2\zeta_{n1}\omega_{n1}s + \omega_{n1}^2}\right)\left(\frac{s^2 + 2\zeta_{n3}\omega_{n3}s + \omega_{n3}^2}{s^2 + 2\zeta_{n2}\omega_{n2}s + \omega_{n2}^2}\right) \quad (7.27).$$

The order of the model is kept at minimum for simplification. The gain K is defined as $K = Ka/J_m$. The inertia of the actuator is known, $J_m = 6.11\times10^{-7}\,kg.m^2$, the rest of the parameters need identification. The data are separated into 2 groups, i.e. 60% for identification, and 40% for validation purposes.

The identification task is formulated as an optimization problem, where the BF-TS and the modified ATS metaheuristics are applied. The search spaces and search parameters for the algorithms appear in Tables 7.20 and 7.21, respectively.

Table 7.20 Boundaries and search parameters of the BF-TS for identification of the hard-disk head actuator.

Search parameters	S	N_c	N_s	α	R	N	$J_{min} \leq$	BT and n_re_back	$count_{max}$	AR
$K = [0 - 1]$, $\varsigma_{n1}, \varsigma_{n2}, \varsigma_{n3} = [0 - 0.5]$, $\omega_{n1} = [8\times10^4, -8.7\times10^4]$ $\omega_{n2}, \omega_{n3} = [1\times10^5 - 1.2\times10^5]$	30	20	4	0.001	50	50	1	5	500	if J<40 then R=25, if J<25 then R=5, if J<1.5 then R=0.5

Table 7.21 Search parameters of the modified ATS for identification of the hard-disk head actuator.

Search parameters	S	N_c	S_2	N_{c2}	N_s	α	α_2	$J_{min} \leq$	$count_{max}$
$K = [0 - 1]$, $\varsigma_{n1}, \varsigma_{n2}, \varsigma_{n3} = [0 - 0.5]$, $\omega_{n1} = [8\times10^4 \; -8.7\times10^4]$, $\omega_{n2}, \omega_{n3} = [1\times10^5 - 1.2\times10^5]$,	30	20	1	50	4	0.001	1000	1	500

The objective function, J representing a multi-objective requirement has a surrogate form as in (7.28). Calculation of the objective function during the search process follows the steps listed below. For more details refer to (Carawukh et al., 2011; Sarasiri et. al., 2011).

Function J = obj ($K, \omega_{n1}, \varsigma_{n1}, \omega_{n2}, \varsigma_{n2}, \omega_{n3}, \varsigma_{n3}$)

Step1: Heuristically assign initial values for damped natural frequencies (rad/sec) based on test data.

$$\omega_{d1} = 8.698\times10^4$$

$$\omega_{d2} = 1.087\times10^5$$

$$\omega_{d3} = 1.181\times10^5$$

Step2: Compute damped natural frequencies using natural frequencies and damping ratios proposed by search algorithms.

$$\omega_{d1\_iden} = \omega_{n1}\sqrt{1-\varsigma_{n1}^2}$$

$$\omega_{d2\_iden} = \omega_{n2}\sqrt{1-\varsigma_{n2}^2}$$

$$\omega_{d3\_iden} = \omega_{n3}\sqrt{1-\varsigma_{n3}^2}$$

Step3: Accept solutions if the errors are bounded within $\pm 5\%$, otherwise assign dummy values to the required solutions.

if $\omega_{d\_iden} = \omega_{d1} \pm 5\%$

$\omega_{d1\_iden} = \omega_{d1\_iden}$

else

$\omega_{d1\_iden} = 1\times 10^{10}$

end

if $\omega_{d2\_iden} = \omega_{d2} \pm 5\%$

$\omega_{d2\_iden} = \omega_{d2\_iden}$

else

$\omega_{d2\_iden} = 1\times 10^{10}$

end

if $\omega_{d3\_iden} = \omega_{d3} \pm 5\%$

$\omega_{d3\_iden} = \omega_{d3\_iden}$

else

$\omega_{d3\_iden} = 1\times 10^{10}$

end

Step4: Normalize ω_{d1_iden}, ω_{d2_iden} and ω_{d3_iden}.

$$\left.\begin{aligned} \omega_{d1\_n} &= \omega_{d1\_iden} / \omega_{d1} \\ \omega_{d2\_n} &= \omega_{d2\_iden} / \omega_{d2} \\ \omega_{d3\_n} &= \omega_{d3\_iden} / \omega_{d3} \end{aligned}\right\} \text{must be within [0.95-1.05]}$$

Step5: Compute root mean square (*RMS*) error values of gain (dB) and phase (deg).

$$\left.\begin{aligned} J_{gain} &= \sqrt{\frac{\sum_{i=1}^{n}\left(gain_{test} - gain_{iden}\right)^2}{n}} \\ J_{phase} &= \sqrt{\frac{\sum_{i=1}^{n}\left(phase_{test} - phase_{iden}\right)^2}{n}} \end{aligned}\right\} \text{must be less than 1}$$

where n = number of data points.

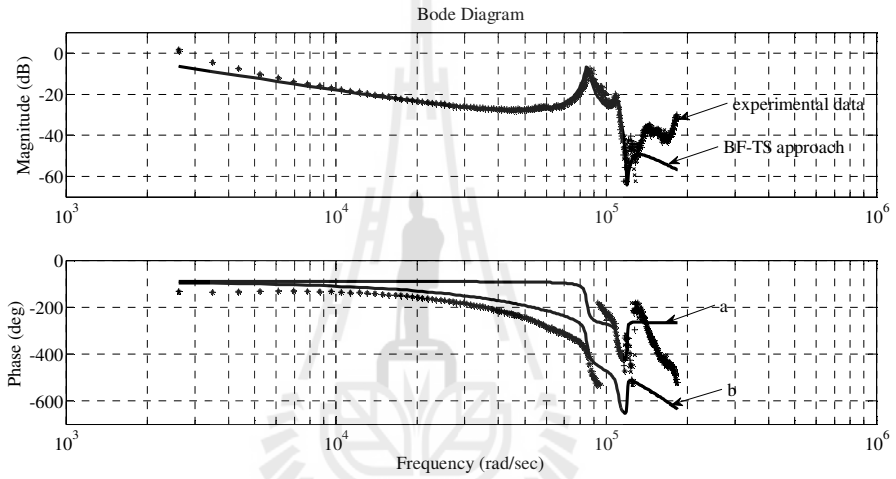
Step6: Compute the objective functions.

$$J = (0.2\times\omega_{d1\_n}) + (0.2\times\omega_{d2\_n}) + (0.2\times\omega_{d3\_n}) + (0.2\times J_{gain}) + (0.2\times J_{phase}) \tag{7.28}$$

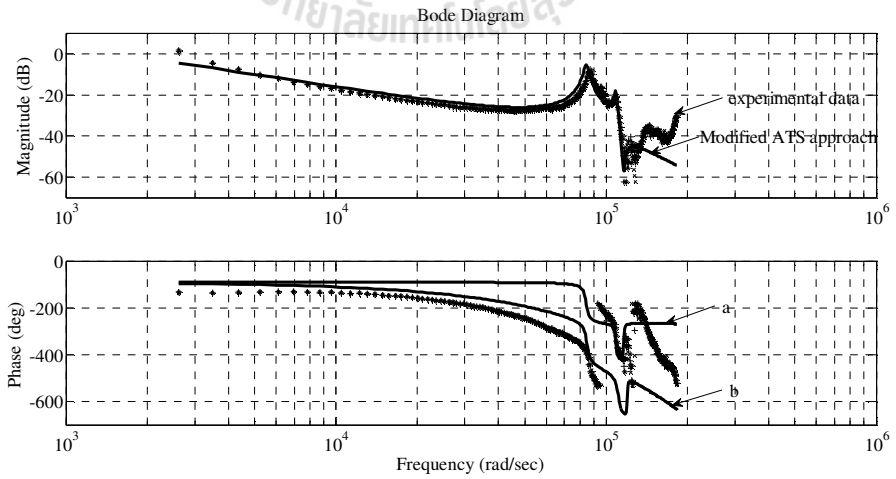
To accept the search results, the cost value must meet the minimum threshold. The damped natural frequencies, ω_d for each term of the *s*-domain model, are normalized. The J_{gain} and J_{phase} are computed as *RMS* errors of gain (dB) and phase (deg), which must be less than 1. The corresponding model parameters obtained from the BF-TS and the modified ATS approaches are summarized in Table 7.22. Their models plotted against the empirical data for validation are illustrated in Figure 7.33.

Table 7.22 Summary of parameters for the actuator model based on the BF-TS and the modified ATS algorithms.

Metaheuristic method	K	ς_{n1}	ω_{n1}	ς_{n2}	ω_{n2}	ς_{n3}	ω_{n3}	*RMS*
BF-TS	0.7673	0.0191	8.5006×10^{4}	0.0183	1.1087×10^{5}	0.0042	1.1984×10^{5}	30.8851
Modified ATS	1.00	0.0200	8.4416×10^{4}	0.0104	1.0863×10^{5}	0.0060	1.1574×10^{5}	30.7127



(a)



(b)

Figure 7.33 Plots of models against recorded data (validation).

Referring to Figure 7.33(a) and 7.33(b), the scattered data represent the 40% of the measured data used for validation. The solid lines represent the model plots obtained from the proposed metaheuristic approaches. There are 2 curves of each approach designated as “a” and “b”. Curve “a” has a rather flat phase characteristic. Curve “b”, when a practical delay of 40 micro-seconds is taken into account, better agrees with the measured phase data. This delay accounts for the processes of data sampling and conversion, control action, and data recording. Both curves (in Figure 7.33(a) and 7.33(b)) show considerable deviations from the measured phase data in high frequency range, where complex (anti) resonances presence. This is probably due to nonlinearity of the real actuator. Noticeably, both search approaches result in parametric models of similar quality, while they hit the maximum number of iteration, i.e. the stop criterion based-on the cost value is never satisfied. Nonetheless, by eye observation, the models of curves “a” for magnitude and phase characteristics are accepted with the cost values of 30.8851 and 30.7127, respectively. Figure 7.34 depicts the comparison of the cost values monitored through out the search.

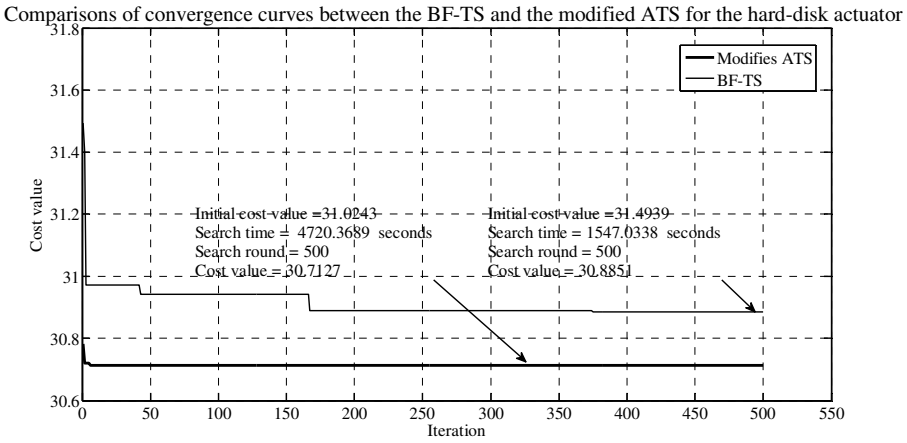


Figure 7.34 Comparisons of convergence curves between the BF-TS and the modified ATS for the hard-disk head actuator.

In addition to this, the modified ATS consumes 3 times longer search time than the BF-TS does, while it converges faster without encountering any local lock. The BF-TS is trapped by local dead locks of 49 times.

7.7 Multi-objective Design Optimization of a Permanent Magnet Synchronous Motor Drive

In recent years, permanent magnet synchronous motors (PMSMs) have been applied for high performance variable speed drive to increase efficiency. The popularity of PMSMs compared with other kinds of motors is due to their desirable features of high torque to inertia ratio, high air gap flux density, high power factor, high acceleration and deceleration rates, lower maintenance cost, simplicity and ruggedness. Replacement of the field winding by a permanent magnet eliminates field copper losses (Koerner et al., 2005). The ideal back-EMF induced in this kind of machines is sinusoidal. As a consequence of the sinusoidal back-EMF and the sinusoidal distribution of the windings, a PMSM produces smooth torque. In addition, constant torque with very low ripple is produced when synchronous sinusoidal currents flow through the stator windings.

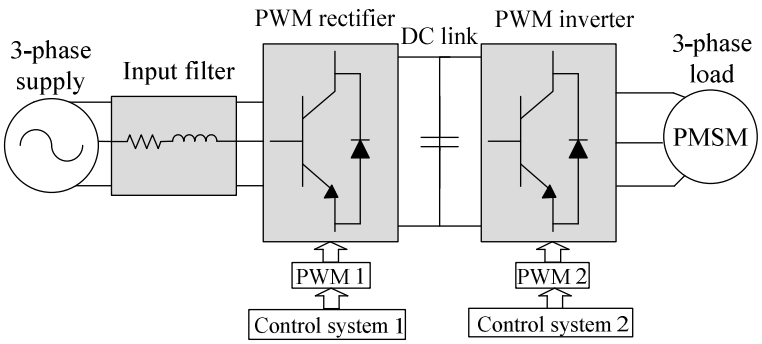


Figure 7.35 Equivalent system of back-to-back converter topology.

One type of well-known drives systems is a back-to-back power converter topology, which has an important role in a wide variety of industrial processes and technologies (Ackermann and Soder, 2002; Noroozian et al., 2003; Rodriguez et al., 2005; Torres et al., 2004; Zhanoqing et al., 2004). The converter structure has some useful characteristics such as the operating capability with sinusoidal input current, etc. The DC-link voltage can be equal or higher than the peak input voltage which is regulated by controlling the power flow to the AC source (Malesani et al., 1995). This topology consists of two three-phase PWM converters with a common DC-link, which decouples the converters as shown an equivalent system in Figure 7.35. One converter acts as a rectifier, and another one as an inverter. Both need separate control systems.

Regarding to the design, input and output current qualities, total power losses, DC-link voltage control, current control loops and motor speed control are taken into account. Conventional designs maybe not cover all of those requirements. Thus, this research has used metaheuristics to obtain several optimum solutions at the same time. The task of finding solutions for such a kind of multiple objective problems is based-on multi-objective optimization. This proposed engineering problem has been carried out in the Department of Electrical and Electronic Engineering, the University of Nottingham, United Kingdom during 1st September 2011 to 29th February 2012 under the supervision of Prof. Dr. Pericle Zanchetta by a financial support from The Thailand Research Fund. The research interest is the applications of metaheuristic algorithms to complex engineering problems. Accordingly, Prof. Dr. Zanchetta assigned the topic "*Multi-objective Design Optimization of a Permanent Magnet Synchronous Motor (PMSM) Drive*", where power converter (back-to-back converter) and electrical machine are considered as a whole system. This research relates to the study of

structure of the back-to-back converter, control loops design based-on *dq* analysis, power losses in power converters and electrical machine and implementation of metaheuristic algorithms. The mathematical models of the three-phase PMSM within the *abc* frame need their parameters to be known are described in Section D.1, Appendix D. To obtain these, the MotorSolve software package (available at the University of Nottingham, UK) was used (detailed in Section D.2, Appendix D).

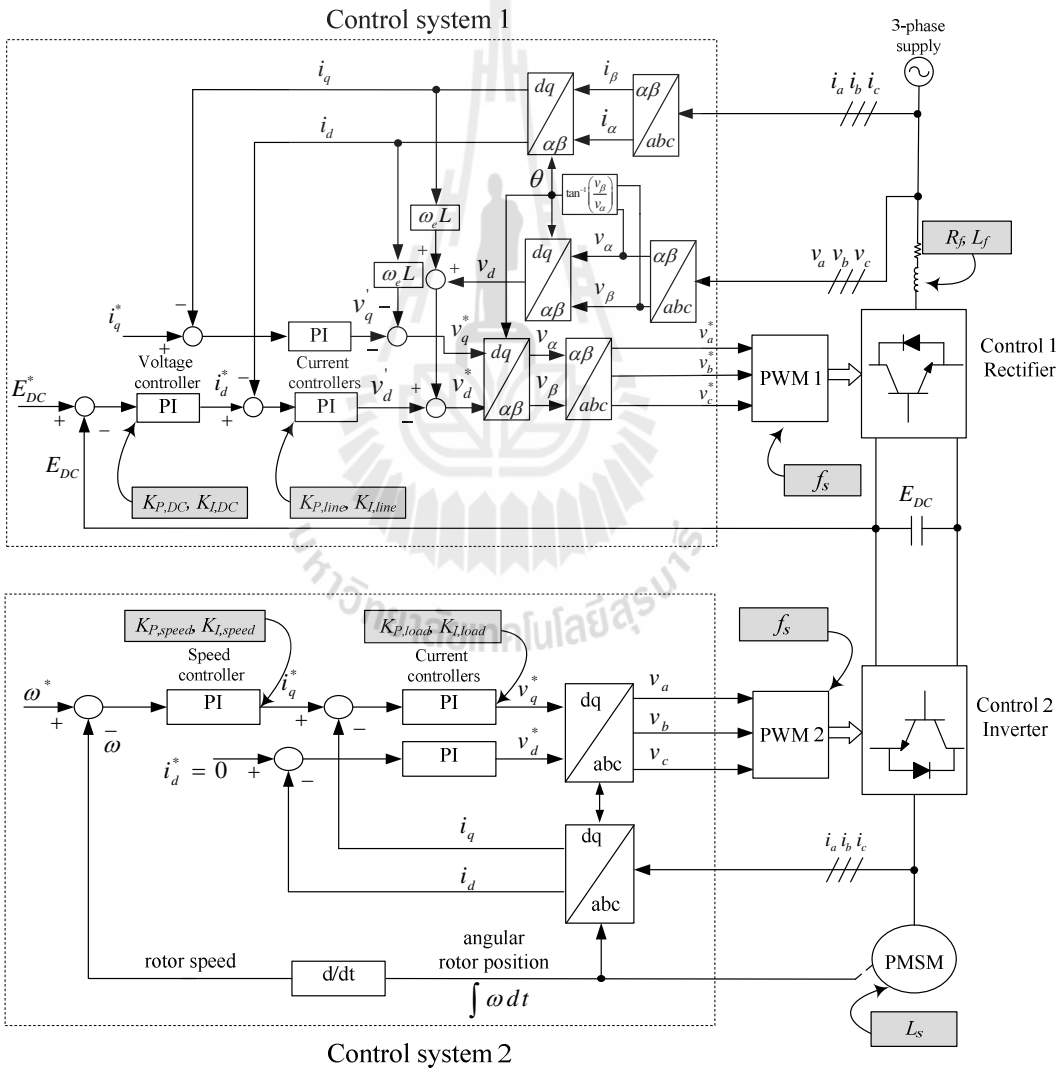


Figure 7.36 A whole control structure based-on the *dq* reference frame of back-to-back converter (Wu et al., 2008).

For the drive to work properly, effective control of both converters are necessary. Figure 7.36 illustrates the block diagrams representing 2 separate closed-loop control systems, one for each converter. The control concept is based-on the *dq* frame. The first control system or control system 1 is for the PWM rectifier, and the second one or control system 2 is for the PWM inverter. Control system 1 consists of two internal control loops for the currents i_d and i_q, and one external control loop for the DC-link voltage, E_{DC}. The purpose of such control systems is to transform the reference voltages into a pulse width modulation pattern for the operation of the rectifier, and to regulate the E_{DC}. Similarly, the control system 2 in Figure 7.36 consists of two internal control loops for the currents i_d and i_q, and one external speed control loop, ω. Such control systems are to transform the reference voltages into a pulse width modulation pattern for gating the inverter, and to govern the motor speed, ω. For more details of the concepts related to the back-to-back converter, input filter design, the *dq* analysis of the PWM rectifier and the PMSM in order to design controllers are explained in Sections D.3-D.5 in Appendix D. The control systems utilize the PI-controller structures which require a very extensive design. Driving the design phase, several technical issues of the converter have to be considered simultaneously. These include the input and output current qualities, the regulated DC link voltage, losses of the converter and the motor, and so on. Therefore, the proposed metaheuristics are applied to obtain the optimum parameters for the design. The implemented objective functions, discussions of the results obtained by the proposed BF-TS algorithm and the modified ATS, and the conventional design are detailed in the next section.

7.7.1 Multi-objective Design Optimization

The thesis aims to apply the proposed metaheuristics to optimize the design of a complicated power drive system. Detailed explanation of the physical system has been given so far in the Section 7.7 and Appendix D. the formulation of all objective functions are revealed in details also in the Appendix D. Multi-objective designs within this Section include the analysis of the input and the output current qualities, the losses both in the converters and the motor, the DC-link voltage, the current control loops, and the motor speed control. These factors need to be considered carefully and synthetically during the design phase. The previous conventional designs have not covered all of those requirements, thus, the optimal solutions of the back-to back converter can be found by metaheuristic methods in order to obtain several optimum solutions at the same time. The task of finding solutions for such a kind of multiple objective problems is known as multi-objective optimization. Generally, multi-objective optimization is not limited to find a unique single solution but a set of solutions called nondominated solutions. Each solution in this set is referred to as a Pareto optimum (Alba, 2005).

There are 12 parameters considered for the back-to-back converter. These are the input filters, R_f and L_f; the stator inductance of motor; L_s, the switching frequency, f_s; the input current controller, $K_{P,line}$ and $K_{I,line}$; the DC-link voltage controller, $K_{P,DC}$ and $K_{I,DC}$; the load current controller, $K_{P,load}$ and $K_{I,load}$; and the speed controller, $K_{P,speed}$ and $K_{I,speed}$. In this particular problem, the objective functions are considered as normalized values. For more detailed descriptions of these functions appear in Section D.6, Appendix D. Each objective function is expressed as follows:

$$f_1(R_f, L_f) = \frac{THD_i, input}{3\%} \tag{7.29}$$

$$f_2(L_s) = \frac{THD_i, motor}{3\%} \tag{7.30}$$

$$f_3(f_s) = \frac{P_{loss,line} + P_{loss,SC} + P_{loss,motor}}{P_{accept,\max}} = \frac{P_{total,loss}}{P_{accept,\max}} \tag{7.31}$$

$$f_4(K_{P,line},\ K_{I,line}) = \frac{RMS_{rectifier}}{RMS_{rectifier,\max}} \tag{7.32}$$

$$f_5(K_{P,DC},\ K_{I,DC}) = \frac{RMS_{DC}}{RMS_{DC,\max}} \tag{7.33}$$

$$f_6(K_{P,load},\ K_{I,load}) = \frac{RMS_{inverter}}{RMS_{inverter,\max}} \tag{7.34}$$

$$f_7(K_{P,speed},\ K_{I,speed}) = \frac{RMS_{speed}}{RMS_{speed,\max}} \tag{7.35}.$$

The formulation of a multi-objective function can be written in terms of combined objective functions with the same priority rational weights as equation (7.36). This is also known as “surrogated form” (Zakian, 2005). Each objective function is weighted equally, and expressed in p.u. The detailed explanation can be found in Appendix D.

$$f_{all} = \frac{1}{7}\left(f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7\right) \tag{7.36}$$

where f_1 and f_2 are functions of the input and output current qualities to obtain R_f, L_f and L_s; f_3 is a function of the total power losses in a 10-pole PMSM to obtain f_s; f_4 is a function of the line input current controller to acquire $K_{P,line}$ and $K_{I,line}$; f_5 is a function of the DC-link voltage controller to return $K_{P,DC}$ and $K_{I,DC}$; f_6 is a function of the load current controller to acquire $K_{P,load}$ and $K_{I,load}$, and f_7 is a function of the speed controller to obtain $K_{P,speed}$ and $K_{I,speed}$.

The penalty functions are applied to the multi-objective optimizations (in equation (7.29)) as shown by the program list in Figure 7.37.

Objective function: $f_{all} = \frac{1}{7}\left(f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7\right)$;

Inequality constraints: InEq=[$f_1, f_2, f_3, f_4, f_5, f_6, f_7$ *];*

Sum square (SS) = 0; $\sigma = 1 \times 10^6$;

```
for     k=1:length(InEq)
        if      InEq(k) ≤ 1
                InEq(k) = 0;
        end
        SS=SS+σ * InEq(k)^2;
end
        J_all = f_all + SS;
```

Figure 7.37 Program lists of the multi-objective optimizations.

The objective function J_{all} will be optimized by using the BF-TS and the modified ATS algorithms for comparison purposes. The boundaries and search parameters of the BF-TS and the modified ATS algorithms are presented in Table 7.23 and Table 7.24, respectively.

Table 7.23 Boundaries and search parameters of the BF-TS for multi-objective design optimization of a PMSM drive.

Search parameters	S	N_C	N_S	α	N	$J_{\min} \leq$	BT and n_re_back	$count_{max}$
R_f=[0.01-0.5], L_f, L_S=[1×10^{-3}-10×10^{-3}] f_S=[1×10^{3}-20×10^{3}], $K_{P,line}$, $K_{P,load}$=[1-100], $K_{I,line}$, $K_{I,load}$=[1×10^{3} – 1×10^{5}], $K_{P,DC}$, $K_{P,speed}$=[0.1 – 100], $K_{I,DC}$, $K_{I,speed}$=[10 – 500],	30	20	4	1×10^{3}	20	0.35	5	500

The search radius set of the BF-TS approach has been used as R = [0.1425, 1.35×10^{-3}, 1.35×10^{-3}, 2850, 14.85, 2.10×10^{4}, 14.985, 73.50, 14.85, 2.10×10^{4}, 14.985, 73.50]; the adaptive search radius are utilized as follows:

if *best_error* < 2.5 then R = [0.0735, 3.375×10^{-4}, 3.375×10^{-4}, 712.50, 3.7125, 5250, 3.75, 18.375, 3.7125, 5250, 3.75, 18.375].

if *best_error* < 1.5 then R = [0.0184, 8.440×10^{-5}, 8.440×10^{-5}, 178.125, 0.928, 1312.50, 0.94, 4.59, 0.928, 1312.50, 0.94, 4.59].

if *best_error* < 0.5 then R= [0.0045, 2.110×10^{-5}, 2.110×10^{-5}, 44.53, 0.2320, 328.125, 0.23, 1.15, 0.232, 328.125, 0.23, 1.15].

Table 7.24 Boundaries and search parameters of the modified ATS for multi-objective design optimization of a PMSM drive.

Search parameters	S	N_c	S_2	N_{C2}	N_S	α	α_2	J_{min} ≤	$count_{max}$
R_f =[0.01-0.5], L_f, L_S=[1×10^{-3}-10×10^{-3}] f_S=[1×10^{3}-20×10^{3}], $K_{P,line}$, $K_{P,load}$=[1-100], $K_{I,line}$, $K_{I,load}$=[1×10^{3} – 1×10^{5}], $K_{P,DC}$, $K_{P,speed}$=[0.1 – 100], $K_{I,DC}$, $K_{I,speed}$=[10 – 500],	30	20	1	20	4	1×10^{3}	1×10^{5}	0.35	500

7.7.2 Results and Discussions

This section presents the results obtained from the conventional design method in comparison with those obtained from the metaheuristic search. Simulation results shown in this section are obtained from using MATLAB/SIMULINK package. The simulation parameters of the back-to-back converter are shown in Table 7.25, and the PMSM parameters are utilized as Table D.1 in Appendix D. Components and parameters of the filter and the controllers for the drive are summarized in Table 7.26.

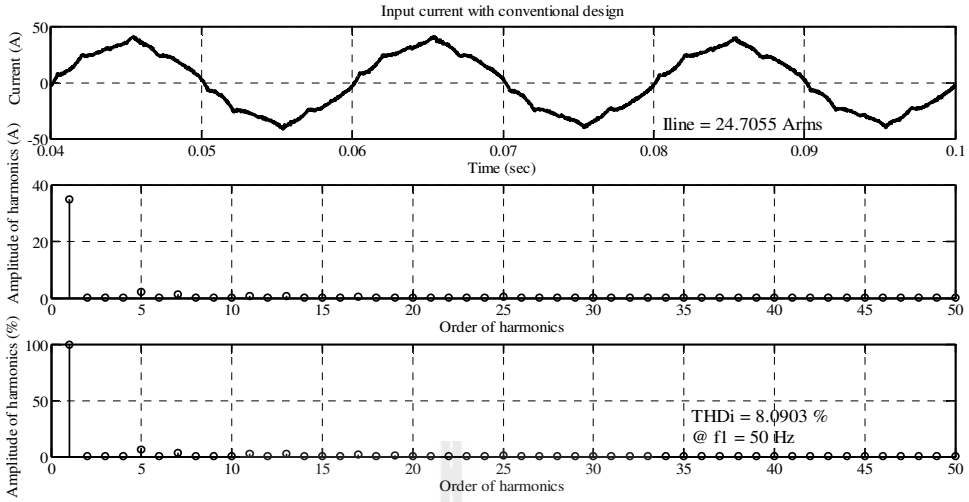
Table 7.25 Parameters of the back-to-back converter for simulation.

Parameters	Values
Three-phase voltage source	240 V_{phase}
DC voltage	1000 V_{DC}
Line impedance: R_{line} and L_{line}	0.05 Ω and 0.10 mH
Input filter: R_f and L_f	0.13 Ω and 3.30 mH
Switching frequency (f_s)	20 kHz

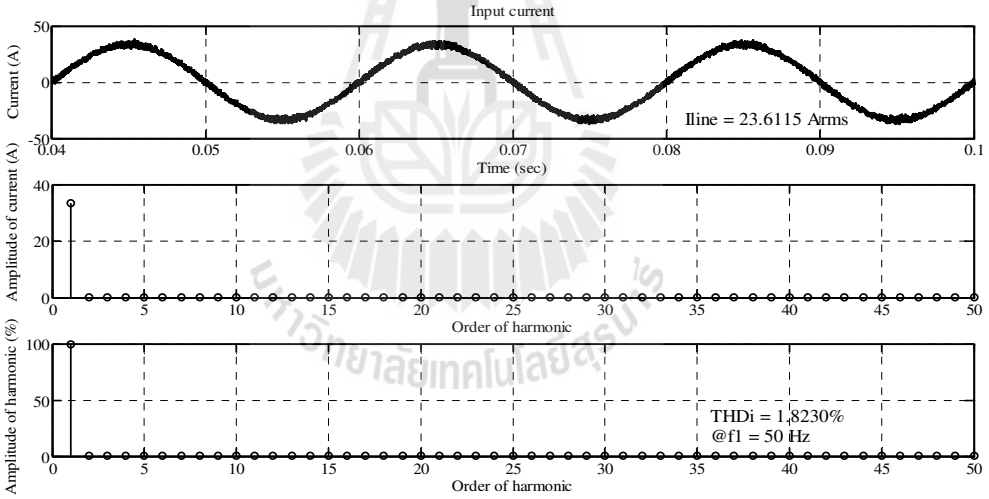
Table 7.26 Summary of the filter components and the controller parameters obtained from searches and conventional designs.

Controllers		Conventional designs	BF-TS	Modified ATS
Input filter:	$R_f(\Omega)$	0.13	0.05	0.015
	L_f (mH)	3.30	1.60	1.58
Stator inductance:	L_s (mH)	1.20	1.21	1.19
Switching frequency:	f_s (kHz)	20	9.71	10.60
Line input current controller:	$K_{P,line}$	16.89	49.87	43.39
	$K_{I,line}$	3.36×10^4	5.41×10^4	6.32×10^4
DC-link voltage controller:	$K_{P,DC}$	0.74	3.54	3.66
	$K_{I,DC}$	73.11	353.02	486.44
Load current controller:	$K_{P,load}$	2.29	36.95	33.31
	$K_{I,load}$	1.89×10^3	1.26×10^3	2.12×10^3
Speed controller:	$K_{P,speed}$	0.21	7.28	7.33
	$K_{I,speed}$	9.05	49.23	57.88

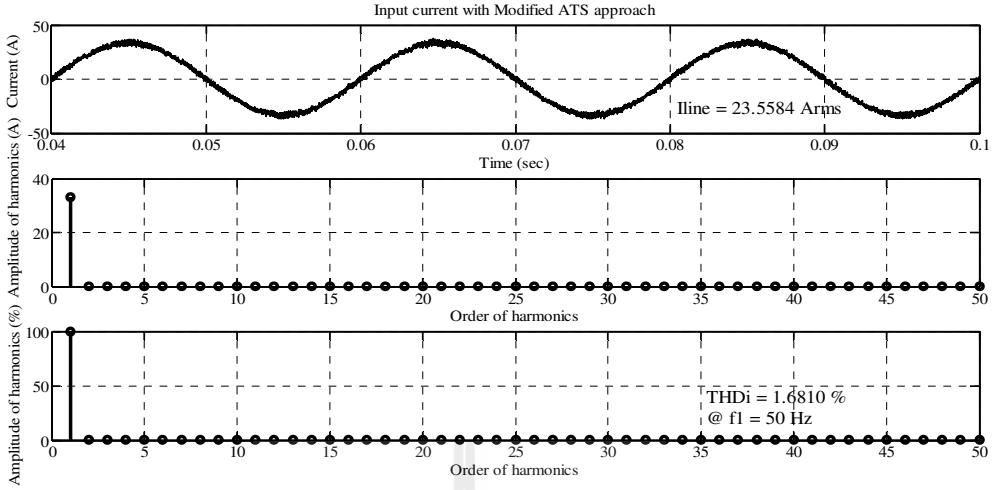
Figures 7.38-7.44 illustrate the simulation results of the input currents, the motor currents, the closed-loop responses of line input current, DC-link voltage responses, the closed-loop responses of motor current and the motor speeds, respectively, which are performed by MATLAB/SIMULINK.


Input current with conventional design
Current (A)
Iline = 24.7055 Arms
Time (sec)
Amplitude of harmonics (A)
Order of harmonics
Amplitude of harmonics (%)
THDi = 8.0903 %
@ f1 = 50 Hz
Order of harmonics

(a)

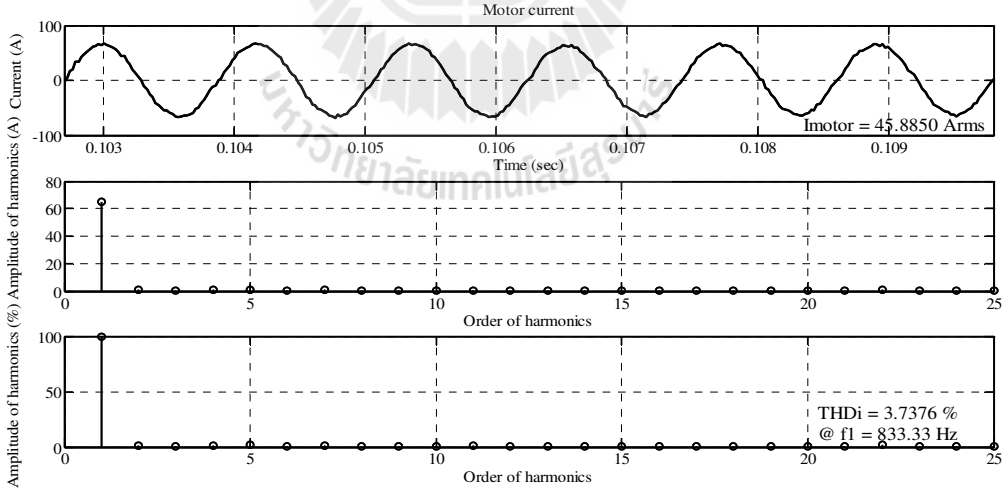

Input current
Current (A)
Iline = 23.6115 Arms
Time (sec)
Amplitude of current (A)
Order of harmonic
Amplitude of harmonic (%)
THDi = 1.8230%
@f1 = 50 Hz
Order of harmonic

(b)

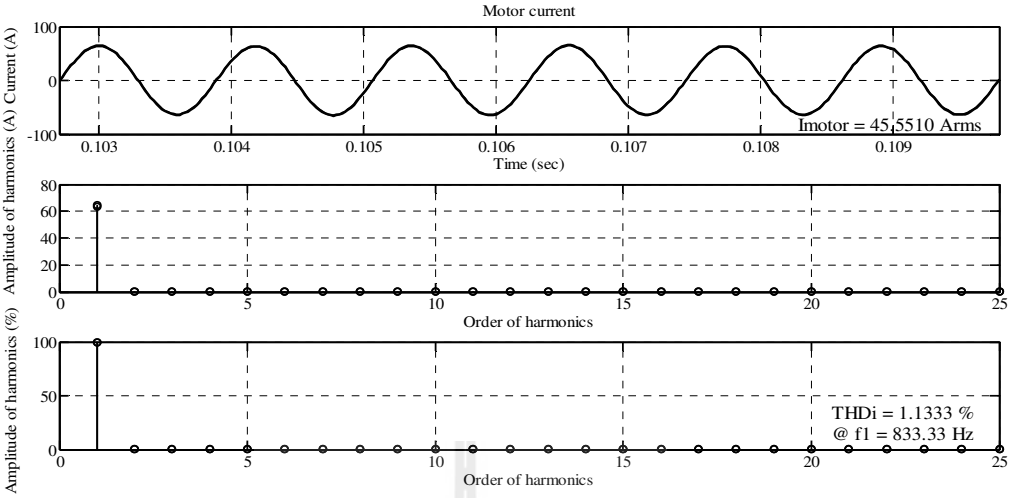


(c)

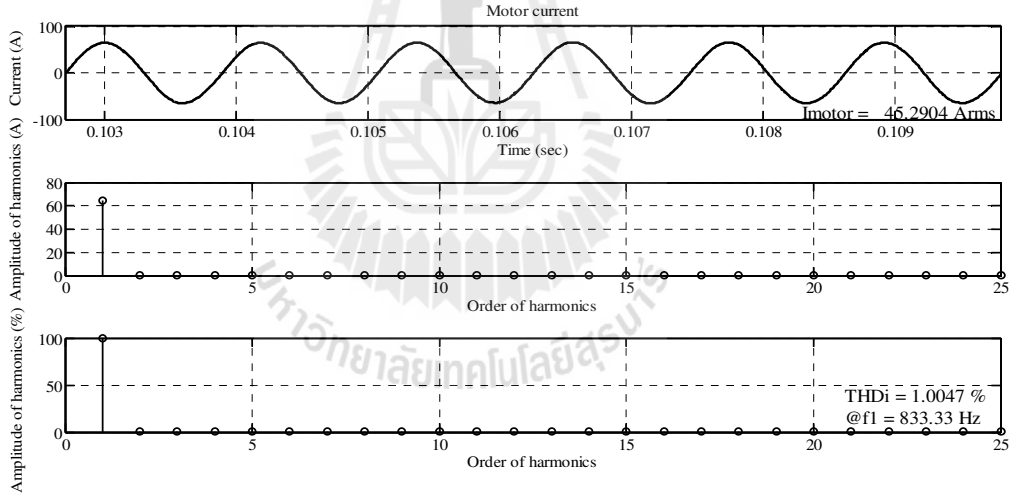
Figure 7.38 Comparisons of the input current qualities: (a) conventional design, (b) BF-TS approach and (c) modified ATS approach.



(a)



(b)



(c)

Figure 7.39 Comparisons of the motor current qualities: (a) conventional design, (b) BF-TS approach and (c) modified ATS approach.

Figure 7.38 shows the input current qualities, where the conventional design as in Figure 7.38(a) provides $THD_{i,input}$ = 8.0903%, whereas the BF-TS and the modified ATS approaches render $THD_{i,input}$ only 1.8230% and 1.6810% as in Figures 7.38(b) and 7.38(c), respectively. Similarly, for the motor current qualities in Figure 7.39, the BF-TS and the modified ATS algorithms (see Figures 7.39(a)-(b)) can provide better results than the conventional design by 3.7376%, where the $THD_{i,motor}$ values are 1.1333%, and 1.0047% for the BF-TS and the modified ATS, respectively.

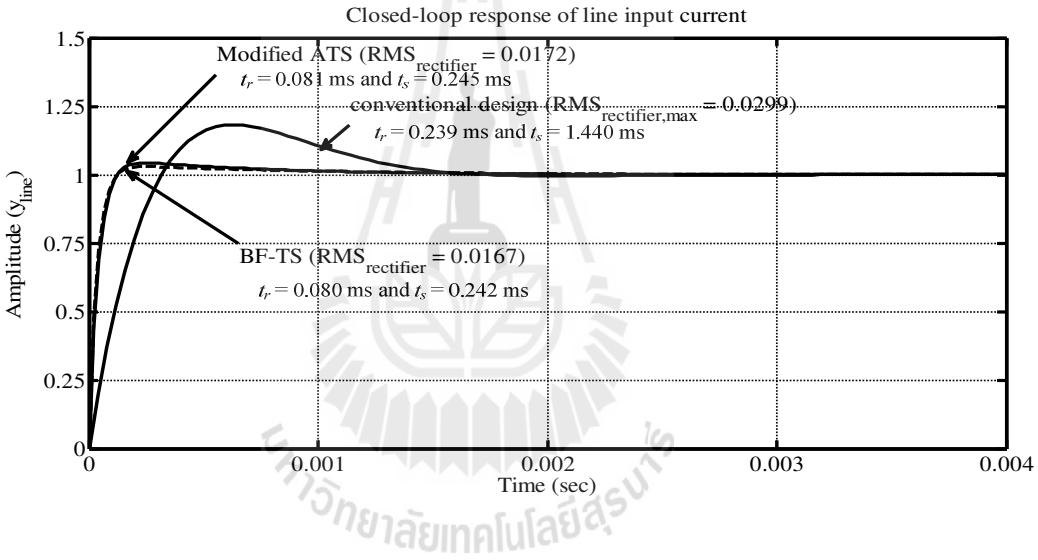


Figure 7.40 Comparison of the closed-loop responses of line input current.

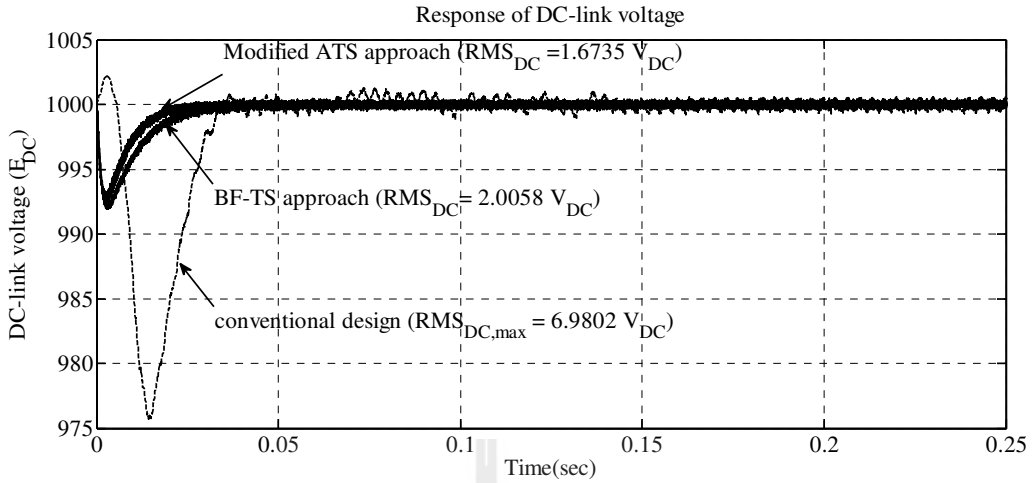


Figure 7.41 Comparison of the DC-link voltage responses.

Regarding the PWM rectifier part of the back-to-back converter, Figures 7.40 and 7.41 show comparisons of the closed-loop responses of line input current and the responses of DC-link voltage. The results of the BF-TS and the modified ATS approaches for line input current provide better rise times and settling times, achieving RMS errors only 0.0167 and 0.0172, respectively. Considering the conventional design, the RMS error obtained is 0.0299. Similarly, the results obtained by the proposed algorithms for DC-link voltage responses present better voltage regulations around 1,000 V_{DC} with fewer ripple voltages than the conventional design, where RMS errors are 2.0058 V_{DC} and 1.6735 V_{DC} for the BF-TS and modified ATS approaches, respectively. Note that the DC-link voltage responses have been considered in steady-state operation.

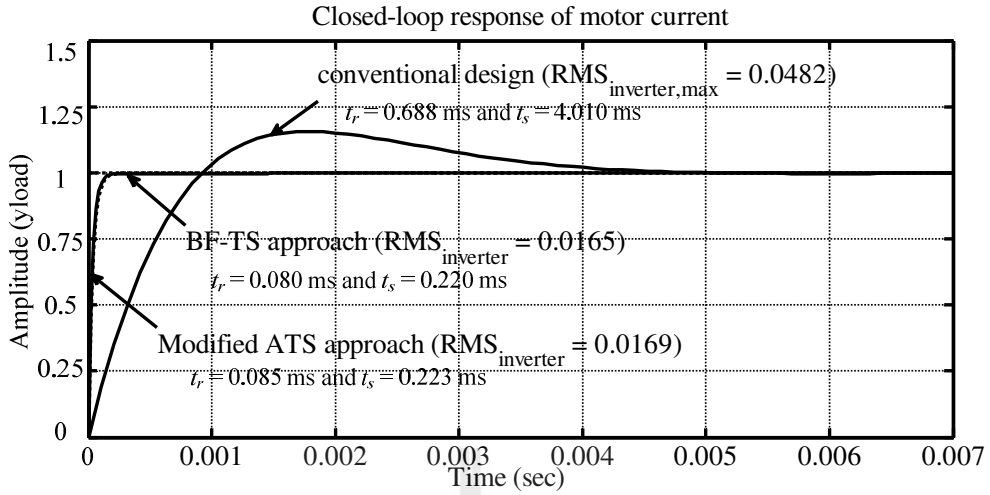


Figure 7.42 Comparison of the closed-loop responses of motor current.

