A HYBRID GENETIC-ARTIFICIAL FISH SWARM ALGORITHM FOR ECONOMIC LOAD DISPATCH WITH VALVE-POINT AND MULTIPLE FUEL EFFECTS

By

MUSA ALIYU YAKUBU

DEPARTMENT OF ELECTRICAL ENGINEERING

FACULTY OF ENGINEERING

AHMADU BELLO UNIVERSITY ZARIA,

NIGERIA

AUGUST, 2018

A HYBRID GENETIC-ARTIFICIAL FISH SWARM ALGORITHM FOR ECONOMIC LOAD DISPATCH WITH VALVE-POINT AND MULTIPLE FUEL EFFECTS

By

Musa Aliyu YAKUBU, B.Sc., (FUT MINNA), 1998

P15EGEE8045

yalimus2004@yahoo.com

A DISSERTATION SUBMITTED TO THE SCHOOL OF POSTGRADUATE STUDIES,

AHMADU BELLO UNIVERSITY, ZARIA

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF A

MASTER OF SCIENCE (M.Sc.) DEGREE IN ELECTRICAL ENGINEERING

DEPARTMENT OF ELECTRICAL ENGINEERING

FACULTY OF ENGINEERING

AHMADU BELLO UNIVERSITY, ZARIA

NIGERIA

AUGUST, 2018

DECLARATION

I declare that the work in this Dissertation entitled "A Hybrid Genetic-Artificial Fish Swarm Algorithm for Economic Load Dispatch with Valve- Point Loading and Multiple Fuel Effects" has been carried out by me in the Department of Electrical Engineering as part of the requirements for the award of degree of Master of Science (M.Sc.) degree in Electrical Engineering. The information derived from the literature has been duly acknowledged in the text and a list of references provided. No part of this dissertation was previously presented for another degree or diploma at this or any other Institution.

Musa Aliyu YAKUBU _____ _____

(Student) Signature Date

CERTIFICATION

This dissertation entitled **"A HYBRID GENETIC-ARTIFICIAL FISH SWARM ALGORITHM FOR ECONOMIC LOAD DISPATCH WITH VALVE-POINT LOADING AND MULTIPLE FUEL EFFECTS"** by Musa Aliyu YAKUBU meets the regulations governing the award of Master of Science (M.Sc.) Degree in Electrical Engineering of the Ahmadu Bello University, and is approved for its contribution to knowledge and literary presentation.

Prof. Boyi Jimoh	_____	_____
(Chairman, Supervisory Committee)	Signature	Date
Dr. Y. Jibril	_____	_____
(Member, Supervisory Committee)	Signature	Date
Dr. Y. Jibril	_____	_____
(Head of Department)	Signature	Date
Prof. A.Z. Abubakar	_____	_____
(Dean, School of Postgraduate Studies)	Signature	Date

DEDICATION

I dedicate this project to Almighty ALLAH (SWT) for his faithfulness to me throughout the entire period of this research and at all times.

ACKNOWLEDGEMENT

My first gratitude goes to Almighty ALLAH (SWT) for guiding me through the research. My thanks go to my supervisors, Prof. Boyi Jimoh and Dr. Yusuf Jibril for the supervision and advisory role they played during the course of the research. I would like to also appreciate Prof. U.O. Aliyu, Prof. M.B. Mu'azu, Dr. S.M. Sani and my able colleague and friend Dr. Abubakar S. Hassan for his support and timely response to my questions.

I would like to also recognize my parents Hajia Ashawu Yakubu and Alh. Yakubu Aliyu of blessed memories, may ALLAH continue to shower HIS blessings upon them. To my friends Dr. Nuhu Suleiman, Engr. Adah I. Friday, John Z. Mullah, Sanusi Ibrahim, Hussein Abdullahi, Yabo M. Mavira and others too numerous to mention I say a very big thank you.

To my beloved wife, Mariam O. Musa whom despite the challenges of time was patient and steadfast in prayers for me, you are dearest to me, and to my amiable and awesome children, Fadila, Fauziya, Ashraf and Anas, may ALLAH continue to protect, guide and shower His blessings upon you all.

Finally, to those who offered useful suggestions and advice throughout the entire period of this research work, I say thank you and may ALLAH reward you abundantly.