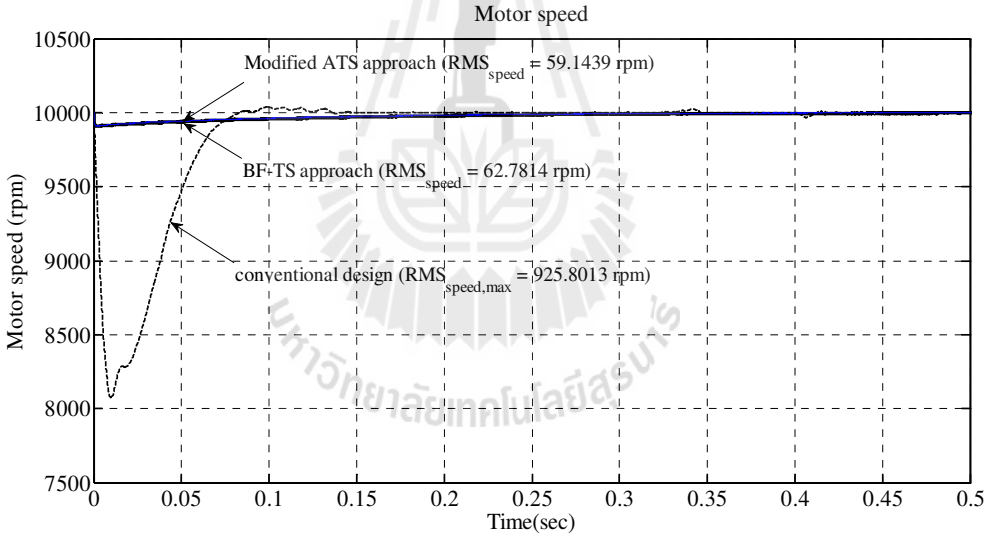


Figure 7.43 Comparison of the motor speeds.

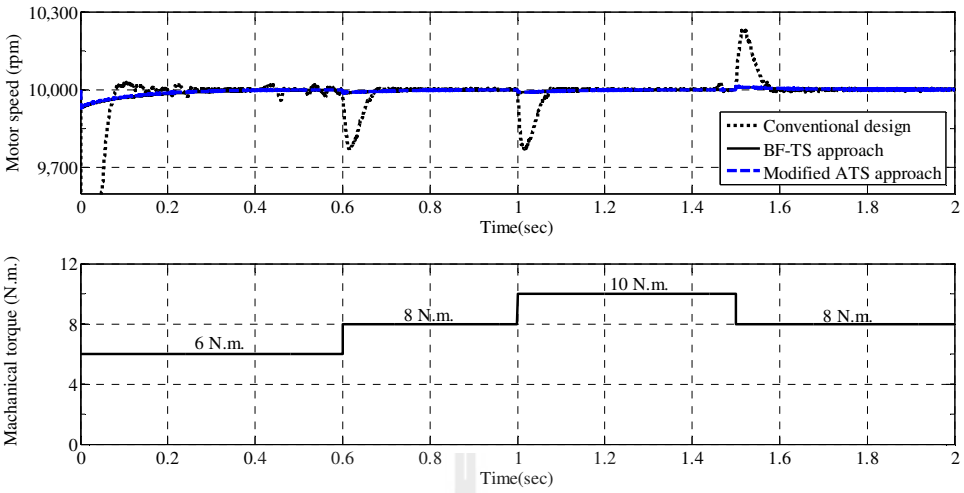


Figure 7.44 Comparison of the motor speeds with variable load torques.

For the designs of the PMSM part, the controller designs of motor current and the motor speed are considered. Figure 7.42 has presented the comparison results between the conventional design and the proposed approaches for the closed-loop responses of motor current. The results of the BF-TS and the modified ATS approaches illustrate similar performances, which provide better rise time, settling time and without overshoot achieving RMS errors equal 0.0165 and 0.0169, respectively. The conventional design provides the RMS error of 0.0482. Similarly, the results obtained by the proposed algorithms for the motor speed provide faster responses with small RMS errors of 62.7814 and 59.1439 rpm for the BF-TS and the modified ATS, respectively, while the conventional one contains RMS error of 925.8013 rpm (see Figure 7.43). The controllers obtained from the proposed algorithms can provide good responses of speed regulations, when the load torques are gradually changed as seen in Figure 7.44. The BF-TS and the modified ATS provide the results of similar quality.

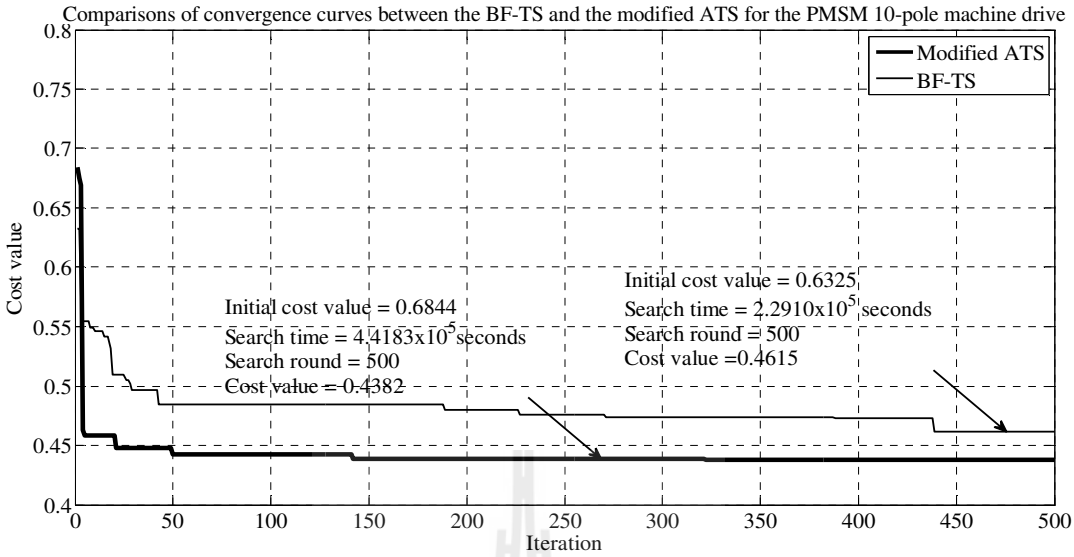


Figure 7.45 Comparisons of convergence curves of the proposed algorithms.

From the convergence curves in Figure 7.45, it can be noticed that the proposed algorithms provide good qualities of initial solutions due to the operation of random walk front-end found in the initial process. In order to obtain the best final solution, this work has defined a very stringent termination criterion of cost value. According to the results, the terminal criteria of multi-objective functions reach the maximum iterations with their cost values, J_{all}, equal to 0.4615 and 0.4382 for the BF-TS and the modified ATS, respectively. Noticeable, all constraint conditions meet the requirements at the beginning of search. As a matter of fact, the search results by the modified ATS can be accepted at the 150^{th} iteration. Although, the BF-TS provides a better initial solution, the modified ATS can converge to the global region faster than the BF-TS does. Meanwhile, the modified ATS spends twice as much search time than the BF-TS approach. Considering the local traps, the BF-TS algorithm encountered local

deadlocks for 55 times, while the modified ATS approach fell into the local deadlocks only 17 times.

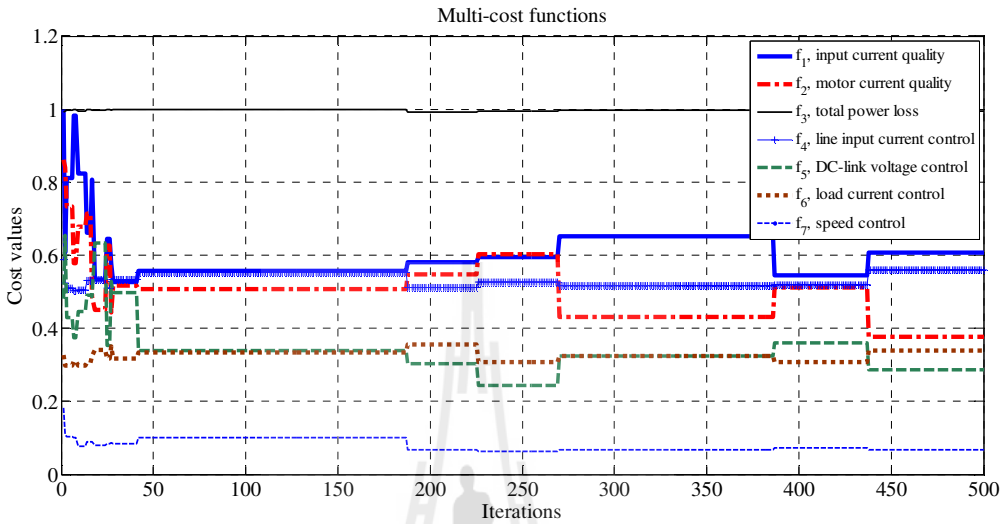


Figure 7.46 Curves of multi-cost functions for the BF-TS algorithms.

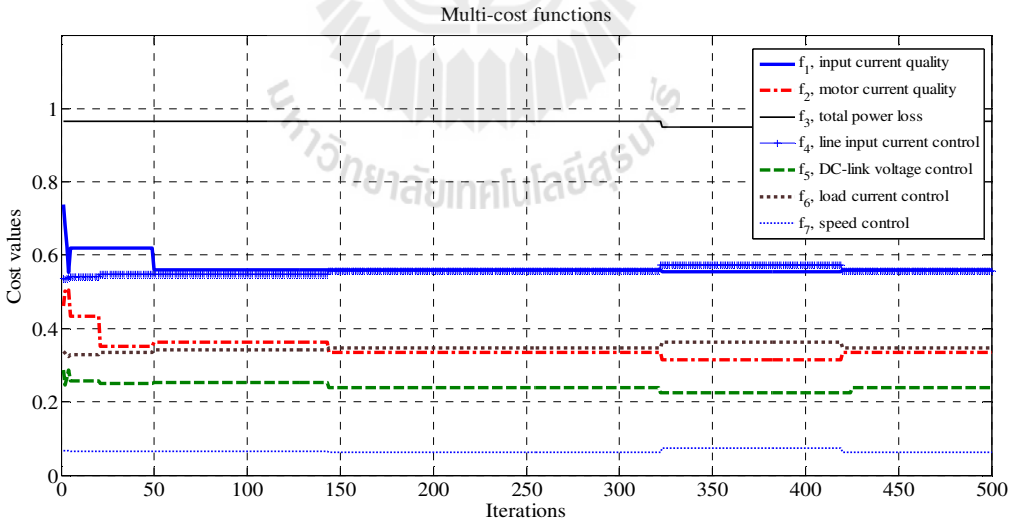


Figure 7.47 Curves of multi-cost functions for the modified ATS algorithms.

As mentioned earlier, each objective function is calculated as a normalized value. This means that the maximum value of its cost should be less than 1. From the results in Figures 7.46 and 7.47, all the objective functions can meet the requirements. Noted that power losses in the drive system are unavoidable due to using switching devices. In this case, the modified ATS being proposed can improve the total power losses, as indicated by f_3 being less than 1 in Figure 7.47. The final cost values obtained from the BF-TS and the modified ATS algorithms are summarized in Table 7.27.

Table 7.27 Summary of the final cost values for each objective function obtained from the BF-TS and the modified ATS algorithms.

Methods	f_1	f_2	f_3	f_4	f_5	f_6	f_7
BF-TS algorithm	0.6077	0.3778	0.9934	0.5591	0.2874	0.3375	0.0678
Modified ATS algorithm	0.5604	0.3349	0.9652	0.5569	0.2398	0.3461	0.0639

As a result comparing with the conventional designs, the proposed metaheuristic algorithms provide better solutions for all objective functions, and the modified ATS is superior.

7.8 Conclusion

Since the proposed BF-TS and the modified ATS algorithms provide outstanding performances, they are applied to some constrained optimization problems. These problems include three abstract mathematical constraint problems, optimal control design of hard-disk heads (single-head and head-stacks), optimal control design

of a truck braking system, stability analysis of nonlinear systems, identification problems of hard-disk head actuator and nonlinear Stribeck friction model, and a power drive system. The results obtained from the proposed metaheuristics were compared with those obtained from the conventional designs. It has been found that the proposed metaheuristics provide superior results in all cases with less complexity.

Concerning with the comparisons between the BF-TS and the modified ATS algorithms, the modified ATS converges to the global solution faster than the BF-TS does in most cases. The search trajectories formed by the modified ATS also encountered a smaller number of local deadlocks, and consume a smaller number of search rounds. Search time per iteration consumed by the modified ATS is longer than that by the BF-TS due to the operation of its random walk front-end, albeit. This drawback can be overcome by parallelization of the algorithms that will be explained in next chapter.

CHAPTER VIII

PARALLELIZATION AND APPLICATIONS

8.1 Introduction

As the size of optimization problems to be solved becomes larger, computing tasks demand a large amount of resources, and consume a long time to complete the solutions. Parallel computation is one of effective techniques to address these difficulties. This parallel or distributed computing can take the advantages of multicore and multiprocessor computers in order to use their full computational power. Many parallel implementations for different search algorithms have been successfully applied in various publications to effectively decrease computing time.

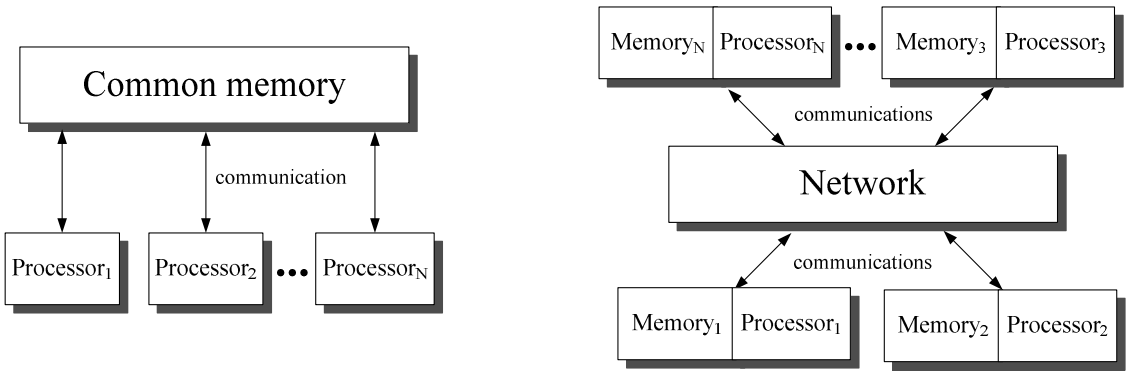
From the previous Chapter, the modified ATS algorithm can provide solutions of high quality with guaranteed convergence. But, it consumes long search time for most applications, especially for the power drive system. In this Chapter, we consider to use parallelization to reduce computing time for a particular power drive problem. Therefore, this Chapter begins Section 8.2 with a brief overview of parallelization including parallel processing (computing), parallel programing along with parallel metaheuristics. The term *parallel* used in this Chapter refers to "simultaneous or concurrent execution of individual tasks being done at the same time" (Bentley, 2000). Subsequently, parallel computing in MATLAB and its significant functions as well as given examples are presented in Section 8.3. In Section 8.4, the implementation of

parallel computing in MATLAB to the modified ATS algorithm for the proposed application is presented, and Section 8.5 conclusions the Chapter.

8.2 Definitions of Parallelization

8.2.1 Parallel Processing or Parallel Computing

Large-scale simulation and data processing task such as mathematical modeling, algorithm development etc., can take an unreasonably long time to be completed or require a lot of computer memory. Such applications involve processing a huge amount of data or performing a large number of iterations (Roosta, 1999). Parallel computing is one of approaches to make the computation feasible by using multiple central processing units (CPUs) at the same time to solve each single problem. While most modern computers currently possess more than one CPU, several computers can be combined together in a cluster manner; these multicore processors allow many computations to be completed in reasonable time or more quickly. Generally, parallel computing or parallel processing is a form of computation in which many calculations are carried out simultaneously (Almasi and Gottlieb, 1994). A large task can be divided into smaller ones, which are solved in parallelization to complete the job in less time than a single machine can do. Each processor works on its section of problem. Furthermore, processors can exchange information among each other within a set of memories based-on distributed computing.



(a) (b)

Figure 8.1 Structure models of parallel computing level: (a) a shared-memory and (b) a distributed-memory.

An important classification of parallel computing is based-on the degree of sharing access to memory. A shared-memory of parallel computing can be classified in two structures, namely, multiprocessor desktop and multiprocessor computer clusters. Multiprocessor desktop is characterized by a structure model of shared-memory in Figure 8.1(a), in which each processor can address locations in a common memory block. Processors in a shared-memory level can communicate with one another via the common memory with a suitable protocol. On the other hand, multiprocessor computer clusters are characterized by a structure mode of distributed-memory in Figure 8.1(b), in which processors can communicate with other processors by sending messages over a network (Eddy and Allman, 2000; Gebali, 2011). Either structure can be chosen according to applications.

8.2.2 Parallel Programming

Parallel programming is a computer programming technique that provides execution of instructions in parallel manner on either a single computer or a cluster of computers (Diaz et al., 2012). In this, several programs can be performed simultaneously as multitasking on one machine employed by multiprocessor desktop or on several machines based-on multiprocessor computer clusters, in which the computing process uses computer resources such as memory blocks and registers. The predominant approaches in the parallel programming are open multiprocessing (OpenMP) for shared memory, and message passing interface (MPI) for distributed memory. The OpenMP is a shared memory application programming interface (API) that supports multiplatform shared memory and multiprocessing programming in C, C++ and Fortran. The MPI is a parallel programming model as a library, where communication between processes is done by interchanging messages over a distributed memory system. This approach can be written in different computer programming languages such as Fortran, C, C++ and Java. These techniques can process both Single Program Multiple Data (SPMD) and Multiple Program Multiple Data (MPMD) programing schemes. Nevertheless, application problems can be run on a computer cluster using both OpenMP and MPI as a hybrid model of parallel programming (Diaz et al., 2012).

The primary goal of most parallel programming is to increase performance and scalability of a CPU instead of using high-cost machines such as mainframe and super computers. This also supports parallel software productivity, which has been becoming increasingly important in recent decades. Due to parallel hardware becoming a low-cost

commodity, it is now to seize this opportunity to develop efficient parallel programing (McKenney, 2014).

8.2.3 Parallel Metaheuristics

Real-world optimization problems are always complex. Metaheuristics, that can provide global solutions within a reasonable time, are efficient solvers for this type of problems. Since standard metaheuristics are sequential in execution, solving large-scale real-world problems still demands very long time to complete the computing process. Parallel metaheuristic is a new advanced technique that has an ability to reduce both the numerical effort and the run time of a metaheuristic. This parallelization approach offers parallel or simultaneous runs of relevant programs to achieve required solutions at high quality. The classification of this parallelization strategy applied to metaheuristics is defined according to the source of parallelism (Alba, 2005) as follows:

- Operation parallelization: This source of parallelism is usually applied within an iteration of the metaheuristic method. The limited functional or data parallelism due to large-scale values can be evaluated in parallel manner. The significant goal of this class of parallelism is to reduce the overall execution time by accelerating a repeated phase of the sequential algorithm, and to achieve higher quality solutions.

- Explicit space decomposition: This approach contains parallelization strategies, which decompose an entire search space into several to many sub-spaces. The sub-spaces are to be searched by different search process in parallel manner. Usually, same metaheuristics are employed by different processes, but using different

metaheuristics is not prohibited. The later may require complicated analysis, software implementation and data management. This class of parallelism is aimed to avoid repetitive search in the parallel implementation, and to diversify the search to different regions within the search space.

- Multi-search threads: This class of parallelization is obtained from multiple concurrent explorations of the search space. Note that the word "thread" is used instead of "task" to clarify that tasks can be interleaved, but thread. Each concurrent thread may execute same or different metaheuristic methods. They may start from same or different initial solutions, and communicate during or at the end of the search to identify the best overall solution. This class of parallel strategies aims to improve the final solution by exploring different parts of the search space, and to increase the search speed.

8.3 Parallel Computing MATLAB Toolbox

So far, the Chapter has described the concepts of parallelization. Parallel computing, parallel programming and parallel metaheuristics have been defined. With today's technology, implementation of parallelization can effectively employ *Parallel Computing Toolbox*^{TM} and *Distributed Computing Server*^{TM} of MATLAB and Simulink. MATLAB and Simulink support parallel computing to run from one MATLAB session (client) to other MATLAB sessions (workers) on multiprocessor desktop and computer clusters. Parallel computing software is very useful for solving computationally intensive problems, and accelerating the processing of repetitive computations with large amounts of data by taking the full computational power of computing resources. MATLAB provides three kinds of parallelism, namely

multithreaded, distributed computing, and explicit parallelism (MathWork, 2008) as follows:

- Multithreaded parallelism: The functions automatically execute on multiple computational threads in a single MATLAB session on a multicore-enabled computer by sharing memory.
- Distributed computing: Multiple instances of MATLAB run multiple independent computations on separate computers with its own memory.
- Explicit parallelism: Several instances of MATLAB run on several processors or computers with separate memory blocks, and simultaneously execute a MATLAB command or m-function.

Parallel Computing Toolbox software helps improve the performance of loop executions of iterative algorithms by letting several MATLAB workers to simultaneously execute individual loop iterations. For instances, a loop of 100 iterations runs concurrently on a computer cluster of 20 MATLAB workers. This means that each worker executes only five iterations of the loop. This may not be 20 times improvement in speed because of communication-time consuming, but the speed-up can be significant. It can be noticed that whether or not the loops take a long time for execution, the loop speed can be improved by distributing iterations to MATLAB workers. Providing, the loop iterations run on the same computer using a multicore or multiprocessor, the speed can be improved even more through local workers. Parallel Computing Toolbox allows distributing a very large array to multiple MATLAB workers. Each worker performs only on its designated part of the array, while workers automatically transfer data among themselves, if this is necessary. Of course, if the size

of an array is not too large to fit in a local memory block, computing in a single MATLAB session is adequate.

In this thesis, we focus on some explicit parallel programing paradigms, i.e. parallel *for*-loop (*parfor*), single program multiple data (SPMD) block and distributed arrays. The description details of these paradigms are explained next.

8.3.1 Parallel *for*-loop

A modification of a simple *for*-loop in parallel MATAB is called parallel *for*-loop or *parfor-loop* conditional command. The basic concept of a *parfor-loop* is similar to the standard *for*-loop. Instead of executing a series of statement over a range of values, the *parfor-loop* is executed on MATLAB client and workers in parallel. To run the parallel loop, the MALAB pool has to be created by using *matlabpool open* command in order to reserve a collection of MATLAB worker sessions on available cores of machine as the structure in Figure 8.2. The MATLAB pool consists of MATLAB sessions running on some local machines or an available cluster.

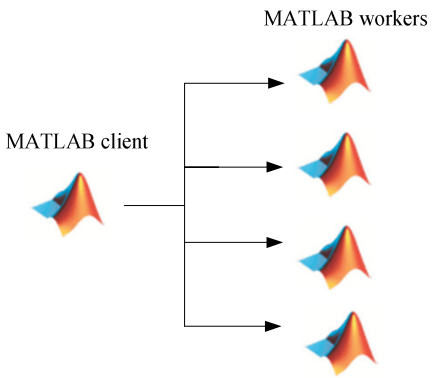


Figure 8.2 MATLAB pool structure.

Each worker evaluates iterations in no particular order and independent of each other. Therefore, each iteration is completely independent with non-sequence operation. During the operation of *parfor*, necessary data are sent from the client to the workers, and results are returned to the client and gathered together. After the process being completed, the array data of all elements are available in the MATLAB client; the MATLAB pool is closed by using *matlabpool close* command, and followed by releasing of all workers. In addition, the *parfor*-loop command allows running parallel Simulink. This feature is useful for an application problem further explained in this Chapter. The multiple models are simulated at the same time on different workers within the *parfor*-loop by using *sim* command, which helps perform multiple simulation runs of the same model. In this thesis, four local workers are used as a default of local pool size available in accordance with a quad-core platform.

8.3.2 Single Program Multiple Data (SPMD) Block

The Parallel Computing Toolbox offers the SPMD construction to achieve a domain decomposition. In SPMD block, each processor concurrently runs the job. Figure 8.3 shows a short code list for an SPMD block in MATLAB environment. Coding is noticeably straightforward.

```
1. matlabpool open
2. spmd
     For each worker do statements
     in this SPMD block.
3. end
4. matlabpool close
```

Figure 8.3 Code list for an SPMD block.

Matlabpool is used to request parallel resources called workers by Parallel Computing Toolbox. It can be said that cores and processors behave as workers in MATLAB. Each worker has a unique identity called *labindex* to customize the execution of parallel jobs. When parallel resources are accessed, parallel jobs are created in an SPMD block, and then *Matlabpool* is closed to release parallel resources at the end of the parallel jobs.

8.3.3 Distributed Array

When complexity of a problem increases, computing resource demanded may be extensive. Under this situation, computing may take many hours or days to be completed. Distributing these computational loads over processors can effectively shorten run-time. MATLAB provides an effective mean to distribute such loads via "distributed and codistributed arrays". In this, a distributed array means an array with its data being distributed from client workspace, and a codistributed one means an array with its data being codistributed within *spmd* statement created in local workers. These data can be stored on the workers of an open parallel pool. The *distributed* and

codistributed commands are used to access elements of such arrays distributed in MATLAB client's workspace, and those of the arrays distributed among workers in parallel within an *spmd* environment, respectively. For example, consider a simple 4-by-8 matrix with ascending element values from 1 to 32, which is created in a client workspace, and then it becomes a distributed array as follow:

```
matlabpool open                  // create MATLAB pool.
   x = reshape(1:32, 4,8);       // create an array on MATLAB client.
   y = distributed(x);    // distribute an array to MATLAB workers.
   spmd
      disp(getLocalPart(y))      // perform on workers in parallel.
                                 // y is codistributed array.
   end
matlabpool close                 // close MATLAB pool
```

Figure 8.4 Instruction of distributed array.

According to the code list in Figure 8.4, the command x = reshape(1:32, 4,8) creates an array having its elements of reshapes among ascending number 1 to 32 into a 4-by-8 matrix. The command y = *distributed*(x) partitions the original array into 4 since our platform is a quad-core. Then, it distributes these partitioned arrays to MATLAB client workspace. Figure 8.5 shows the original array with its associated sub-arrays. These sub-arrays are stored in parts on the workers of the open parallel pool. Inside the body of an *spmd* statement, each MATLAB worker has a unique value of *labindex* to execute the block in parallel. To display the actual data in local segment of array from the body

of an *spmd* statement, *getLocalPart*(y) returns the local portion of a codistributed array. Then, the parallel pool is closed by using *matlabpool close* command.

worker 1		worker 2		worker 3		worker 4	
1	5	9	13	17	21	25	29
2	6	10	14	18	22	26	30
3	7	11	15	19	23	27	31
4	8	12	16	20	24	28	32

Figure 8.5 A result obtained from distributed array.

According Figure 8.5, the full array in the client workspace is equally distributed into each worker as an array segment. Worker 1 stores columns 1 and 2 among an array segment from 1 to 8; worker 2 stores columns 3 and 4 including an array segment from 9 to 16, and so on.

Considering, the codistributed array, a large array structure replicated on all workers can be partitioned into segments, and distributed across the workers. Each segment stores in the workspace of a different worker, which has its own array segment to perform. This aims to reduce the size of array from the client workspace, when the memory to store an initial replicated array is sufficient, and to provide faster processing, especially for large data sets. The codistributed array is normally used in the parallel environment within *spmd* statement. For example, A is a 3-by-10 matrix with ascending element values from 11 to 40, and the *codistributed* array is created inside *spmd* statement running in parallel as shown in Figure 8.6.

```
matlabpool open                 // create MATLAB pool.
  spmd
  A = [11:20; 21:30; 31:40]; // replicate array on each worker.
  D = codistributed(A) // distribute array to MATLAB workers.
        getLocalPart(D)       // each worker operates on its data.
  end
matlabpool close                // close MATLAB pool.
```

Figure 8.6 Instruction of codistributed array.

As an example in Figure 8.6, a 3-by-10 matrix is replicated on each worker in its own workspace under an *spmd* statement, and assigned to the variable A. D = *codstributed*(A) distributes a replicated array A using the default *codstributed* command inside an *spmd* statement. Noticeably, arrays A and D are of the same size (3-by-10), and the array A exists in its full size on each worker, while only a segment of array D exists on each worker. When an array distributes to a number of workers, the MATLAB partitions the array A into segments and assigns one segment of the array to each worker as evenly as possible. The *getLocalPart* function returns the local portion of a codistributed array. At the end of code list, the parallel pool is released all of workers by using *matlabpool closed* command. The result of distributed array D is also 3-by-10 in size, but only a segment of the full array resides on each worker as Figure 8.7.

worker 1			worker 2			worker 3		worker 4	
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40

Figure 8.7 A result obtained from codistributed array.

According Figure 8.7, the full size on each worker is uniformly assigned into segments on each worker through the *codistributed* command. Worker 1 stores columns 1, 2 and 3, worker 2 contains column 4, 5 and 6, and so on.

Moreover, the explicit indexing of the distributed dimension for the codistributed array also includes using a *for*-loop over a distributed range (*for-drange*). The *for-drange* construct requires loop iterations to perform independently of each other, which each worker can be executed on the portion of range of it owns. It means that *in* a *for-drange* loop can access only the portion of a distributed array located to each worker. To illustrate this characteristic, Figure 8.8 shows an example code of using a *for-drange* loop.

```
matlabpool open                    // create MATLAB pool.
n = 1000;
y = zeros(1, n, codistributor());  // create a codistributed array.
for x = drange (1:n)       // use a for-loop over a distributed range.
    y(x) = x^2;            // calculation performed on workers.
end
    y = gather(y,1);       // collect distributed arrays into worker 1.
matlabpool close                   // close MATLAB pool.
```

Figure 8.8 An example code of using a *for-drange* loop.

According to the code list in Figure 8.8, the loop increases from 1 to n. The loop x is partitioned by *codistributed* command into segments. Each segment becomes an iteration for a conventional for-loop on an individual worker. Thus, the calculation of y is performed on workers. After getting the computational results, an array data from each worker is collected into worker 1 using *gather* command.

As the above mentioned, some commands of Parallel Computing Toolbox software for MATLAB have been presented. These significant commands of MATLAB parallel computing will be applied to improve the proposed modified ATS algorithm in order to speed-up the process. This is to further optimize the power drive system problem in the next section.

8.4 Applications of Parallel Computing Toolbox to the Modified ATS Algorithm

From the previous section, we realize that the Parallel Computing Toolbox executes their commands in no particularly order. Its execution process is unreached by ordinary users. Some parallel command functions may not be applicable to the proposed algorithms because their main processes need to perform sequentially to update the current solution from the previous one, which is dependent on each other. Particularly, the tabu list storage must sort the elite solutions in order. If the parallel command functions are carelessly utilized, the results obtained will be erroneous. The *parfor-loop* and *for-drange* commands may not be appropriated for this case. Regarding to the proposed application, the power drive system involves time-consuming computing with large data sets. The ability of parallelization on one computer platform is particularly suitable in order to improve computing speed. Thus, the thesis works will gain benefit from the Parallel Computing Toolbox in local parallel computing point-of-view. The implemented algorithms are performed by an Intel(R) quad-core CPUs, which contains four workers to execute in MATLAB sessions. This computing platform runs at clock speed of 2.40GHz with 4GB RAM. Our implementation is explained next.

8.4.1 Implementation of Parallel Process

From the previous results of the power drive system problem obtained from the proposed algorithms, we recall that the solutions cover all constraint requirements superior to the conventional designs. Unfortunately, the search process spent very long time to obtain the optimum solutions because of the following reasons: (i) the objective functional model is very complicated, (ii) there are many constraints, and (iii) the random-walk front-end is invoked many times. During search, it was necessary to frequently evaluate the objective model, and generate a population of initial solutions via the random-walk front-end. To reduce the search time, these two major activities are considered for parallelization. The SPMD construction and codistributed arrays have been applied to these problems. The SPMD block instructs each worker to concurrently compute each function through *labindex* command in parallel. We further use the *codistributed* command to distribute the sets of initial solutions indicated by S into each worker. This process independently generates a population of solutions, and each worker can effectively update the random movement indicated by N_c for each location on each worker. The computing process of the proposed parallel version of the modified ATS is represented by the diagram in Figure 8. 9, in which the detail of random walk front-end appears in Figure 4.6 of Chapter 4.

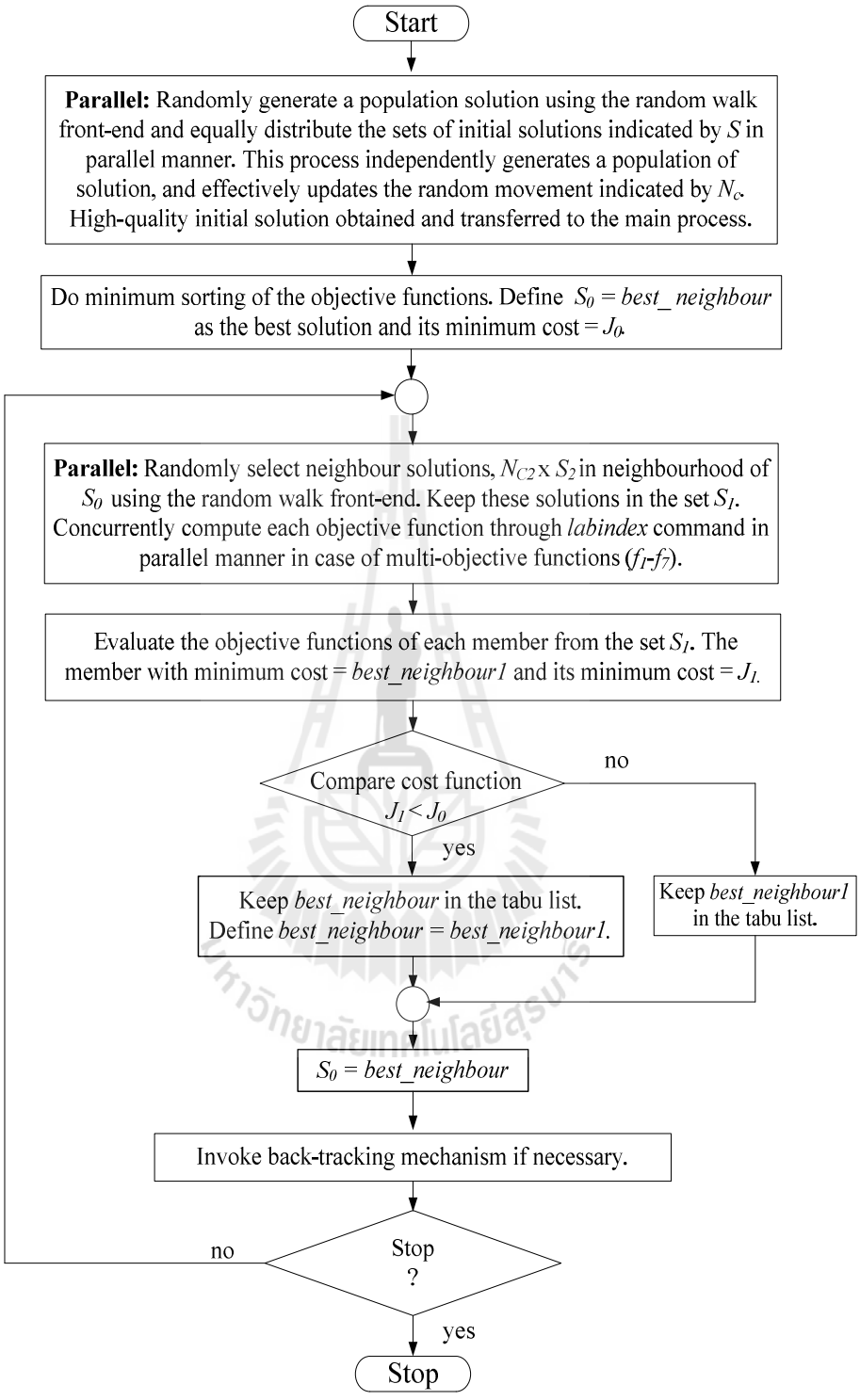


Figure 8.9 Flow diagram of the parallel version of the modified ATS.

We consider parallelization of the computation of the objective functions, first. Since MATLAB does not allow one simulink model to be divided into multi sections, or to run the model in parallel, it is necessary to run the whole simulink model on each worker and the MATLAB client. In this, our simulink model is run on the MALAB client. Subsequently, we allocate the data obtained from the simulation to all workers' codistributed arrays. Afterward, the *labindex* command is invoked to simultaneously compute the objective functions as indicated by the code list in Figure 8.10. Note that, the order of the objective functions computed is decisively placed to minimize the computing time.

```
matlabpool open                 // create MATLAB pool.
spmd (4)                        // 4 workers are used.
// all data assigned inside the spmd construction performed as codistributed
// arrays to all workers.
labBarrier;                     // to ensure all workers are synchronized,
                                // and start their timed work together.
    if labindex==1              // worker 1 indicated.
        J_THDf=THDf/3;          // compute objective function 1: THDi of input
                                // current.
    end
    if labindex==2              // worker 2 indicated.
        PL_total=PL_line + PL_SD + PL_motor;
        J_PL_total=PL_total/Paccept_max;
                                // compute objective function 3: Total power loss.
        J_RMS_recti=RMS_recti/RMSrecti _max;
                                // compute objective function 4: RMS of current
                                // control loops for back-to-back converter.
        J_RMS_DC= RMS_DC/ RMS_DC_max
                                // compute objective function 5: RMS of DC-link
                                // voltage control loop.
    end
    if labindex==3              // worker 3 indicated.
        J_THDi_motor=THDf_motor/3;
                                // compute objective function 2: THDi of motor
                                // current.
    end
    if labindex==4              // worker 4 indicated.
        J_RMS_inverter=RMS_inverter/ RMS_inverter_max
                                // compute objective function 6:  RMS of current
                                // control loops for PMSM.
        J_RMS_Speed=RMS_Speed/ RMS_Speed_max
                                // compute objective function 7: RMS of speed
                                // control loop for PMSM.
    end

    end                         // end SPMD.

    f1=J_THDf{1};               // Transfer array from worker 1 to client.
    f3=J_PL_total{2};           // Transfer array from worker 2 to client.
    f4=J_RMS_recti{2};
    f5=J_RMS_DC{2};
    f2=J_THDi_motor{3};         // Transfer array from worker 3 to client.
    f6=J_RMS_inverter{4};       // Transfer array from worker 4 to client.
    f7=J_RMS_Speed{4};

matlabpool close                // close MATLAB pool.
```

Figure 8.10 Code list for parallel computing of multi-objective functions.

Referring to Figure 8.10, the *matlabpool* is used as an open command in order to reserve a collection of MATLAB workers on the available four CPU cores. Subsequently, all workers are activated under the *spmd* command. The identity of each worker called *labindex* has been assigned to compute 7 objective functions simultaneously. For instances, worker 1 computes the function of $THD_{i,input}$ for input current, worker 2 computes the functions of the total power loss, the RMS value of the current control loops for the back-to-back converter, and the RMS value of the DC-link control loop, and so on. After the completion of such calculations, the results are returned to the MATLAB client, and then computing follows in a similar manner to what described in Figure 7.37, Chapter 7. As a result, our parallel program consumes computing time of 31.9172 seconds, while the sequential one consumes 43.5739 seconds. It is clear that the objective function parallelization results in 26.75% run-time savings. Note that this part of the program was tested independently from the others by assigning some real values to relevant variables, instructing it to run, and monitoring the computing time.

```
function [min_cost_function, X_Y_best]= Initial_generation(p,S,Nc,Ns,alpha,xlimit)
matlabpool open                                  // create MATLAB pool.

parfor m1=1:S                                    // use parfor-loop to generate random
                                                 // solutions
   P(:,m1,1)= ((xlimit(1,:)-xlimit(2,:)).*rand(1,p))+xlimit(2,:);
 end

spmd
   DIST=codistributor1d(2,[S/4 S/4 S/4 S/4]);    // assign array of solution on each worker.
   P_codis=codistributed(P,DIST);                // replicate array on each worker.
   P=getLocalPart(P_codis);                      // each worker operates on its data.

for j=1:Nc                                       // start for-loop of random movement.
    for i=1:S/4                                  // start for-loop of a number of solution.
        J(i,j)=obj(P(:,i,j));                    // recall obj function to compute cost value.
        Delta(:,i)=(2*round(rand(p,1))-1).*rand(p,1);   // compute constant value.
        C(i,j)=abs(J(i,j))/((abs(J(i,j)))+alpha);       // compute step size.
        P(:,i,j+1)=P(:,i,j)+C(i,j)*Delta(:,i)/sqrt(Delta(:,i)'*Delta(:,i));
                                                 // update solutions.
        J(i,j+1)=obj(P(:,i,j+1));                // recall obj function to update cost value.
     .
     .
     .
    end                                          // end loop of a number of solution, S.
end                                              // end loop of random movement, Nc.
end                                              // end spmd construction.
Jgather=[J{1}; J{2}; J{3}; J{4}];                // Return all array of cost values, J from
                                                 // workers to client.
reproduction = Jgather (:,1:Nc);                 // select cost value.
[jlastreproduction,iter] = min(reproduction,[],2); // find minimum cost value and its position
                                                   // among random movement, Nc.
[min_cost_function,back_ID] = min(jlastreproduction);
                                                 // find minimum cost value and its position
                                                 // among each number of solution, S.
Pgather =[P{1} P{2} P{3} P{4}];                  // Return all array of solutions, P from
                                                 // workers to client.
X_Y_best=Pgather (:,back_ID,iter(back_ID,:));    // select an elite solution.
matlabpool close                                 // close MATLAB pool.
```

Figure 8.11 Code list for parallel computing of generating an elite initial solution.

Second, we consider parallelization of the program part to generate an elite initial solution. Figure 8.11 shows the program codes of this part. Recall that the process at the beginning of the modified ATS evaluates a group of solutions based-on repeated number of solutions, S and random movement, N_c, to obtain an elite solution. The code starts from using a MATLAB pool, and then invoking the *parfor*-loop to randomly generate a group of initial solutions equal to the maximum number of S for the first movement. The initial solutions have been equally assigned to codistributed arrays on 4 workers under the *spmd* construction. Noticeable, the number of initial solutions, S should be integer that can be divided by 4 in order to be distributed equally among 4 MATLAB workers. $S = 32$ is applied here. One is not recommended to use an unequal work distribution, e.g. S =30 etc., because an unbalanced computing load on each worker needs to wait the others to finish the tasks completely. This imposes a long wait-time, in turn a long computing time. Next the process runs parallelly to update the random movement indicated by N_c for each location on each worker until the maximum number of random movement, N_c is reached. The solutions (P) and their corresponding cost values (J) are then returned to the MATLAB client for evaluating the optimum solution. This is an end of the process, and the MATLAB pool is released by using the *matlabpool* command.

As a preliminary test, this section of our program was run independently by setting $S = 32$ and $N_c = 20$ in order to compare the computing time between the parallel and sequential versions. As a result, the sequential version took 4.3156×10^4 seconds to compute the job, while the parallel one required 2.3356×10^4 seconds. Therefore, 45.88% of computing time savings are brought into attention.

Applications of our parallel version of the modified ATS to the power drive problem have been conducted. The results and discussions are presented next.

8.4.2 Results and Discussions

The same power drive system as presented in Chapter 7 is now considered. The search problem is assigned to our new parallel version of the modified ATS with the same values of all search parameters and the same multi-objective functions. Comparisons of the results and computing time obtained from the parallel and the sequential programs are of our interest. Readers are reminded that our parallel version of the modified ATS contains two parts of parallelization denoted as the parallel elite initial solution generation and the parallel multi-objective functions, respectively.

Firstly, we consider the filter components and the controller parameters as being obtained from various design approaches. Table 8.1 summarizes those values for comparison purposes. For low-pass input filter, the obtained components are somewhat different. As a practical requirement, the *THDi* must be less than 3%. This results in the cut-off frequency at -3dB or $f_{-3dB} \le 250\,\text{Hz}$ in order to effectively sift the 5^{th}-order harmonic since it appears most in the system. Considering the R_f and L_f values tabulated, the conventional design, the BF-TS, the sequential and the parallel version of the modified ATS result in the following cut-off frequencies: 70,145, 145 and 90 Hz, respectively. The following THD_i values of the input currents or $THD_{i,input}$ are obtained: 8.0903%, 1.8230%, 1.6810% and 1.314%, respectively. Figure 8.12 illustrates the corresponding frequency responses of the obtained filers. It can be said that all the methods being considered provide similar results of harmonics suppression for the

input current. However, the R_f values as low as 10-50 $m\Omega$ seems to be good in terms of low power losses, but they may be impractical in power system field because they lead to large L_f s required. In practice, the filter obtained from the conventional design is recommended.

Table 8.1 Summary of the filter components and the controller parameters obtained from conventional designs and the proposed algorithms.

Controllers		Conventional designs	BF-TS	Modified ATS	Parallel modified ATS
Input filter:	R_f (Ω)	0.13	0.050	0.015	0.027
	L_f (mH)	3.30	1.60	1.58	2.46
Stator inductance:	L_s (mH)	1.20	1.21	1.19	2.05
Switching frequency:	f_s (kHz)	20	9.71	10.60	10.56
Line input current controller:	$K_{P,line}$	16.89	49.87	43.39	55.26
	$K_{I,line}$	3.36×10^4	5.41×10^4	6.32×10^4	7.28×10^4
DC-link voltage controller:	$K_{P,DC}$	0.74	3.54	3.66	3.84
	$K_{I,DC}$	73.11	353.02	486.44	515.94
Load current controller:	$K_{P,load}$	2.29	36.95	33.31	30.44
	$K_{I,load}$	1.89×10^3	1.26×10^3	2.12×10^3	2.69×10^3
Speed controller:	$K_{P,speed}$	0.21	7.28	7.33	14.57
	$K_{I,speed}$	9.05	49.23	57.88	36.07

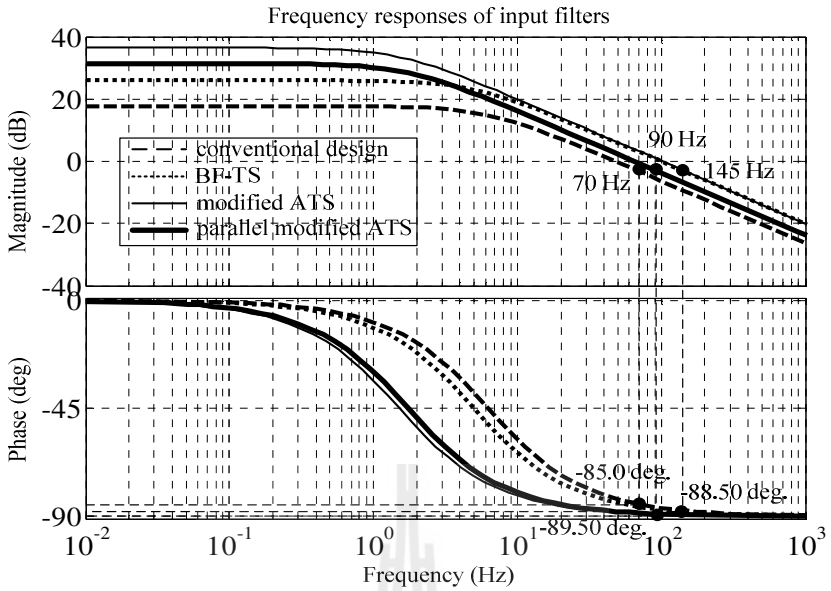


Figure 8.12 Comparison of frequency responses of input filters obtained from different methods.