ABSTRACT

The restructuring of the electrical power industry has given rise to a high degree of vibrancy and competitive market, which changed many features of the power industry. Energy resources become scarce, the cost of power generation increases, environmental concerns are raised, and an ever-increasing demand for electrical energy characterizes this now-altered scenario. In this perspective, Economic Load Dispatch (ELD) is necessitated. Strong heuristic techniques can go a long way in determining the optimum solution to such technical problems having large number of possible solutions. In the proposed research work, two heuristic algorithms namely: Genetic Algorithm (GA) and Artificial Fish Swarm Algorithm (AFSA) are hybridized to yield a more robust technique called "Hybrid Genetic-Artificial Fish Swarm Algorithm", (HGAFSA) that is suitable for solving complex ELD problems. The technique is then applied to solve a multi-objective ELD problem involving higher order cost functions that includes the effects of valve-point loading and multiple fuel cost function. The proposed approach was validated using five standard IEEE test systems for 13, 40, 110, 140, and 160 generating unit systems. Testing of the developed HGAFSA based ELD algorithm (HGAFSAELDA) yielded reduction in fuel cost by 1.53%, 0.03%, 0.07%, 0.00012% and 1.37% for the 13, 40, 110, 140 and 160 generating units respectively. An annual savings in fuel cost of $3.254e+06, $3.8235e+05, $2135.7, $9.5563e+06, and $1.1588e+06 for the 13, 40, 110, 140, and 160-generating-units respectively were achieved over the existing best costs presented in (Pradhan *et al*., 2017). HGAFSA based optimization curves and the Cumulative Power Generation curves are also presented to demonstrate how the inequality constraints are satisfied by each of the generating units.