Now, we consider the switching frequency and the stator inductance as tabulated in Table 8.1. Recall from Chapter 7 that the stator resistance is known, $R_s = 0.124\,\Omega$. An optimum inductance could be searched for. Practically, it helps reduce the motor current harmonic. The values of L_s shown in Table 8.1 are closed to each other except that found by the parallel version of the modified ATS, which is about twice as much. For the switching frequency (f_s), the conventional design gives 20 kHz, which is about two times the others. The frequency, f_s does not have a role on harmonic, but L_s. Notice that the modified ATS gives L_s = 1.19 mH, which is minimum. This results in the minimum $THD_{i,motor}$ = 1.0047% very much less than 3% of the standard requirement. In addition, the BF-TS and the parallel version of the modified ATS render $THD_{i,motor}$ = 1.1330% and 1.2452%, respectively. In terms of total

power losses, f_S has a significant contribution. While the conventional design gives f_S = 20 kHz, the BF-TS, the modified ATS and the parallel version provide f_S = 9.71, 10.60 and 10.56 kHz, respectively. The corresponding power losses to these frequencies are 1.91, 1.69, 1.64 and 1.65 kW, respectively. f_S provided by search methods can result in a significant reduction in the total losses by 14% approximately.

We now consider the parameters of the PI-controllers summarized by Table 8.1. In all cases, the search approaches provide the values of K_P and K_I much higher than those obtained from the conventional design. This means that responses of the system are faster due to high K_P, and smoother due to high K_I, as a result. However, the response curves, i.e. current, voltage and speed responses, are very similar to those illustrated in Chapter 7. Such curves are omitted herein. Table 8.2 summarizes THD_is, power losses and RMS responses corresponding to the search methods. It can be observed that the parallel version of the modified ATS gives solutions resulting in the smoothest responses and the minimum current distortions.

Table 8.2 Comparison of constraint results obtained from the proposed algorithms.

Methods	$THD_{i,input}$ (%)	$THD_{i,motor}$ (%)	$P_{total,loss}$ (kW)	$RMS_{rectifier}$	RMS_{DC} (Vdc)	$RMS_{inverter}$	RMS_{speed} (rpm)
BF-TS	1.8230	1.1333	1.69	0.0167	2.0058	0.0165	62.7814
Modified ATS	1.6810	1.0047	1.64	0.0172	1.6735	0.0169	59.1439
Parallel modified ATS	1.3138	1.2452	1.65	0.0180	1.5734	0.0198	37.8697

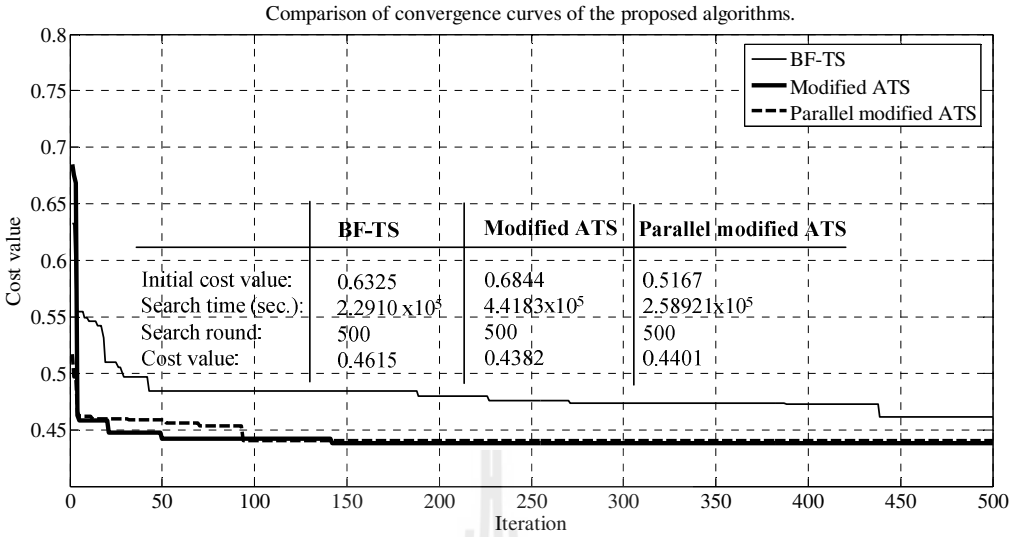


Figure 8.13 Comparison of convergence curves of the proposed algorithms.

Secondly, we consider the convergent behaviour of each search method applied to the power drive problem. This can be represented by the convergence curves with the corresponding data shown in Figure 8.13. Notice that the parallel version cuts the search time of the modified ATS by 41.40% approximately. The sequential and the parallel versions of the modified ATS provide better quality solutions than those given by the BF-TS. This can be observed from the cost values according to the ATS-based methods reach the minimum by the 150th iteration. As a matter of fact, the ATS-based methods could effectively cease their searches very much faster than the BF-TS could. The multi-cost values of the problem during search by the parallel version of the modified ATS are shown in Figure 8.14 and Table 8.3 to confirm the quality of the obtained solutions. Notice that, all the constraints are met due to $\left| f_k \right|_{k=1,2,...,7} < 1$.

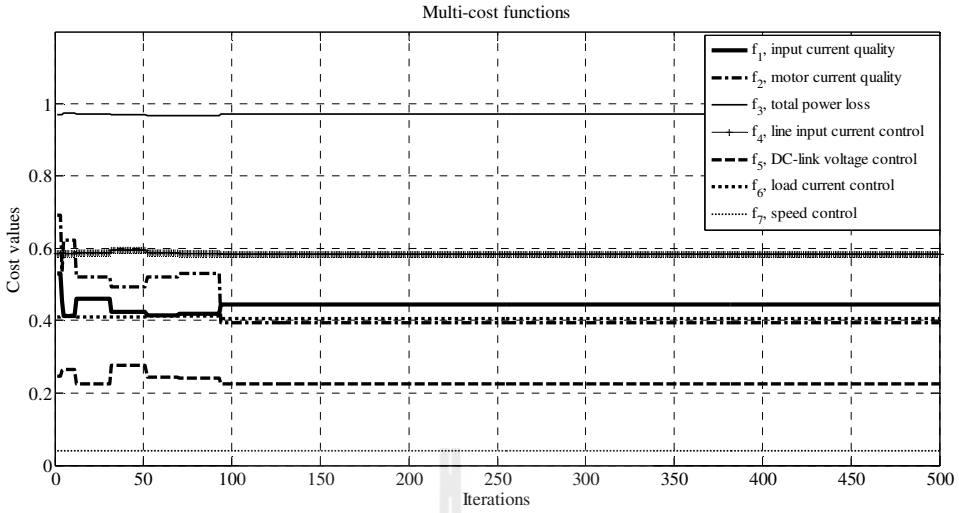


Figure 8.14 Curves of multi-cost functions for the parallel version of modified ATS.

Table 8.3 Comparison of the final cost values for each objective function obtained from the proposed algorithms.

Methods	f_1	f_2	f_3	f_4	f_5	f_6	f_7
BF-TS	0.6077	0.3778	0.9934	0.5591	0.2874	0.3375	0.0678
Modified ATS	0.5604	0.3349	0.9652	0.5569	0.2398	0.3461	0.0639
Parallel modified ATS	0.4379	0.4151	0.9718	0.5841	0.2254	0.4055	0.0409

8.5 Conclusion

So far, the Chapter has presented a successful approach to significantly reduce computing time or search time of metaheuristics particularly when the problem of interest is a large-scale complex real-world problem. In general, this type of problems is nonlinear, subject to several to many constraints, possesses both continuous and discrete variables, and needs complex simulation models to evaluate the problem costs.

Parallelization is a useful technique to achieve the search time reduction as the Chapter begins explaining its concepts and terminologies. Implementation of parallel algorithms is quite straightforward using the *Parallel Computing Toolbox*^{TM} with MATLAB. We have explained our parallel implementation of the modified ATS in details, and applied it to a complex power drive system successfully. As a result, the parallel version of the algorithms reduces the search time approximately by 41.40%, in other words 1.71 times faster than the sequential version.

CHAPTER IX

CONCLUSIONS

9.1 Conclusions

The aims of this research were to examine and improve the search performance of metaheuristics in order to achieve an efficient computational time. Adaptive tabu search (ATS), adaptive bacterial foraging (ABFO), invasive weed optimization (IWO), and genetic algorithm (GA) were considered, and their performances were investigated via some unconstrained surface optimization problems. After thorough studies of their outstanding capabilities, the ABFO and the ATS algorithms were combined in order to obtain new hybrid or cooperative metaheuristics denoted as the cooperative bacterial foraging-tabu search or BF-TS and the modified ATS algorithm, respectively. To demonstrate potential, usefulness and capability of the proposed metaheuristic algorithms, they were applied to various complex engineering problems, and reorganized to run in parallel manner under a quad-core machine.

At the beginning of this thesis, a literature review of two groups of hybrid metaheuristic algorithms (population-based and single-solution based metaheuristics) to solve various optimization problems was presented in Chapter II. Here, the main metaheuristic algorithms for this research, ABFO and ATS approaches, were characterized in order to expound the advantages of combining both methods. According to the literature review, a pure population-based metaheuristic like the

ABFO is sometimes not well suited for fine-tuned search, and in some cases, the exploitation of the solution found is weak when used to solve complex combinatorial problems. Moreover, single-solution based metaheuristic such as the ATS is poor in exploration of entire search space. Therefore, the ability of the ABFO in terms of exploration into the search space was combined with the ATS, which has the ability of exploitation in order to increase the convergence rate, and improve the quality of the final solutions.

Background to metaheuristics was presented in Chapter III. This chapter also introduced different algorithms for solving combinatorial optimization problems, including classical optimization algorithms and generic algorithms of some important metaheuristics. In addition, a historical overview and a classification of the metaheuristics were presented. The development of the metaheuristic methods indicates that they will be more popular in the future as optimization problems are increasing in size and complexity.

The algorithms considered in this work were thoroughly described in Chapter IV. The descriptions of procedural lists and flowcharts for the algorithms previously mentioned (ATS, ABFO, IWO and GA) were given. In addition, this chapter introduced two cooperative approaches: the bacterial foraging-tabu search (BF-TS) and the modified ATS algorithms. The algorithm denominated bacterial foraging-tabu search (BF-TS) is a manner of combining the ABFO and ATS algorithms. In this method, the adaptive random movement of bacteria leads to an improvement of the quality of initial solution of the ATS algorithm. The modified ATS approach is a new cooperative algorithm which integrates ABFO and ATS. Basically, in this algorithm, the adaptive random movement of bacteria allows the algorithm to produce neighbour solutions

around the global solution in order to enhance the search performance without the limit of radius.

In Chapter V, an analysis of the convergence of the modified ATS algorithm (included an initial part of BF-TS algorithm) based-on mathematical model was presented. This algorithm can be divided into two parts, the random walk front-end and the ATS without search radius adjustment. Since the convergence of the TS with the *BT* mechanism is known to have convergence property, the convergence analysis was primarily focused on the convergence of the random walk front-end. This analysis was carried out by employing Lyapunov's stability concepts. By intuition, the convergence of the proposed algorithms resulted from combination of the two methods was concluded.

Comparisons of search performances for the cooperative approaches and the other aforementioned algorithms were carried out in Chapter VI. The comparison was in terms of number of local locks, quality of initial solutions, quality of global solutions obtained, search time, number of search rounds and time consumed per search round. These aspects reflect the performance of the proposed algorithms. Five unconstrained optimization problems, also called benchmark functions, were utilized to test the performances, namely Bohachevsky (BF), Shekel's fox-holes (SF), Rastrigin (RF), Shwefel (ShcF) and Schubert (ShuF) functions. Each set of algorithms was tested by undergoing 50 trial runs. The obtained results were averaged in finding search parameters so as to obtain better performances. Those parameters were applied to a vast number of computing tasks for collecting the results. According to the results, the proposed modified ATS algorithm exhibited the best performance, when considering quality of initial solutions, quality of global solutions, number of local locks and

number of search rounds. This was mainly owing to the high-quality initial solutions generated by the random walk front-end. These initial solutions lead to high-quality global solutions in a significantly low number of search rounds. Furthermore, the chemotaxis allowed the algorithm to avoid oscillations in the searched region. However, since the modified ATS approach required invoking the random walk front-end frequently, the search process of this approach consumed longer time in comparison with the standard ATS and ABFO algorithms. In addition, based on experimental results, population-based metaheuristic methods like IWO and GA showed moderate search performances compared with the single-solution based ones.

In Chapter VII, applications of the proposed algorithms to various constrained real-world problems and some abstract mathematical ones were carried out. Such problems include three abstract mathematical constraint problems, optimal control design of hard-disk heads (single-head and head-stacks), optimal control design of a truck braking system, stability analysis of nonlinear systems, identification problems of hard-disk head actuator and nonlinear Stribeck friction model, and a power drive system. The results were compared between the proposed metaheuristic algorithms and conventional designs. It was found that the proposed metaheuristic algorithms provided superior results in all cases with less design complexity. In addition, the comparison results between the BF-TS and the modified ATS were discussed. In most cases, faster convergences, a smaller number of local deadlocks and a smaller number of search rounds were found in the case of the modified ATS algorithm. In contrast, search time per iteration consumed by the BF-TS was shorter than that by the modified ATS because the modified ATS evaluated the objective functions more frequent than the BF-

TS did. This was due to the modified ATS spending time to execute its random walk front-end.

Since real-world optimization problems have become increasingly complicated, solving such problems require considerably long CPU time. Meanwhile, personal computers today have multiple cores that enable multiple threads to be executed simultaneously. To take advantage of the hardware, parallel computing is considered as a technique working with metaheuristics on multicore processors. Chapter VIII explains to various meaning of parallelization. Definitions of parallel processing or parallel computing, parallel programing and parallel metaheuristics have been described. Furthermore, parallel computing in MATLAB and its significant functions employed by *Parallel Computing Toolbox*^{TM} were exhibited with some simple examples given. The parallel computing with MATLAB was applied to the modified ATS for solving a complex design problem of an electric drive in order to reduce computing time. As a result of running our parallel MATLAB program on a 4-core desktop computer, the computing time was significantly decreased by 41.40%.

9.2 Future Works

From extensive studies of the thesis, several possible future works are identified. They can be classified into five groups namely (i) algorithm development, (ii) search performance assessment, (iii) convergence of algorithm, (iv) parallel computer clusters and (v) applications. These are elaborated as follows.

9.2.1 Algorithm Development

In recent years, some new groups of metaheuristics have emerged. They are denoted as swarm intelligence, bio-inspired metaheuristics (without swarming) and physics/chemistry-based metaheuristics. Each group has salient characteristics which are able to enhance each other. It will be beneficial to the field of computational intelligence providing such algorithms are explored. New metaheuristics could be developed as cooperative algorithms and new metaheuristics. The newly developed algorithms will be very useful for solving hard combinatorial optimization problems.

Furthermore, developing the modified ATS algorithms proposed by the thesis into "parallel metaheuristics" is very much encouraged. It is very challenging to combine operation parallelization, space decomposition and multi-threads together to achieve high performance algorithms. The algorithms to be obtained will be interestingly tested on "parallel computer cluster" proposed in 9.2.4.

9.2.2 Search Performance Assessment

A new set of algorithms necessarily needs a proper assessment in terms of search performance. As a common practice, geometrical functions have been recommended for the purpose. However, multi-dimensional functions sometimes used are difficult to be visualized. Some researchers organize them into many groups of three dimensional databases indexed by some variables of the same functions referred to as pointers. We consider the approaches as theoretic because the functions and variables bear no physical meanings. Therefore, we suggest for performance assessment that researchers should conduct two main tasks. One is based-on geometrical functions, and

another on real-world problems. The real-world problems must be complicated enough for this purpose. Here are our suggestions but not limited to:

- Engineering problems: stability analysis of a large-scale complex nonlinear system, optimization problems in a whole system of rail transports, a large-scale power system (i.e. optimization of power flow and power quality, etc.), and a large-scale water management.

- Science problems: identification of some molecular structures, and computational synthesis of new materials.

- Management problems: optimization of logistic routes over a very large area, and risk assessment and management of a stock market or a large organization.

9.2.3 Convergence of Algorithms

It is rather unfortunate that most researchers in the field of computational intelligence usually end-up with their proposed algorithms without a provision of convergence property. We believe that algorithms with their convergence guaranteed are very solid for being used by potential users. Based-on our survey, we have found three possible approaches for convergence analysis such that we recommend researchers to conduct one of the following ways:

- probability-based method,
- Markov-process-based method, or
- energy-based (stability) method.

9.2.4 Parallel Computer Clusters

This thesis has demonstrated a parallelization of the proposed algorithms on a single hardware platform. Hence, computing resources are clearly limited. For very large-scale complex systems, more powerful hardware with more resources may be necessary. One possible future work is to develop parallel computer clusters to work with *Distributed Computing Server™ Toolbox*. This will require an up-to-date hardware development based-on existing PCs, notebooks, CPU-boards or the like.

9.2.5 Applications

As demonstrated by the thesis, metaheuristics are powerful tools for solving very complicated problems. We have observed to date some challenging problems that can be suitably tackled by metaheuristics are as follows:

- Optimal design of a matrix converter with nonlinear loads,
- Development of smart grid technology,
- Optimal placement of distributed energy resources,
- Optimal power flow problem,
- Electrical machine design with unusual requirements on characteristics, and so on.

REFERENCES

Abdelhadi B., Benoudjit A., and Nait-Said N. (2005). Application of Genetic Algorithm with a Novel Adaptive Scheme for the Identification of Induction Machine Parameters. **IEEE Transactions on Energy Conversion**. 20(2). pp. 284-391.

Ackermann T. and Soder L. (2002). An Overview of Wind Energy Status. **Renewable and Sustainable Energy Reviews**. 6(1-2), pp. 67-127.

Alba E. (2005). **Parallel Metaheuristics: A New Class of Algorithms**. Published by John Wiley & Sons, Inc., Hoboken, New Jersey.

Almasi G. S. and Gottlieb A. (1994). **Highly parallel computing**. 2nd ed., Redwood City, Calif., Benjamin/Cummings Pub. Co.

Apap M., Clare J.C., Wheeler P.W., Bland M., Bradley K. (2003). Comparison of Losses in Matrix Converters and Voltage Source Inverters. **IEE Seminar on Matrix Converters**. pp. 4/1-4/6.

Armstrong-Helouvry B. (1993). Stick Slip and Control in Low-Speed Motion. IEEE **Transactions on Automatic Control**. 38(10). pp. 1483-1495.

Armstrong-Helouvry B., Dupont P. and Cadudas de Wit, C. (1994). A Survey of Model, Analysis Tools and Compensation Methods for the Control of Machines with Friction. **Automatica**. 30(7). pp.1083-1138.

Bada A.T. (1987). Robust Brake Control for a Heavy-Duty Truck. **IEE Proceedings**. 134(1). pp. 1-8.

Bagis A. (2006). Performance Comparison of Genetic and Tabu Search Algorithms for System Identification. **LNAI4251-Part I, Springer-Verlag**, Berlin Heidelberg. pp. 94-101.

Battiti R. and Tecchiolli G. (1994). The Reactive Tabu Search. **ORSA Journal on Computing**. 6(2). pp. 126-140.

Bentley J. E. (2000). SAS Multi-Process Connect: What, When, Where, How, and Why". **Conference Proceedings in SUGI26**, Cary, NC: SAS Institute Inc.

Biswas A, Dasgupta S., Das S, Abraham A. (2007). Synergy of PSO and Bacterial Foraging Optimization: A Comparative Study on Numerical Benchmarks, in: E. Corchado et al. (Eds.), Second International Symposium on Hybrid Artificial Intelligent Systems (HAIS 2007), **Advances in Soft computing Series**, Springer Verlag, Germany, Innovations in Hybrid Intelligent Systems, ASC 44, pp. 255-263.

Blum C. and Roli A. (2003). Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. **ACM Computing Surveys**. 35(3). pp. 268-308.

Brown D. H., Christensen T.C, Cunningham E. A. and Rogelstad W. A. (2000). SELF SERVO WRITING FILE. **U.S. Patent**. 6040955.

Cadudas de Wit C., Olsson H., Armstrong K.J. and Lischinsky P. (1995). A New Model for Control of Systems with Friction. **IEEE Transactions on Automatic Control**. 40(3). pp. 419-425.

Carawukh S., Sarasiri N., Boonpranchoo V., Kumbla K. and Sujitjorn S. (2011). Application of Hybrid BF-TS Algorithms to Identification of HDD Actuator Model, **The 2011 World Congress on Engineering and Technology**, Shanghai, China, October 28-30, pp. 1-4.

Carl B. (2002). Searching for Lyapunov Functions Using Genetic Programming. http://www.aerojockey.com/files/lyapunovgp.pdf. pp. 1-13.

Casanellas F. (1994). Losses in PWM Inverters using IGBTs. **IEE Proceedings on Electric Power Applications**. 141(5), pp. 235-239.

Chiang W.C. and Chiang C. (1998). Intelligent Local Search Strategies for Solving Facility Layout Problems with the Quadratic Assignment Problem Formulation. **European Journal of Operational Research**. 106. pp.457-488.

Chen B.M., Lee T.H., Peng K. and Venkataramanan V. (2003). Composite Nonlinear Feedback Control for Linear Systems with Input Saturation: Theory and An Application. **IEEE Transactions on Automatic Control**. 48(3). pp. 427-439.

Chen B.M., Lee T.H., Peng K. and Venkataramanan V. (2006). **Hard Disk Drive Servo Systems**. 2nd edition, Springer-Verlag, London.

Chen H., Zhu Y. and Hu K. (2009). Cooperative Bacterial Foraging Optimization. **Dynamics in Nature and Society**. Hindawi Publishing Corporation Discrete 2009(815247). pp. 1-17.

Chen H., Zhu Y. and Hu K. (2010). Multi-Colony Bacteria Foraging Optimization with Cell-to-Cell Communication for RFID Network Planning. **Applied Soft Computing**. 10. pp. 539-547.

Cohen G. H. and Coon G. A. (1953). Theoretical Consideration of Retarded Control. **Transactions on ASME**. 76. pp. 827-834.

Cook S.A. (1971). The Complexity of Theorem Proving Procedures. **Conference Proceedings, Third Annual ACM Symposium on the Theory of Computing**, New York. pp. 151-158.

Courant, R. (1943). Variational Methods for the Solution of Problems of Equilibrium and Vibrations. **Bulletin of the American Mathematical Society**. 49. pp. 1-23.

Cupertino F., Mininno E., Naso D., Turchiano B., Salvatore L. (2004). Online Genetic Design of Anti-Windup Unstructured Controllers for Electric Drives with Variable Load. **IEEE Transactions on Evolutionary Computation**. 8(4). pp. 347-364.

Dadalipour B., Mallahzadeh A.R. and Davoodi-Rad Z. (2008). Application of the Invasive Weed Optimization Technique for Antenna Configurations. **IEEE Conference on Antennas and Propagation**. pp. 425-428.

Das S., Dasgupta S., Biswas A. and Abraham A. (2009). On Stability of the Chemotactic Dynamics in Bacterial-Foraging Optimization Algorithm. **IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans**. 39(3). pp. 670-679.

Dasgupta S. Das S. Abraham A. and Biswas A. (2009). Adaptive Computational Chemotaxis in Bacterial Foraging Optimization: An Analysis. **IEEE Transactions on Evolutionary Computation**. 13(4). pp. 919-941.

Dasgupta S., Das S., Biswas A. and Abraham A. (2010). Automatic Circle Detection on Digital Images with An Adaptive Bacterial Foraging Algorithm. **Soft Computing,** Springer-Verlag. 14. pp. 1151-1164.

Davies B. (2002). **Heaviside Step Function**. Integral Transforms and their Applications. 3rd edition. Springer.

Di L. and Ze T. (2011). A Genetic Algorithm with Tabu Search for Multi-Objective Scheduling Constrained Flexible Job Shop. **IEEE Conference on Cross Strait Quad-Regional Radio Science and Wireless Technology**. pp. 1662-1665.

Diaz J., Munoz-Caro C. and Nino A. (2012). A Survey of Parallel Programming Models

and Tools in the Multi and Many-Core Era. **IEEE Transactions on Parallel and Distributed Systems**. 23(8). pp. 1369-1386.

Dong, F.D., Ang, K. and Wang, Y.Y. (2006). TMR Analysis and Controller Design for Self-Servo Writing in High-Density Hard Disk Drives. **IEEE Proceedings of the International Conference on Control, Automation, Robotics and Vision**. pp. 1-7.

Dong W., Zanchetta P. and Thomas David W.P. (2008). Identification of Electrical Parameters in a Power Network Using Genetic Algorithms and Transient Measurements. **IEEE Conference on Power Electronics and Motion Control**. pp. 1716-1721.

Dorf R.C. and Bishop R.H. (1995). **Modern Control Systems** (7th ed.). Massachusetts: Addison-Wesley.

Dorigo M., Maniezzo V. and Colorni A. (1996). Ant System: Optimization by a Colony of Cooperating Agents. **IEEE Transactions on Systems, Man and Cybernetics-Part B**. 26(1) pp. 29-41.

Du H. and Nair S. S. (1999). Modeling and Compensation of Low-Velocity Friction with Bounds. **IEEE Transactions on Control Systems Technology**. 7(1). pp. 110-121.

Eddy W. M. and Allman M. (2000). Advantages of Parallel Processing and the Effects of Communications Time. **Research Report-NASA Glenn Research Center Report**, Number CR-209455, Ohio Univ.; Athens, OH United States. pp. 1-10.

Forgel L.J. (1962). Toward Inductive Inference Automata. **Proceedings of the International Federation for Information Processing Congress**, Munich. pp. 395-399.

Gandomi A. H. and Alavi A. H. (2012). Krill Herd: A New Bio-Inspired Optimization Algorithm. **Communication on Nonlinear Science and Numerical Simulation**. 17(12). pp. 4831-4845.

Gebali F. (2011). **Algorithms and Parallel Computing**. Published by John Wiley & Sons, Inc. Hoboken, New Jersey and Canada.

Genesio R. and Vicino A. (1984). New Techniques for Constructing Asymptotic Stability Regions for Nonlinear Systems. **IEEE Transactions on Circuit and Systems**. 31(6). pp. 574-581.

Glover F. (1986). Future Paths for Integer Programming and Links to Artificial Intelligence. **Computers and Operations Research**. 13. pp. 533-549.

Glover F. (1989). Tabu Search – Part I. (1989). **ORSA Journal on Computing**. 1(3). pp. 190-206.

Glover F. (1990). Tabu search – Part II. (1990). **ORSA Journal on Computing**. 2(1). pp. 4-32.

Glover F., Laguna M. and Marti R. (2000). Fundamentals of Scatter Search and Path Relinking. **Control and Cybernetics**. 39(3) pp. 653-684.

Goh T. B, Li Z., Chen B. M., Lee T. H. and Huan T. (2001). Design and Implementation of a Hard Disk Drive Servo System Using Robust and Perfect Tracking Approach. IEEE Transactions on Control Systems Technology. 9(2). pp. 221-233.

Golnaraghi F. and Kuo B.C. (2009). **Automatic Control Systems** (9th ed.). Wiley & Sons, Incorporated, John

Golub G.H., Nash S. and Van Loan C. (1979). A Hessenberg-Schur Method for Problem AX+XB=C. **IEEE Transactions on Automatic Control**. Ac-24(6). pp. 909-913.

Graovac D. and Purschel M. (2009). IGBT Power Losses Calculation Using the Data-Sheet Parameters. **Application Note in Infineon**. 1.1, pp. 1-17.

Grosman B. and Lewin D. R. (2008). Lyapunov-based Stability Analysis Automated by Genetic Programming. **Automatica**. 45. pp.252-256.

Hamalainen W. (2006). **Class NP, NP-complete and NP-hard problems**. Lecture online. pp. 1-7.

Holland J.H. (1975). **Adaptation in Natural and Artificial Systems**. Ann Arbor, USA: The University of Michigan Press.

Holland J.H. (1992). **Genetic Algorithms**. Scientific American, July. pp. 62-72.

Huang X., Horowitz R. and Li Y. (2005). Track Following Control with Active Vibration Damping and Compensation of a Dual-Stage Servo System. **Microsystem Technology**. pp.1276-1286.

IGW60T120 IGBT. (2009). **Infineon Power Semiconductors**.

Infolytica Corporation©. (2010). http://www.infolytica.com.

Isayed B. M. and Hawwa M. A. (2007). A Nonlinear PID Control Scheme for Hard Disk Drive Servosystems. **IEEE Conference on Control and Automation**, ISBN: 978-1-4244-1282-2. pp. 1-6.

Jat S. A. and Yang S. (2011). A Hybrid Genetic Algorithm and Tabu Search Approach for Post Enrolment Course Timetabling. **Journal of Scheduling**. 14(6). pp. 617-637.

Kann V. (1992). **On the Approximability of NP-complete Optimization Problems**. Doctoral thesis, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden.

Karimkashi S. and Kishk A.A. (2010). Invasive Weed Optimization and its Features in Electromagnetics. **IEEE Transactions on Antenna and Prograntion**. 58(4). pp. 1269-1278.

Katsigiannis Y. A., Georgilakis P.S. and Karapidakis E. S. (2012). Hybrid Simulated Annealing-Tabu Search Method for Optimal Sizing of Autonomous Power Systems with Renewables. **IEEE Transactions on Sustainable Energy**. 3(3). pp. 330-338.

Kennedy J. and Eberhart R. (1995). Particle Swarm Optimization. **Proceedings of IEEE International Conference on Neural Networks IV**. pp. 1942-1948.

Khalil H. K. (1992). **Nonlinear Systems**, New York: MacMillan.

Khwan-on S. (2011). **Fault Tolerant Matrix Converter Motor Drives**. Doctoral thesis, University of Nottingham, Nottingham, UK.

Kim D. H., Abraham A. and Cho J. H. (2007). A Hybrid Genetic Algorithm and Bacterial Foraging Approach for Global Optimization. **Information Sciences**. 177(18). pp. 3918-3937.

Kirkpatrick S., Gellat C. and Vecchi M. (1983). Optimization by Simulated Annealing. **Science**. 220. pp. 671-680.

Kluabwang J., Puangdownreong D. and Sujitjorn S. (2009). Performance Assessment of Search Management Agent Under Asymmetrical Problems and Control Design Applications. **WSEAS Transactions on Computers**. 8(4). pp. 691-704.

Kluabwang J. Thomthong T. (2012). Solving Parameter Identification of Frequency Modulation Sounds Problem by Modified Adaptive Tabu Search under Management Agent. **International Conference on Advances in Computational Modeling and Simulation**. 31. pp. 1006-1011.

Koerner O., Brand J., and Rechenberg K. (2005). Energy Efficient Drive System for a Diesel Electric Shunting Locomotive. **Conference proceedings on Power Electronics and Applications**. pp. 1-10.

Krause P.C., Wasynczuk O. and Sudhoff S.D. (2002). **Analysis of Electric Machinery and Drive Systems**. Wiley-IEEE Press.

Krishnan R. 2010. **Permanent Magnet Synchronous and Brushless DC Motor Drives**. CRC Press-Taylor & Francis Group, LLC.

Kulworawanichpong T. and Sujitjorn S. (2002). Optimal Power Flow Using Tabu Search. **IEEE Power Engineering Review**. 22(6). pp. 37-40.

Kulworawanichpong T., Areerak K.-N, Areerak K.-L. and Sujitjorn S. (2004). Harmonic Identification for Active Power Filters via Adaptive Tabu Search Method. **LNAI3215-PartIII**. Springer-Verlag. pp. 687-694.

Lenwari W., Sumner M., and Zanchetta P. (2009). The Use of Genetic Algorithms for the Design of Resonant Compensators for Active Filters. **IEEE Transactions on Industrial Electronics**. 56(8). pp. 2852-2861.

Lidozzi A., Solero L., Crescimbini F., Burgos R. (2008). Vector Control of Trapezoidal Back-EMF PM Machine using Pseudo-Park Transformation. **IEEE Conference on Power Electronics Specialists**. pp. 2167-2171.

Li M. S., Tang W. J., Tang W. H., Wu Q. H. and Saunders J. R. (2007). Bacteria Foraging Algorithm with Varying Population for Optimal Power Flow. **LNCS4448-Applications of Evolutionary Computing**. Springer-Verlag. pp. 32-41.

Lin W., Cheng F. and Tsay M. (2001). Nonconvex Economic Dispatch by Integrated Artificial Intelligence. **IEEE Transactions on Power Systems**. 16(2). pp. 307-311.

Liu X., Liu J. and Lim C.K. (2003). Fem and Experimental Analysis of the Actuator 'Butterfly' Mode in a Hard-Disk Drive. **Mechanical Systems and Signal Processing.** 17(5). pp. 955-964.

Liu Y. and He X. (2005). Modeling identification of power plant thermal process based on PSO algorithm. **Proceedings of the 2005 American Control Conference**, Portland, OR, USA. 7. pp. 4484-4489.

Liu Y. and Passino K.M. (2002). Biomimicry of Social Foraging Bacteria for Distributed Optimization: Model, Principle and Emergent Behaviors. **Journal of Optimization Theory and Applications**. 115(3). pp. 603-628.

Li Y., Horowitz R. and Evans R. (2003). Vibration Control of a PZT Actuated Suspension Dual-Stage Servo System Using a PZT Sensor. **IEEE Transactions on Magnetics**. 39(2). pp. 932-937.

Majhi R., Panda G., Majhi B. and Sahoo G. (2009). **Efficient Prediction of Stock Market Indices Using Adaptive Bacterial Foraging Optimization (ABFO) and BFO Based Techniques**. Elsevier. 36(6). pp. 1-8.

Malesani L., Rossetto L., Tenti P. and Tomasin P. (1995). AC/DC/AC PWM Converter with Reduced Energy Storage in the DC Link, **IEEE Transactions on Industry Applications**. 31(2). pp. 287-292.

Mallahzadeh A. R., Es'haghi S., and Alipour A. (2009). Design of an E-Shaped MIMO Antenna Using IWO Algorithm for Wireless Application at 5.8 GHz. **Progress in Electromagnetics Research**. pp. 187-203.

Mamun A. A., Lee T.H. and Low T.S. (2002). **Frequency Domain Identification of Transfer Function Model of A Disk Drive Actuator**. Elsevier. 12(4). pp. 563-574.

Martin O.C. and Otta S.W. and Felten E.W. (1992). Large-Step Markov Chains for the TSP: Incorporating Local Search Heuristics. **Operation Research Letter**, 11. pp. 219-224.

MathWorks. (2005). **Genetic Algorithm and Direct Search Toolbox: for Use with MATLAB, User's Guide,** version 1. MathWorks Inc., Natick, MA.

MathWorks. (2008). **Parallel Computing Toolbox™ User's Guide**, version 4, MathWorks Inc., Natick, MA.

McKenney P.E. (2014). **Is Parallel Programming Hard, And, If So, What Can You Do About It?**. Linux Technology Center IBM Beaverton.

Mehrabian A.R and Lucas C. (2006). A Novel Numerical Optimization Algorithm Inspired from Weed Colonization. **Ecological Informatics**. 1. pp. 355-366.

Mehrabian A.R. and Yousefi-Koma. (2007). Optimal Positioning of Piezoelectric Actuators on a Smart in Using Bio-Inspired Algorithms. **Aerospace Science and Technology**. 11. pp. 174–182.

Mehrabian A.R. and Yousefi-Koma. (2009). A Novel Technique for Optimal Placement of Piezoelectric Actuators on Smart Structures. **Journal of the Franklin Institute** 348(1). pp. 12-23.

Mishra S. (2005). A Hybrid Least Square-Fuzzy Bacterial Foraging Strategy for Harmonic Estimation. **IEEE Transactions on Evolutionary Computation**. 9(1). pp. 61-73.

Mishra S. and Bhende C. N. (2007). Bacterial Foraging Technique-Based Optimized Active Power Filter for Load Compensation. **IEEE Transactions on Power Delivery**. 22(1). pp. 457-465.

Munoz M. A., Lopez J. A. and Caicedo E. (2007). Bacteria Foraging Optimization for Dynamical Resource Allocation in a Multi Zone Temperature Experimentation Platform, **Analysis and Design of Intelligent Systems Using Soft Computing** Techniques. 41, pp.427-435.

MUR8100E diode. 2002. **Fairchild Semiconductor datasheet**.

Ngernbaht W., Areerak K.-L. and Sujitjorn S. (2009). Resonance and Friction Compensation in a Micro Hard Drive. **WSEAS Transactions on Systems**. 8(2). pp. 179-188.

Normey-Rico J.E. and Camacho E.F. (2007). **Control of Dead-Time Processes**. Springer-Verlag, London.

Noroozian M., Edris A.-A., Kidd D. and Keri A.J.F. (2003). The Potencial use of Voltage-Sourced Converter-Based Back-to-Back Tie in Load Restorations, **IEEE Transaction on Power Delivery**. 18. pp. 1416-1421.

Nowicki E. and Smutnicki C. (1996). A Fast Tabu Search Algorithm for the Flow Shop Problem. **European Journal of Operational Research**. 91. pp. 160-175.

Oftadeh R., Mahjoob M.J. and Shariatpanahi M. (2010). A Novel Meta-Heuristic Optimization Algorithm Inspired by Group Hunting of Animals: Hunting Search. **Computers and Mathematics with Applications**. 60 (2010). pp. 2087-2098.

Panikhom S., Sarasiri N. and Sujitjorn. S. (2010). Hybrid Bacterial Foraging and Tabu Search Optimization (BTSO) Algorithms for Lyapunov's Stability Analysis of Nonlinear Systems. **International Journal of Mathematics and Computes in Simulation**. 4(3). pp. 81-89.

Passino K.M. (2002). Biomimicry of Bacterial Foraging for Distributed Optimization and Control. **IEEE Control Systems Magazine**. 22. pp. 52-67.

Pena R., Clare J.C., Asher G. M. (1996). Doubly Fed Induction Generator using Back-to-Back PWM Converters and Its Application to Variable-Speed Wind-Energy Generation**. IEE Proceedings on Electric Power Applications**. 143(3). pp. 231-241.

Peng K., Chen B. M., Cheng G. and Lee T.H. (2005). Modeling and Compensation of Nonlinearities and Friction in A Micro Hard Disk Drive Servo System with Nonlinear Feedback Control. **IEEE Transactions on Control Systems Technology**. 13(5). pp. 708-721.

Puangdownreong D., Areerak K.-N., Areerak K.-L., Kulworawanichpong T. and Sujitjorn S. (2005). Application of Adaptive Tabu Search to System Identification. **Proceedings on IASTED International Conference**. pp. 178-183.

Puangdownreong D., Kulworawanichpong T. and Sujitjorn S. (2004). Finite Convergence and Performance Evaluation of Adaptive Tabu Search. **LNCS3215-Knowledge-Based Intelligent Information and Engineering Systems**. pp 710-717.

Puangdownreong D. and Sujitjorn S. (2006). Image Approach to System Identification. **WSEAS Transactions on Systems**. 5(5). pp. 930-938.

Puangdownreong D., Sujitjorn S. and Kulworawanichpong T. (2004). Convergence Analysis of Adaptive Tabu Search. **Science Asia: Research Article**. 30(2004). pp. 183-190.

Puangdownreong D., U-Thaiwasin C. and Sujitjorn S. (2006). Optimized Performance of a 2-mass Rotary System Using Adaptive Tabu Search. **WSEAS Transactions on Systems**. 5(3). pp. 339-345.

Rodriguez J., Dixon J., Espinoza J. and Lezana P. (2005). PWM Regenerative Rectifiers: State of the Art, **IEEE Transactions on Industrial Electronics**. 52. pp. 5-22.

Roosta S.H. (1999). **Parallel Processing and Parallel Algorithms: Theory and Computation**. Springer-Verlag, New York, Berlin Heidelberg.

Russell S. and Norvig P. (1995). **Artificial Intelligence: A Modern Approach**. Upper Saddle River: Prentice Hall.

Saber A. Y. and Venayagamoorthy G. K. (2008). Economic Load Dispatch using Bacterial Foraging Technique with Particle Swarm Optimization Biased Evolution. **Proceedings IEEE Swarm Intelligence Symposium**, St. Louis MO, USA. pp. 1-8.

Sarasiri N. and Sujitjorn S. (2011). Control Design Optimization of Truck Braking System using Bacterial-Foraging-Tabu-Search Metaheuristics. **World Academy of Science**, **Engineering and Technology**. 80. pp. 1189-1193.

Sarasiri N., Carawukh S., Boonpranchoo V., Kumbla K. and Sujitjorn S. (2011). Identification of Hard-Disk Head Actuator Model Using Bacterial Foraging-Tabu Search Metaheuristics. **American Journal of Scientific and Industrial Research**. 2(4). pp. 686-689.

Sarasiri N., Srikaew A. and Sujitjorn S. (2010). Dynamic Compensation of Hard-Disk R/W Head and Head-Stack, **WSEAS Transactions on Systems,** 9(7), pp. 764-773.

Sarasiri N., Suthamno K. and Sujitjorn S. (2012). Bacterial Foraging-Tabu Search Metaheuristics for Identification of Nonlinear Friction Model. **Journal of Applied Mathematics**. 2012(238563). pp. 1-23.

Shao L. and Chen L. (2009). Motif Discovery Using Evolutionary Algorithms. **Proceedings International Conference on Soft Computing and Pattern Recognition**. Malacca. pp. 420-425.

Slotine J.J.E. and Li W. (1991). **Applied Nonlinear Control**. Prentice Hall Inc., Englewood Cliffs.

Sorensen D. C. and Zhou Y. (2003). Direct Method for Matrix Sylvester and Lyapunov Equations. **Journal of Applied Mathematics**. 6. pp.277-303.

Sriyingyong N. and Attakitmongcol K. (2006). Wavelet-Based Audio Watermarking Using Adaptive Tabu Search. **Proceedings IEEE. 1st International Symposium on Wireless Pervasive Computing**. pp. 1-5.

Sujitjorn S. (2003). **Automatic Control**. Pearson Education Indochina. (in Thai)

Sujitjorn S. and Khawn-on S. (2006). Learning Control via Neuro-Tabu-Fuzzy Controller. **LNAI4251**. pp. 833-840.

Sujitjorn S., Kluabwang J., Puangdownreong D. and Sarasiri N. (2010). Adaptive Tabu Search and Management Agent, **ECTI Transactions on Electrical Engineering, Electronics and Communications**, 8(1), pp. 1-10.

Sujitjorn S., Kulworawanichpong T., Puangdownreong D. and Areerak K-N. (2006). Adaptive Tabu Search and Applications in Engineering Design. **Book Chapters in Integrated Intelligent Systems for Engineering Design (ed. X. F. Zha and R.J. Howlett)**, **IOS Press**, The Netherlands. pp. 233-257.

Suthamno K. (2004). **Parameter Estimation of Nonlinear Friction Model for A Linear Slide Bed**. Master's Thesis, School of Electrical Engineering, Suranaree University of Technology, Nakhon Ratchasima, Thailand.

Taghirad H. D. and Jamei E. (2008). Robust Performance Verification of Adaptive Robust Controller for Hard Disk Drives. **IEEE Transactions on Industrial Electronics**. 55(1). pp. 448-456.

Talbi E.G. (2002). A Taxonomy of Hybrid Metaheuristics, **Journal of Heuristic**, 8. pp. 542-564.

Talbi E.G. (2009). **Metaheuristics**. John Wiley & Sons, Hoboken, New Jersey, USA.

Tang K. S., Man K. F., Kwong S., and He Q. (1996). Genetic Algorithms and their Applications. **IEEE Signal Processing Magazine**. 13(6). pp. 22-37.

Tang W. J., Li M. S., Wu Q. H. and Saunders J. R. (2008). Bacterial Foraging Algorithm for Optimal Power Flow in Dynamic Environments. **IEEE Transactions on Circuits and Systems** 55(8). pp. 2433-2442.

Thangiah S.R. (1999). A Hybrid Genetic Algorithm, Simulated Annealing and Tabu Search Heuristic for Vehicle Routing Problems with Time Windows. **Practical Hand book of Genetic algorithms**, Complex Coding Systems.

Torres M., Espinoza R. and Ortega R. (2004). Modeling and Control of a High Voltage Direct Current Power Transmission System Based on Active Voltage Source Converters, **Proceedings IEEE Power Electronics Specialists Conference**. pp. 1726-1731.

Tripathy M. and Mishra S. (2007). Bacteria Foraging-Based to Optimize Both Real Power Loss and Voltage Stability Limit. **IEEE Transactions on Power Systems**. 22(1). 240-248.

Tripathy M., Mishra S., Lai L.L. and Zhang Q.P. (2006). Transmission Loss Reduction Based on FACTS and Bacteria Foraging Algorithm. **LNCS4193- Parallel Problem Solving from Nature IX.**, Springer-Verlag. pp. 222-231.

Uematsu, Y., Fukushi, M. and Taniguchi, K. (2001). Development of the Pushpin Free STW. **IEEE Transactions on Magnetics**. 37(2). pp. 964-968.

Ulagammai, L., Vankatesh, P., Kannan, P.S. and Padhy N.P. (2007). Application of Bacteria Foraging Technique Trained and Artificial and Wavelet Neural Networks in Load Forecasting. **Neurocomputing**. 2659-2667.

Vilanova R. and Serra I. (1997). Realisation of Two-Degrees-of Freedom Compensators. **IEE Proceedings on Control Theory and Application**. 144(6). pp. 589-595.

Viswanathan G. M ., Afanasyev V., Buldyrev S. V., Havlin S., Luz M. G. E., Raposo E. P. and H.Eugene Stanley. (2000). Lévy Flights in Random Searches. **Physica A: Statistical Mechanics and its Applications** 282(1-2). pp. 1-12

Wang X. A. and Dayawansa. W. (1999). On Global Lyapunov Functions of Nonlinear Autonomous Systems. **Proceedings Conference on Decision and Control**. pp.1635-1639.

Wu T., Bozhko S., Asher G., Wheeler P. and Thomas D. (2008). Fast Reduced Functional Models of Electromechanical Actuators for More-Electric Aircraft Power System Study. **SAE Technical Paper**. pp. 1-10.

Xiang L., Ji-feng C., Jian-xum Q. and Shang-dong Y. (2007). Energy Transmission Modes based on Tabu Search and Particle Swarm Hybrid Optimization Algorithm. **Journal of Central South University of Technology**, Springer. 14(1). pp 144-148.

Xiuli Z., Yanchi G. (2012). An improved GRASP for Irregular Flight Recovery. **Proceedings International Conference on System Science and Engineering**, June 30-July 2, 2012, Dalian, China. pp 465-469.

Yang X. S. (2010). A New Metaheuristic Bat-Inspired Algorithm. **Journal on Nature Inspired Cooperative Strategies for Optimization**. 284. pp. 65-74.

Yang X. S. (2009). Firefly algorithms for multimodal optimization. **Stochastic LNCS5792-Algorithms: Foundations and Applications**. Springer-Verlag pp. 169-178.

Yang X. S. (2009). Harmony Search as a Metaheuristic Algorithm. **Music-Inspired Harmony Search Algorithm Studies in Computational Intelligence**. 191, pp 1-14.

Yang X. S. and Deb S. (2009). Cuckoo Search via Lévy Flights. **Proceedings IEEE World Congress on Nature & Biologically Inspired Computing**. pp. 210-214**.**

Yao X., Karady G.G., Farmer R.G. and Agrawal B.L. (2001). Estimation of Generator Excitation-System Parameters by Tabu Search. **Proceedings IEEE Power Engineering Society-Winter Meeting**. 3. pp. 1185-1190.

Zakian V. (2005). **Control systems design-A new framework**. Springer-Verlag London.

Zanchetta P., Wheeler P., Empringham L. and Clare J. (2009). Design Control and Implementation of a Three-Phase Utility Power Supply Based on the Matrix Converter. **IET Power Electronics**. 2(2). pp. 156-162.

Zhang G., Habenicht W and SpieB W.E.L. (2003). Improving the Structure of Deep Frozen and Chilled Food Chain with Tabu Search Procedure. **Food Engineering**. 60(1). pp. 67-79.

Zhanoqing H., Chengxiong M. and Jiming L. (2004). A Novel Control Strategy for VSC Based HVDC in Multi-Machine Power Systems. **Journal of Electrical and Electronics Engineering**. 4. pp. 1183-1190.

Zhaolu T. and Chuanqing G. (2008). A Numerical Algorithm for Lyapunov Equations. **Applied Mathematics and Computation** 202. pp.44-53.

Zheng C. and Wang P. (1996). Parameter Structure Identification using Tabu Search and Simulated Annealing. **Advances in Water Resources**. 19(4). pp. 215-224.

Ziegler J. G. and Nichols N. B. (1942). Optimum Setting for Automatic Controllers. **Transactions on ASME**. 64. pp. 759-768.

Zou G., Zhao Z., Yuan L., Wang X. and Lu T. (2011). Optimal Design of the Back-to-Back IGBT-Based Converter with the Concept of Systematic Safe Operating Area. **Proceedings International Conference on Electrical Machines and Systems (ICEMS)**. pp. 1-5.

APPENDIX A

HARD DISK DRIVE CONVENTIONAL CONTROL DESIGNS AND ANTI-RESONANCE FILTERS

A.1 Notch Filter for Single R/W Head

The frequency responses of the R/W heads contain several resonance modes as illustrated by Figure 7.6(b). If they are not properly treated beforehand, some of them will remain after compensations, and seriously destabilize the R/W heads due to high frequency noise and disturbance. In particular, the suspension's structural vibration occurs during the track-following mode since the resonance is excited during the track-seeking operation (Chen et al., 2006; Huang et al., 2005). One approach to solve this resonance problem is to use a notch or an anti-resonance filter (Golnaraghi and Kuo, 2009) of the form as in equation (A.1).

$$G_n(s) = \frac{s^2 + 2\zeta_n \omega_0 s + \omega_0^2}{s^2 + 2\zeta_0 \omega_0 s + \omega_0^2} \tag{A.1}$$

where ω_0 is an estimated resonant frequency, and ζ_0, ζ_n are dominator and numerator damping factors, respectively. The design herein follows the procedures described by Golnaraghi and Kuo, 2009. Referring to the responses in Figure 7.6(b), the dominant resonant frequencies are at 5.89, 6.64 and 14.10 kHz, respectively. Substitute $\zeta_0 = 1$, $\zeta_n = 0.1$ and the resonant frequencies into equation (A.1), the obtained notch filter can be expressed by

$$G_{nsh}(s) = \left(\frac{s^2 + 7402s + 1.37\times10^9}{s^2 + 74020s + 1.37\times10^9}\right)\left(\frac{s^2 + 8344s + 1.741\times10^9}{s^2 + 83440s + 1.741\times10^9}\right)\left(\frac{s^2 + 1.772\times10^4 s + 7.849\times10^9}{s^2 + 1.772\times10^5 s + 7.849\times10^9}\right) \tag{A.2}$$

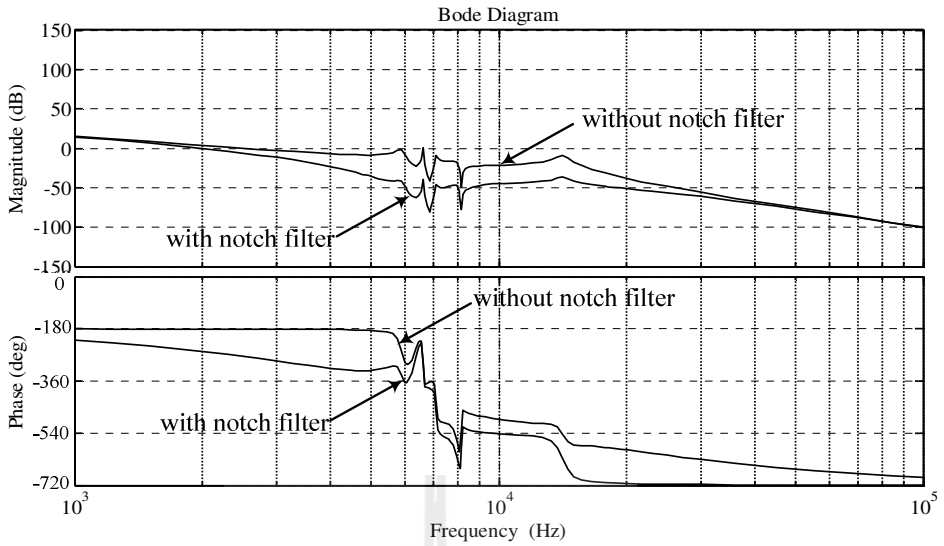


Figure A.1 Frequency responses of the single R/W head without and with notch filter.

Figure A.1 illustrates the frequency responses of the single head. It can be clearly seen that the designed notch filter properly suppress the severe resonant frequencies. In the resonant frequency ranges after compensation, the magnitudes of the frequency responses are well below 0 dB. However, phase instability is still another problem, which is addressed in the next section.

A.2 Compensation Design for Single R/W Head

According to the use of notch or anti-resonance filter for the single R/W head, the gain and phase margins are inf. dB and -78.30 deg., respectively. This unstable phase margin must be compensated for by a phase-lead compensator. The design procedures are referred to Dorf, Bishop and Sujitjorn (Dorf and Bishop, 1995; Sujitjorn, 2003).

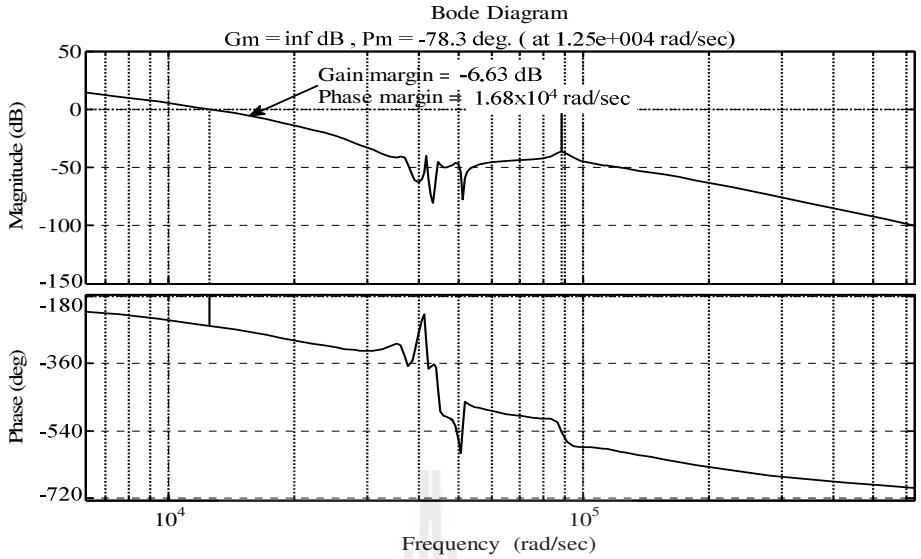


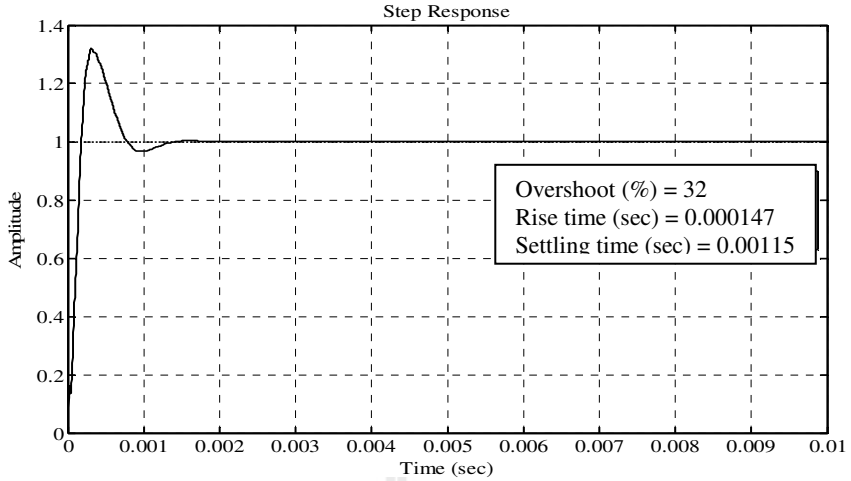
Figure A.2 Frequency responses of the single R/W head with notch filters.

Firstly, the maximum phase, $\phi_{\max 1}$, is determined. A safety factor of 5 deg. is used. The phase margins of this uncompensated system (PM_{uncomp}) and the specified phase margins (PM_{comp}) are -78.30 deg. and 50 deg., respectively. Thus, the required $\phi_{\max 1}$ is approximately 140 deg. (practically, $\phi_{\max}$ should be ≤ 75 deg.) which is considered as an excessive phase demanded. Only one compensator structure cannot handle this instability. Therefore, a multi-stage compensator is need. In this work, we consider to use 3 first-order compensators in cascade connection. Next α is calculated from $\alpha = \left(1+\sin\phi_{\max 1}\right)/\left(1-\sin\phi_{\max 1}\right) = 4.60$ in accordance with $\phi_{\max 1} = 140$ deg. The corresponding magnitude can be determined as $-10\log\alpha = -6.63$ dB, which occurs approximately at $\omega_{m1} = 1.68\times 10^4$ rad/sec (see Figure A.2). The compensator's pole and zero can be obtained from $p = \omega_{m1}\sqrt{\alpha} = 4.23\times 10^4$ rad/sec, and $z = p/\alpha = 6.67\times 10^3$ rad/sec, respectively.

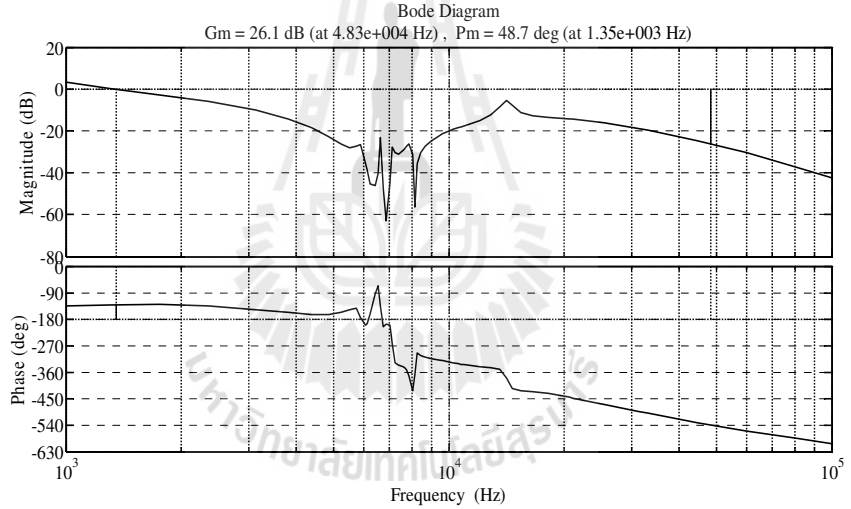
Hence, the ratio $p/z = 6.34$. This compensator somewhat theoretic renders $GM_{comp} = 21.80$ dB and $PM_{comp} = -55.20$ deg. To stabilize the phase instability, it is expected to require at least another two compensators of the same class.

The design of phase-lead compensators follows similar procedures as mentioned earlier. The specified phase margin (PM_{comp}) is used 25 deg. in this case, thus the maximum phase shifts $\phi_{\max 2}$ and $\phi_{\max 3}$ for the expressions (7.3) and (7.4) (see Chapter 7) are approximately 60 deg. and 80 deg.; their corresponding α values are 13.93 and 130.65, respectively. The magnitudes in the form of $-10\log\alpha$ are subsequently calculated equal to -11.45 dB and -21.16 dB, which are stated at $\omega_{m2} = 8.87\times10^4$ rad/sec and $\omega_{m3} = 9.78\times10^4$ rad/sec for the expressions (7.3) and (7.4), respectively. Expression (A.3) declares these 3 compensators in series connection. The total constant gain is calculated equals to 1.16×10^4 but for the appropriate one, in this system has been used 1.80×10^3 in order to decrease a high overshoot value. The third order phase-lead compensator can be expressed by equation (A.3). Step and frequency responses of the R/W single head with anti-resonance filters and the third-order phase-lead compensators are shown in Figures A.3(a) and A.3(b), respectively.

$$G_{PL1} = 1.80\times10^3\left(\frac{s + 6.67\times10^3}{s + 4.23\times10^4}\right)\left(\frac{s + 2.377\times10^4}{s + 3.311\times10^5}\right)\left(\frac{s + 8.557\times10^3}{s + 1.118\times10^6}\right) \qquad \text{(A.3)}$$



(a)



(b)

Figure A.3 Responses of the single R/W head with anti-resonance filters and the third-order phase-lead compensator; (a) step response and (b) bode plot.

Referring to Figure A.3(a), the step response of the compensated head is quite fast with zero steady-state errors although it overshoot of 32% is quite high. The

corresponding bode plot in Figure A.3(b) indicates the gain margin of 26.10 dB and the phase margin of 48.70 deg.

A.3 Anti-Resonance Filters for R/W Head-Stacks

The anti-resonance filters for the R/W head-stacks are designed as in the form of equation (A.1). Referring to Figure 7.12(b) for the head-stack, only 2 resonant frequencies are considered, which are 1.85 and 4.82 kHz, respectively. The damping factors $\zeta_0 =$ 1 and $\zeta_n =$ 0.01, are also used. The filter transfer function is shown in equation (A.4), and bode plots for the head-stacks with and without filters are illustrated in Figure A.4. It can be easily observed that notch filters suppress the resonant magnitudes to well below 0 dB.

$$G_{nhs} = \left(\frac{s^2 + 232.5s + 1.351\times10^8}{s^2 + 23250s + 1.351\times10^8} \right) \times \left(\frac{s^2 + 605.7s + 9.172\times10^8}{s^2 + 60570s + 9.172\times10^8} \right) \tag{A.4}$$

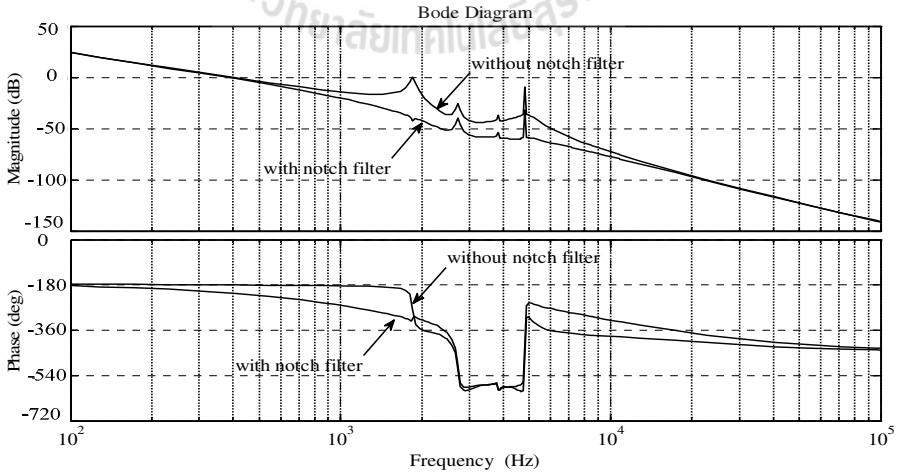
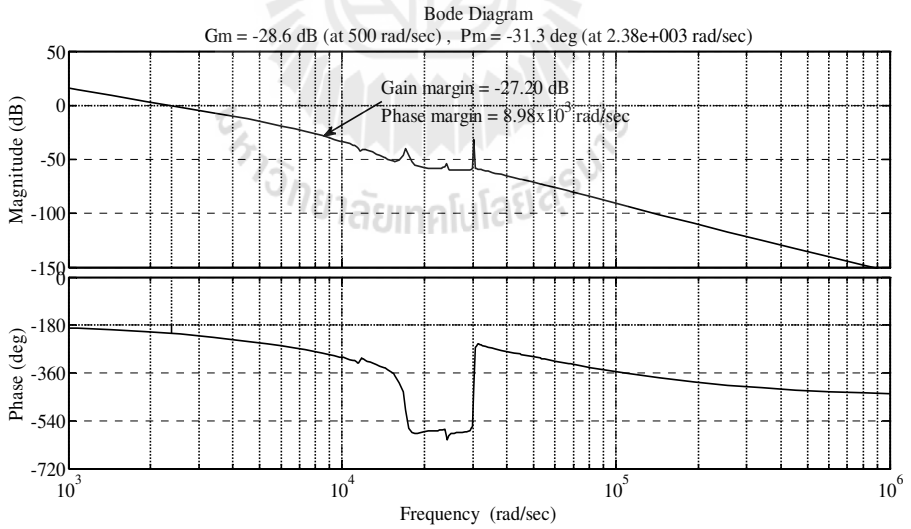


Figure A.4 Bode plots of the R/W head-stacks with and without notch filters.

A.4 Compensation design for R/W Head-Stacks

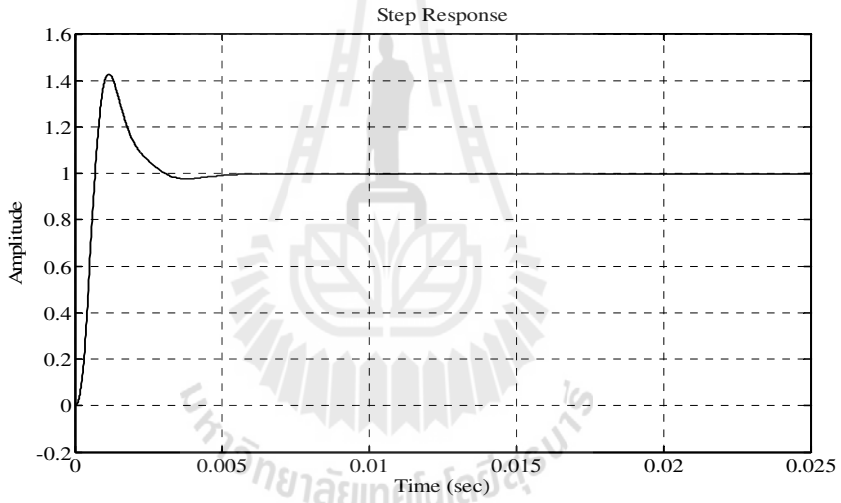
Referring to the previous design of the filters for the R/W head-stacks, this compensated system renders the gain and phase margins of -28.60 dB and -31.30 deg., respectively. These margins need good stabilization. Again, three-stage phase-lead compensators are considered. Their designs follow similar procedures described previously. So, detailed explanation is omitted herein. We start with the phase demand from the 1st compensator is set to be 85 deg. = $\phi_{\max 1}$. This results in $\alpha =$ 524.58, $\omega_{m1} = 8.98\times10^{3}$ rad/sec, $p =$ 2.78×10^{5} rad/sec, $z =$ 2.82×10^{2} rad/sec and $p/z = 958.52$. Bode plot in Figure A.5 accompanies the design. It is found that $GM_{comp} = 4.86$ dB and $PM_{comp} = 32.90$ deg. We shall incorporate two more phase-lead compensators in order to increase the *PM* up to 40 deg. or higher.



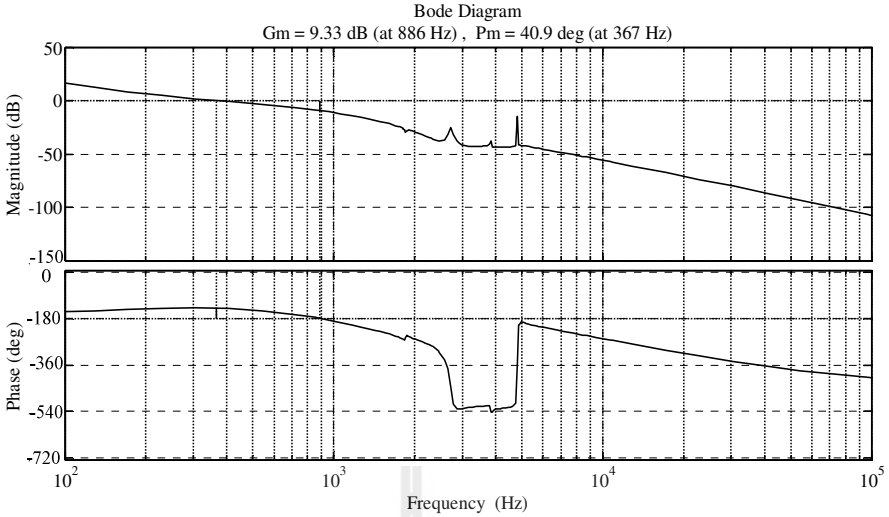
A.5 Bode plot of the R/W head-stacks with notch filters.

Regarding this need, two phase-lead compensators are designed to provide $\phi_{\max 2}$ and $\phi_{\max 3}$ = 40 deg. and 30 deg. (from expressions (7.11) and (7.14) in Chapter 7), respectively. The α values are 4.60 and 3.0 with the associative frequencies at $\omega_{m2} = 3.37\times10^{3}$ rad/sec and $\omega_{m3} = 3.08\times10^{3}$ rad/sec, respectively. Therefore, the overall transfer function of the required compensator is described by (A.5).

$$G_{PL2} = 50\left(\frac{s + 2.82\times10^{2}}{s + 2.78\times10^{5}}\right)\left(\frac{s + 1.570\times10^{3}}{s + 7.225\times10^{3}}\right)\left(\frac{s + 1.780\times10^{3}}{s + 5.341\times10^{3}}\right) \quad \text{(A.5)}$$



(a)



(b)

Figure A.6 Responses of the R/W head-stacks with anti-resonance filters and the 3-stage phase-lead compensator: (a) time-domain and (b) bode plot.

Referring to Figure A.6, the step response of the compensated head-stacks is fast with zero steady-state errors, and a large overshoot of 41% approximately. From the bode plot, we achieve $GM_{comp} = 9.33$ dB and $PM_{comp} = 40.90$ deg.

APPENDIX B

CONVENTIONAL CONTROL DESIGN FOR A HEAVY-DUTY TRUCK

The manufacturer's design specifications call for the simplest controller form, but a proportional controller is unsuitable because the contribution to the input from a sloping road, in the form of a ramp, generates a steady state error. This is avoided by choosing the controller to be of proportional-plus-integral (PI) form which can decrease the rise time and eliminate the steady-state error. In order to design the controllers, the conditioning circuits and disturbance force are omitted. For comparisons, the designs of the PI-controller apply the simple internal model control (SIMC) method, Ziegler-Nichol method, Cohen-Coon method, respectively.

B.1 SIMC method (Normey-Rico and Camacho, 2007)

The simplified model of the truck braking system is shown in Figure B.1.

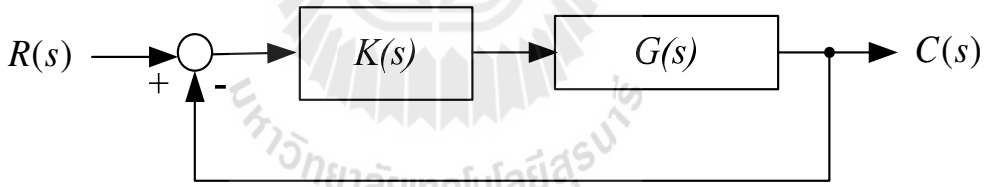


Figure B.1 Standard feedback control system.

The transfer function *K(s)* represents the controller. The transfer function *G(s)* includes the air servo valve, the power changer and the braking force generator. Due to the plant is in second-order-plus-delay-time (SOPDT) form, so a PI controller based on

the SIMC method is adapted in this case. According to the SIMC method, the plant representation is

$$G(s) = \frac{k_p e^{-Ls}}{(1+sT_1)(1+sT_2)} \tag{B.1}$$

where k_p is a nominal values of the gain defined as 1; L is dead-time equal to 0.125 seconds; T_1 is a time constant of the servo valve equal to 0.4 seconds and T_2 is a time constant of the power changer equal to 0.1 seconds.

The method considers the PI-controller of the following form

$$K_{SIMC}(s) = k_c\left(\frac{1+sK_I}{sK_I}\right) \tag{B.2}$$

in which $k_c = T_1/(2Lk_p) = 1.6$, and $K_I = \min(T_1, 8L) = 0.4$. Therefore,

$$K_{SIMC}(s) = 1.6\left(\frac{1+0.4s}{0.4s}\right) \tag{B.3}$$

B.2 Ziegler-Nichol method (Ziegler and Nichols, 1942)

The Ziegler-Nichol method is the most famous tuning method, which can be found in many introductory textbooks. Tuning PI parameters using the Ziegler-Nichol method is as follows

$$K_{ZN}(s) = K_P + \frac{K_I}{s} \tag{B.4}$$

Gains K_P and K_I are computed from

$$K_P = 0.45K_C \tag{B.5}$$

$$K_I = \frac{K_P}{T_i} \tag{B.6}$$

where K_C is the proportional gain giving a sustained oscillation. Using the proportional control action as in Figure B.2, slowly increase k_p from 0 to a critical value at which the output, *c(t)*, first exhibits sustained oscillations as Figure B.3. Thus, the critical gain, K_C and the corresponding period (time), T_0, are experimentally determined, 4.97 and 0.7 seconds, respectively. The relation $T_i = 0.83T_0$ is to obtain the time-constant, T_i.

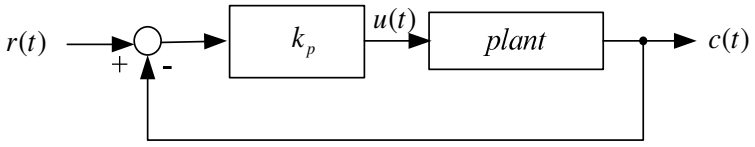


Figure B.2 Close loop system for tuning sustained oscillation.

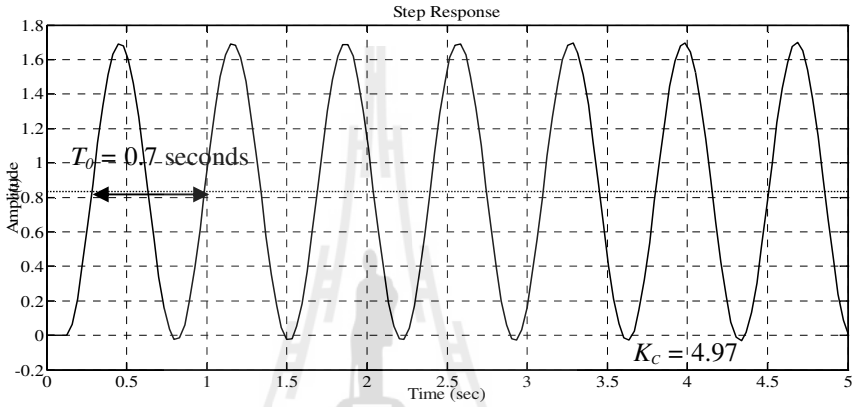


Figure B.3 Sustained oscillation with period, T_0.

Therefore,

$$K_{ZN}(s) = 2.24 + \frac{3.85}{s} \qquad \text{(B.7).}$$

B.3 Cohen-Coon method (Cohen and Coon, 1953)

Another well-known method used by industry for controller tuning is the Cohen–Coon method. This method is similar to the Ziegler-Nichols method, but

provides better results when the controller has a large dead-time relative to the time constant. The Cohen-Coon tuning is as follows:

$$K_p = \frac{0.9}{a}\left(1+\frac{0.92\tau}{1-\tau}\right) \tag{B.8}$$

$$T_i = L\left(\frac{3.3-3\tau}{1+1.2\tau}\right) \tag{B.9}$$

where $a = K_C L/T_0 = 0.8875$ and the dead-time, $\tau = L/(L+T) = 0.1515$ seconds. Note that K_C, L and T_0 are the same parameters as those in the Ziegler-Nichol method. Therefore,

$$K_{CC}(s) = 1.81 + \frac{3.92}{s} \tag{B.10}$$

APPENDIX C

NONLINEAR FRICTION MODEL AND EXPERIMENTAL SETUP

C.1 Nonlinear Friction Model

A stick-slip phenomenon occurs, when two solid materials translating over one another at very low velocity. This phenomenon is caused by nonlinear friction characteristics also known as Stribeck's effect (Armstrong-Helouvry, 1993). An effective model describing the friction can be represented by the curve in Figure C.1, and is referred to as complex friction model or Stribeck model (Armstrong-Helouvry, 1994; Cadudas, et al., 1995; Du, and Nair, 1999). When an applied force to a mass cannot overcome the static friction, which is represented by either F_{S+} or F_{S-} depending on the direction of motion, the mass cannot move. This situation is referred to as stick model, and described by the stick-friction force

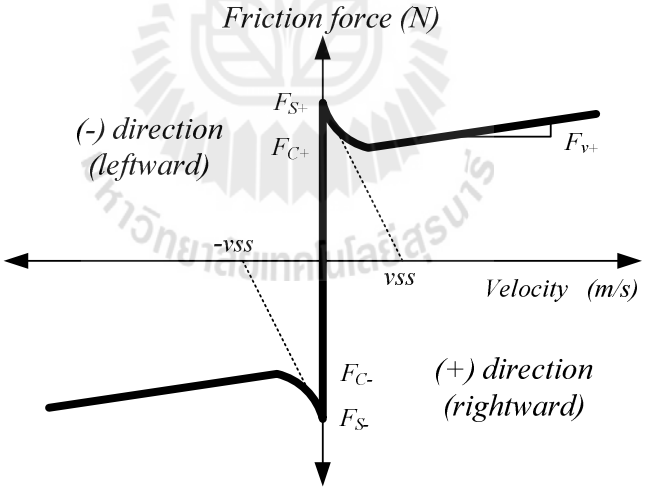


Figure C.1 Stribeck friction curve.

$$F_{stick}(F_{in}) = \begin{cases} F_{s+} & , F_{in} > F_{s+} \\ F_{in} & , F_{s-} \le F_{in} \le F_{s+} \\ F_{s-} & , F_{in} < F_{s-} \end{cases} \quad \text{(C.1)}$$

Once the applied force is greater than the static friction, the mass begins moving. After a certain period of time, the mass keeps up a higher velocity during which it encounters both coulomb and viscous frictions. This situation is known as slip mode, and described by the slip-friction force

$$F_{slip}(v) = \left(F_C + \left(F_S - F_C \right).e^{-\left(v/vss\right)} \right).\text{sgn}(v) + F_v.v \quad \text{(C.2)}$$

To cover the whole velocity range, the friction force can be expressed in a compact form as

$$F_f\ (v, F_{in}) = \begin{cases} F_{stick}(F_{in}) & , |v| = 0 \\ F_{slip}(v) & , |v| \neq 0 \end{cases} \quad \text{(C.3)}$$

C.2 Experimental Setup

A closed loop position control system is a necessary test bed for monitoring stick-slip phenomenon. The diagram in Figure C.2 represents the experimental setup (Suthamno, 2004). The linear slide bed is the controlled plant consisting of a dc motor, a threaded rod, a reflector, and an ultrasonic transducer (UC3000-UIE2). The effective moving range of the reflector is 0-400 mm with the home position at the middle. In test mode, the motion control circuit performs an initial test move of the reflector for the

whole range and, eventually, places the reflector at the home position. For the reflector to follow a ramp command, a closed loop position control has been built. The hardware components consist of a PC as a P-controller, a 12-bit ADC, a 2Q-drive circuit, a current sensor, an ultrasonic transducer, a 2nd-order differentiator producing a speed signal from a position signal, and a few signal conditioning circuits including zero-span circuits and a bipolar voltage generator, respectively, and a dc power supply. In control mode, the motion follows an up-down ramp command directing the reflector to move rightward (positive direction, ramp-up command) and leftward (negative direction, ramp-down command). The reflector moves in the range of 50-350mm in the control mode. A desired speed can be set via the keyboard of the PC functioning as a P-controller. Position of the reflector during several test runs is measured and recorded.

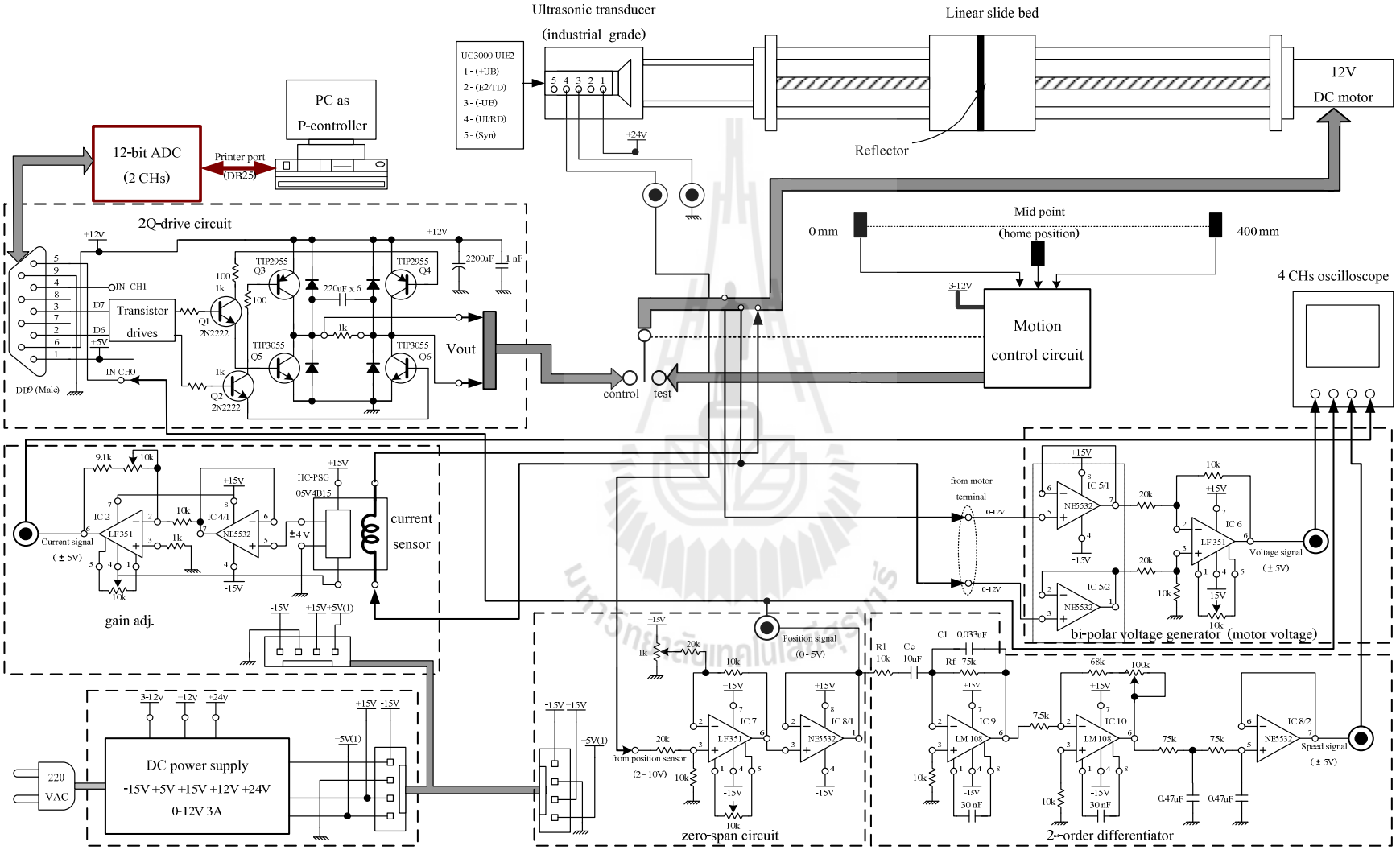


Figure C.2 Circuit diagram representing the experimental setup courtesy of K. Suthamno, 2004.

APPENDIX D

SYSTEM OF PERMANENT MAGNET SYNCHRONOUS MOTOR DRIVE

This appendix presents mathematical models of main components of the drive system. Fundamental concepts are also explained. The models are for design, simulation and optimization tasks carried out by the author.

D.1 Permanent Magnet Synchronous Motor

The three-phase windings of a PMSM are Y-connected. In the following analysis, the assumption of sinusoidally distributed windings and sinusoidal flux linkage are considered. The equivalent model of the three-phase PMSM is illustrated in Figure D.1, and the voltage equations for the stator windings are given by equations (D.1)-(D.3).

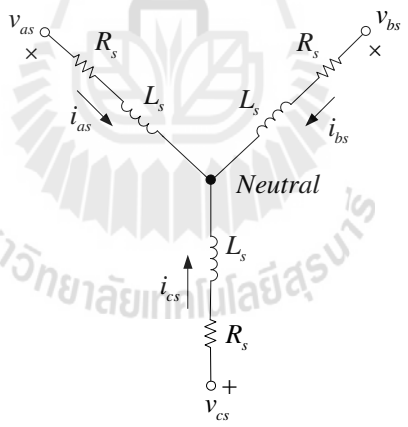


Figure D.1 The Y-connected model of the three-phase PMSM

$$v_{as} = R_s i_{as} + \frac{d\psi_{as}}{dt} \tag{D.1}$$

$$v_{bs} = R_s i_{bs} + \frac{d\psi_{bs}}{dt} \tag{D.2}$$

$$v_{cs} = R_s i_{cs} + \frac{d\psi_{cs}}{dt} \tag{D.3}$$

where v_{as}, v_{bs} and v_{cs} are the phase stator voltages; i_{as}, i_{bs} and i_{cs} are the phase stator currents; ψ_{as}, ψ_{bs} and ψ_{cs} are the flux linkages, and R_s is the resistance of each stator winding. L_s also appears in the Figure D.1, and represents the inductance of each winding. The three-phase flux linkages can be written in terms of self- and mutual-flux linkages, and the corresponding inductances and currents as

$$\psi_{as} = L_{aa} i_{as} + L_{ab} i_{bs} + L_{ac} i_{cs} + \psi_m \sin(\theta_r) \tag{D.4}$$

$$\psi_{bs} = L_{ba} i_{as} + L_{bb} i_{bs} + L_{bc} i_{cs} + \psi_m \sin(\theta_r - \frac{2\pi}{3}) \tag{D.5}$$

$$\psi_{cs} = L_{ca} i_{as} + L_{cb} i_{bs} + L_{cc} i_{cs} + \psi_m \sin(\theta_r + \frac{2\pi}{3}) \tag{D.6}$$

where L_{aa}, L_{bb} and L_{cc} are self-inductances of the a-, b- and c-phase stator windings, respectively; L_{ab}, L_{ac}, L_{ba}, L_{bc}, L_{ca} and L_{cb} are the mutual inductances between the stator phases. ψ_m is the magnitude of the flux linkage, and θ_r is the angular rotor position. The self- and mutual-inductances are found below

$$L_{aa} = L_{ls} + \bar{L}_m - L_{\Delta m} \cos(2\theta_r) \tag{D.7}$$

$$L_{bb} = L_{ls} + \bar{L}_m - L_{\Delta m}\cos(2\theta_r - \frac{4\pi}{3}) \tag{D.8}$$

$$L_{cc} = L_{ls} + \bar{L}_m - L_{\Delta m}\cos(2\theta_r + \frac{4\pi}{3}) \tag{D.9}$$

$$L_{ab} = L_{ba} = -\frac{1}{2}\bar{L}_m - L_{\Delta m}\cos(2\theta_r - \frac{2\pi}{3}) \tag{D.10}$$

$$L_{ac} = L_{ca} = -\frac{1}{2}\bar{L}_m - L_{\Delta m}\cos(2\theta_r + \frac{2\pi}{3}) \tag{D.11}$$

$$L_{bc} = L_{cb} = -\frac{1}{2}\bar{L}_m - L_{\Delta m}\cos(2\theta_r) \tag{D.12}$$

where L_{ls} is the stator leakage inductance; $\bar{L}_m$ is the average value of the magnetizing inductance, and $L_{\Delta m}$ is half the amplitude of the sinusoidal variation of the magnetizing inductance.

The most importance output variable is the electromagnetic torque which determines the mechanical dynamics of the machine, such as the rotor position and speed (Krishnan, 2010; Khwan-on, 2011). The electromagnetic torque is expressed as

$$T_e = \frac{P}{2}\psi_m\left[i_{as}\cos(\theta_r) + i_{bs}\cos(\theta_r - \frac{2\pi}{3}) + i_{cs}\cos(\theta_r + \frac{2\pi}{3})\right] \tag{D.13}$$

where P is the number of poles. Based-on the Newton's second law, the mechanical equation of the motor is given by

$$J\frac{d\omega_m}{dt} = T_e - B_m\omega_m - T_L \tag{D.14}$$

$$\frac{d\theta_m}{dt} = \omega_m \tag{D.15}$$

where ω_m is the mechanical angular velocity of the rotor; θ_m is the mechanical angular position of the rotor; J is the motor moment of inertia; B_m is the viscous coefficient; T_e is the electromagnetic torque produced by the motor, and T_L is the external load torque.

D.2 Parameters of Permanent Magnet Synchronous Motors

Engineers from many industries use Infolytica™ software for magnetic, electric, and thermal analyses. The software is used for the design and analysis of electromechanical devices, non-destructive tests (NDT), induction heating, power electronics, sensors, and industrial transformers (http://www.infolytica.com). Therefore, the parameters of a PMSM are generated by using MotorSolve, a software created by (Infolytica Corporation©, 2010). Table D.1 shows the parameters of a 10-pole PMSM.

Table D.1 Parameters of the permanent magnet synchronous motor by MotorSolve.

Parameters	10-pole machine
Magnetic flux (ψ_m)	0.0313 Wb
Mechanical torque (T_m)	10 Nm
Moment of inertia (J)	0.000538 kgm^2
Motor speed (N_m)	1,0000 rpm
Stator leakage inductance (L_s)	1.2 mH
Stator resistance (R_s)	0.124 Ω
Viscous friction coefficient (B_m)	0.0051 Nm.s

D.3 Back-to-Back Converter

The structure of a back-to-back converter is shown in Figure D.2. This research considers the 3-phase loads 10-pole permanent synchronous motors (PMSM). The basic concepts related to the back-to-back converter are analyzed. The first part of this chapter is related to the input filter design. The following sections present the *dq* analysis of the PWM rectifier and the PMSM which are used for designing the current control loops, the DC-link voltage control and the motor speed control. The conventional design explained in this thesis is used to compare with the optimum designs by using the proposed metaheuristic algorithms in Chapter 7.

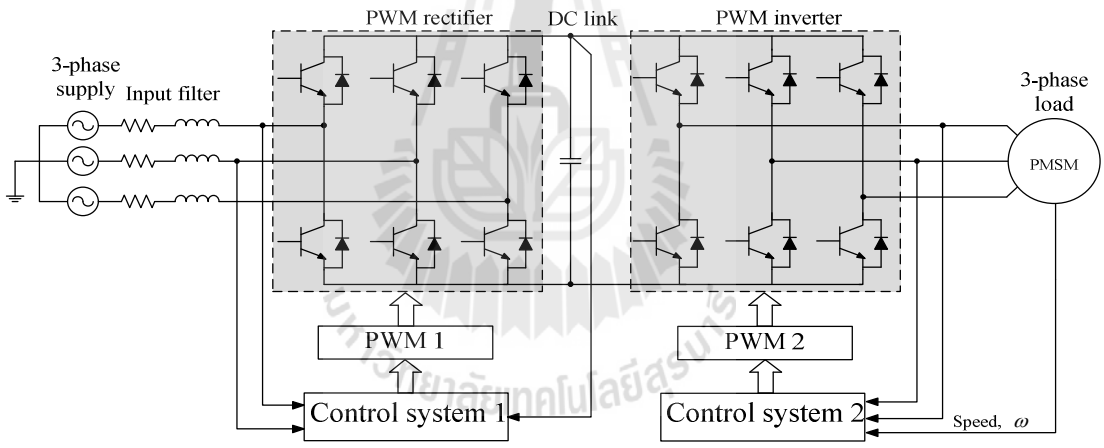


Figure D.2 Back-to-back converter topology.

The input-side filters are necessary in order to reduce the harmonic contents of the input current. The effect of power factor can be maintained at almost unity. As shown in the Figure D.2, usually series inductors-resistors are implemented to filter out the harmonics generated by the converters. Since the converters operate at high switching frequencies, the distorted input currents contain high-frequency components.