TABLE OF CONTENTS

DECLARATION i

CERTIFICATION ii

DEDICATION iii

ACKNOWLEDGEMENT iv

ABSTRACT v

TABLE OF CONTENTS vi

LIST OF FIGURES x

LIST OF ABBREVIATIONS xii

NOMENCLATURE xiv

CHAPTER ONE: GENERAL INTRODUCTION 1

1.1 Background of Research 1

1.2 Problem Statement 5

1.3 Aim and Objectives 6

1.4 Scope of Work and Limitation 6

1.5 Dissertation Organization Outline 7

CHAPTER TWO: LITERATURE REVIEW 8

2.1 Introduction 8

2.2 Review of Fundamental Concepts 8

2.2.1 Thermal power plant 8

2.2.2 Generator operating cost 9

2.2.3 Fuel efficiency 10

2.2.4 Incremental *cost (*IC*)* 10

2.2.5 Economic Load Dispatch (ELD) formulation 11

2.2.6 The cost function 11

2.2.7 Solution by Lagrange method 14

2.2.8 Dynamic programming 16

2.2.9 Quadratic programming method 16

2.2.10 Non-linear programming 17

2.2.11 Newton's method 18

2.2.12 Heuristic approaches 18

2.2.12.1 Particle swarm optimization 19

2.2.12.2 Artificial fish swarm algorithm 19

2.2.12.3 Genetic algorithm 25

2.2.12.4 Hybrid algorithms 32

2.3 Review of Similar Works 33

CHAPTER THREE: METHODOLOGY 42

3.1 Introduction 42

3.2 Formulation of the Proposed HGAFSA 42

3.3 Decoder Function 43

3.4 Encoder Function 44

3.5 Population Update 46

3.6 Model of the ELD Problem 50

3.7 The Proposed HGAFSA based Higher Order ELD Algorithm 54

CHAPTER FOUR: RESULTS AND DISCUSSION 56

4.1 Introduction 56

4.2 Simulation Setup 56

4.3Test system 1 57

4.4Test system 2 60

4.5 Test system 3 62

4.6 Test system 4 65

4.7 Test system 5 68

CHAPTER FIVE: CONCLUSION AND RECOMMENDATION 73

Introduction 73

5.1 Conclusion 73

5.2 Significant Contributions 74

5.3 Recommendations 74

5.4 Limitation 75

REFERENCES 76

APPENDICES 85

Appendix A1: Table 4a: 13-Units Economic Load Dispatch Data ……85

Appendix A2: Table 4b: 40-Units Economic Load Dispatch Data ……....86

Appendix A3: Table 4c: 110-Units Economic Load Dispatch Data …….....87

Appendix A4: Table 4d: 140-Units Economic Load Dispatch Data …….90

Appendix A5: Table 4e: 160-Units Economic Load Dispatch Data …….93

Appendix B: M-file MATLAB codes for the proposed model …….96

Appendix B1: M-file Hybrid Genetic Artificial Fish Swarm Algorithm …….....96

Appendix B2: M-file for ELD encoder algorithm ……..110

Appendix B3: M-file for ELD objective function ……..113

Appendix B4: M-file for GA and AFSA parameters …….....114

Appendix B5: M-file for GA reproduction ……....117

Appendix B6: M-file for GA mutation ……..118

Appendix B7: M-file for GA crossover……..119

Appendix B8: M-file for GA code …….120

Appendix B9: M-file for AFSA preying ……...121

Appendix B10: M-file for AFSA swarming …….....123

Appendix B11: M-file for AFSA chasing ……126

LIST OF FIGURES

Figure 2.1: Energy conversion in a thermal power plant (Vanita & Thanushkodi, 2011). 9

Figure 2.2: Heat rate curve (Vlachos, 2011). 10

Figure 2.3: Incremental Fuel Cost Curve of a Typical Generating Unit (Vanita & Thanushkodi, 2011) 13

Figure 2.4: The visual and step of artificial fish (Azad *et al*., 2014) 20

Figure 2.5: Flowchart of Artificial Fish Swarm Algorithm (Fang *et al*., 2014) 25

Figure 2.6: Flow chart of Genetic Algorithm (Dhebar & Deb, 2017) ……. 32

Figure 3.1: Illustration of Chromosome-Fish Conversion and vice versa 45

Figure 3.2: Flow Chart of Proposed Hybrid GA AND AFSA (HGAFSA)……...50

Figure 3.3: Evaluating the Fitness of a Chromosome/Fish 52

Figure 3.4: HGAFSA based ELD Algorithm 55

Figure 4.1: 13-Units: (a) Optimization Curve (b) Cumulative Power Generated..... 59

Figure 4.2: 40-units: (a) Optimization Curve (b) Cumulative Power Generated 62

Figure 4.3: 110-Units: (a) Optimization Curve (b) Cumulative Power Generated..... 64

Figure 4.4: 140-Units: (a) Optimization Curve (b) Cumulative Power Generated..... 67

Figure 4.5: 160-Units: (a) Optimization Curve (b) Cumulative Power Generated..... 70

Figure 4.6: Annual Savings in Fuel Cost for Systems 1 to 5 72

LIST OF TABLES

Table 4.1: HGAFSA Simulation Parameter Settings..... 58

Table 4.2: Comparison of Results for 13-Units System……...60

Table 4.3: Comparison of Results for 40-Units System 62

Table 4.4: Comparison of Results for 110-Units System 63

Table 4.5: Comparison of Results for 140-Units System 66

Table 4.6: Comparison of Results for 160-Units System 69

Table 4.7: Summary of Results for the Comparison of HGAFSA and OGWO/OIWO 72

LIST OF ABBREVIATIONS

Acronyms	Definition
AFSA	Artificial Fish Swarm Algorithm
ANN	Artificial Neural Network
BBO	Biography Based Optimization
DSP	Digital Signal Processor
ELD	Economic Load Dispatch
EPUSPSO	Efficient Population Utilization Strategy for Particle Swarm Optimization
FA	Firefly Algorithm
FM	Frequency Modulation
GA	Genetic Algorithm
GAAPI	Genetic Algorithm -Ant Colony Algorithm for Continuous domains
GWO	Grey Wolf Optimization
HGAFSA	Hybrid Genetic-Artificial Fish Swarm Algorithm
HGAFSAELDA	Hybrid Genetic Artificial Fish Swarm Based Economic Load Dispatch Algorithm
ICA-PSO	Improved Coordinated Aggregation Based Particle Swarm Optimization
KHA	Krill Herd Algorithm
LIM	Lambda Iteration Method