The input filters help smooth the input currents, and reduce the input voltage distortion at the same time. The line input filter is connected between the converter and the input source. The input filter inductance can be designed using equation (D.16) (Zou et al., 2011).

$$L_f \geq \frac{(2E_{DC} - 3V_s)V_s}{2E_{DC}\Delta i_{\max} f_s} \tag{D.16}$$

where E_{DC} and V_s are the DC-link voltage (1,000 V_{dc}) and input voltage (240 $V_{rms(ph)}$), respectively. According to simulation by MATLAB/SIMULINK, the input current waveforms of 10-pole machine without the input filters are shown in Figure D.3. The maximum input-side harmonic currents $\Delta i_{\max}$ appear in the fifth-order harmonic (*h*), 2.11 A_{rms}. To design the input filter inductance for this machine, the maximum $\Delta i_{\max}$ value equal to 2.11 A_{rms} is applied. The switching frequency, f_s is 20 kHz. According to equation (D.16), the input filter inductance is $L_f \geq$ 3.26 mH, and hence 3.30 mH is used. The internal resistance of this inductor is equal to 0.13 Ω (a small resistance is used in order to obtain minimum power loss) which is related to the quality factor expression $Q = X_L h / R_f = 2\pi f h L / R_f$. Most of inductors have Q values that are less than 100 when used in their intended frequency ranges, so the quality factor is selected to be 40.

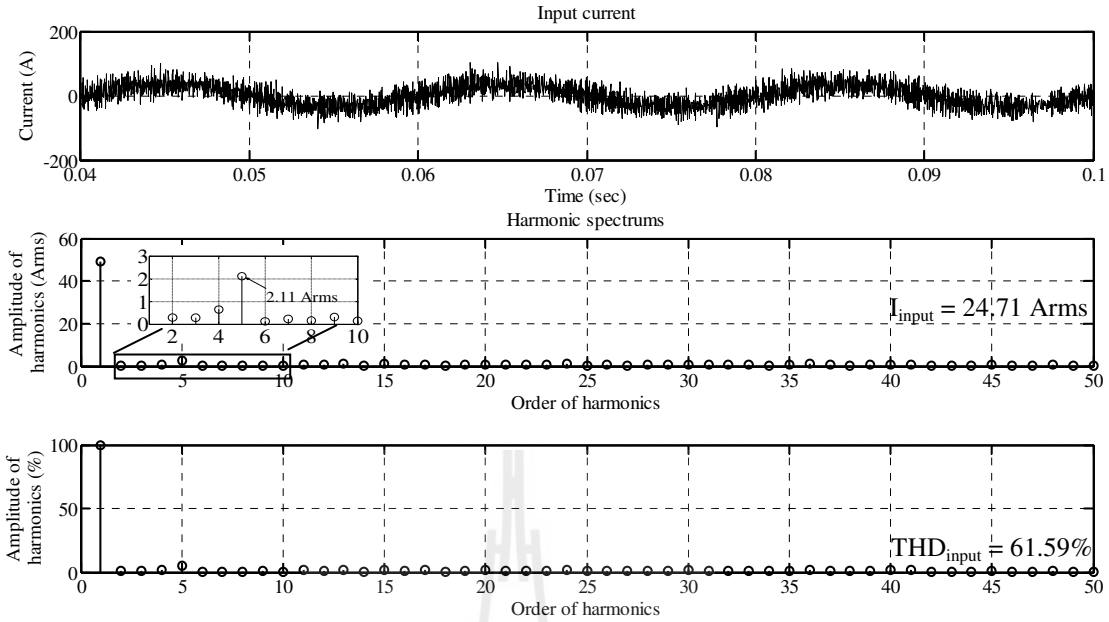


Figure D.3 Simulated input current waveform of a 10-pole machine.

D.4 DQ Analysis Model of PWM Rectifier and PMSM

The transformation matrix to convert three-phase (*abc*) to the *dq* frame with zero-sequence component can be obtained in the following forms (Rodriguez et al., 2005; Lidozzi et al., 2008)

$$\begin{bmatrix} f_\alpha \\ f_\beta \\ f_0 \end{bmatrix} = \frac{2}{3}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix}\begin{bmatrix} f_a \\ f_b \\ f_c \end{bmatrix} \tag{D.17}$$

$$\begin{bmatrix} f_d \\ f_q \\ f_0 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_\alpha \\ f_\beta \\ f_0 \end{bmatrix} \tag{D.18}$$

and the inverse transformations are given by

$$\begin{bmatrix} f_\alpha \\ f_\beta \\ f_0 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_d \\ f_q \\ f_0 \end{bmatrix} \tag{D.19}$$

$$\begin{bmatrix} f_a \\ f_b \\ f_c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & 1 \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} & 1 \end{bmatrix} \begin{bmatrix} f_\alpha \\ f_\beta \\ f_0 \end{bmatrix} \tag{D.20}$$

where f_α, f_β and f_0 represent the $\alpha-$, $\beta-$ and $0-$axis components; f_a, f_b and f_c denote either current or voltage in the three-phase *abc* frame; f_d, f_q and f_0 are the $d-$, $q-$ and $0-$axis components to be obtained. The angle θ corresponding to the supply voltage vector is calculated by

$$\theta = \tan^{-1}\frac{f_\beta}{f_\alpha} \tag{D.21}$$

To calculate the angle θ for sinusoidal waveforms, f_α and f_β can be either voltage or current. These *dq*-transformations are utilized for the control of the back-to-back converter and the PMSM.

D.4.1 The *dq* Analysis Model of PWM Rectifier

The designs these control system of the PWM rectifier is first explained in this section. The control configuration is shown in Figure D.4. The control structure consists of two internal current loops, i_d, i_q with one external DC-link voltage, E_{DC}. The three-phase supply and the input currents are transformed into the two-axis $\alpha\beta$ reference frame, and then into the rotating *dq* reference frame which is synchronized with the input impedances. E_{DC} is compared with the reference voltage E_{DC}^*, and the difference is processed by a PI-voltage controller. The reference *dq* currents, i_d^* and i_q^* are compared with the transformed currents, i_d and i_q, and the differences are processed by PI-current controllers such that v_d^* and v_q^* be generated. The reference voltages from the controllers are transformed by *dq*-/$\alpha\beta$-frame into the three-phase system. The last stage of the control system is to transform the reference voltages into a pulse width modulation pattern designed by the block PWM.

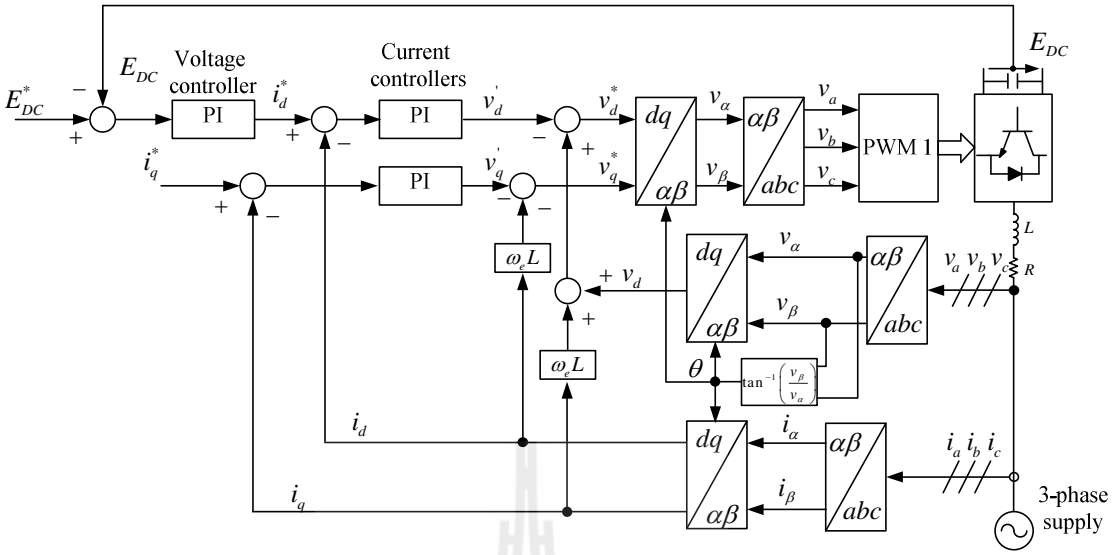


Figure D.4 Control structure for a 3-phase PWM rectifier based-on the *dq* reference frame (Wu et al., 2008).

The error between the reference and the measured DC link voltage produces the reference current, i_d^*, for the *d*-axis current. The error between this reference and the measured *d*-axis currents (i_d) generates v_d'. The voltage balance equations across the input line impedance (*R*-*L*) are expressed in terms of the input source-connected reference frame as follows

$$\begin{bmatrix} v_a \\ v_b \\ v_c \end{bmatrix} = R\begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} + L\frac{d}{dt}\begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} + \begin{bmatrix} v_a^* \\ v_b^* \\ v_c^* \end{bmatrix} \tag{D.22}$$

Using the transformation matrix to convert three-phase (*abc*) to *dq* rotating reference frame at ω_e,

$$v_d = Ri_d + L\frac{d}{dt}i_d - \omega_e Li_q + v_d^* \tag{D.23}$$

$$v_q = Ri_q + L\frac{d}{dt}i_q + \omega_e Li_d + v_q^* \tag{D.24}$$

where ω_e is the angular frequency of the rotating *dq* reference frame, v_d^* and v_q^* are the input terminal voltages in the *dq*-frame. Let $v_d^{'} = Ri_d + Ldi_d / dt$ and $v_q^{'} = Ri_q + Ldi_q / dt$, the equations (D.23) and (D.24) can be rewritten as

$$v_d^* = -v_d^{'} + \omega_e Li_q + v_d \tag{D.25}$$

$$v_q^* = -v_q^{'} - \omega_e Li_d + v_q \tag{D.26}$$

When aligning the direct axis of the rotating reference frame along the input voltage vector, v_q naturally becomes zero, and the amplitude of the supply voltage v_d is constant (Pena et al., 1996; Wu et al, 2008). Thus,

$$v_d^* = -v_d^{'} + \omega_e Li_q + v_d \tag{D.27}$$

$$v_q^* = -v_q^{'} - \omega_e Li_d \tag{D.28}$$

The above two equations support parts of the control structure are shown in Figure D.4 where v_d^* and v_q^* are generated.

D.4.2 The *dq* Analysis Model of PMSM

The control system of PWM inverter is based-on the *dq* analysis model of the PMSM. The transformation matrix to convert the three-phase variable stationary components to the *dq* frame ones is given as follows

$$\begin{bmatrix} f_d \\ f_q \\ f_0 \end{bmatrix} = \frac{2}{3}\begin{bmatrix} \cos(\theta) & \cos\left(\theta-\frac{2\pi}{3}\right) & \cos\left(\theta+\frac{2\pi}{3}\right) \\ -\sin(\theta) & -\sin\left(\theta-\frac{2\pi}{3}\right) & -\sin\left(\theta+\frac{2\pi}{3}\right) \\ 0.5 & 0.5 & 0.5 \end{bmatrix}\begin{bmatrix} f_a \\ f_b \\ f_c \end{bmatrix} \tag{D.29}$$

The inverse transformation is shown in equations (D.30).

$$\begin{bmatrix} f_a \\ f_b \\ f_c \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 1 \\ \cos\left(\theta-\frac{2\pi}{3}\right) & -\sin\left(\theta-\frac{2\pi}{3}\right) & 1 \\ \cos\left(\theta+\frac{2\pi}{3}\right) & -\sin\left(\theta+\frac{2\pi}{3}\right) & 1 \end{bmatrix}\begin{bmatrix} f_d \\ f_q \\ f_0 \end{bmatrix} \tag{D.30}$$

where θ is the angle between the rotor *d*-axis and the *q*-axis of the three-phase reference frame or a flux position angle as shown in Figure D.5.

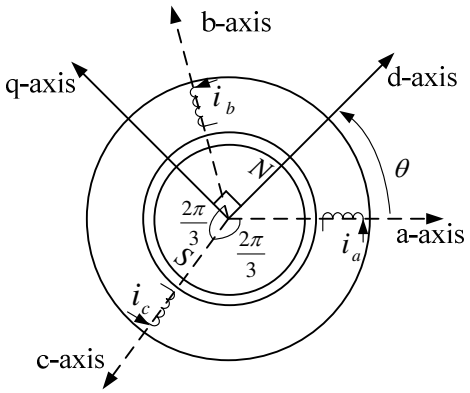


Figure D.5 Transformation between the *abc* stationary reference frame and the *dq* rotating reference frame.

Using the transformation in equation (D.29), the machine voltage equations in the *dq* rotating frame can be written as (Krause et al., 2002)

$$v_d = R_s i_d + \frac{d}{dt}\psi_d - \omega\psi_q \tag{D.31}$$

$$v_q = R_s i_q + \frac{d}{dt}\psi_q - \omega\psi_d \tag{D.32}$$

where v_d, v_q are the *d*- and the *q*-axis stator voltages; i_d, i_q are the *d*- and the *q*-axis stator currents; ψ_d, ψ_q are the *d*- and the *q*-axis stator flux quantities; R_s is the stator resistance, and ω is the angular velocity of the rotor. The flux linkages of the stator and the rotor circuits are given by

$$\psi_d = L_d i_d + \psi_m \tag{D.33}$$

$$\psi_q = L_q i_q \tag{D.34}$$

where L_d, L_q are the *d*- and the *q*-axis inductances, and ψ_m is the flux linkage of the permanent magnet mounted on the rotor shaft.

The electromagnetic torque is determined as (Wu et al., 2008)

$$T_e = \frac{3P}{4}\left[\left(L_d - L_q\right) i_d i_q + \psi_m i_q\right] = k_t i_q \tag{D.35}$$

where T_e is the electromagnetic torque of the motor; *P* is the number of poles of motor, and k_t is the torque constant of the motor.

The above *dq*-frame models of the PMSM are useful for control of an inverter driving a PMSM. Following (Wu et al., 2008), the closed-loop control system is represented by the block diagram in Figure D.6. This block diagram is the second control system according to (Wu et al., 2008) in Figure D.2. There are two internal current control loops (i_d, i_q), and one external speed control loop, ω.

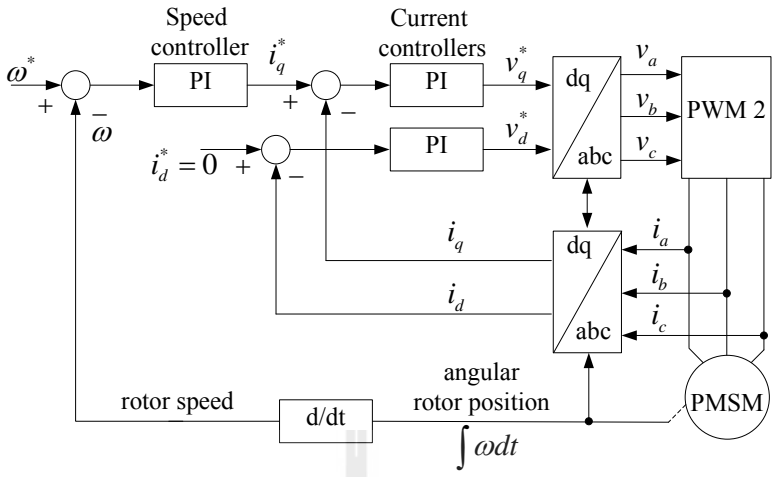


Figure D.6 Block diagram of three-phase PWM inverter controllers for the PMSM drive based-on the *dq* reference frame (Wu et al., 2008).

The speed reference signal, ω^* is compared to the actual speed of the motor, which is derived from the rotor position. The motor speed is controlled by a PI controller generating the reference *q*-axis current, i_q^*. The reference *dq* currents, i_d^* and i_q^* are compared with the transformed currents, i_d and i_q, and the errors are processed by the PI-current controllers such that v_d^* and v_q^* be generated. To obtain the maximum electromagnetic torque of the motor, i_d^* is equal to zero (Pena et al., 1996; Wu et al., 2008). After the reference voltages v_d^* and v_q^* are transformed into three-phase frame, the last stage of the control system is to transform the reference voltages into a pulse width modulation pattern in order to generate the gate signals for the converter.

D.5 Control Designs of Back-to-Back Converter and PMSM

D.5.1 Control of Back-to-Back Converter

As in Figure D.4, the reference voltages, v_d^* and v_q^* are produced by the corresponding current controllers. Since the *d*-axis of the *dq* rotating reference frame is aligned with the input voltage, which forces v_q to become zero, the active and reactive power will be proportional to i_d and i_q, respectively. The DC-link voltage control should be stabilized regardless of the output power, therefore, the DC-link voltage error is processed by a proportional-integral (PI) voltage controller to obtain the *d*-current reference. The *q*-axis current reference is set to zero if unity power factor is required.

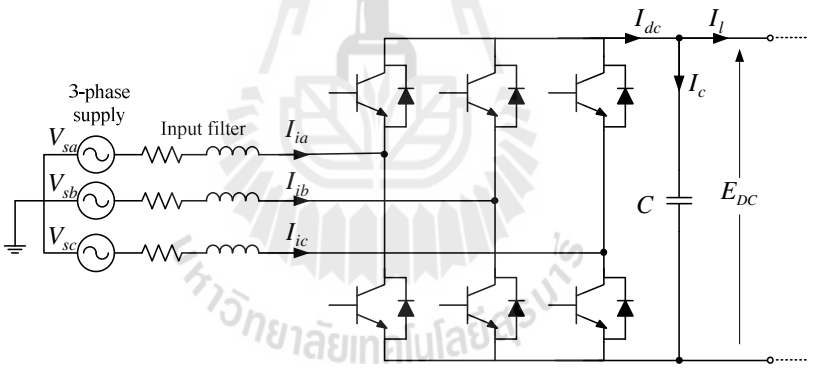


Figure D.7 Three-phase PWM rectifier.

The relationship of the DC side of the PWM rectifier and the DC link capacitor is considered as represented by the diagram in Figure D.7. The power balance of the PWM rectifier without losses can be defined as

$$E_{DC} I_{dc} = 3 v_d i_d \tag{D.36}$$

The modulation index (M_I) formula can be calculated using (D.37)

$$M_I = \frac{2\sqrt{2}v_d}{E_{DC}} \tag{D.37}$$

Obtaining,

$$I_{dc} = \frac{3M_I i_d}{2\sqrt{2}} \tag{D.38}$$

According to

$$I_{dc} = I_c + I_l \tag{D.39}$$

$$C\frac{dE_{DC}}{dt} = I_{dc} - I_l \tag{D.40}$$

The design of the DC-link voltage controller can be carried out in the continuous domain, and it is assumed that the inner i_d loop is ideal and in which I_l is represented as a disturbance (Pena et al., 1996). From equation (D.40), the effective transfer function of the plant is derived as the following

$$C\frac{dE_{DC}}{dt} = I_{dc} \tag{D.41}$$

Taking the Laplace transformation to (D.41) with zero initial conditions, one can obtain

$$CsE_{dc}(s) = I_{dc}(s) \tag{D.42}$$

$$\frac{E_{dc}(s)}{I_{dc}(s)} = \frac{1}{Cs} \tag{D.43}$$

Substituting I_{dc} from (D.38) into (D.43) results in

$$\frac{E_{dc}(s)}{i_d(s)} = \frac{3M_I}{2\sqrt{2}Cs} \tag{D.44}$$

The closed-loop block diagram can be sketched as in Figure D.8.

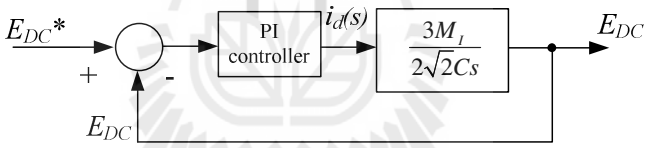


Figure D.8 DC-link voltage control loop for back-to-back converter.

The controller speed depends on the bandwidth or the nominal closed-loop natural frequency. If the bandwidth is low, the speed of the controller will be slow. Mostly, the current control loop is designed to operate at higher speed than the voltage control loop. Hence, the bandwidth used in the current controller must be greater than that in the voltage controller. The current controller design can be simplified by taking into account the fact that normally the current loops are designed with a high bandwidth of up to 300-500Hz. This requires the bandwidth of the current control loop

to be at least ten times faster than the voltage control loop (Wu et al., 2008; Zanchetta et al., 2009). Using a bandwidth for the voltage control loop, f_n = 25 Hz, damping factor, ζ = 0.8, E_{DC}^{*} = 1000 V_{dc}, V_s = 240 $V_{rms(ph)}$, modulation index, M_I = 0.7, capacitor for DC-link, C = 2.20 mF. The PI controller can be designed by using the closed-loop characteristic equation. Since the controller $G_c(s) = K_P + K_I / s$, the characteristic equation of the DC-link voltage control can be expressed as

$$s^2 + \left(\frac{3K_p M_I}{2\sqrt{2}C} \right) s + \frac{3K_I M_I}{2\sqrt{2}C} = 0 \tag{D.45}$$

Compare (D.45) to the characteristic equation of the second-order system in (D.46)

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0 \tag{D.46}$$

where ω_n is natural frequency, $2\pi f_n$, one can conclude that the controller gains are

$$K_P = \frac{4\sqrt{2}C\zeta\omega_n}{3M_I} \tag{D.47}$$

$$K_I = \frac{2\sqrt{2}C\omega_n^2}{3M_I} \tag{D.48}$$

The values of K_p and K_I of the voltage controller can be obtained as 0.74 and 73.11, respectively.

The parameters of the current control loop are determined by using the bandwidth, f_n = 500 Hz, the damping factor, ζ = 0.8, total line inductance, $L = L_{line} + L_f = 3.40\ mH$, and total line resistance $R = R_{line} + R_f = 0.18\ \Omega$. In practical system, L_{line} = 0.10 mH and $R_{line} = 0.05\,\Omega$. The system can be represented by the block diagram in Figure D.9, which is applied for both i_d and i_q.

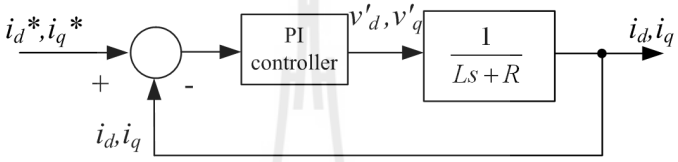


Figure D.9 Current control loops for back-to-back converter.

Similar to the voltage control loop design, the characteristic equation of the current control is considered as in (D.49)

$$s^2 + \left(\frac{K_P + R}{L}\right)s + \frac{K_I}{L} = 0 \tag{D.49}$$

Compared (D.49) to the characteristic equation of the second-order system in (D.46), the parameter gains, K_P and K_I for the current controllers can be obtained as

$$K_P = 2\zeta\omega_n L - R \tag{D.50}$$

$$K_I = \omega_n^2 L \tag{D.51}$$

Therefore, the gain values, K_P and K_I, of the current controller are 16.89 and 33,556.65, respectively.

D.5.2 Control of PMSM

Control of motor speed in various applications is usually implemented using a cascade-control principle with internal current control loops, which are commonly proportional-integral (PI) types. According to the block diagram in Figure D.6, the reference signal for the *d*-axis current component i_d^* is defined as zero to obtain the maximum electromagnetic torque of the motor. It could be assumed that the rotor flux density distribution of the PMSM is sinusoidal. Consequently, the inductances, L_d and L_q can be equal. The block diagram of speed control is shown in Figure D.10 consists of a PI controller, torque constant (k_t) and the PMSM *s*-domain mechanical model. The load torque (T_L) is considered as disturbance.

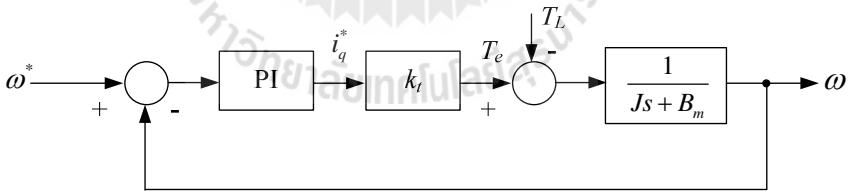


Figure D.10 PI speed control loop for PMSM.

For tracking control purposes, the speed response due to $\omega^*(s)$ is expressed by

$$\omega(s) = \frac{K_P k_t s + K_I k_t}{Js^2 + B_m s + K_P k_t s + K_I k_t} \cdot \omega^*(s) \tag{D.52}$$

The closed-loop characteristic equation is

$$s^2 + \left(\frac{B_m + K_p k_t}{J} \right) s + \frac{K_I k_t}{J} = 0 \tag{D.53}$$

where B_m is the motor's viscous coefficient, J is the motor's moment of inertia, and k_t is the motor's torque constant ($k_t = \frac{3}{2}\frac{P}{2}\psi_m$). Comparing (D.46) to (D.53), the controller gains, K_P and K_I, for the speed controller can be obtained as

$$K_P = \left(2\zeta\omega_n - \frac{B_m}{J} \right) \left(\frac{J}{k_t} \right) \tag{D.54}$$

$$K_I = \frac{\omega_n^2 J}{k_t} \tag{D.55}$$

For the speed command tracking control, the damping factor (ζ) and the bandwidth of the speed controller are chosen as 0.8 and 10 Hz, respectively. Therefore, [K_P, K_I] = [0.21, 9.05] for a 10-pole machine are obtained.

In general, for the design of current control loop the bandwidth should be at least ten times greater than the speed control loop to ensure that the currents are accurately regulated at the references with fast dynamic responses. The block diagram of the current control for *d*- and *q*-axis is shown in Figure D.11.

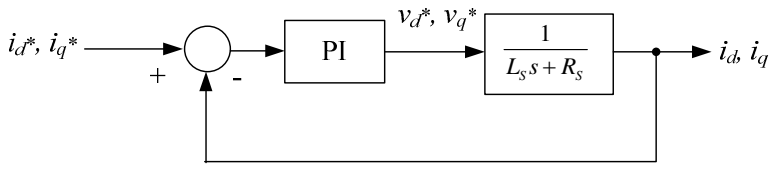


Figure D.11 PI current control loops for PMSM.

Similarly to the voltage/current control previously described, the characteristic equation of the closed-loop control is expressed by

$$s^2 + \left(\frac{K_p + R_s}{L_s} \right) s + \frac{K_I}{L_s} = 0 \tag{D.56}$$

Compare equation (D.56) to (D.46), therefore, the expressions for the controller gains can be obtained as

$$K_P = 2\zeta\omega_n L_s - R_s \tag{D.57}$$

$$K_I = \omega_n^2 L_s \tag{D.58}$$

For the design of the current controller, the bandwidth and the damping factor are chosen as 200 Hz and 0.8, respectively. Therefore, the controller parameters $[K_P,\ K_I] = [2.29,\ 1894.96]$ for a 10-pole machine are obtained.

D.6 Multi-objective Function

The multi-objective optimization is converted into one single objective using weights and summation of each objective function as briefly given by equation (D.59)

$$f_{all}(x) = \sum_{i=1}^{L} w_i f_i(x) \tag{D.59}$$

where $f_{all}(x)$ is the objective function in a surrogate form, w_i are the real non-negative weights such that $\sum_{i=1}^{n} w_i = 1$, and $f_i(x)$ is each objective function.

According to the weights, they can be chosen either arbitrarily or to satisfy some criteria. The principle of inequality (Zakian, 2005) lays a foundation of the multi-objective optimization, in which each objective, $f_i(x)$, is constrained in the form of (D.60).

$$f_i(x) \le \varepsilon_i \tag{D.60}$$

where each constant ε_i is the largest acceptable or permissible value of the objective $f_i(x)$.

To solve a multi-objective optimization problem, an effective approach is to convert the constrained problem to an unconstrained one. The unconstrained function is formed by adding the objective function, $f_{all}(x)$ with a penalty parameter, σ_k, and a measure of constrained condition, $g(f_i(x))$. In minimization case, the measure of the constrained condition can be zero, when the constraints are not violated or nonzero,

when the constraints are violated. These multi-objective functions can be formulated as a series of unconstrained minimization in (D.61).

$$\min J_{all}(x) = f_{all}(x) + \sigma_k \sum_{i=1}^{k} g(f_i(x)) \tag{D.61}$$

where $g(f_i(x))$ is the penalty function calculated by $g(f_i(x)) = \min(0, f_i(x))^2$ and σ_k is the coefficients (Courant, 1943).

Referring to power drive system, the analysis of the input and the output current qualities, the losses in the converters and the motor, the DC-link voltage, the current control loops, and the motor speed control are concerned. Thus, the task of finding solutions for multiple objective problems is necessary. As mentioned earlier, the multi-objective optimization will be applied in order to obtain several optimum solutions at the same time. The details of each objective function are described in the next sections.

D.6.1 Input and Output Current Qualities

The input and output current qualities are represented by the total harmonic distortion (THD) of the input and the motor currents.

$$THD_i = \sqrt{\frac{\sum_{n=2}^{N} I_{n,rms}^2}{I_{1,rms}^2}} \times 100 \tag{D.62}$$

where $I_{n,rms}$ denotes the rms current of the *n*-th current harmonic, n = 1 is the fundamental, and N is the total number of harmonics. A high current quality is indicated by a low THD_i which should be less than 3% following industrial requirement (a very stringent one). The R_f, L_f and L_s are obtained from the objective functions given by (D.63)-(D.64).

$$f_1(R_f, L_f) = \frac{THD_{i,input}}{3\%} \tag{D.63}$$

$$f_2(L_s) = \frac{THD_{i,motor}}{3\%} \tag{D.64}$$

Note that the fundamental frequencies have been used for line input and motor currents are 50 Hz and 833.33 Hz (referred to $N_m = 120 \times f / P$), respectively.

D.6.2 Total Power Losses

The total power losses are considered in the terms of line input loss, switching loss, conduction loss and motor losses. The details are described as follows:

- **Line input loss:**

The power loss of a 3-phase line input is calculated using (D.65)

$$P_{loss,line} = 3I_{rms}^2 \left(R_{line} + R_f \right) \tag{D.65}$$

where I_{rms} is the input line current, R_{line} and R_f are the phase resistances of the input line and the input filter, respectively. According by R_{line} and R_f are $0.05\,\Omega$ and $0.13\,\Omega$, respectively, and the input line current is $24.7055\text{Arms}_{\text{ph}}$. Thus, the calculated power line input losses for three-phase are 329.60W.

- **Switching and conduction losses:**

The power losses in an IGBT and diode can be divided in three groups; conduction losses (P_{cond}), switching losses (P_{sw}) and blocking (leakage) losses (P_b) (normally being neglected). Therefore the total losses can be described by

$$P_{loss,SC} = P_{cond} + P_{sw} + P_b \approx P_{cond} + P_{sw} \tag{D.66}$$

where conduction losses depend on both the collector-emitter saturation voltage drop of the IGBT and the forward voltage drop of a diode used to block reverse conduction of the IGBT. The IGW60T120 IGBT (Infineon Power Semiconductors, 2009) and the MUR8100E diode (Fairchild Semiconductor, 2002) are used in this research. The linearized characteristic for the conduction voltage drop as a function of current for the series combination of an IGBT and a diode is

$$P_{cond} = P_{cond,sw} + P_{cond,diode} = \left(V_{CE0} + V_{F0}\right) I_{crms} + (r_{CE} + r_D) I_{c,rms}^2 \tag{D.67}$$

where V_{CE0} is the collector-emitter voltage drop, V_{F0} is the diode forward voltage drop, r_{CE} is the linearized on-state resistance of the IGBT obtained from the datasheet diagram in terms of $\Delta V_{CE}/\Delta I_C = 0.0161\Omega$ $(V_{GE} = 15V)$, r_D is the literalized on-state

resistance of the diode according to $\Delta V_F / \Delta I_F = 0.02\Omega$ ($temperature = 25^{\circ}C$) and $I_{c,rms}$ is the collector current (Apap et al., 2003; Casanellas, 1994; Graovac and Purschel, 2009; Zanchetta et al., 2009).

The turn-on and turn-off energy losses for an IGBT and diode can be assumed to vary linearly with the change in voltage across the IGBT during the switching transients, which are considered from the device datasheets for the IGBT and diode. It is reasonable to assume that the turn-on and turn-off energies vary linearly with the collector current. The switching losses in the IGBT and diode can be calculated in a similar manner. The turn-off energy losses in the diode are normally neglected ($E_{offD} \approx 0$). Therefore:

$$P_{sw} = P_{sw,sw} + P_{sw,diode} = (E_{onT} + E_{offT}) \cdot f_s + E_{onD} \cdot f_s \tag{D.68}$$

The turn-on and turn-off energy losses of IGBT are computed by equations (D.69)-(D.70), respectively.

$$E_{onT} = e_{onT} \Delta V_{ce} \Delta I_c \tag{D.69}$$

$$E_{offT} = e_{offT} \Delta V_{ce} \Delta I_c \tag{D.70}$$

e_{onT} and e_{offT} are the turn-on and turn-off switching energy losses respectively per unit voltage and unit current for the specified devices. ΔV_{ce} is the change in the collector-emitter voltage during the switching transient, 20V, and ΔI_c is the change in the

collector current during the switching transient, 60A. Similarly the diode recovery loss is calculated by

$$E_{onD} = e_{onD} \Delta V_d \Delta I_d \tag{D.71}$$

where e_{onD} is the switch-on energy loss in the diode per unit voltage and unit current. ΔV_d is the change in the voltage across the diode, 20V, and ΔI_d is the change in the diode current, 1 A (Apap et al., 2003; Casanellas, 1994; Graovac and Purschel, 2009).

Table D.2 Parameters of conduction and switching losses from the datasheets.

V_{CE0} (V)	V_{F0} (V)	r_{CE} (Ω)	r_D (Ω)	e_{onT} (μJ/VA)	e_{offT} (μJ/VA)	e_{onD} (μJ/VA)
2.4	1.8	0.0161	0.02	0.1194	0.1444	2.50

According the details in Table D.2, the conduction losses of a 10-pole PMSM is calculated by using equation (D.67), thus, the conduction loss value is 754.89W. For the switching losses in the IGBT and diode can be calculated from equation (D.68) equal to 43.9872W.

- **Motor loss:**

The loss of PMSM in this thesis is considered to be only copper loss as this is the most significant loss. The copper loss is a function of the phase current and the phase resistance as given by the following expression

$$P_{loss,motor} = I^2_{motor,rms} R_s \tag{D.72}$$

From the stator resistances of a 10-pole PMSM is $0.1240\,\Omega$ and the motor current values is 45.8850A. Therefore, the calculated motor power losses according to (D.72) are 783.22W.

As mentioned above, the total power loss can be calculated by

$$P_{total,loss} = P_{loss,line} + P_{loss,SC} + P_{loss,motor} \tag{D.73}$$

where gives the total power losses of a 10-pole PMSM equal to 1.91kW. To obtain the minimum total power losses, the accepted maximum values, $P_{accept,\max}$ of the total power losses must be considered to be lower 10-20% which is 1.70 kW. The switching frequency f_s is the obtained variable from this objective function. This objective function is shown in equation (D.74).

$$f_3(f_s) = \frac{P_{total,loss}}{P_{accept,\max}} \tag{D.74}$$

D.6.3 Line Input Current Controller

To obtain the line input current controllers, $K_{P,line}$ and $K_{I,line}$, the objective function is considered in term of root mean square error (RMS) of closed-loop response which includes the plant and the PI controller. The RMS calculation is shown in equation (D.75).

$$RMS_{rectifier} = \sqrt{\frac{\sum_{i=1}^{N}\left(1 - y_{i,line}\right)^2}{N}} \tag{D.75}$$

where $y_{i,line}$ is the response of the line input current control and N is a number of the line input current response.

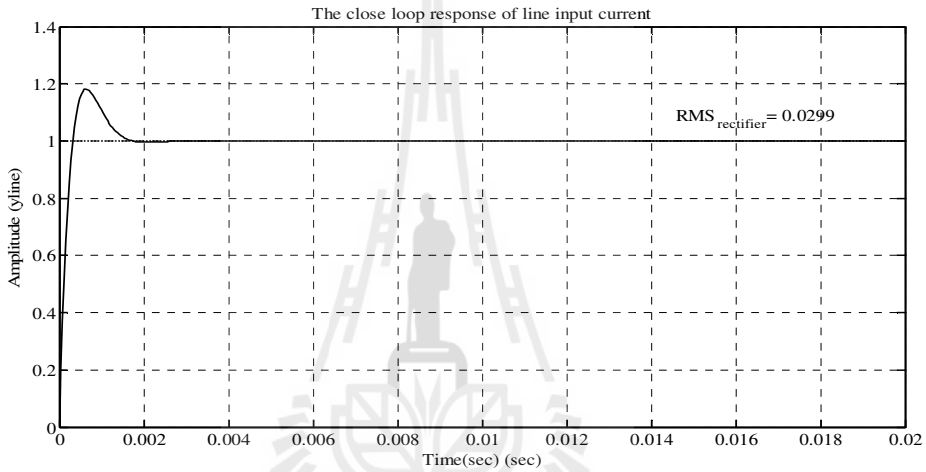


Figure D.12 Closed-loop response of line input current.

According to the closed-loop response of the conventional control design as shown in Figure D.12, the RMS value is equal to 0.0299 which is set as the maximum reference, $RMS_{rectifier,\max}$. In order to obtain a better performance of the line input current, an objective function is formulated as

$$f_4(K_{P,line},\ K_{I,line}) = \frac{RMS_{rectifier}}{RMS_{rectifier,\max}} \tag{D.76}$$

D.6.4 DC-Link Voltage Controller

The DC-link voltage controller consists of $K_{P,DC}$ and $K_{I,DC}$ parameters in order to be regulated to 1,000 V_{DC} – level. From a conventional design, we obtain a voltage response curve as shown in Figure D.13. Equation (D.77) is used for a calculation of an RMS value of the DC-link voltage.

$$RMS_{DC} = \sqrt{\frac{\sum_{i=1}^{N}\left(1{,}000 - y_{i,DC}\right)^2}{N}} \tag{D.77}$$

, where $y_{i,DC}$ is the DC-link voltage, and N is a number of the DC-link voltage data. According to the response shown in Figure D.13, the RMS value is equal to 6.9802 V_{DC} for a 10-pole PMSM, which is set as the maximum reference, $RMS_{DC,\max}$.

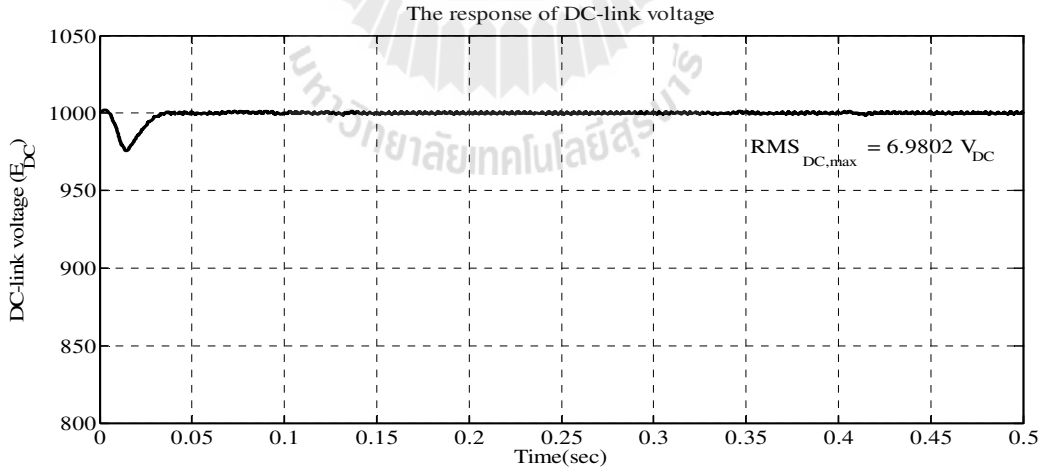


Figure D.13 DC-link voltage of a 10-pole PMSM.

For an optimized design of a DC-link voltage controller, an objective function is formulated as

$$f_5(K_{P,DC},\ K_{I,DC}\ ) = \frac{RMS_{DC}}{RMS_{DC,\max}} \tag{D.78}$$

D.6.5 Load Current Controller

In a similar manner, the PI-controller parameters of a load current controller consist of $K_{P,load}$ and $K_{I,load}$. The RMS value of a closed-loop response can be calculated by

$$RMS_{inverter} = \sqrt{\frac{\sum_{i=1}^{N}\left(1 - y_{i,load}\right)^2}{N}} \tag{D.79}$$

where $y_{i,load}$ is the response of the load current control and N is a number of the load current response data. From the responses of conventional control design as in Figure D.14, the RMS value is equal to 0.0482, which is set as the maximum reference $RMS_{inverter,\max}$.

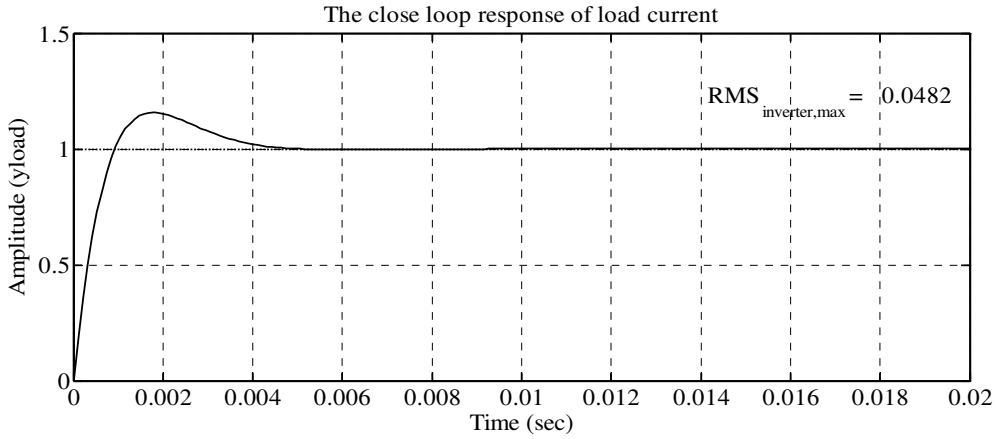


Figure D.14 Closed-loop current responses of a 10-pole PMSM.

To obtain a better performance of load current control, the objective function is formulated by

$$f_6(K_{P,load},\ K_{I,load}\ ) = \frac{RMS_{inverter}}{RMS_{inverter,\max}} \tag{D.80}$$

D.6.6 Speed Controller

The speed controller consists of $K_{P,speed}$ and $K_{I,speed}$ parameters. The objective function is considered in terms of the RMS of the motor speed, which is regulated to 10,000 rpm as shown in Figure D.15. The RMS value of speed can be calculated by

$$RMS_{speed} = \sqrt{\frac{\sum_{i=1}^{N}\left(10{,}000 - y_{i,speed}\right)^2}{N}} \tag{D.81}$$

where $y_{i,speed}$ is the speed response of motor and N is number of the speed response data. According to the responses of conventional control design, the RMS value is equal to 925.8013 rpm for a 10-pole PMSM, which is set as the maximum reference, $RMS_{speed,\max}$.

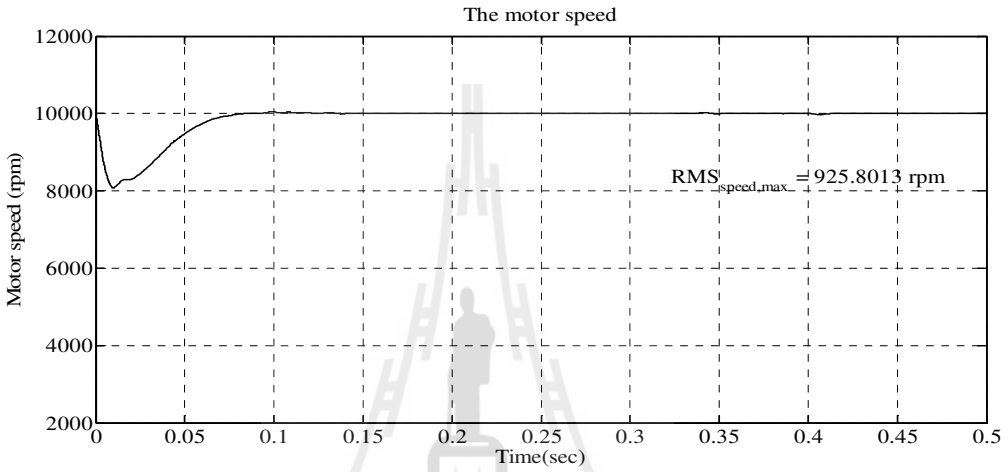


Figure D.15 Speed of a 10-pole PMSM.

The objective function is formulated as

$$f_7(K_{P,speed},\ K_{I,speed}\ ) = \frac{RMS_{speed}}{RMS_{speed,\max}} \tag{D.82}$$

APPENDIX E

PROGRAM-CODE LISTS

Computer programs have been used in this thesis based-on MATLAB coding. The program lists of the proposed BF-TS, the modified ATS, and the parallel modified ATS algorithms are detailed as follows:

E.1 Code List of the BF-TS Algorithm

The program list of the proposed BF-TS algorithm is shown in this appendix (an optimization problem of Bohachevsky 's function). This experimental program consists of BF-TS.m, initial_generation.m, obj.m, objective.m, rand1.m, bact_cellcell_attract_func.m and random_neigh.m.

```
% This BF-TS program has been developed by using MATLAB™ by Nuapett Sarasiri.
% School of Electrical Engineering, Suranaree University of Technology.

Program of BF-TS.m

function [min_cost_function,X_Y_best,overall_best_error,overall_best_neighbor,…
n_back_tracking,time,count]=BF_TS(p)
                                        % This main program uses p as a input, where p is a
                                        % number of parameter to be optimized. There are 7
                                        % output parameters: min_cost_function is an initial
                                        % objective function obtained from random-walk
                                        % frontend. X_Y_best is an initial solution obtained
                                        % from random-walk frontend. overall_best_error is
                                        % optimal cost function. overall_best_neighbor is
                                        % global optimum solution. n_back_tracking is a
```

```
                                    % number of solution occurred in local deadlock.
                                    % time is total time. count is total iteration.
rand('state',sum(100*clock));       % Initialize rand to a different state each time.
% Search parameter
S=10;                               % Number of solution in the population.
Nc=250;                             % Number of chemotactic steps.
Ns=4;                               % Limits the length of a swim when it is on a
                                    % gradient.
flag=0;                             % If flag=0 indicates that will have nutrients and
                                    % cell-cell attraction. If flag=1 indicates that will
                                    % have no (zero) nutrients and only cell-cell
                                    % attraction. If flag=2 indicates that will have
                                    % nutrients and no cell-cell attraction.
alpha=10;                           % A positive constant for step size expression.
max_count=10000;                    % A maximum iteration.
radius=0.2;                         % Search radius.
Number_neighb=30;                   % A maximum number of neighbour solution.
n=0;                                % Initial number of repeated solution.
t=0;                                % Initial sliding of local.
tt=0;                               % Initial sliding of tabu_list.
ttt=0;                              % Initial sliding of best_error_list.
count=0;                            % Initial number of iteration.
n_back_tracking=0;                  % Initial number of back tracking.
best_error=0;                       % Initial number of best_error.
best_neighbor=0;                    % Initial number of best_neighbor.
xlimit=[2 2;-2 -2];                 % Search space, row 1 of xlimit is a maximum of
                                    % parameter.
                                    % Row 2 of xlimit is a minimum of parameter.
tic;                                % Start search time.
```

```
% Randomly initial solutions from random-walk frontend by using Initial_generation function.
[min_cost_function, X_Y_best]= Initial_generation(p,S,Nc,Ns,flag,alpha,xlimit);
                                        % Input of this function are p, S, Nc, Ns, flag, alpha,
                                        % xlimit and output are min_cost_function and
                                        % X_Y_best.
S0=X_Y_best';                           % Store initial solution in S0.
best_neighbor=S0;                       % Store initial solution in best_neighbor
best_error= min_cost_function;          % Define best_error = min_cost_function.
overall_best_error=best_error;          % Define overall_best_error=best_error.
overall_best_neighbor=best_neighbor;    % Define overall_best_neighbor=best_neighbor.
t=t+1;                                  % Increase sliding of local value.
tt=tt+1;                                % Increase sliding of tabu_list.
ttt=ttt+1;                              % Increase sliding of best_error_list.
local(t,1)=count;                       % Store count in the first column of local.
local(t,2:3)=best_neighbor;             % Store best_neighbor in the second column and the
                                        % third column of local.
local(t,4)=best_error;                  % Store best_error in the fourth column of local.
tabu_list(tt,1)=count;                  % Store count in the first column of tabu_list.
tabu_list(tt,2:3)=best_neighbor;        % Store best_neighbor in the second column and the
                                        % third column of tabu_list.
tabu_list(tt,4)=best_error;             % Store best_error in the fourth column of tabu_list.
best_error_list(ttt,1)=count;           % Store count in the first column of best_error_list.
best_error_list(ttt,2:3)=best_neighbor; % Store best_neighbor in the second column and the
                                        % third column of best_error_list.
best_error_list(ttt,4)=best_error;      % Store best_error in the fourth column of
                                        % best_error_list.
while (count<max_count)                 % Process loop.
   count=count+1;
   S1=random_neigh(Number_neighb,radius,xlimit,S0);
```

```
                                % Recall random_neigh function to random among
                                % neighbour solutions around S0 within current sub
                                % search space. There are 4 input parameters such
                                % as Number_neighb, radius, xlimit and S0 and one
                                % output, S1.
[best_error1,best_neighbor1,best_error,best_neighbor]=objective(S1,best_error,S0);
                                % Recall objective function to evaluate random
                                % neighbour solutions. Inputs are S1, best_error and
                                % S0 and outputs are best_error1, best_neighbor1,
                                % best_error and best_neighbor.
 if (best_error1-best_error)>1e-18
                                % Check repeated solution by compare between
                                % best_error1 and best_error.
   n=n+1;
else
   n=0;
end
tt=tt+1;                        % Increase sliding of tabu_list.
tabu_list(tt,1)=count;          % Store count in the first column of tabu_list.
tabu_list(tt,2:3)=best_neighbor1;
                                % Store best_neighbor1 in the second column and
                                % the third column of tabu_list.
tabu_list(tt,4)=best_error1;    % Store best_error1 in the fourth column of tabu_list
ttt= ttt+1;                     % Increase sliding of best_error_list.
best_error_list(ttt,1)=count;   % Store count in the first column of best_error_list.
best_error_list(ttt,2:3)=best_neighbor;
                                % Store best_neighbor in the second column and the
                                % third column of best_error_list.
best_error_list(ttt,4)=best_error;
```

```
                                    % Store best_error in the fourth column of
                                    % best_error_list.
% Start AR mechanism
   if best_error<1e-1               % Condition 1 if best_error<1e-1 then radius=2e-3.
   radius=2e-3;
   end
   if best_error<=1e-3              % Condition 2 if best_error<=1e-3 then radius=2e-4.
   radius=2e-4;
   end
   if (overall_best_error<1e-9)     % Check terminal criterion.
   t=t+1;                           % Increase sliding of local.
   tt=tt+1;                         % Increase sliding of tabu_list.
   local(t,1)=count;                % Store count in the first column of local.
   local(t,2:3)=best_neighbor;      % Store best_neighbor in the second column and the
                                    % third column of local.
   local(t,4)=best_error;           % Store best_error in the fourth column of local.
   overall_best_error=best_error;   % Define overall_best_error=best_error.
   overall_best_neighbor=best_neighbor;
                                    % Define overall_best_neighbor=best_neighbor.
   break;                           % Stop search
end
% Start BT mechanism
      if n>=5                       % When number of repeated solution is equal to 5.
      n_back_tracking=n_back_tracking+1;
                                    % Increase n_back_tracking value.
      TEMP=tabu_list(count-3:count+1,:);
                                    % Rank among cost values and among 5 solutions
                                    % and then store cost values and 5 solutions in
                                    % TEMP.
```

```
    [MAX,INDEX] = max(TEMP(:,4));   % Find maximum cost value and its position.
    RANK(5,:) = TEMP(INDEX,:);      % Define maximum cost value and its position in
                                    % the fifth of RANK.
    TEMP(INDEX,4)=0;                % Replace maximum cost value in TEMP by zero.
    [MAX,INDEX] = max(TEMP(:,4));   % Find maximum cost value and its position.
    RANK(4,:) = TEMP(INDEX,:);      % Define maximum cost value and its position in
                                    % the fourth of RANK.
    TEMP(INDEX,4)=0;                % Replace maximum cost value in TEMP by zero.
    [MAX,INDEX] = max(TEMP(:,4));   % Find maximum cost value and its position.
    RANK(3,:) = TEMP(INDEX,:);      % Define maximum cost value and its position in
                                    % the third of RANK.
    TEMP(INDEX,4)=0;                % Replace maximum cost value in TEMP by zero.
    [MAX,INDEX] = max(TEMP(:,4));   % Find maximum cost value and its position.
    RANK(2,:) = TEMP(INDEX,:);      % Define maximum cost value and its position in
                                    % the second of RANK.
    TEMP(INDEX,4)=0;                % Replace maximum cost value in TEMP by zero.
    [MAX,INDEX] = max(TEMP(:,4));   % Find maximum cost value and its position.
    RANK(1,:) = TEMP(INDEX,:);      % Define maximum cost value and its position in
                                    % the first of RANK.
    TEMP(INDEX,4)=0;                % Replace maximum cost value in TEMP by zero.
     neighbor=RANK(5,2:3);          % Replace neighbor by the fifth solutions in RANK.
    S0=neighbor;                    % Define S0=neighbor.
    best_error=RANK(5,4);           % Replace best_error by the fifth cost value of
                                    % RANK
    n=0;                            % Assign n=0
    else                            % Otherwise.
    S0=best_neighbor;               % Define S0=best_neighbor.
    best_error=best_error;          % Define best_error=best_error.
end
```

```
        if overall_best_error>best_error
          overall_best_error=best_error;      % Define overall_best_error=best_error and define
                                              % overall_best_neighbor=best_neighbor, if
                                              % overall_best_error>best_error.
          overall_best_neighbor=best_neighbor;
       end
end
       time=toc;                              % Stop search time.
return
```

Program of initial_generation.m

```
function  [min_cost_function, X_Y_best]=Initial_generation(p,S,Nc,Ns,flag,alpha,xlimit)
                                              % This program is to generate an elite solution as an
                                              % initial solution for BF-TS program. There are 7
                                              % input parameters: p, S, Nc, Ns, flag, alpha and
                                              % xlimit providing 2 output sets:
                                              % min_cost_function and X_Y_best.
% Initial population
P(:,:,:)=0*ones(p,S,Nc);                      % Allocate needed memory.
C=0*ones(S,Nc);                               % Initialize the parameters of step size
                                              % govern part of the random movement.
J=0*ones(S,Nc);                               % Allocate memory for cost function.
for m1=1:S                                    % Generate initial solution.
   P(:,m1,1)= ((xlimit(1,:)-xlimit(2,:)).*rand(1,p))+xlimit(2,:);
 end
```

```
% Swim/tumble (chemotaxis) loop:
for j=1:Nc                                  % Start for-loop of random movement.
    for i=1:S                               % Start for-loop of a number of solution.
                J(i,j)=obj(P(:,i,j));       % Compute the cost value at the current location of
                                            % each solution set
% Next, add on cell-cell attraction effect:
                J(i,j)=J(i,j)+bact_cellcell_attract_func(P(:,i,j),P(:,:,j),S,flag);
% Tumble:
                Jlast=J(i,j);               % Initialize the cost value to be the first
                                            % at the tumble.
                Delta(:,i)=(2*round(rand(p,1))-1).*rand(p,1);
                                            % Generate a random direction.
                C(i,j)=abs(J(i,j))/((abs(J(i,j)))+alpha);
                                            % Updating variable run length unit.
 % Next, move all the solution by a small amount in the direction that the tumble resulted in
                P(:,i,j+1)=P(:,i,j)+C(i,j)*Delta(:,i)/sqrt(Delta(:,i)'*Delta(:,i));
                                            % This adds a unit vector in the random direction,
                                            % scaled by the step size C(i,j).
% Swim (for solutions that seem to be headed in the right direction):
% Check parameters in range of search space.
while    (P(1,i,j+1))> xlimit(1,1) | P(1,i,j+1)<xlimit(2,1) | P(2,i,j+1)>xlimit(1,2)|…
         P(2,i,j+1)<xlimit(2,2)
         Delta(:,i)=(2*round(rand(p,1))-1).*rand(p,1);
         C(i,j)=abs(J(i,j))/((abs(J(i,j)))+alpha);
         P(:,i,j+1)=P(:,i,j)+C(i,j)*Delta(:,i)/sqrt(Delta(:,i)'*Delta(:,i));
end
                J(i,j+1)=obj(P(:,i,j+1));   % Compute cost value for each solution after
                                            % a small step (used by the solution to decide if it
                                            % should keep swimming).
```

```
% Next, add on cell-cell attraction effect:
                J(i,j+1)=J(i,j+1)+bact_cellcell_attract_func(P(:,i,j+1),P(:,:,j+1),S,flag);
    m=0;                                % Initialize counter for swim length.
    while m<Ns                          % While climbing a gradient.
      m=m+1;
      if J(i,j+1)<Jlast                 % Check a new cost value and move it further in
                                        % same direction.
      Jlast=J(i,j+1);                   % First, save the cost value at current location.
% Next, extend the run in the same direction since it climbed at the last step:
      P(:,i,j+1)=P(:,i,j+1)+C(i)*Delta(:,i)/sqrt(Delta(:,i)'*Delta(:,i));
      J(i,j+1)=obj(P(:,i,j+1));         % Compute cost value at where it swam to and give
                                        % it new cost value.
% Next, add on cell-cell attraction effect:
      J(i,j+1)=J(i,j+1)+bact_cellcell_attract_func(P(:,i,j+1),P(:,:,j+1),S,flag);
      else                              % It did not move up the gradient so stop the run for
                                        % this solution.
      m=Ns;
    end
end                                     % Test if should end run step.
end                                     % End run for number of solution, S.
end                                     % End run for random movement Nc.
reproduction = J(:,1:Nc);               % Select cost values and assign in reproduction.
[jlastreproduction,iter] = min(reproduction,[],2);
                                        % Find minimum cost value and its position
                                        % among random movement, Nc.
[min_cost_function,back_ID] = min(jlastreproduction) ;
                                        % Evaluate minimum cost value and its position
                                        % among each number of solution, S, and
                                        % assign to min_cost_function and back_ID.
X_Y_best=P(:,back_ID,iter(back_ID,:));  % Assign optimum an elite solution in X_Y_best.
return
```

Program of obj.m

```
function J=obj(para)                    % This program is to compute cost values as an
                                        % example of Bohachevsky 's function .
x=para(1);                              % This consists of 2 parameters, x and y.
y=para(2);
J=(x.^2)+(2*y.^2)-(0.3*cos(3*pi*x))-(0.4*cos(4*pi*y))+0.7;
                                        % Bohachevsky 's function.
return
```

Program of objective.m

```
function [best_error1,best_neighbor1,best_error,best_neighbor]=objective(S1,best_error,S0)
                                        % Objective function program is to evaluate
                                        % neighbour solutions around S0. Input parameters
                                        % are S1,best_error and S0 and output parameters
                                        % are best_error1, best_neighbor1, best_error and
                                        % best_neighbor.
    error=[];
    for k=1:size(S1,1)
        ysim=obj(S1(k,:));              % Evaluate cost values of S1 by calculating obj
                                        % function.
        error(k,1)=ysim;                % Store all cost values of neighbourhood.
    end
    [best_error1,index]=min(error);     % Find minimum cost values and it index of
                                        % neighbourhood, and define in best_error1 and its
                                        % location.
    best_neighbor1=S1(index,:);         % Define neighbour solution with minimum cost
                                        % value in best_neighbor1.
```

```
        if best_error1<best_error
                best_error=best_error1;
                best_neighbor=S1(index,:); % Define best_error=best_error1 and
                                           % best_neighbor=S1(index,:) if best_error1<
                                           % best_error.
    else                                   % Otherwise
                best_neighbor=S0;          % Define best_neighbor=S0.
    end
return
```

Program of rand1.m

```
function x=rand1(a,b)                      % rand1 function program is to random parameter
                                           % within a and b based on expression, a+(b-a)rand
x=a+rand*(b-a);
return
```

Program of bact_cellcell_attract_func.m

```
% Bacteria cell to cell attraction function
function [Jar]=bact_cellcell_attract_func(x,theta,S,flag)
                                       % Given locations of all bacteria, find Jar for all S
                                       % Note that theta rows are dimensions of the
                                       % optimization problem, while the columns are
                                       % the S different bacteria.
if flag==2                             % Test to see if main program indicated cell-cell
                                       % attraction.
        Jar=0;
        return
end
```

```
depthattractant=0.1;                    % Sets magnitude of secretion of attractant by a cell.
widthattractant=0.2;                    % Sets how the chemical cohesion signal diffuses.
heightrepellant=1*depthattractant;      % Sets repellant (tendency to avoid nearby cell).
widthrepellant=1;                       % Makes small area where cell is relative to
                                        % diffusion of chemical signal.
Jar=0;
for j=1:S
        Ja=-depthattractant*exp(-widthattractant*((x(1,1)-theta(1,j))^2+(x(2,1)-theta(2,j))^2))
                                        % Set how the cell attracts other cells via secretions
                                        % of diffusible attractants.
        Jr=+heightrepellant*exp(-widthrepellant*((x(1,1)-theta(1,j))^2+(x(2,1)-theta(2,j))^2));
                                        % Set how the cell repels other cells since it eats in
                                        % its own region (and since an intact cell is
                                        % apparently not food for another cell).
        Jar=Jar+Ja+Jr;                  % Next, set the combined effect.
end
return
```

Program of random_neigh.m

```
function S1=random_neigh(Number_neighb,radius,xlimit,S0)
                                        % This program is to random neighbour solutions
                                        % around S0. Input parameters are Number_neighb,
                                        % radius, xlimit and S0 and output parameter is S1.
  for u=1:Number_neighb
    for k=1:size(xlimit,2)
      S1(u,k)=S0(1,k)+(radius*(xlimit(1,k)-xlimit(2,k))*rand1(-1,1));
                                        % Random neighbour solutions.
```

```
        while ( S1(u,k)>xlimit(1,k) | S1(u,k)<xlimit(2,k) )
                                        % Check limit of neighbour solution values.
          S1(u,k)=S0(1,k)+(radius*(xlimit(1,k)-xlimit(2,k))*rand1(-1,1));
        end
      end
   end
return
```

E.2 Code List of the Modified ATS Algorithm

The program list of the modified ATS algorithm is shown in this appendix (an abstract mathematical constraint problem of Constrained function 1). This experimental program consists of Modified_ATS.m, initial_generation.m, obj.m, objective.m and random_neigh.m. The initial_generation.m and objective.m files can be referred to above BF-TS algorithm's program.

```
% This Modified_ATS program has been developed by using MATLAB™ by Nuapett Sarasiri.
% School of Electrical Engineering, Suranaree University of Technology.
```

Program of Modified_ATS.m

```
function [min_cost_function,X_Y_best,overall_best_error,overall_best_neighbor,…
n_back_tracking,time,count]=modified_ATS(p)
                                        % This main program uses p as a input, where p is a
                                        % number of parameter to be optimized. There are 7
                                        % output parameters: min_cost_function is an initial
                                        % objective function obtained from random-walk
                                        % frontend. X_Y_best is an initial solution obtained
```

```
                                    % from random-walk frontend. overall_best_error is
                                    % optimal cost function. overall_best_neighbor is
                                    % global optimum solution. n_back_tracking is a
                                    % number of solution occurred in local deadlock.
                                    % time is total time. count is total iteration.
rand('state',sum(100*clock));       % Initialize rand to a different state each time.
S=30;                               % Number of solution in the population.
Nc=100;                             % Number of chemotactic steps.
Ns=4;                               % Limits the length of a swim when it is on a
                                    % gradient.
alpha=1e3;                          % A positive constant for step size expression C1.
alpha2=1e4;                         % A positive constant for step size expression C2.
S2=100;                             % Number of neighbour solution.
Nc2=1000;                           % Number of chemotactic steps for random
                                    % neighbour solution.
max_count=50000;                    % A maximum iteration.
n=0;                                % Initial number of repeated solution.
t=0;                                % Initial sliding of local.
tt=0;                               % Initial sliding of tabu_list.
ttt=0;                              % Initial sliding of best_error_list.
count=0;                            % Initial number of iteration.
n_back_tracking=0;                  % Initial number of back tracking.
best_error=0;                       % Initial number of best_error.
best_neighbor=0;                    % Initial number of best_neighbor.
xlimit=[10  10;-10 -10];            % Search space, row 1 of xlimit is a maximum of
                                    % parameter.
                                    % Row 2 of xlimit is a minimum of parameter.
tic;                                % Start search time
% Randomly initial solutions from random-walk frontend by using Initial_generation function.
```

```
[min_cost_function, X_Y_best]= Initial_generation(p,S,Nc,Ns,alpha,xlimit);
                                        % Input of this function are p, S, Nc, Ns, alpha,
                                        % xlimit and output are min_cost_function and
                                        % X_Y_best.
S0=X_Y_best';                           % Store initial solution in S0.
best_neighbor=S0;                       % Store initial solution in best_neighbor
best_error= min_cost_function;          % Define best_error = min_cost_function.
overall_best_error=best_error;          % Define overall_best_error=best_error.
overall_best_neighbor=best_neighbor;    % Define overall_best_neighbor=best_neighbor.
t=t+1;                                  % Increase sliding of local value.
tt=tt+1;                                % Increase sliding of tabu_list.
ttt=ttt+1;                              % Increase sliding of best_error_list.
local(t,1)=count;                       % Store count in the first column of local.
local(t,2:3)=best_neighbor;             % Store best_neighbor in the second column and the
                                        % third column of local.
local(t,4)=best_error;                  % Store best_error in the fourth column of local.
tabu_list(tt,1)=count;                  % Store count in the first column of tabu_list.
tabu_list(tt,2:3)=best_neighbor;        % Store best_neighbor in the second column and the
                                        % third column of tabu_list.
tabu_list(tt,4)=best_error;             % Store best_error in the fourth column of tabu_list.
best_error_list(ttt,1)=count;           % Store count in the first column of best_error_list.
best_error_list(ttt,2:3)=best_neighbor; % Store best_neighbor in the second column and the
                                        % third column of best_error_list.
best_error_list(ttt,4)=best_error;      % Store best_error in the fourth column of
                                        % best_error_list.
while (count<max_count)                 % Process loop.
   count=count+1;
   S1=random_neigh(p,S2,Nc2,Ns,xlimit,S0,alpha2);
                                        % Recall random_neigh function to random among
```

```
                                        % neighbour solutions around S0 within current sub
                                        % search space. There are 7 input parameters such
                                        % as  p, S2, Nc2, Ns, xlimit, S0 and alpha2, and one
                                        % output, S1.
[best_error1,best_neighbor1,best_error,best_neighbor]=objective(S1,best_error,S0);
                                        % Recall objective function to evaluate random
                                        % neighbour solutions. Inputs are S1, best_error and
                                        % S0 and outputs are best_error1, best_neighbor1,
                                        % best_error and best_neighbor.
  if (best_error1-best_error)>1e-18
                                        % Check repeated solution by compare between
                                        % best_error1 and best_error.
     n=n+1;
  else
     n=0;
  end
  tt=tt+1;                              % Increase sliding of tabu_list.
  tabu_list(tt,1)=count;                % Store count in the first column of tabu_list.
  tabu_list(tt,2:3)=best_neighbor1;
                                        % Store best_neighbor1 in the second column and
                                        % the third column of tabu_list.
  tabu_list(tt,4)=best_error1;          % Store best_error1 in the fourth column of tabu_list
  ttt= ttt+1;                           % Increase sliding of best_error_list.
  best_error_list(ttt,1)=count;         % Store count in the first column of best_error_list.
  best_error_list(ttt,2:3)=best_neighbor; % Store best_neighbor in the second column and the
                                        % third column of best_error_list.
  best_error_list(ttt,4)=best_error;
                                        % Store best_error in the fourth column of
                                        % best_error_list.
```

```
   if (overall_best_error<=1.3935)          % Check terminal criterion.
   t=t+1;                                   % Increase sliding of local.
   tt=tt+1;                                 % Increase sliding of tabu_list.
   local(t,1)=count;                        % Increase sliding of tabu_list.
   local(t,2:3)=best_neighbor;              % Store best_neighbor in the second column and the
                                            % third column of local.
   local(t,4)=best_error;                   % Store best_error in the fourth column of local.
   overall_best_error=best_error;           % Define overall_best_error=best_error.
   overall_best_neighbor=best_neighbor;     % Define overall_best_neighbor=best_neighbor.
   break;                                   % Stop search
 end
% Start BT mechanism                        % When number of repeated solution is equal to 5.
     if n>=5
     n_back_tracking=n_back_tracking+1; % Increase n_back_tracking value.
     TEMP=tabu_list(count-3:count+1,:);
                                            % Rank among cost values and among 5 solutions
                                            % and then store cost values and 5 solutions in
                                            % TEMP.
     [MAX,INDEX] = max(TEMP(:,4));          % Find maximum cost value and its position.
     RANK(5,:) = TEMP(INDEX,:);             % Define maximum cost value and its position in
                                            % the fifth of RANK.
     TEMP(INDEX,4)=0;                       % Replace maximum cost value in TEMP by zero.
     [MAX,INDEX] = max(TEMP(:,4));          % Find maximum cost value and its position.
     RANK(4,:) = TEMP(INDEX,:);             % Define maximum cost value and its position in
                                            % the fourth of RANK.
     TEMP(INDEX,4)=0;                       % Replace maximum cost value in TEMP by zero.
     [MAX,INDEX] = max(TEMP(:,4));          % Find maximum cost value and its position.
     RANK(3,:) = TEMP(INDEX,:);             % Define maximum cost value and its position in
                                            % the third of RANK.
```

```
      TEMP(INDEX,4)=0;                  % Replace maximum cost value in TEMP by zero.
      [MAX,INDEX] = max(TEMP(:,4));     % Find maximum cost value and its position.
      RANK(2,:) = TEMP(INDEX,:);        % Define maximum cost value and its position in
                                        % the second of RANK.
      TEMP(INDEX,4)=0;                  % Replace maximum cost value in TEMP by zero.
      [MAX,INDEX] = max(TEMP(:,4));     % Find maximum cost value and its position.
      RANK(1,:) = TEMP(INDEX,:);        % Define maximum cost value and its position in
                                        % the first of RANK.
      TEMP(INDEX,4)=0;                  % Replace maximum cost value in TEMP by zero.
       neighbor=RANK(5,2:3);            % Replace neighbor by the fifth solutions in RANK.
      S0=neighbor;                      % Define S0=neighbor.
      best_error=RANK(5,4);             % Replace best_error by the fifth cost value of
                                        % RANK
      n=0;                              % Assign n=0
      else                              % Otherwise.
      S0=best_neighbor;                 % Define S0=best_neighbor.
      best_error=best_error;            % Define best_error=best_error.
      end
    if overall_best_error>best_error
        overall_best_error=best_error;  % Define overall_best_error=best_error and define
                                        % overall_best_neighbor=best_neighbor, if
                                        % overall_best_error>best_error.
        overall_best_neighbor=best_neighbor;
      end
end
      time=toc;                         % Stop search time.
return
```

Program of Obj.m

```
function J=obj(x)
                                        % This program is to compute cost values as an
                                        % example of Constrained function 1.
    x1=x(1);                            % This consists of 2 parameters, x1 and x2.
    x2=x(2);
    f=(x1-2).^2+ (x2-1).^2;             % Objective function.
    g1=x1-(2.*x2)+1;                    % Equality constraint g1=0.
    g2=-((x1.^2)./4)-(x2.^2)+1;         % Inequality constraint g2>=0.
      if g1<=1e-3                       % Check condition of constraint g1.
        g1=0;
      else
        g1=1e10;
      end

      if g2<=1e-3                       % Check condition of constraint g2.
        g2=0;
      else
        g2=1e10;
      end
      SSP=(1e10*(g2.^2));
J=f+SSP+1e10*(g1.^2);                   % Minimization function with penalty value.
return
```

Program of random_neigh.m

```
function S1=random_neigh(p,S2,Nc2,Ns,xlimit,S0,alpha2)
                                        % This program is to generate neighbour solutions
                                        % for Modified_ATSprogram. There are 7 input
```

```
                                        % parameters: p, S2, Nc2, Ns, xlimit, S0, alpha2
                                        % with one output parameter: S1.
P(:,:,:)=0*ones(p,S2,Nc2);              % Allocate needed memory.
C=0*ones(S2,Nc2);                       % Initialize the parameters of step size
                                        % govern part of the random movement.
J=0*ones(S2,Nc2);                       % Allocate memory for cost function.
for m1=1:S2
   P(:,m1,1)= S0;                       % Assign solution from S0 to P.
end
% Swim/tumble (chemotaxis) loop:
for j=1:Nc2                             % Start for-loop of random movement.
  for i=1:S2                            % Start for-loop of a number of solution.
% Compute the nutrient concentration at the current location of each solution
                J(i,j)=obj(P(:,i,j));
% Tumble:
                Jlast=J(i,j);           % Initialize the cost value to be the first
                                        % at the tumble.
                Delta(:,i)=(2*round(rand(p,1))-1).*rand(p,1);
                                        % Generate a random direction.
                C(i,j)=abs(J(i,j))/((abs(J(i,j)))+alpha2);
                                        % Updating variable run length unit.
% Next, move all the solution by a small amount in the direction that the tumble resulted in
                P(:,i,j+1)=P(:,i,j)+C(i,j)*Delta(:,i)/sqrt(Delta(:,i)'*Delta(:,i));
                                        % This adds a unit vector in the random direction,
                                        % scaled by the step size C(i,j).
% Swim (for solutions that seem to be headed in the right direction):
% Check parameters in range of search space.
while    (P(1,i,j+1))>xlimit(1,1) | P(1,i,j+1)<xlimit(2,1) | P(2,i,j+1)>xlimit(1,2)|…
         P(2,i,j+1)<xlimit(2,2)
```

```
        Delta(:,i)=(2*round(rand(p,1))-1).*rand(p,1);
        C(i,j)=abs(J(i,j))/((abs(J(i,j)))+alpha2);
        P(:,i,j+1)=P(:,i,j)+C(i,j)*Delta(:,i)/sqrt(Delta(:,i)'*Delta(:,i));

end
                J(i,j+1)=obj(P(:,i,j+1));    % Compute cost value for each solution after
                                             % a small step (used by the solution to decide if it
                                             % should keep swimming).
        m=0;                                 % Initialize counter for swim length
        while m<Ns                           % While climbing a gradient.
                m=m+1;
        if J(i,j+1)<Jlast                    % Check a new cost value and move it further in
                                             % same direction.
        Jlast=J(i,j+1);                      % First, save the cost value at current location.
% Next, extend the run in the same direction since it climbed at the last step:
        P(:,i,j+1)=P(:,i,j+1)+C(i)*Delta(:,i)/sqrt(Delta(:,i)'*Delta(:,i));
        J(i,j+1)=obj(P(:,i,j+1));            % Compute cost value at where it swam to and give
                                             % it new cost value.
        else                                 % It did not move up the gradient so stop the run for
                                             % this solution.
        m=Ns;
      end
end                                          % Test if should end run step.
end                                          % End run for number of solution, S2.
end                                          % End run for random movement Nc2.
reproduction = J(:,1:Nc2);                   % Select cost values and assign in reproduction.
[jlastreproduction,iter] = min(reproduction,[],2);
                                             % Find minimum cost value and its position
                                             % among random movement, Nc2.
[min_cost_function,back_ID] = min(jlastreproduction);
```

```
                                        % Evaluate minimum cost value and its position
                                        % among each number of solution, S2, and
                                        % assign to min_cost_function and back_ID.
X_Y_best=P(:,back_ID,iter(back_ID,:));  % Assign optimum initial solution in X_Y_best.
S1=X_Y_best';                           % Assign optimum an elite solution in S1.
return
```

E.3 Code List of the Parallel Version of the Modified ATS Algorithm

The program list of the parallel modified ATS algorithm has been utilized to optimize the optimal design of power drive system. This experimental program consists of Modified_ATS.m, parallel_initial_generation.m, parallel_obj.m, objective.m, random_neigh.m and drive_system.mdl (Simulink file) as shown in Figure E.1-E.2. The main program of Modified_ATS.m (with 12 parameters), objective.m and random_neigh.m files can be referred to above modified ATS algorithm's program.


```
% This Modified_ATS program with parallel computing has been developed by using

% MATLAB™ and Parallel Computing Toolbox™ by Nuapett Sarasiri.
% School of Electrical Engineering, Suranaree University of Technology.
```

Program of parallel_obj.m

```
function J=obj(para)
%  Define parameters
Rr=para(1);                             % input filter components.
Lr=para(2);
fs=para(3);                             % switching frequency.
```

```
Ls=para(4);                          % inductance of mortor.
KPline=para(5);                      % control parameter PI for rectifier.
KIline=para(6);
KPrectifi=para(7);                    % control parameter PI for DC-link.
KIrectifi=para(8);
KPcurrent= para(9);                  % control parameter PI for inverter.
KIcurrent=para(10);
KPspeed=para(11)  ;                  % control parameter PI for speed control.
KIspeed=para(12);
 %  parameters
Lline=0.1e-3;                        % line inductance.
Rline= 0.05;                         % line resistance.
Rs=0.124;                            % stator resistance.
time=0.11;                           % time (sec.).
assignin('base','Rr',para(1));       % assign parameter values to simulink model. There
assignin('base','Lr',para(2));       % are 12 parameters; Rr, Lr, fs, Ls, KPlline, KIline,
assignin('base','fs',para(3));       % KPrectifi, KIrectifi, KPcurrent, KIcurrent, KPspeed
assignin('base','Ls', para(4));      % and KIspeed.
assignin('base','KPline', para(5));
assignin('base','KIline',para(6));
assignin('base','KPrectifi',para(7));
assignin('base','KIrectifi', para(8));
assignin('base','KPcurrent',para(9));
assignin('base','KIcurrent',para(10));
assignin('base','KPspeed',para(11));
assignin('base','KIspeed',para(12));
model = drive_system ';
```

```
load_system(model);                          % load model as the structure in Figure E.1-E.2.
set_param('drive_system ','SimulationCommand','update'); % simulation
[tout,xout,I_out] = sim('drive_system ',linspace(0,0.11,5000));
                                             % I_out(:,1) is a line current.
                                             % I_out(:,2)is a motor current.
                                             % I_out(:,3) is a DC voltage.
                                             % I_out(:,4) is speed of motor.
close_system('drive_system.mdl ',0)          % Close simulink model.
%% Parallel computing
if matlabpool('size') == 0                   % checking to see if matlabpool is already open.
   matlabpool open                           % open matlabpool.
end
spmd (4)                                     % Open single program multi data worked on 4
                                             % workers. Inside spmd structure, the parameters:
                                             % Lline, Rline, Rs, time and I_out performed as
                                             % codistributed arrays on each worker.
labBarrier;                                  % labBarrier is to ensure all workers are synchronized
                                             % and start their timed work together.
if labindex==1                               % Indicate worker 1.
%% Compute harmonics both of current to find Rr and Lr
fa=50;                                       % fundamental frequency.
T=3;                                         % periods.
s=0.02*T;                                    % sampling periods.
Tn=fa*s;                                     % periods.
Iline_out=I_out(:,1);
xnn= Iline_out(1813:4540);                   % sampling data.
xn=xnn';
```

```
N=length(xn);                          % samples of data.
t=linspace(0.04,0.1,N);                % time start from 0 to number of sampling periods.
[x,y]=dft(xn,N,Tn);                    % export data to DFT function.
ii=find(x==1);                         % find fundamental data at the first order
fund=y(ii);                            % storage fundamental value in fund.
x1=[0:Tn:Tn*x(end)];
x1=x1+1;
for i=1:length(x1)
   aa=x1(i);                           % integer order of harmonics.
   cc(i)=x(aa);
   bb(i)=y(aa);                        % each harmonic value at integer order of harmonics.
end
ynew=bb;
y1=ynew.^2;
y2=y1(3:end);                          % harmonic values without fundamental value.
THDf=100*sqrt(sum(y2))./sqrt(fund^2); %  THD of input current.
J_THDf=THDf/3;                         % Objective function of THDi,input.
    xnn=labBroadcast(1,xnn);           % Distribute sampling data to all workers.
else
   xnn=labBroadcast(1);
end                                    % end worker 1.


if labindex==2                         % Indicate worker 2.
% Compute Total losses to find fs
Iline_rms=sqrt ((sum(xnn.^2)) /length(xnn)); % Input current (rms).
I_load=I_out(:,2);
I_motor=I_load(4667:4993);
```

```
t_motor=linspace(0.1027,0.1098,length(I_motor)); % start time 0.1027 to 0.1098sec 6 periods.
Irms_motor= sqrt ((sum(I_motor.^2)) /length(I_motor)); % Mortor current (rms).
VCE0=2.4;
VF0=1.8;
rCE=(3-2)/(137-75);                    % slope of Vce-Ic.
rD=(1.9-1.7)/(20-10);                  % slope of Vf-If.
EonT=(0.1194e-6)*(20*60);              % EonT(uJ/VA)=eonT*delta(Vce)*delta(Ic).
Eoff=(0.1444e-6)*(20*60);              % EoffT(uJ/VA)=eoffT*delta(Vce)*delta(Ic).
EonD=(2.5e-6)*(20*1);                  % EonD(uJ/VA)=eonD*delta(Vd)*delta(Id).
DeltaV=10/100;                         % ripple voltage.
PL_line= 3*((Iline_rms^2)*(Rline+Rr));% line input loss.
Pcond=6*(((VCE0+VF0)*Iline_rms)+((rCE+rD)*Iline_rms^2)); % multiply by 6 because
                                       % switch don't not operate in the same time (just a half
                                       % from 12).
Psw=  6*(((EonT+Eoff)*fs)+(EonD*fs));
PL_SD=Pcond+Psw;                       % Conduction and switching losses.
PL_motor= 3*(Irms_motor^2)*Rs;         % loss of motor.
PL_total=PL_line + PL_SD + PL_motor; % total losses.
J_PL_total=PL_total/1.7e3;              % objective funcion of total losses.
%% Line current control
t_step=linspace(0,time,5000);          % smpling time 0-0.11 sec.
plant_line_gain= tf([1],[(Lline+Lr) (Rline+Rr)]); % Plant of input current.
PI_line=tf([KPline KIline],[1 0]);     % PI controller
plant_line_control=feedback(plant_line_gain*PI_line,1,-1); %Plant of input current close loop.
[Amp_line t_step]=step(plant_line_control,t_step);
RMS_recti_a=(1-Amp_line).^2;           % RMS of input current close loop.
RMS_recti_b=sum(RMS_recti_a)/length(Amp_line);
```

```
RMS_recti=sqrt(RMS_recti_b);
J_RMS_recti=RMS_recti/0.0309;          % objective funcion of input current close loop.
%  DC-link voltage control
VDC_error=I_out(:,3);
RMS_DC_a=(1000-VDC_error).^2;      % RMS of DC voltage.
RMS_DC_b=sum(RMS_DC_a)/length(VDC_error);
RMS_DC=sqrt(RMS_DC_b);
J_RMS_DC= RMS_DC/6.9802;           % objective funcion of DC voltage.
B=I_motor;
I_motor=labBroadcast(2,B);             % Distribute mortor current to all workers.
B1=t_step;
t_step = labBroadcast(2,B1);           % Distribute sampling time to all workers.
else
   I_motor = labBroadcast(2);
   t_step = labBroadcast(2);
end                                    % end worker 2.

if labindex==3                         % Indicate worker 3.
%  THD mortor
fa_motor=833.33;                       % fundamental frequency of motor (Hz).
T_motor=6;                             % periods.
s_motor=(1/833.33)*T_motor;            % sampling periods.
Tn_motor=fa_motor*s_motor;             % periods.
xnn_motor=I_motor;                     % sampling data.
xn_motor=xnn_motor';
N_motor=length(xn_motor);              % samples of data.
[x_motor,y_motor]=dft_motor(xn_motor,N_motor,Tn_motor);
                                       % export data to DFT function
```

```
ii_motor=find(x_motor==1);              % find fundamental data at the first order.
fund_motor=y_motor(ii_motor);           % storage fundamental value in fund.
x1_motor=[0:Tn_motor:Tn_motor*x_motor(end)];
x1_motor=x1_motor+1;
for i=1:length(x1_motor)
   aa_motor=x1_motor(i);                % integer order of harmonics.
   cc_motor(i)=x_motor(aa_motor);
   bb_motor(i)=y_motor(aa_motor);       % each harmonic value at integer order of harmonics.
end
ynew_motor=bb_motor;
y1_motor=ynew_motor.^2;
y2_motor=y1_motor(3:end);               % harmonic values without fundamental value.

THDf_motor=100*sqrt(sum(y2_motor))./sqrt(fund_motor^2); %  %THD of motor current.
J_THDi_motor=THDf_motor/3;              % Objective function of THDi,motor.
end                                     % end work 3.

if labindex==4                          % Indicate worker 4.
%  PWM current Conroller
plant_motor_gain=tf([1],[Ls Rs]);       % plant of motor current control.
PI_inverter=tf([KPcurrent KIcurrent],[1 0]); %  PI controller
plant_inverter_control=feedback(plant_motor_gain*PI_inverter,1,-1);
                                        % close loop control of motor current.
[Amp_inverter t_step]=step(plant_inverter_control,t_step);
RMS_inverter_a=(1-Amp_inverter).^2;  %  RMS of inverter current
RMS_inverter_b=sum(RMS_inverter_a)/length(Amp_inverter);
RMS_inverter=sqrt(RMS_inverter_b);
J_RMS_inverter=RMS_inverter/0.0488; % objective function of inverter current.
```

```
% Speed control
Speed_error=I_out(:,4);
RMS_Speed_a=(10000-Speed_error).^2; % RMS of speed.
RMS_Speed_b=sum(RMS_Speed_a)/length(Speed_error);
RMS_Speed=sqrt(RMS_Speed_b);
J_RMS_Speed=RMS_Speed/925.8013;  % objective function of RMS of speed.
end                              % end work 4.
end                              % close spmd.
% receive functions
J_THDf=J_THDf{1};                % Return data from each worker to client.
J_PL_total=J_PL_total{2};        % These parameters are the class of composite
J_THDi_motor=J_THDi_motor{3};    % arrays on each worker. These contain
J_RMS_recti=J_RMS_recti{2};      % 7 objective functions: J_THDf, J_PL_total,
J_RMS_DC=J_RMS_DC{2};            % J_THDi_motor, J_RMS_recti, J_RMS_DC,
J_RMS_inverter=J_RMS_inverter{4};% J_RMS_inverter and J_RMS_Speed.
J_RMS_Speed=J_RMS_Speed{4};
fall=(1/7)*(J_THDf+J_PL_total+J_THDi_motor+J_RMS_recti+J_RMS_DC+…
J_RMS_inverter+J_RMS_Speed);     % Compute multi objective function.
InEq= [J_THDf; J_PL_total; J_THDi_motor; J_RMS_recti; J_RMS_DC; J_RMS_inverter; …
 J_RMS_Speed];                   % inequality constraints.
SSR=0;                           % allocate memory.
rho=1e6;                         % penalty factor.
for kk=1:length(InEq)
   if InEq(kk)<1                 % if all constraint conditions below than 1.
      InEq(kk)=0;                % all constraint conditions equal to 0.
   end                           % end condition.
   SSR=SSR+rho*InEq(kk)^2;       % sum square value multiply by penalty factor.
```

```
end
J = fall+SSR;                          % Objective value.
return
```

Program of parallel_initial_generation.m

```
function  [min_cost_function, X_Y_best]= Initial_generation(p,S,Nc,Ns,alpha,xlimit)
                                       % This program is to generate an elite solution as an
                                       % initial solution for the parallel computing of the
                                       % modified ATS program. There are 6
                                       % input parameters: p, S, Nc, Ns, alpha and
                                       % xlimit providing 2 output sets:
                                       % min_cost_function and X_Y_best.
%%Initial population
P(:,:,:)=0*ones(p,S,Nc);               % Allocate needed memory.
C=0*ones(S/4,Nc);                      % Initialize the parameters of step size
                                       % govern part of the random movement.
J=0*ones(S/4,Nc);                      % Allocate memory for cost function.
% parallel computing
if matlabpool('size') == 0             % Checking to see if matlabpool is already open.
   matlabpool open                     % open matlabpool.
end
% Generate initial solution
 parfor m1=1:S                         % Use parfor-loop to generate random solutions
   P(:,m1,1)= ((xlimit(1,:)-xlimit(2,:)).*rand(1,p))+xlimit(2,:);
 end
```

```
spmd                                    % Open single program multi data.
   DIST=codistributor1d(2,[S/4 S/4 S/4 S/4]);
                                        % Assign array of solution on each worker.
   P_codis=codistributed(P,DIST);       % Replicate array on each worker.
   P=getLocalPart(P_codis);             % Each worker operates on its data.
% Swim/tumble (chemotaxis) loop:
for j=1:Nc                              % Start for-loop of random movement on each worker.
        for i=1:S/4                     % Start for-loop of number of solution on each
worker.
% Compute the nutrient concentration at the current location of each solution
        J(i,j)=obj(P(:,i,j));
% Tumble:
        Jlast=J(i,j);                   % Initialize the cost value to be the first at the tumble.
        Delta(:,i)=(2*round(rand(p,1))-1).*rand(p,1);
                                        % Generate a random direction.
        C(i,j)=abs(J(i,j))/((abs(J(i,j)))+alpha);
                                        % Updating variable run length unit.
 % Next, move all the solution by a small amount in the direction that the tumble resulted in
        P(:,i,j+1)=P(:,i,j)+C(i,j)*Delta(:,i)/sqrt(Delta(:,i)'*Delta(:,i));
                                        % This adds a unit vector in the random direction,
                                        % scaled by the step size C(i,j).
% Swim (for solutions that seem to be headed in the right direction):
% Check parameters in range of search space.
while   P(1,i,j+1)> xlimit(1,1) | P(1,i,j+1)< xlimit(2,1) | ...
        P(2,i,j+1)> xlimit(1,2) | P(2,i,j+1)< xlimit(2,2) | ...
        P(3,i,j+1)> xlimit(1,3) | P(3,i,j+1)< xlimit(2,3) | ...
        P(4,i,j+1)> xlimit(1,4) | P(4,i,j+1)< xlimit(2,4) | ...
        P(5,i,j+1)> xlimit(1,5) | P(5,i,j+1)< xlimit(2,5) | ...
        P(6,i,j+1)> xlimit(1,6) | P(6,i,j+1)< xlimit(2,6) | ...
        P(7,i,j+1)> xlimit(1,7) | P(7,i,j+1)< xlimit(2,7) | ...
```

```
        P(8,i,j+1)> xlimit(1,8)   | P(8,i,j+1)< xlimit(2,8) | ...
        P(9,i,j+1)> xlimit(1,9)   | P(9,i,j+1)< xlimit(2,9) | ...
        P(10,i,j+1)> xlimit(1,10) | P(10,i,j+1)< xlimit(2,10) | ...
        P(11,i,j+1)> xlimit(1,11) | P(11,i,j+1)< xlimit(2,11) | ...
        P(12,i,j+1)> xlimit(1,12) | P(12,i,j+1)< xlimit(2,12)
        Delta(:,i)=(2*round(rand(p,1))-1).*rand(p,1);
        C(i,j)=abs(J(i,j))/((abs(J(i,j)))+alpha);
        P(:,i,j+1)=P(:,i,j)+C(i,j)*Delta(:,i)/sqrt(Delta(:,i)'*Delta(:,i));
end
        J(i,j+1)=obj(P(:,i,j+1));          % Compute cost value for each solution after
                                           % a small step (used by the solution to decide if it
                                           % should keep swimming).
        m=0;                               % Initialize counter for swim length
        while m<Ns                         % While climbing a gradient.
                m=m+1;
                if J(i,j+1)<Jlast          % Check a new cost value and move it further in
                                            % same direction.
                Jlast=J(i,j+1);            % First, save the cost value at current location.
% Next, extend the run in the same direction since it climbed at the last step:
                P(:,i,j+1)=P(:,i,j+1)+C(i)*Delta(:,i)/sqrt(Delta(:,i)'*Delta(:,i));
                J(i,j+1)=obj(P(:,i,j+1));  % Compute cost value at where it swam to and give
                                           % it new cost value.
                else                       % It did not move up the gradient so stop the run for
                                           % this solution.
                m=Ns;
                end
        end                                % Test if should end run step.
        end                                % End run for number of solution, S.
        end                                % End run for random movement Nc.
```

```
Jgather=[J{1}; J{2}; J{3}; J{4}];          % Return all cost values, J from workers to client.
reproduction = Jgather (:,1:Nc);           % Select cost value.
[jlastreproduction,iter] = min(reproduction,[],2);
                                           % Find minimum cost value and its position
                                           % among random movement, Nc.
[min_cost_function,back_ID] = min(jlastreproduction);
                                           % Find minimum cost value and its position
                                           % among each number of solution, S.
Pgather =[P{1} P{2} P{3} P{4}];            % Return all solutions, P from workers to client.
X_Y_best=Pgather (:,back_ID,iter(back_ID,:));
                                           % Select an elite solution and store in X_Y_best.
matlabpool close                           % close MATLAB pool.
return
```

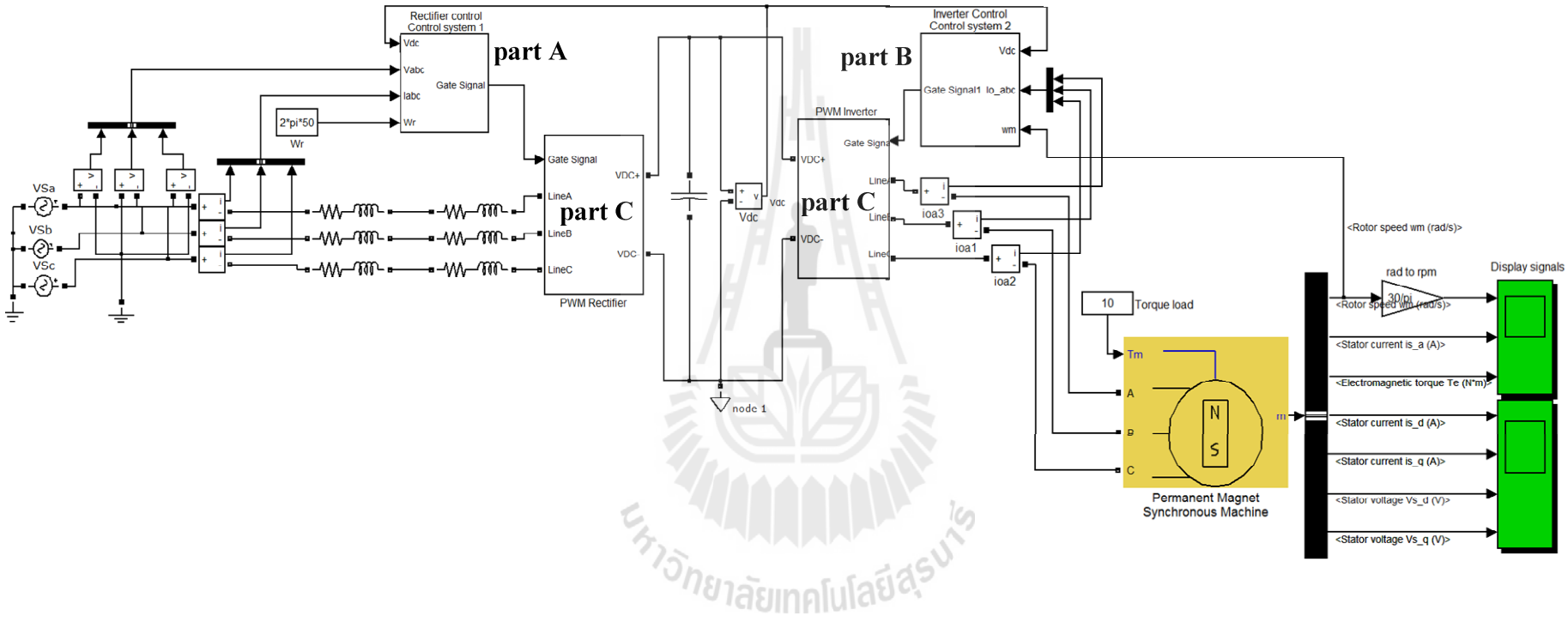
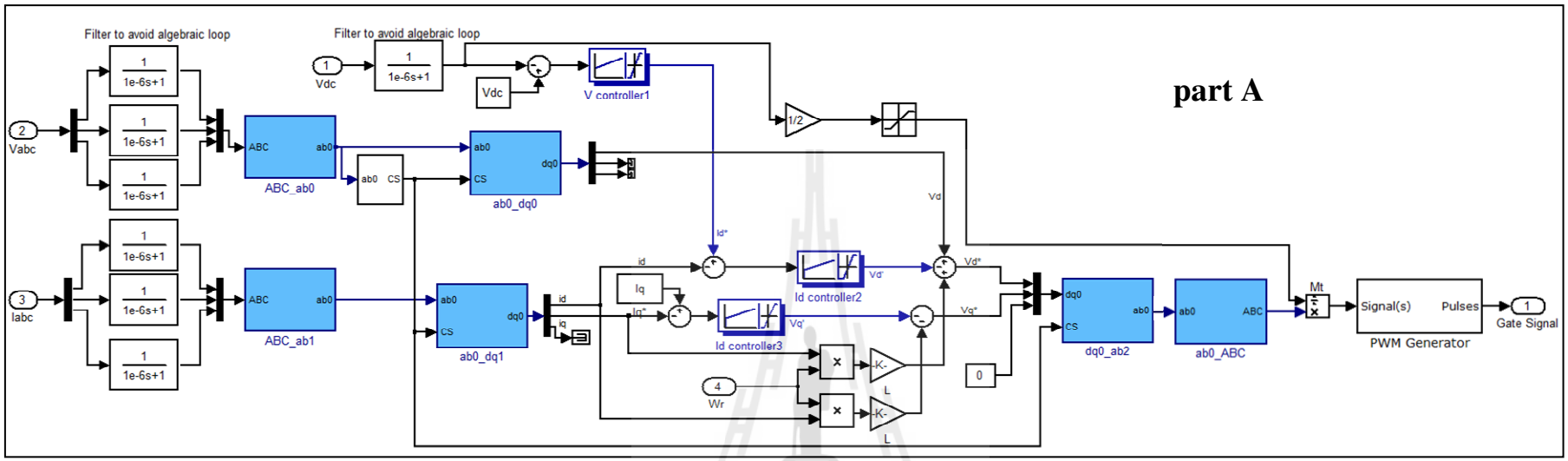
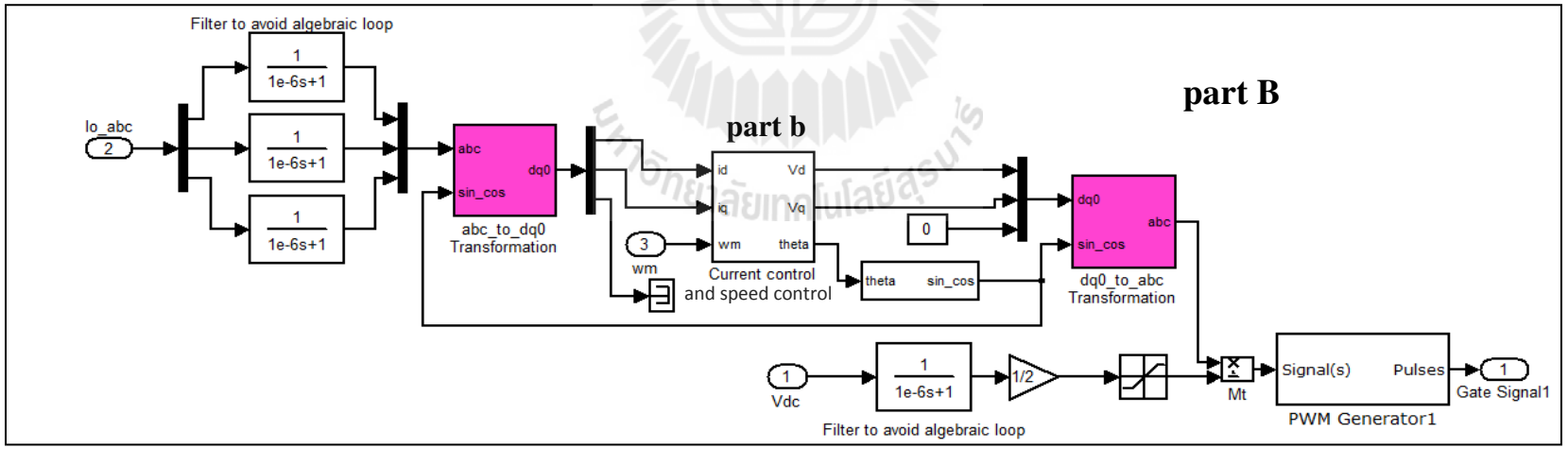


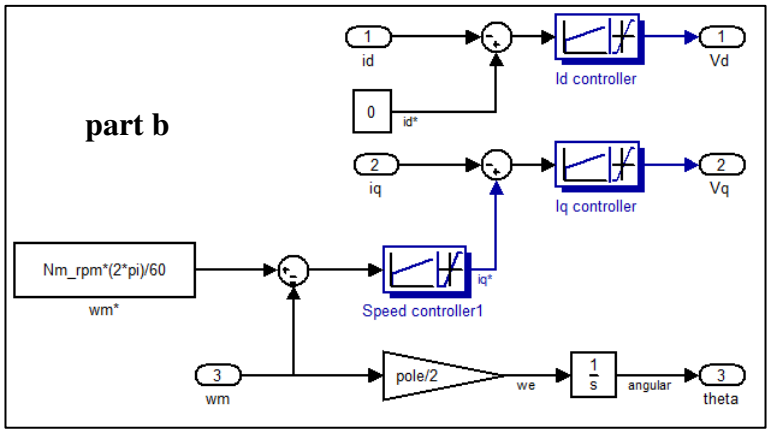
Figure E.1 Simulink model of whole drive system.


part A
Filter to avoid algebraic loop
Filter to avoid algebraic loop
Vabc
Vdc
V controller1
ABC_ab0
ab0_dq0
Id controller2
Id controller3
ABC_ab1
ab0_dq1
labc
Wr
dq0_ab2
ab0_ABC
Mt
Signal(s)
Pulses
PWM Generator
Gate Signal

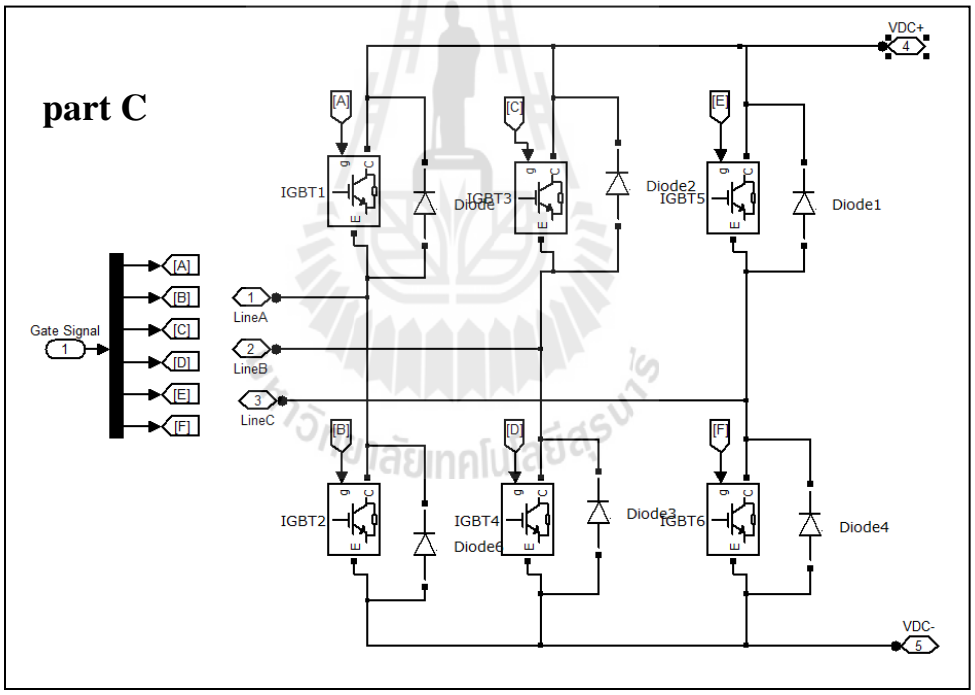
(a)


part B
part b
Filter to avoid algebraic loop
Io_abc
abc_to_dq0
Transformation
wm
Current control
and speed control
dq0_to_abc
Transformation
Vdc
Filter to avoid algebraic loop
Mt
Signal(s)
Pulses
PWM Generator1
Gate Signal1

(b)



(c)



(d)

Figure E.2 Sub-models of whole drive system: (a) control system 1 for PWM rectifier, (b) control system 2 for PWM inverter, (c) current and speed control loops and (d) switching system.

APPENDIX F

RECOMMENDATION OF SUPERVISOR FROM THE UNIVERSITY OF NOTTINGHAM

Dr. Pericle Zanchetta

01159515592
pericle.zanchetta@nottingham.ac.uk

The University of Nottingham

Department of Electrical and Electronic Engineering
University Park
Nottingham
NG7 2RD
Tel +44 (0)115 951 5600
Tel +44 (0)115 951 5616
Head of Department:
Professor Trevor Benson BSc PhD MIEEE

To: Whom it may concern

28th February 2012

Final statement for Ms Nuapett Sarasiri research visit

Dear Sir/Madam

Ms **Nuapett Sarasiri** has spent a 6 month research period working under my supervision in the PEMC (Power Electronics machines and Control) research group of the University of Nottingham, UK, from September 1st 2011 to February 29th 2012. She was involved in a quite challenging project investigating the multi objective design optimization of a Permanent Magnet motor drive, where power converter and electrical machine are considered as a whole. This is an interesting issue given the different dynamics of the two systems and the incredible amount of computational power needed. This required substantial theoretical and modelling skills and the capability of deeply understanding power electronics and electrical machines subjects. For this project she used a novel meta-heuristic optimization algorithm developed by her during her PhD studies in Thailand.

She showed a great independence since from the start; she was able to undertake the work with minimal supervision, demonstrating an excellent capability to manage and plan her work. This capability is far in advance the average PhD student. The achieved final results of this project are excellent, and Nuapett will continue with some more refinements of the method when back home in Thailand. A practical rig is being set up at the moment and experimental tests will be undertaken in the next month to validate **Nuapett** work.

The excellent quality and novelty of her research project achievements up to date are worth of publication and she is now preparing a scientific paper for a IEEE journal in the power electronics/electrical machines subjects.

Nuapett has improved her English language capabilities substantially during her stay in Nottingham. She is of a pleasant personality, very mature as a student and a very polite person; she communicates and relates very well with staff and colleagues. She is trustworthy and hardworking, willing to learn new concepts and very motivated. I am very happy having worked with her and willing to keep collaborating with her and her home institution in the future.

Yours faithfully

Pericle Zanchetta

Dr Pericle Zanchetta
Associate Professor
Power Electronics, Machines and Control Group
University of Nottingham, University Park, NG7 2RD Nottingham

APPENDIX G

LIST OF PUBLICATIONS

List of Publications

Sujitjorn S., Kluabwang J., Puangdownreong D. and Sarasiri N. (2010). Adaptive Tabu Search and Management Agent. **ECTI Transaction on Electrical Eng. Electronics and Communications**, 8(1). pp. 1-10. (Thai JIF = 0.021)

Sarasiri N., Srikaew A. and Sujitjorn S. (2010). Dynamic Compensation of Hard-Disk R/W Head and Head-Stack. **WSEAS Transactions on Systems**. 9(7). pp. 764-773. (SJR = 0.420)

Panikhom S., Sarasiri N. and Sujitjorn S. (2010). Hybrid Bacterial Foraging and Tabu Search Optimization (BTSO) Algorithms for Lyapunov's Stability Analysis of Nonlinear Systems. **International Journal of Mathematics and Computes in Simulation**. 4(3). pp. 81-89. (SJR = 0.240)

Sarasiri N., Carawukh S., Boonpranchoo V., Kumbla K. and Sujitjorn S. (2011). Identification of Hard-Disk Head Actuator Model Using Bacterial Foraging-Tabu Search Metaheuristics. **American Journal of Scientific and Industrial Research**. 2(4). pp. 686-689.

Sarasiri N. and Sujitjorn S. (2011). Control Design Optimization of Truck Braking System using Bacterial-Foraging-Tabu-Search Metaheuristics. **World Academy of Science**, **Engineering and Technology**. 80. pp. 1189-1193. (SJR = 0.120)

Sarasiri N., Suthamno K. and Sujitjorn S. (2012). Bacterial Foraging-Tabu Search Metaheuristics for Identification of Nonlinear Friction Model. **Journal of Applied Mathematics**. 2012(238563). pp. 1-23. (JIF = 0.720)

*Note that JIF stands for Journal Impact Factor, and SJR stands for Scientific Journal Rankings.

Panikhom S., Sarasiri N. and Sujitjorn S. (2010). Numerical Approach to Lyapunov's Stability Analysis of Nonlinear Systems Using Threshold Accepting Algorithms, **ICT-CSCC International Conferences**. Pattaya. Thailand. July 4-7. 2010. pp. 811-814.

Sarasiri N., Srikaew A. and Sujitjorn S. (2010). Dynamic Compensations for Hard Disk Heads, **Proceedings of the 4th International Conference on Circuits, Systems and Signals**. Corfu Island, Greece. pp. 54-57.

Sarasiri N. and Sujitjorn S. (2010). Bacterial Foraging Optimization and Tabu Search: Performance Issues and Cooperative Algorithms. **WSEAS International Conferences**. ISBN: 978-960-474-218-9. Taipei, Taiwan. pp. 186-191.

Carawukh S., Sarasiri N., Boonpranchoo V., Kumbla K. and Sujitjorn S. (2011). Application of Hybrid BF-TS Algorithms to Identification of HDD Actuator Model. **The 2011 World Congress on Engineering and Technology**. Shanghai, China. pp. 1-4.

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2012, Article ID 238563, 23 pages
doi:10.1155/2012/238563

Research Article

Bacterial Foraging-Tabu Search Metaheuristics for Identification of Nonlinear Friction Model

Nuapett Sarasiri, Kittiwong Suthamno, and Sarawut Sujitjorn

Power Electronics, Machines, and Control Research Group, Control and Automation Research Unit, School of Electrical Engineering, Suranaree University of Technology, Nakhon Ratchasima 30000, Thailand

Correspondence should be addressed to Sarawut Sujitjorn, sarawut@sut.ac.th

Received 20 February 2012; Accepted 28 April 2012

Academic Editor: Hak-Keung Lam

This paper proposes new metaheuristic algorithms for an identification problem of nonlinear friction model. The proposed cooperative algorithms are formed from the bacterial foraging optimization (BFO) algorithm and the tabu search (TS). The paper reports the search comparison studies of the BFO, the TS, the genetic algorithm (GA), and the proposed metaheuristics. Search performances are assessed by using surface optimization problems. The proposed algorithms show superiority among them. A real-world identification problem of the Stribeck friction model parameters is presented. Experimental setup and results are elaborated.

1. Introduction

AI-based algorithms have been successfully applied to solve optimal solutions of complex and NP-hard problems in engineering. For some examples, dated back to 1995, the genetic algorithm was proposed to solve line-balancing problem to minimize the cycle time of the line for a given number of workstations [1], similar approach was used to optimize the determination of an optimal control sequence in model-based predictive control [2], the simulated annealing and genetic algorithms were used to solve nonlinear controller parameter optimization for the diving and heading motions of a submarine model [3], and recently the particle swarm optimization algorithm was applied to optimize the parameters of a fuzzy control system that was used for the vibration control problem of a flexible structure [4], and so forth. Of a particular interest in control, current motion control technologies demand very high precision in positioning of an object, for example, in CNC machines, robots, and so forth. Difficulties caused by nonlinear friction arise under very low velocity motions, in which stick-slip phenomenon pronouncedly exhibits. Control techniques can

be used to compensate for the problem, and as such an accurate friction model is needed. To obtain the model it requires careful experimental setup, accurate measurement and appropriate identification algorithms. Due to strong nonlinearity of the friction model, the conventional regressive approach is inappropriate.

An identification task can be formulated as an optimization problem solvable by available optimization algorithms. Artificial intelligence- (AI-) based methods are efficient candidates of the present technology. There is a wide range of algorithms which has been applied to solve identification problems. For some instances, the genetic algorithm has been applied to various problems including the modelling of a laboratory scale process involving a coupled water tank system and the identification of a helicopter rotor speed controller [5] and the identification of induction machine parameters [6, 7]; the tabu search and simulated annealing algorithms were applied to identify the optimal parameter structure for groundwater models [8]; the adaptive tabu search was applied to harmonic identification for an active power filter [9]; the particle swarm optimization was applied to solve various problems including the identification of thermal power plant [10], the calculation of deformations in soil or rock in geotechnical engineering [11], and the identification of MIMO FIR systems [12]; the ant colony optimization algorithm was used to identify the aquifer parameters for the underground water control engineering [13]; the proportionate affine projection algorithm was applied to the identification of sparse impulse response [14]; recently, the bacterial foraging algorithm was applied for the radiofrequency identification (RFID) communication system [15].

Among those metaheuristics, evolutionary and bioinspired algorithms have gained major interests since they are not hard to understand, and programming according to the procedural lists is not so strict compared with conventional scientific programming. It also opens a new route to effectively obtain an optimal or suboptimal solution for a complex system. This paper proposes the use of the tabu search (TS) and the bacterial foraging optimization (BFO) algorithm in a cooperative manner. Due to the dominant explorative property of the BFO algorithm, the modified TS with the BFO built-in is able to start searching with an elite initial solution. While the sharp focusing property of the TS remains, the proposed algorithms can move towards the solution very rapidly. Section 2 of the paper explains the algorithms. The search performance was investigated using some well-known unconstrained optimization problems on Pentium IV, 2.4 GHz, 640 Mbytes SD-RAM. The search results are compared among the BFO, the TS, the genetic algorithm (GA), and the modified TS denoted as bacterial foraging-tabu search (BTS). Since the GA is well known, review of the algorithms is omitted herein. The comparative results of performance studies are presented in Section 3. The proposed algorithms are applied to identify the parameters of the nonlinear Stribeck friction model. Section 4 presents the experimental setup, results, and discussions. Conclusions follow in Section 5.

2. Bacterial Foraging-Tabu Search Metaheuristics

2.1. Tabu Search

Tabu search (TS) originated by Glover [16, 17] has become one of the most efficient metaheuristic methods. It incorporates two major strategies, namely, intensification and diversification, respectively. Successful applications of the TS have appeared in various fields, for example, food processing [18], optimal power flow [19], flow shop problem [20], and so forth. For some complex systems containing many local optima, the simplistic TS is

usually unable to release the search move from a local entrapment. This problem has been overcome by different modifications made to the TS. These include the reactive tabu search [21], probabilistic tabu search [22] and adaptive tabu search (ATS) [23]. The ATS has found various successful applications such as identification [9], control [24], and signal processing [25].

The ATS consists of two major additional strategies made to the conventional TS. These are back-tracking (BT) and adaptive search radius (AR) mechanisms, respectively. The former assists the TS to release itself from being locked by a local solution. It looks up the tabu list (TL), that is, short-term memory, for a visited elite solution, and uses this solution for starting a new search move. The later enhances the focusing characteristic of the TS. This strategy decreases the search radius gradually when the search comes close to a solution of high quality having a potential of being the optimal one. However, too short of the search radius could result in a slow search. Recommendations for selection of search parameters are in [23]. The ATS algorithm is presented as a part of the proposed algorithms in Section 2.3.

2.2. *Bacterial Foraging Optimization*

In 2002, Passino developed a new bioinspired optimization algorithm called bacterial foraging optimization (BFO) [26, 27]. The algorithm imitates the foraging behavior of *E. coli* bacteria; the computer codes appear in http://www2.ece.ohio-state.edu/~passino/ICbook/ic_code.html. The BFO has been applied to various optimization problems including estimation of harmonics [28], active power filter design [29], transmission loss reduction [30], and optimal power flow [31]. Unfortunately, the BFO sometimes does not converge to a high-quality solution, particularly when applied to complicated problems. The difficulty has been resolved by some researchers [32, 33] who modified the chemotaxis step of the BFO to become an adaptive mechanism and, hence, the name adaptive bacterial foraging optimization, or ABFO. It incorporates an adaptive jumping step denoted as $C(i)$ under a basic concept of a long step corresponding to a large deviation of the cost value from the targeted one, and vice versa. The ABFO algorithm consists of 4 main mechanisms as follows.

Chemotaxis

This mechanism imitates the swimming movement of a bacterium. The position of a bacterium is denoted as θ^i.

Swarming

When one bacterium presents itself in an elite position, that is, a local hill or valley, it attracts the others. Simultaneously, each bacterium tries to repel the others nearby. The attractive and the repellent effects are modeled as weighted summation of exponential terms presenting the objective function, J_{CC}. The weighting factors are d_{attract}, w_{attract}, $h_{\text{repellant}}$, and $w_{\text{repellant}}$ and can be chosen arbitrarily.

Reproduction

The bacteria are classified during the computing process as healthy and unhealthy due to their cost values. Only the healthy ones reproduce by duplicating themselves at the same positions.

Elimination and Dispersal

The mechanism allows the unhealthy bacteria to be discarded. The healthy ones are dispersed randomly over the search space with the probability P_{ed}.

The original BFO and the ABFO algorithms run iteratively and terminate on the maximum iteration criterion. The solutions obtained from search are stored in a memory and eventually sorted to find the optimal solution. From testing the ABFO algorithm, it demonstrates a strong explorative (or diversification) property. This property is commonly found in population-based algorithms, and the ABFO algorithm is one of them. In contrast, single-solution-based algorithms, such as the TS, have strong exploitative property [34–36]. Therefore, both algorithms complement each other in the senses that the ABFO is useful for provision of an elite initial solution to the TS, and the TS is an efficient tool to track down a global solution rapidly. The ABFO algorithm is presented as a part of the algorithms in the next section.

2.3. Bacterial Foraging-Tabu Search

As mentioned, the TS has a dominant focusing characteristic, while the ABFO is strong in explorative operation. Such properties can complement each other. Since the TS has straightforward procedures, and moves rapidly towards a local solution, the method forms the hunting steps for a satisfied solution to the problem. The two algorithms are combined to form new metaheuristics working in a cooperative manner. The new algorithms are referred to as bacterial foraging-tabu search or BTS in short. In this new algorithmic form, it is unnecessary to employ the reproduction mechanism of the ABFO part because ranking the available solutions to single out one with the minimum cost is an important step. This specific solution is transferred to the TS part as an initial solution. The procedural list of the BTS algorithms is as follows.

Step 1. Initialize search parameters: p, *search space*, S, N_c, N_s, α, d_{attract}, w_{attract}, $h_{\text{repellant}}$, $w_{\text{repellant}}$, R, N, TL, $\text{count}_{\max}$, BT, *n_re_back*, *best_neighbor1*, *best_error*, R_i and ε_i.

Step 2. Randomly or heuristically select an initial solution θ^i from the search space. Set θ^i as the current solution.

Step 3. Compute objective functions $J(i,j)$ according to (2.1) ($i = 1, 2, \ldots, S$). Set $J_{\text{last}} = J(i,j)$; $j = 1, 2, \ldots N_c$,

$$
\begin{aligned}
J(i,j) &= J(i,j) + J_{\text{CC}}\left(\theta^i(j), P(i,j)\right) \\
J_{\text{CC}}\left(\theta^i, P(i,j)\right) &= \sum_{i=1}^{S} J_{\text{CC}}^i\left(\theta, \theta^i(j)\right) \\
&= \sum_{i=1}^{S}\left[-d_{\text{attract}} \exp\left(-w_{\text{attract}} \sum_{m=1}^{p}\left(\theta_m - \theta_m^i\right)^2\right)\right] \\
&\quad + \sum_{i=1}^{S}\left[h_{\text{repellent}} \exp\left(-w_{\text{repellent}} \sum_{m=1}^{p}\left(\theta_m - \theta_m^i\right)^2\right)\right].
\end{aligned}
\tag{2.1}
$$

Step 4. Generate randomly ([−1, 1]) the elements of the random vector $\Delta_p(i) \in \Re^p$, then compute the adaptive step size, $C(i, j)$ using (2.2), and update the solution $\theta^i(j+1)$ according to (2.3). Compute the objective function for $j = j + 1$ according to (2.1). Set $m = 0$,

$$C(i,j) = \frac{|J(i,j)|}{|J(i,j)| + \alpha} = \frac{1}{1 + \alpha/|J(i,j)|}, \tag{2.2}$$

$$\theta^i(j+1) = \theta^i(j) + C(i,j)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}. \tag{2.3}$$

Step 5. If $J(i, j + 1) < J_{\text{last}}$ then $J_{\text{last}} = J(i, j + 1)$; use the direction of the same random vector $\Delta(i)$ to compute $\theta^i(j + 1)$ and $J(i, j + 1)$. Update m and repeat Step 5 until $m > N_s$.

Step 6. If $j \leq N_c$, go to Step 3.

Step 7. Do minimum sorting of the objective functions J. Define *best_θ* as the solution with the minimum J. Set $S_0 = best\_\theta$.

Step 8. Generate a neighbourhood around S_0 within an initial search radius R. Set N solutions as the members of the set $S_1(r)$.

Step 9. Evaluate the objective function of each member belonging to $S_1(r)$. Define $S_1 =$ *best_neighbor1* as a solution with the minimum cost, J_1.

Step 10. If $J_1 < J_0$, store S_0 in the TL, assign $S_0 = S_1$, otherwise, store S_1 in the TL.

Step 11. Invoke the BT when a solution deadlock occurs (the current solution has been repeated many times as defined by $n = 1, 2, \ldots,$ BT) (Algorithm 1).

Step 12. If the termination criterion based on the J values is met or count > $\text{count}_{\max}$ (count = $1, 2, \ldots, \text{count}_{\max}$) exit with the global solution.

Step 13. Invoke the AR when the current solution is relatively close to a local minimum (Algorithm 2).

Step 14. Updated count. If count ≤ $\text{count}_{\max}$ then go to Step 8.

Referring to Step 12, one termination criterion is the maximum number of iterations. There are other approaches the users may employ, that is, maximum CPU time, maximum iteration with or without improvement in solution quality, and a solution of sufficient quality [35]. Another termination criterion is the sufficient solution quality concept, which is represented by the cost J. The value of the cost J depends on application. For engineering problems, the cost J can be set from design specifications, component tolerances, and so forth. The maximum number of iterations ($\text{count}_{\max}$) can be determined from the ratio of search boundary to search radius of the TS. Some presearch trials are useful to determine an adjustment to the value of $\text{count}_{\max}$. The above algorithms are general enough for various optimization problems. Specific alterations for the problem of friction model identification will be discussed in Section 4.

BT: if $n \geq BT$
  $n = n + 1$
  $best\_error = RANK(TL)$
look back in the TL, then retrieve the $n\_re\_back^{th}$ solution from the TL.
else
  $n = 0$
define $S_0 = best\_neighbor$
  $best\_error = best\_error$
end if

Algorithm 1

AR: if $best\_error < \varepsilon_1$
$R = R_1$ where $R < R_1$.
  end
  if $best\_error < \varepsilon_2$
$R = R_2$ where $R_2 < R_1$ and $\varepsilon_2 < \varepsilon_1$.
  end
...
  if $best\_error < \varepsilon_n$
$R = R_n$ where $R_n < R_{n-1}$ and $\varepsilon_n < \varepsilon_{n-1}$.
  end

Algorithm 2

3. Search Performance

This section presents the performance comparison studies among the following algorithms: adaptive tabu search (ATS), adaptive bacterial foraging optimization (ABFO), bacterial foraging-tabu search metaheuristics (BTS), and genetic algorithm (GA). Review of the GA is omitted since the algorithm is well known. Good sources that readers may refer to are [37–39]. Each of these algorithms performs search on several test functions for 50 trials, and the results are averaged. Each search trial begins the search with different initial solutions, while search parameters are kept the same for all trials.

This approach is commonly referred to as multiple-points-single-strategy (MPSS) in metaheuristic contexts. The test functions adapted are well-known unconstrained problems for testing optimization algorithms. These include Bohachevsky function (BF), Rastrigin function (RF), Shekel's fox-holes function (SF), Schwefel function (SchF), and Shubert function (ShuF), respectively. Table 1 summarizes these test functions in which $J_{\min}$ is the minimum cost required to terminate the search. Search parameter settings for the ATS follow [23], the for ABFO follow [26], and for the GA follow MATLAB-GA Toolbox [39]. Tables 2 and 3 summarize the search parameters of the ATS and the ABFO, respectively. These parameters are adapted for the BTS with $N_c = 20$, in particular.

Table 4 summarizes the average results over 50 trials. There are 2 groups of data denoted as average search time and average search rounds, respectively. Since the ATS, ABFO, and GA have different algorithmic approaches, comparisons of their average search rounds are not meaningful. On the contrary, the ABFO is combined to the ATS in order to

Table 1: Summary of the unconstrained problems used for performance test.

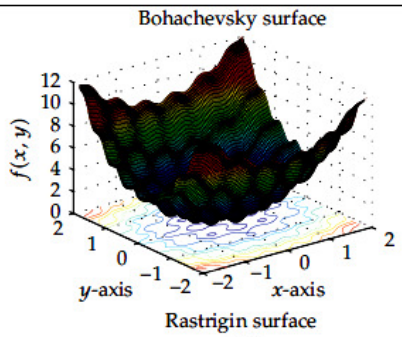
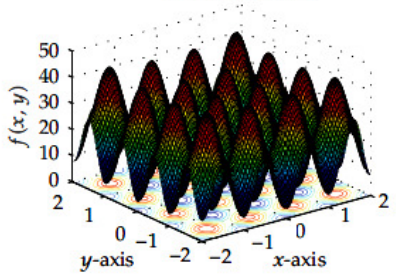
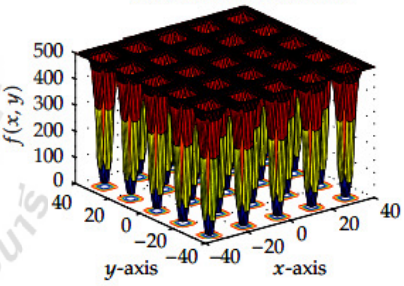
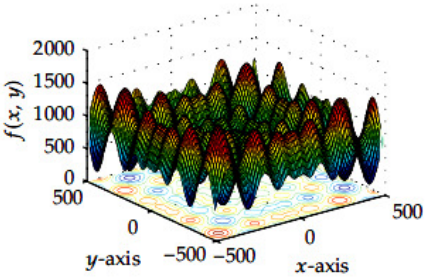
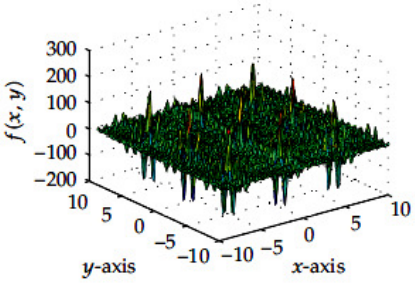
Test functions	Equations	Surfaces in 3D
BF	$f(x,y) = x^2 + 2y^2 - 0.3\cos(3\pi x) - 0.4\cos(4\pi y) + 0.7$ global minimum of 0 at (0, 0), search space: [2 2; −2 −2] and $J_{\min} \leq 1 \times 10^{-9}$.	
RF	$f(x,y) = x^2+y^2-10\cos(2\pi x)-10\cos(2\pi y)+20$ global minimum of 0 at (0, 0), search space: [2 2; −2 −2] and $J_{\min} \leq 1 \times 10^{-8}$.	
SF	$f(x_1,x_2) = [1/500 + \sum_{j=1}^{25}(1/(j + \sum_{i=1}^{2}(x_i - a_{ij})^6))]^{-1}$ when $a_{ij} = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \cdots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \cdots & 32 & 32 & 32 \end{pmatrix}$. global minimum of 0.998 at (−32, −32), search space: [40 40; −40 −40] and $J_{\min} \leq 0.9990$.	
SchF	$f(x) = 418.9829n - \sum_{i=1}^{n}(x_i \sin\sqrt{\lvert x_i \rvert}); n = 2$ global minimum of 0 at (420.9687, 420.9687), search space: [500 500; −500 −500] and $J_{\min} \leq 1 \times 10^{-1}$.	
ShuF	$f(x_1,x_2) = \sum_{i=1}^{5} i\cos((i+1)x_1 + i) * \sum_{i=1}^{5} i\cos((i+1)x_2 + i)$ when $-10 \leq x_1, x_2 \leq 10$. global minima of equal values −186.7309 at 18 different locations, search space: [10 10; −10 −10] and $J_{\min} \leq -186.73$.	

Table 2: ATS parameters.

Test functions	N	$\text{Count}_{\max}$	R	BT, *n_re_back*	AR		
					Stage I	Stage II	Stage III
BF	30	10,000	0.2	5	$J < 1 \times 10^{-1}$,	$J < 1 \times 10^{-3}$,	—
RF	30	10,000	0.2	5	$R = 2 \times 10^{-3}$	$R = 2 \times 10^{-4}$	—
SF	30	10,000	0.8	5	$J < 5$, $R = 0.5$	$J < 2$, $R = 0.1$	—
SchF	30	10,000	50	5	$J < 100$, $R = 25$	$J < 10$, $R = 10$	$J < 1$, $R = 1$
ShuF	30	10,000	1.0	5	$J < 1$, $R = 1 \times 10^{-2}$	$J < -1$, $R = 1 \times 10^{-3}$	—

Table 3: ABFO parameters.

Test functions	ABFO parameters										
	S	N_c	N_S	N_{re}	N_{ed}	P_{ed}	α	d_{attract}	$h_{\text{repellant}}$	w_{attract}	$w_{\text{repellant}}$
BF	30	20	4	4	2	0.25	10	0.1	0.1	0.2	1
RF	30	200	4	4	2	0.25	100	0.1	0.1	0.2	1
SF	30	500	4	4	2	0.25	0.01	0.1	0.1	0.2	10
SchF	30	2000	4	4	2	0.25	1	0.1	0.1	0.2	10
ShuF	30	1000	4	4	2	0.25	1	0.1	0.1	0.2	10

reduce search rounds and search time. So, it is meaningful to compare the search rounds consumed by the ATS and the BTS. In average, the proposed BTS consumes search rounds of 63.45% less than the ATS does. Referring to the search time data in Table 4, the BTS spends search time of 37.15% less than the ATS does, and 68.31% less than the ABFO does as averages. Moreover, the BTS consumes 58.21% less search time than the GA does. In terms of the number of local entrapment, the BTS encounters the entrapment of 73.43% less than the ATS does, and produces high-quality solution to the problem. Convergence curves are shown in Figure 1 for comparison purposes.

Table 5 summarizes the solutions obtained from different methods. It can be noticed that the ABFO provides solutions with the best quality in an exchange of a considerably long search time (see Table 4). The proposed BTS provides solutions with second to the best quality within the shortest search time (see Table 4). Note that the solutions found by the BTS meet the criterion of minimum cost. This outstanding performance of the BTS is achieved due to the explorative characteristic of the ABFO, the exploitative characteristic of the TS, and the deadlock releasing property of the ATS.

Figure 2 shows a representation of bacteria movements on the search space of the test functions. Noticeably, large areas of search spaces are explored by the bacteria. In other words, the BTS can provide a high-quality solution rapidly because the ABFO provides an elite initial solution to the ATS, and the ATS effectively releases the search from a deadlock or local entrapment while rapidly focuses the search to the solution.

The BTS has been applied to a constrained parametric search problem, that is, an identification of the nonlinear friction model. In the next section, experimental setup, and identification results are presented.

Table 4: Summary of the results (averaged over 50 trials).

Test functions	Average search time (seconds)				Average search rounds			
	ATS	ABFO	BTS	GA	ATS	ABFO (N_C)	BTS	GA
BF	11.66	5.75	6.83	49.12	616.48	20	151.20	1177.18
RF	14.60	48.63	5.81	11.78	868.28	200	323.30	225.54
SF	4.18	146.67	3.69	8.53	139.36	500	25.70	141.28
SchF	408.58	728.87	172.52	868.32	6889.42	2000	1469.14	354.44
ShuF	3.28	182.83	2.80	3.39	68.06	1000	55.28	34.24

Table 5: Solutions obtained from different approaches.

Objective function	ATS	ABFO	BTS	GA
BF:				
Average	4.5090e−10	1.0833e−10	5.30112e−10	6.0074e−08
Min	7.3184e−11	2.2204e−16	3.569e−12	1.1830e−11
Max	9.4151e−10	8.8759e−10	9.97229e−10	2.9855e−06
Std.	2.7476e−10	2.0742e−10	3.13515e−10	4.2216e−07
RF:				
Average	5.3478e−09	1.1534e−10	5.2460e−09	4.9642e−09
Min	2.9051e−10	3.5527e−15	5.9587e−10	1.1486e−10
Max	9.3771e−09	6.7094e−10	9.6620e−09	9.5235e−09
Std.	2.4328e−09	1.4942e−10	2.6721e−09	2.6740e−09
SF:				
Average	0.9982	0.9981	0.9983	0.9983
Min	0.9980	0.9980	0.9980	0.9980
Max	0.9989	0.9988	0.9990	0.9990
Std.	0.0002	0.0002	0.0003	0.0003
SchF:				
Average	0.0456	0.0139	0.0273	0.0439
Min	0.0006	2.5455e−05	0.0002	0.0008
Max	0.0998	0.0768	0.0944	0.0982
Std.	0.0278	0.0240	0.0341	0.0281
ShuF:				
Average	−186.7305	−186.7305	−186.7305	−186.7304
Min	−186.7309	−186.7309	−186.7309	−186.7309
Max	−186.7300	−186.7300	−186.7301	−186.7300
Std.	0.0003	0.0003	0.0003	0.0002

4. Identification Results

4.1. Experimental Setup

A closed loop position control system is a necessary test bed for monitoring stick-slip phenomenon. The diagram in Figure 3 represents the experimental setup. The linear slide bed

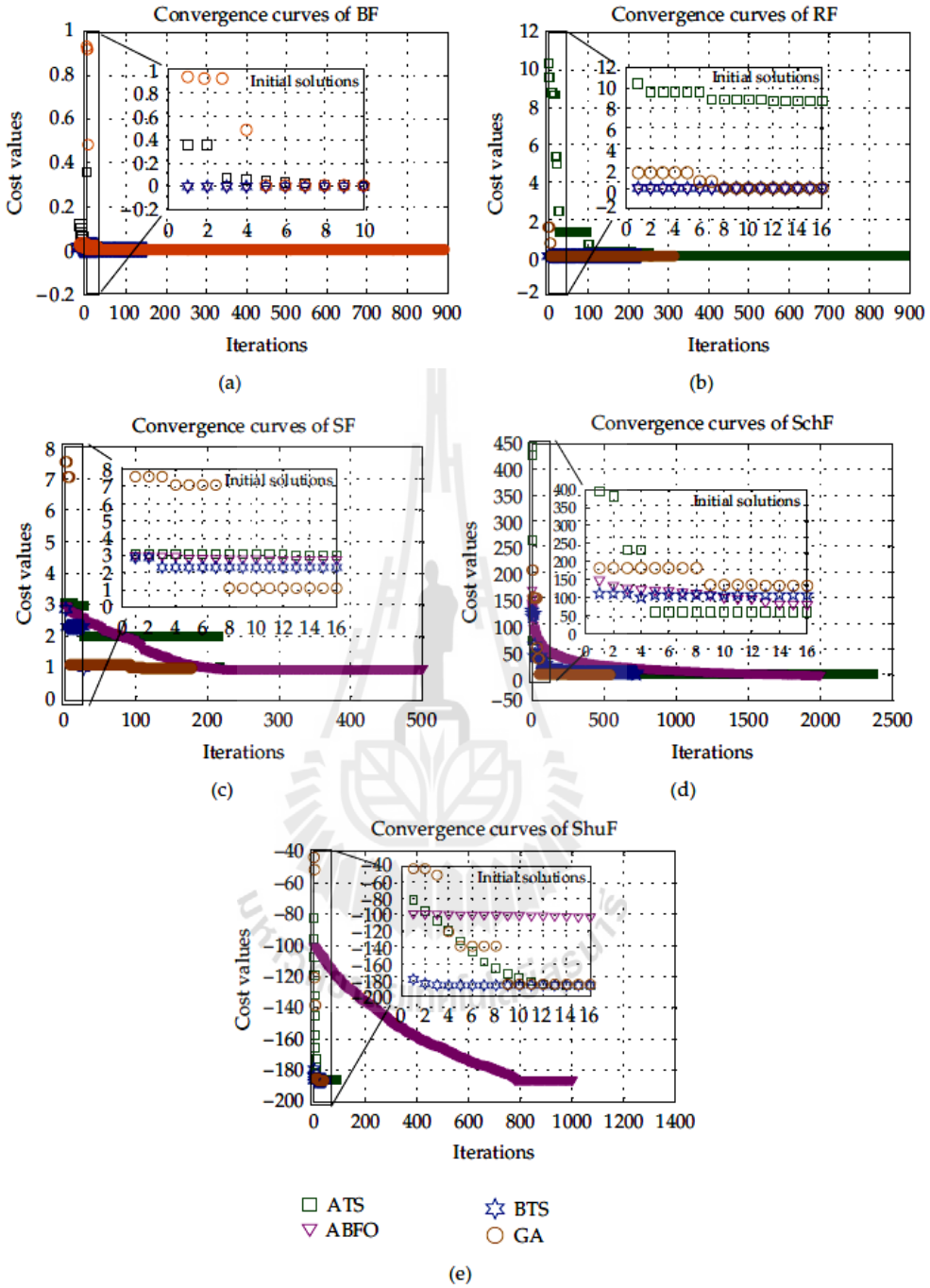


Figure 1: Convergence curves—(a) BF, (b) RF, (c) SF, (d) SchF, and (e) ShuF.

is the controlled plant consisting of a dc motor, a threaded rod, a reflector, and an ultrasonic transducer (UC3000-UIE2). The effective moving range of the reflector is 0–400 mm with the home position at the middle. In test mode, the motion control circuit performs an initial test move of the reflector for the whole range and, eventually, places the reflector at the home position.

For the reflector to follow a ramp command, a closed loop position control has been built. The hardware components consist of a PC as a P-controller, a 12-bit ADC, a 2Q-drive

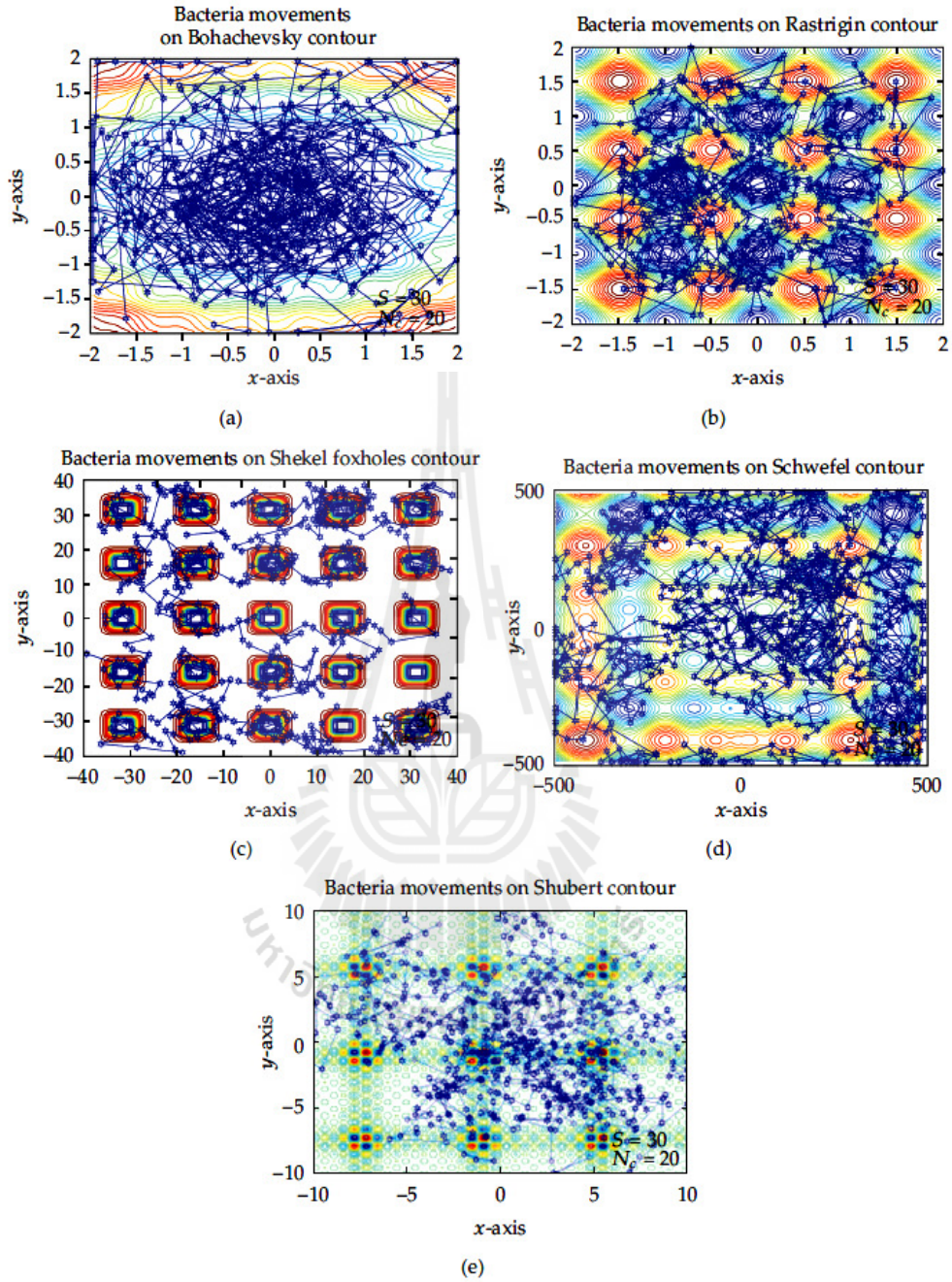


Figure 2: Bacterial search movements—(a) BF, (b) RF, (c) SF, (d) SchF, and (e) ShuF.

circuit, a current sensor, an ultrasonic transducer, a 2nd-order differentiator producing a speed signal from a position signal, and a few signal conditioning circuits including zero-span circuits and a bipolar voltage generator, respectively, and a dc power supply. In control mode, the motion follows an up-down ramp command directing the reflector to move rightward (positive direction, ramp-up command) and leftward (negative direction, ramp-down command). The reflector moves in the range of 50–350 mm in the control mode. A desired speed can be set via the keyboard of the PC functioning as a P-controller.

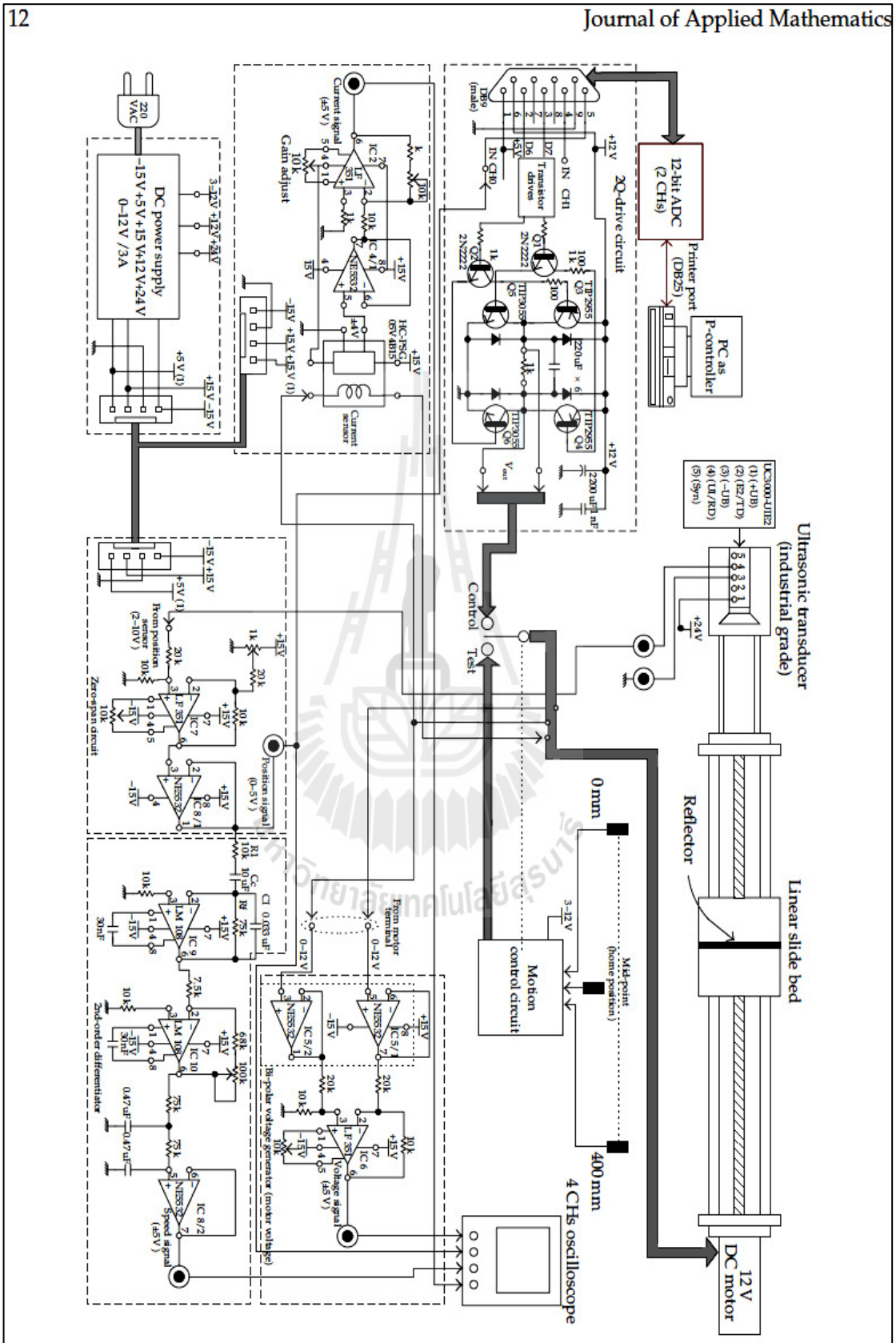


Figure 3: Circuit diagram representing the experimental setup.

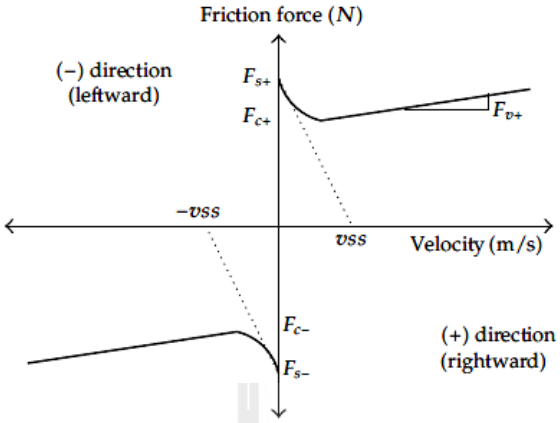


Figure 4: Stribeck friction curve.

4.2. Nonlinear Friction Model

When two solid materials translating over one another at very low velocity, a stick-slip phenomenon occurs. This phenomenon is caused by nonlinear friction characteristics also known as Stribeck's effect [40]. An effective model describing the friction can be represented by the curve in Figure 4, and is referred to as complex friction model or Stribeck model [41–43]. When an applied force to a mass cannot overcome the static friction, which is represented by either F_{S+} or F_{S-} depending on the direction of motion, the mass cannot move.

This situation is referred to as stick mode, and described by the stick-friction force

$$F_{\text{stick}}(F_{\text{in}}) = \begin{cases} F_{s+}, & F_{\text{in}} > F_{s+} \\ F_{\text{in}}, & F_{s-} \le F_{\text{in}} \le F_{s+} \\ F_{s-}, & F_{\text{in}} < F_{s-}. \end{cases} \tag{4.1}$$

Once the applied force is greater than the static friction, the mass begins moving. After a certain period of time, the mass keeps up a higher velocity during which it encounters both Coulomb and viscous frictions. This situation is known as slip mode and described by the slip-friction force

$$F_{\text{slip}}(v) = \left(F_C + (F_S - F_C) \cdot e^{-(v/vss)}\right) \cdot \text{sgn}(v) + F_v \cdot v. \tag{4.2}$$

To cover the whole velocity range, the friction force can be expressed in a compact form as

$$F_f(v, F_{\text{in}}) = \begin{cases} F_{\text{stick}}(F_{\text{in}}), & |v| = 0 \\ F_{\text{slip}}(v), & |v| \neq 0. \end{cases} \tag{4.3}$$

4.3. Objective Function Implementation

During the search process to identify the friction model parameters, an objective function (J) has to be evaluated repeatedly. To calculate the objective function, it is assumed that the translational dynamic can be represented by the mass-spring model as follows:

$$m\ddot{x} = k_{\text{spring}}(x_i - x_d) - F_f(v, F_{\text{in}}) + F_{\text{ex}}, \tag{4.4}$$

in which the parameters F_S, F_C, F_V, vss, and k_{spring} are to be identified. Below is the procedural list for objective function calculation.

Step 1. Calculate an average displacement $x_i = (\sum_{j=1}^{50} x_{\text{test}}(j))/50$.

For ramping-up motion, calculate an approximated force $F_{\text{motor}} = \hat{k}(x_i - x_d) \approx 3.5(x_i - x_d)$.

For ramping-down motion, calculate an approximated force $F_{\text{motor}} = \hat{k}(x_i - x_d) \approx 2.8(x_i - x_d)$.

An approximated force $F_{\text{motor}} = \hat{k}(x_i - x_d) \approx 2.8(x_i - x_d)$.

Step 2. If $(F_{\text{motor}} \geq F_{\text{fm}})$ then $(F_{f\_\text{motor}} = F_{\text{fm}})$.
If $(F_{\text{motor}} \leq -F_{\text{fm}})$ then $(F_{f\_\text{motor}} = -F_{\text{fm}})$.
If $(-F_{\text{fm}} < F_{\text{motor}} < F_{\text{fm}})$ then $(F_{f\_\text{motor}} = F_{\text{motor}})$.

Step 3. Calculate $F_{m\_\text{motor}}(v) = K_{\text{kt}}\ (J_m(dv/dt) + D_m v)$.
If $(F_{m\_\text{motor}} > 30\,\text{N})$ then $(F_{m\_\text{motor}} = 30\,\text{N})$.
If $(F_{m\_\text{motor}} < -30\,\text{N})$ then $(F_{m\_\text{motor}} = -30\,\text{N})$.

Step 4. Calculate the following forces:

externally applied force—$F_{\text{ex}} = F_{\text{motor}} - F_{f\_\text{motor}}(F_{\text{motor}}) - F_{m\_\text{motor}}(v)$,

spring force—$F_{\text{sp}} = k_{\text{spring}}(x_i - x_d)$,

internally applied force—$F_{\text{in}} = F_{\text{ex}} + F_{\text{sp}}$,

stick friction force—if $(F_{\text{in}} > F_S)$ then $(F_{\text{stick}} = F_S)$, if $(F_{\text{in}} < -F_S)$ then $(F_{\text{stick}} = -F_S)$, if $(-F_S \leq F_{\text{in}} \leq F_S)$ then $(F_{\text{stick}} = F_{\text{in}})$, and slip friction force—$F_{\text{slip}}(v) = (F_C + (F_S - F_C) \cdot e^{-(v/vss)}) \cdot \text{sgn}(v) + F_v \cdot v$.

Step 5. Calculate velocity and displacement of the mass:
$P = \int [F_{\text{in}} - F_f(v, F_{\text{in}})]dt$.
If $(-dp < P < dp)$ then $(v = 0)$ otherwise $(v = P/m)$.
$x_d = \int v dt$.

Step 6. Calculate the objective function:

$$J = \sqrt{\frac{\sum_{i=1}^{n} (x_{\text{test}} - x_d)^2}{n}}. \tag{4.5}$$

Step 7. Return to main search.

4.4. Algorithm Implementation

Regarding this identification problem, the mass (m) is known, $m = 10.90$ kg. There are 5 parameters to be searched, that is, F_S, F_C, F_V, vss, and k_{spring}, respectively. The termination criterion is either $J < 4.5$ or $count_{max} = 1,000$. The procedural list below presents the implemented algorithm for this identification problem.

Step 1. Initialization: search parameters: {$p = 5$, $S = 30$, $N_c = 20$, $N_S = 4$, $\alpha = 1 \times 10^3$, $d_{attract} = 0.1$, $w_{attract} = 0.2$, $h_{repellant} = 0.1$, $w_{repellant} = 0.1$, $R = 0.15$, $N = 40$, $count_{max} = 1000$, BT = 5 and *n_re_back* = 5}, search spaces: {F_S = [100–180], F_C = [40–80], F_V = [0.4–1.0], vss = [1–7], and k_{spring} = [0.1–1.5]}.

Step 2. Randomly assign real values to the parameters to be searched for S sets. Calculate the corresponding objective functions. Select the solution set having the best objective function, and store it in the variable θ^i.

Step 3. Calculate the objective functions, $J(i, j)$, with J_{cc} taken into account according to (2.1). Assign $J_{last} = J(i, j)$.

Step 4. Random the value of $\Delta(i)$ in $[-1, 1]$. Use (2.2) to calculate $C(i, j)$. Calculate the next parameters, $\theta^i(j + 1)$, according to (2.3). Calculate the objective functions $J(i, j + 1)$.

Step 5. Evaluate the objective functions: if $(J(i, j+1) \leq J_{last})$ then $(J_{last} = J(i, j+1))$, otherwise J_{last} remains unchanged. Update $J(i, j + 1)$ by using $\Delta(i)$ until iteration count = N_S.

Step 6. Repeat Steps 3 to 5 for N_c times.

Step 7. Evaluate the objective functions, J. Assign *best_θ* = minimum value of J just found. Assign the current best solutions as the initial solutions, S_0, and their corresponding objective functions as the initial J_0 values.

Step 8. In the neighborhood of S_0 with the search radius R, create randomly N sets of solutions and store them in the set $S_1(r)$. Calculate the objective functions for all solutions according to the procedures described in Section 4.3.

Step 9. Based on the objective functions, do minimum sorting for the solutions in $S_1(r)$. Assign *best_neighbor1* = solutions with minimum objective functions and their objective functions = J_1.

Step 10. If $(J_1 < J_0)$ then (store S_0 in the 2nd–6th columns of TL, and store J_0 in the 7th column of TL), otherwise (store S_1 and J_1 in the TL).

Step 11 (backtracking mechanism). If the frequency of solution cycling occurrence is equal to BT, do minimum sorting for the previous solutions stored in the TL, retrieve the 5th backward solution set, and assign it as the initial solution set for the next search move.

Step 12. If ($J < 2$ or count = $count_{max}$) then (terminate the search, exit and render the best solutions).

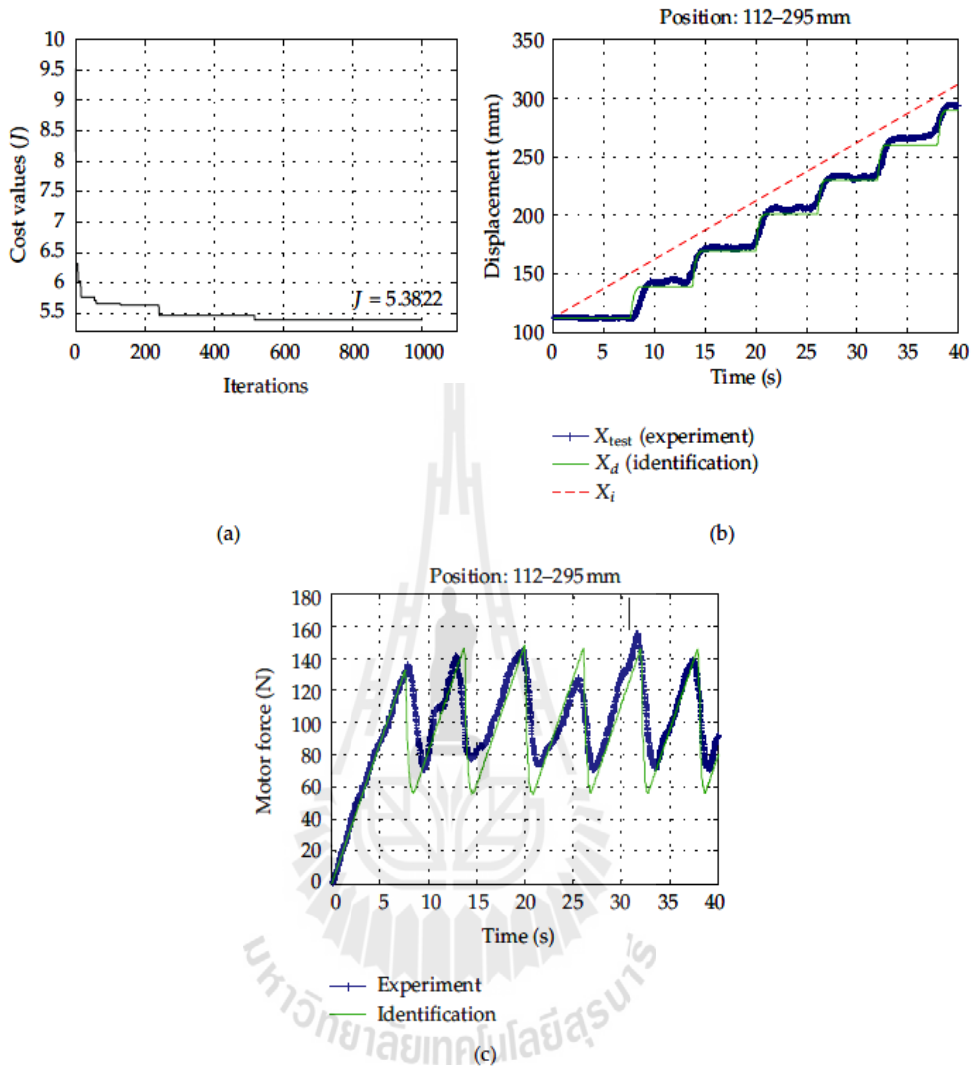


Figure 5: Identification results of ramp-up command at 5 mm/s—(a) convergence curve, (b) displacement, and (c) force exerted by motor. (Note: positions in the range of 112–295 mm).

Step 13 (Adaptive search radius mechanism). If (*best_error* < 15) then ($R = 0.0375$).
If (*best_error* < 8) then ($R = 0.0095$).
If (*best_error* < 3) then ($R = 0.0025$).

Step 14. Go to Step 8 until computing expires.

4.5. Results and Discussions

4.5.1. Identification

Due to the strong nonlinearity in friction force, it is necessary to identify two sets of model parameters corresponding to rightward and leftward motions. Referring to Figure 5, the illustrated graphs correspond to the rightward motion, that is, ramp-up command of 5 mm/s,

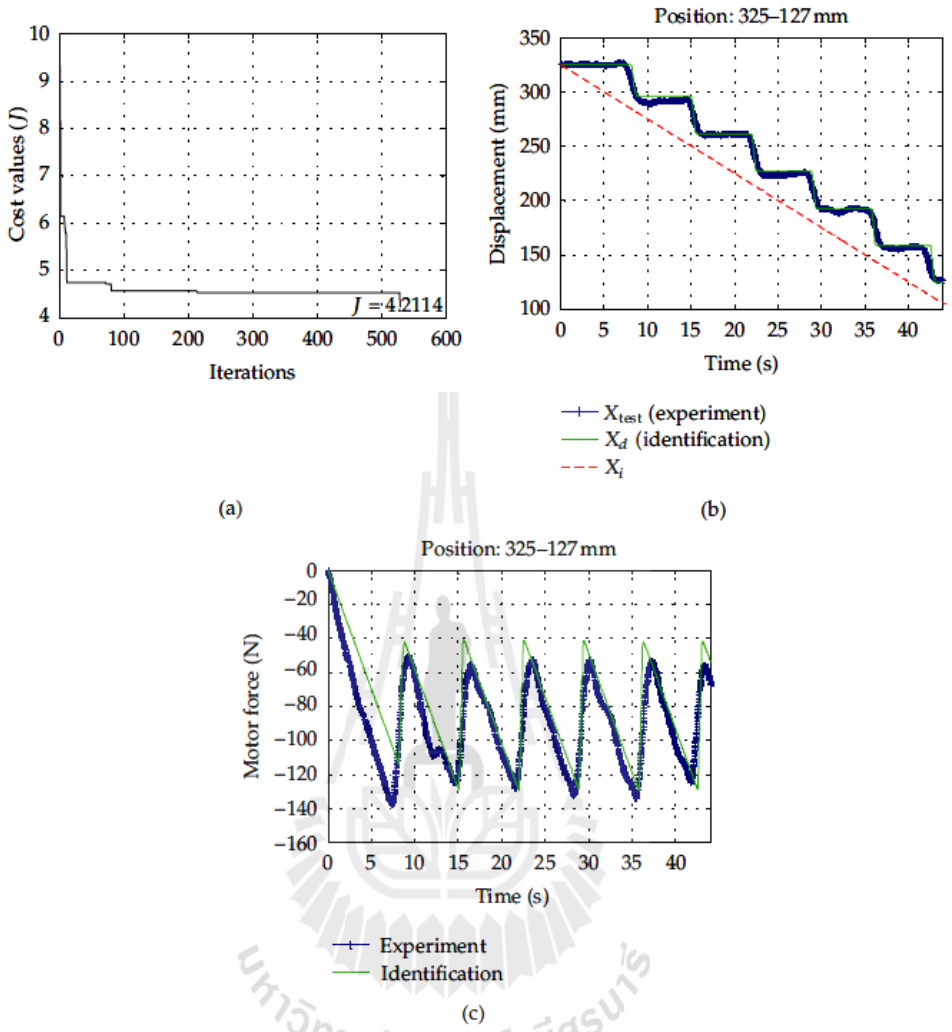


Figure 6: Identification results of ramp-down command at −5 mm/s—(a) convergence curve, (b) displacement, and (c) force exerted by motor. (Note: positions in the range of 325–127 mm).

that uses the data in the range of 112–295 mm for identification. The convergence curve in Figure 5(a) indicates that the search terminated by $\text{count}_{\max} = 1000$. The cost of the returned results is $J = 5.3822$. The obtained parameters are as follows: $\{F_S = 144.5463\,\text{N}$, $F_C = 47.8514\,\text{N}$, $F_V = 0.9583\,\text{Ns/mm}$, $vss = 1.0964\,\text{mm/s}$, $k_{\text{spring}} = 0.96798\,\text{N/mm}\}$. The experimental data and the model plots for displacement and force exerted by motor are illustrated in Figures 5(b) and 5(c), respectively. Good agreement between the experiment and the model can be observed.

For the leftward motion, that is, ramp-down command of −5 mm/s, the graphical displays of identification results are shown in Figure 6. The data used for identification are in the range of 325–127 mm. As indicated by the convergence curve in Figure 6(a), the search terminated at the 528th iteration and returned the solutions with the cost $J = 4.2114$. The obtained parameters are as follows: $\{F_S = -152.2804\,\text{N}$, $F_C = -40.4153\,\text{N}$, $F_V = -0.9757\,\text{Ns/mm}$, $vss = -3.21475\,\text{mm/s}$, $k_{\text{spring}} = 0.55384\,\text{N/mm}\}$. The experimental data and the model plots in Figures 6(b) and 6(c) show a good agreement.

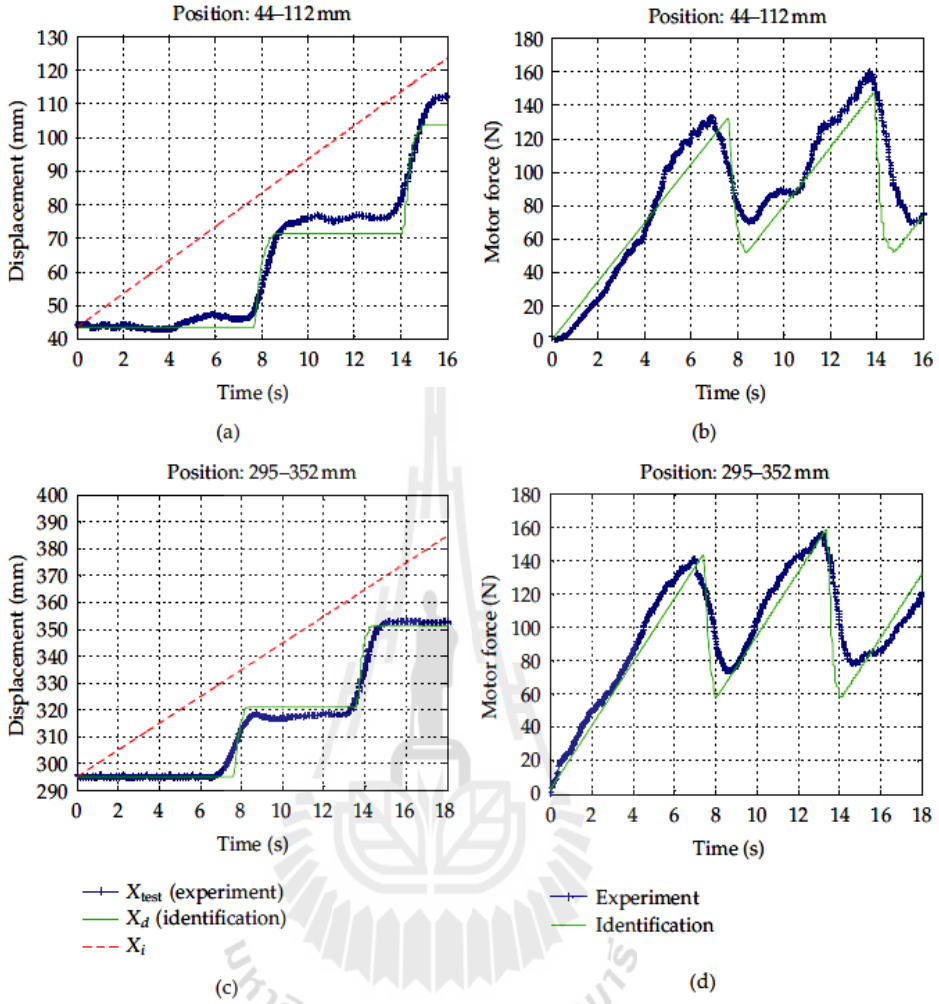


Figure 7: Validation results of ramp-up command—(a) displacement (44–112 mm), (b) force exerted by motor (44–112 mm), (c) displacement (295–352 mm), and (d) force exerted by motor (295–352 mm).

4.5.2. Validation

Model validation was conducted for both directions of motion. Figure 7 illustrates the experimental data and the model plots for the rightward direction covering two ranges, that is, 44–112 mm and 295–352 mm. Figures 7(a) and 7(b) display the plots of the displacement and the force exerted by motor for 44–112 mm range. Similarly, the results for 295–352 mm range are shown in Figures 7(c) and 7(d). For the leftward direction covering 352–325 mm and 127–68 mm ranges, similar graphical displays are illustrated in Figures 8(a) and 8(b). Very good agreement among the practical and the theoretical results can be observed.

Furthermore, the friction curves based on model plots are shown against the experimental data in Figure 9. Very good agreement between the two can be observed. Therefore, the identified models are very good representations of the nonlinear friction forces.

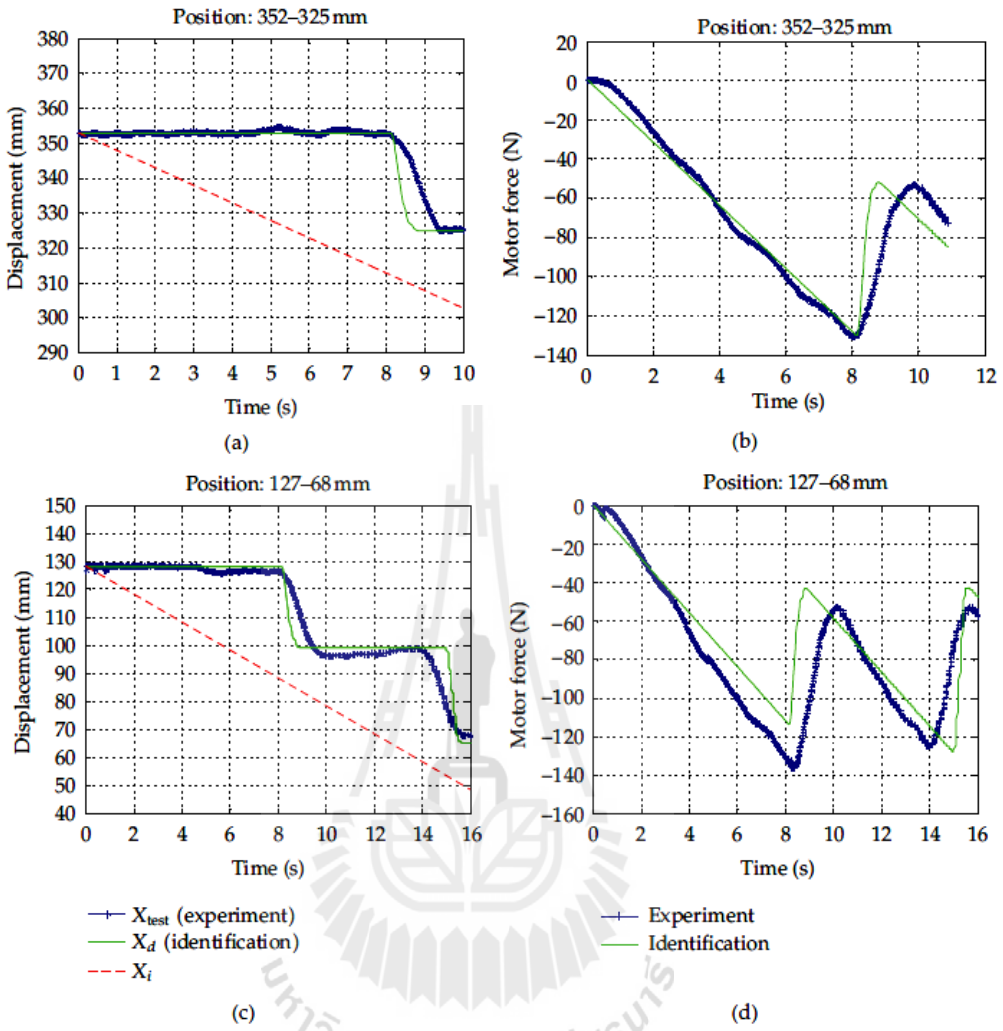


Figure 8: Validation results of ramp-down command—(a) displacement (352–325-mm), (b) force exerted by motor (352–325 mm), (c) displacement (127–68 mm), and (d) force exerted by motor (127–68 mm).

5. Conclusions

This paper has proposed new metaheuristics denoted as bacterial foraging-tabu search (BTS), which are formed from the adaptive bacterial foraging optimization algorithm (ABFO) and the adaptive tabu search (ATS). The paper has elaborated the search performance assessment among the ABFO, ATS, GA, and BTS. The proposed BTS algorithms provide superior search performances as the presentation appears in Section 3. The algorithms have been applied to identify 5 parameters of the Stribeck friction model. An experimental bed of a closed-loop position control of a linear slide bed was constructed at the laboratory. The system setup is described in Section 4. Several test runs of ramp command following control were conducted for the slide bed to follow ±5 mm/s commands such that the slide bed pronouncedly exhibited stick-slip. The experimental data were split into 2 groups for identification and validation purposes. Section 4 also elaborates important issues of objective function and algorithm implementations as well as identification results. As a result of model validation, very satisfactory model parameters have been identified by the proposed metaheuristics.

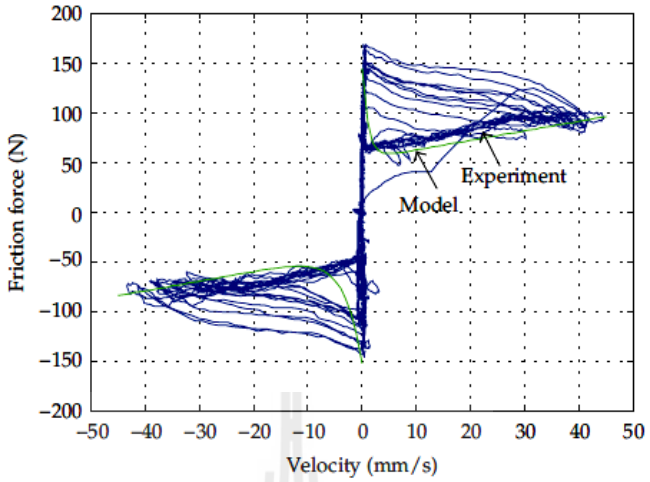


Figure 9: Plots of friction force curves (ramp command of ±5 mm/s).

Nomenclature

(A) Algorithms

$\text{count}_{\max}$:	Maximum iteration
d_{attract}:	Coefficient representing the depth of attractant released
$h_{\text{repellant}}$:	Coefficient representing the height of the repellant effect
n_re_back:	*k*th backtracking solution retrieved from the TL
p:	Number of parameters to be optimized
w_{attract}:	Coefficient representing the width of the attractant signal
$w_{\text{repellant}}$:	Coefficient representing the width of the repellant by the cell
AR:	Adaptive radius
BT:	Frequency of solution cycling
$C(i, j)$:	Step size taken in random direction specified by the tumble
$J(i, j)$:	Cost value of ith bacterium
N:	Number of the neighbourhood
N_c:	Number of iterations to be carried out in a chemotactic loop
N_{ed}:	Maximum number of elimination and dispersal events
N_{re}:	Number of reproduction loop
N_S:	Swimming length after which tumbling of bacteria in a chemotactic loop
P_{ed}:	Probability with which the elimination and dispersal continues
R:	Search radius
S:	Number of bacteria in the population
S_r:	A half of number of bacteria ($S/2$)
TL:	Tabu list
α:	A positive constant
$\Delta(i)$:	Random vector on $[-1, 1]$
θ^i:	Position of ith bacterium.

(B) Test Functions

BF: Bohachevsky function
RF: Rastrigin function
SchF: Schwefel function
ShuF: Shubert function
SF: Shekel's fox-holes function.

(C) Identification Problem

a: Gear ratio = 5.9
i: Motor current (A)
k_{spring}: Gravity constant (N/mm)
l: Ball screw lead = 5 mm
m: Mass (kg)
n: Number of data
v: Velocity (mm/s)
vss: Crossover velocity (mm/s)
x_t: Displacement of spring (mm)
x_d: Displacement of mass (mm)
D_m: Viscous friction coefficient = 2.60×10^{-6} Nm/rad/sec
F_{ex}: External input force (N)
F_f: Friction force (N)
F_{f_motor}: Friction force of motor (N)
F_{in}: Internal input force (N)
F_{m_motor}: Force equivalent to the inertia of motor (Nm)
J_m: Inertia of motor = 4.17×10^{-6} Kg·m^2
K_{kt}: Inertia to force conversion factor = $K_n \times K_v = (2\pi a/l)^2 \eta_b \eta_c$
K_n: Linear to angular velocity conversion factor = $2\pi a/l$ (rad/m)
K_t: Torque constant of motor = 18.2×10^{-3} N/A
K_v: Torque to ball screw force conversion factor = $2\pi \eta_b \eta_c a/l$
F_C: Coulomb friction (N)
F_S: Static friction (N)
F_V: Viscous friction (Ns/mm)
P: Moment (Nm)
X_{test}: Displacement from measured (mm)
$\pm dv$: Velocity band around zero velocity = ± 0.1 mm/s
$\pm dp$: Notation for the term $\pm dv \times m$
η_b: Gear box efficiency = 0.81
η_c: Ball screw efficiency = 0.925
$\widehat{k}$: Proportional controller gain.

Acknowledgments

The authors are thankful to the Royal Golden Jubilee Ph.D. Program under Grant PHD/0091/2551 for the research grants as well as some partial fundings available from Suranaree University of Technology.

References

[1] T. Watanabe, Y. Hashimoto, I. Nishikawa, and H. Tokumaru, "Line balancing using a genetic evolution model," *Control Engineering Practice*, vol. 3, no. 1, pp. 69–76, 1995.

[2] C. Onnen, R. Babuška, U. Kaymak, J. M. Sousa, H. B. Verbruggen, and R. Isermann, "Genetic algorithms for optimization in predictive control," *Control Engineering Practice*, vol. 5, no. 10, pp. 1363–1372, 1997.

[3] E. W. McGookin and D. J. Murray-Smith, "Submarine manoeuvring controller's optimisation using simulated annealing and genetic algorithms," *Control Engineering Practice*, vol. 14, no. 1, pp. 1–15, 2006.

[4] M. Marinaki, Y. Marinakis, and G. E. Stavroulakis, "Fuzzy control optimized by PSO for vibration suppression of beams," *Control Engineering Practice*, vol. 18, no. 6, pp. 618–629, 2010.

[5] J. G. Gray, D. J. Murray-Smith, Y. Li, K. C. Sharman, and T. Weinbrenner, "Nonlinear model structure identification using genetic programming," *Control Engineering Practice*, vol. 6, no. 11, pp. 1341–1352, 1998.

[6] B. Abdelhadi, A. Benoudjit, and N. Nait-Said, "Application of genetic algorithm with a novel adaptive scheme for the identification of induction machine parameters," *IEEE Transactions on Energy Conversion*, vol. 20, no. 2, pp. 284–291, 2005.

[7] F. Alonge, F. D'Ippolito, and F. M. Raimondi, "Least squares and genetic algorithms for parameter identification of induction motors," *Control Engineering Practice*, vol. 9, no. 6, pp. 647–657, 2001.

[8] C. Zheng and P. Wang, "Parameter structure identification using tabu search and simulated annealing," *Advances in Water Resources*, vol. 19, no. 4, pp. 215–224, 1996.

[9] T. Kulworawanichpong, K.-L Areerak, K.-N Areerak, and S. Sujitjorn, "Harmonic identification for active power filters via adaptive tabu search method," *Lecture Notes in Computer Science*, vol. 3215, part 3, pp. 687–694, 2004.

[10] Y. Liu and X. He, "Modeling identification of power plant thermal process based on PSO algorithm," in *Proceedings of the American Control Conference (ACC 05)*, pp. 4484–4489, Portland, Ore, USA, June 2005.

[11] J. Meier, W. Schaedler, L. Borgatti, A. Corsini, and T. Schanz, "Inverse parameter identification technique using PSO algorithm applied to geotechnical modeling," *Journal of Artificial Evolution and Applications*, vol. 2008, Article ID 574613, 14 pages, 2008.

[12] V. Khanagha, A. Khanagha, and V. T. Vakili, "Modified particle swarm optimization for blind deconvolution and identification of multi channel FIR Filters," *Eurasip Journal on Advances in Signal Processing*, vol. 2008, Article ID 280635, 6 pages, 2010.

[13] X. He and J. J. Liu, "Aquifer parameter identification with ant colony optimization algorithm," in *Proceedings of the International Workshop on Intelligent Systems and Applications (ISA '09)*, pp. 1–4, May 2009.

[14] L. Liu, M. Fukumoto, S. Saiki, and S. Zhang, "A variable step-size proportionate affine projection algorithm for identification of sparse impulse response," *Eurasip Journal on Advances in Signal Processing*, vol. 2009, Article ID 150914, 10 pages, 2009.

[15] H. Chen, Y. Zhu, and K. Hu, "Multi-colony bacteria foraging optimization with cell-to-cell communication for RFID network planning," *Applied Soft Computing Journal*, vol. 10, no. 2, pp. 539–547, 2010.

[16] F. Glover, "Tabu search—part I," *ORSA Journal on Computing*, vol. 1, pp. 190–206, 1989.

[17] F. Glover, "Tabu search—part II," *ORSA Journal on Computing*, vol. 2, pp. 4–32, 1990.

[18] G. Zhang, W. Habenicht, and W. E. L. Spieß, "Improving the structure of deep frozen and chilled food chain with tabu search procedure," *Journal of Food Engineering*, vol. 60, no. 1, pp. 67–79, 2003.

[19] T. Kulworawanichpong and S. Sujitjorn, "Optimal power flow using tabu search," *IEEE Power Engineering Review*, vol. 22, no. 6, pp. 37–40, 2002.

[20] E. Nowicki and C. Smutnicki, "A fast tabu search algorithm for the permutation flow-shop problem," *European Journal of Operational Research*, vol. 91, no. 1, pp. 160–175, 1996.

[21] R. Battiti and G. Tecchiolli, "The reactive tabu search," *ORSA Journal Computing*, vol. 6, no. 2, pp. 126–140, 1994.

[22] Y. A. Kochetov and E. N. Goncharov, "Probabilistic tabu search algorithm for the multi-stage uncapacitated facility location problem," in *Operations Research Proceedings*, pp. 65–70, Springer, 2001.

[23] S. Sujitjorn, T. Kulworawanichpong, D. Puangdownreong, and K.-N. Areerak, "Adaptive tabu search and applications in engineering design," in *Integrated Intelligent Systems for Engineering Design*, X. F. Zha and R. J. Howlett, Eds., pp. 233–257, IOS Press, Amsterdam, The Netherlands, 2006.

[24] S. Sujitjorn and S. Khwan-on, "Learning control via neuro-tabu-fuzzy controller," *Lecture Notes in Computer Science*, vol. 4251, pp. 833–840, 2006.

[25] N. Sriyingyong and K. Attakitmongcol, "Wavelet-based audio watermarking using adaptive tabu search," in *Proceedings of the 1st International Symposium on Wireless Pervasive Computing*, pp. 1–5, January 2006.

[26] K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *IEEE Control Systems Magazine*, vol. 22, no. 3, pp. 52–67, 2002.

[27] Y. Liu and K. M. Passino, "Biomimicry of social foraging bacteria for distributed optimization: models, principles, and emergent behaviors," *Journal of Optimization Theory and Applications*, vol. 115, no. 3, pp. 603–628, 2002.

[28] S. Mishra, "A hybrid least square-fuzzy bacterial foraging strategy for harmonic estimation," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 1, pp. 61–73, 2005.

[29] S. Mishra and C. N. Bhende, "Bacterial foraging technique-based optimized active power filter for load compensation," *IEEE Transactions on Power Delivery*, vol. 22, no. 1, pp. 457–465, 2007.

[30] M. Tripathy, S. Mishra, L. L. Lai, and Q. P. Zhang, "Transmission loss reduction based on FACTS and bacteria foraging algorithm," *Lecture Notes in Computer Science*, vol. 4193, pp. 222–231, 2006.

[31] W. J. Tang, M. S. Li, Q. H. Wu, and J. R. Saunders, "Bacterial foraging algorithm for optimal power flow in dynamic environments," *IEEE Transactions on Circuits and Systems*, vol. 55, no. 8, pp. 2433–2442, 2008.

[32] T. Datta, I. S. Misra, B. B. Mangaraj, and S. Imtiaj, "Improved adaptive bacteria foraging algorithm in optimization of antenna array for faster convergence," *Progress in Electromagnetics Research*, vol. 1, pp. 143–157, 2008.

[33] R. Majhi, G. Panda, B. Majhi, and G. Sahoo, "Efficient prediction of stock market indices using adaptive bacterial foraging optimization (ABFO) and BFO based techniques," *Expert Systems with Applications*, vol. 36, no. 6, pp. 10097–10104, 2009.

[34] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: overview and conceptual comparison," *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308, 2003.

[35] E. Alba, *Parallel Metaheuristics*, Wiley-Interscience, New Jersey, NJ, USA, 2005.

[36] E. G. Talbi, *Metaheuristics*, John Wiley & Sons, New Jersey, NJ, USA, 2009.

[37] J. H. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, 1975.

[38] J. H. Holland, "Genetic algorithms," *Scientific American*, vol. 267, no. 1, pp. 66–72, 1992.

[39] MathWorks, "*Genetic Algorithm and Direct Search Toolbox: for Use with MATLAB*," User's Guide, Version 1, MathWorks, Natick, Mass, USA, 2005.

[40] B. Armstrong-Hélouvry, "Stick slip and control in low-speed motion," *Institute of Electrical and Electronics Engineers. Transactions on Automatic Control*, vol. 38, no. 10, pp. 1483–1496, 1993.

[41] B. Armstrong-Helouvry, P. Dupont, and C. Cadudas de Wit, "A survey of model, analysis tools and compensation methods for the control of machines with friction," *Automatica*, vol. 30, no. 7, pp. 1083–1138, 1994.

[42] C. Canudas de Wit, H. Olsson, K. J. Åström, and P. Lischinsky, "A new model for control of systems with friction," *Institute of Electrical and Electronics Engineers. Transactions on Automatic Control*, vol. 40, no. 3, pp. 419–425, 1995.

[43] H. Du and S. S. Nair, "Modeling and compensation of low-velocity friction with bounds," *IEEE Transactions on Control Systems Technology*, vol. 7, no. 1, pp. 110–121, 1999.

BIOGRAPHY

Miss Nuapett Sarasiri was born on September 17, 1984 in Muang District, Chachoengsao Province. She attended high school at Datdaruni School in Chachoengsao, and finished her high school education in 2003. She received her Bachelor and Master of Engineering degrees in Electrical Engineering from Suranaree University of Technology, Nakhon Ratchasima Province, in 2007 and 2009, respectively. After graduating, she continued to study for a Doctoral degree at the School of Electrical Engineering, Institute of Engineering, Suranaree University of Technology. During her graduate work, she was awarded a Royal Golden Jubilee (RGJ) PhD Program Scholarship from the Thailand Research Fund (TRF) in 2009. In September 2011, she was a Visiting Research Scholar in the Power Electronics, Machines and Control (PEMC) Research Group in the Department of Electrical and Electronic Engineering at the University of Nottingham, UK, with Professor Dr. Pericle Zanchetta. Her research interests include power systems, power electronics, control, and search algorithms.