LR	Lagragian Relaxation
MATLAB	Matrix Laboratory
MFCFE	Multiple Fuel Cost Function Effect
MVMOS	Mean Variance Mapping Optimization Strategy
MSSA	Modified Social Spider Algorithm
NLP	Non Linear Programming
NTA	Novel TANAN's Algorithm
OGWO	Oppositional Grey Wolf Optimization
OIWO	Oppositional Invasive Weed Optimization
PSO	Particle Swarm Optimization
QOTLBO	Quasi- Oppositional Teaching Learning Based Optimization
QP	Quadratic Programming
RCGA	Real Coded Genetic Algorithm
SDE	Shuffled Differential Evolution
SSA	Social Spider Algorithm
VPLE	Valve Point Loading Effect

NOMENCLATURE

Symbol	Meaning
Ω	Set of updated population
S	List of steps to be executed in chronological order
P	Combined population of GA and AFSA
K	Population counter
S	Step counter
P	No. of parameters
Q	Quantization level
q_{norm}	Quantization level normalization
N	No. of bits in chromosome (X)
N_{bits}	No. of bits per parameter
X	Chromosome (main parameter of decoder function)
χ_{min}	Lower boundary of desired decoded output
χ_{max}	Higher boundary of desired decoded output

CHAPTER ONE

GENERAL INTRODUCTION

1.1 BACKGROUND OF RESEARCH

The efficient and optimum economic operation and planning of electric power generation systems have always occupied a vital position in the electric power industry (Gargeya & Pabba, 2013). Economic load dispatch (ELD) is a process of allocating generation levels to dispersed generating power plants so that the system is fully supplied in the most economical way (Harpreet Kaur *et al*., 2015). It can also be defined as the operation of generation facilities to produce energy at the lowest cost to reliably serve consumers recognizing any operational limits of generation and transmission facilities. ELD is simply a technique used to schedule the outputs of available generating units for a particular time that minimizes the total production cost while satisfying equality and inequality constraints (Pothiya *et al*., 2008), Prior to 1973 and the oil embargo that caused the rapid increase in fuel prices, electric utilities in the United States spent about 20% of their total income on fuel for the production of electrical energy (Alsumait *et al*., 2010). An idea of magnitude of the amounts of money was under consideration, and could be obtained by considering the annual operating expenses of a large utility for buying fuel. Based on assumption proposed by Gargeya & Pabba, 2013, the following parameters for a moderately large power system:

i. Annual peak load= 10,000MW;
ii. Annual load factor= 60%;
iii. Average annual heat rate for converting fuel to electric energy= 10,550.56KJ/kWh;
iv. Average fuel cost= $3.00/1.055GJ, corresponding to oil price at $18/Bbl.

With these assumptions, the total annual fuel cost for the system is as follows:

I. Annual energy produced $=10^7$ MW * 8760h/year * 0.60 = 5.256 * 10^{10}kWh;

II. Annual fuel consumption= 10,550.56KJ/kWh * 5.256 * 10^{10} kWh = 55.45 * 10^{13}KJ;

III. Annual fuel cost = 55.45* 10^{13} * 3/1.055* 10^{-9} $/J = $1.5767million.

This cost represents a direct requirement for revenues for the average customer of the system of 3.15cents/kWh aimed at recovering the expense for fuel. A savings in the operation of the system of small percent represents a significant reduction in operating cost, as well as in the quantities of fuel consumed. It is not surprising that this area has warranted a great deal of attention from the engineers through the years.

However, periodic changes in basic fuel price levels serve to accentuate the problem and increase its economic significance. Inflation also causes problems in developing and presenting methods, techniques, and examples of economic operation of electric power generating systems (Al-Othman & El-Naggar, 2008).

Moreover, rapid growth in power system size and electrical power demand has resulted into a problem of reducing the operating cost while maintaining voltage security and thermal limits of transmission line branches (Alsumait et al., 2010). A large number of mathematical Optimization Technique such as: GA based ELD; PSO based ELD; Hybrid GA-PSO based ELD; Dynamic Programming based ELD; Evolutionary Programming based ELD; to mention but a few have been applied to solve ELD problems. In most general formulation, the ELD problem is modeled as a non-linear, non-convex, large scale, static optimization problem with both continuous and discrete control variables (Burns & Gibson, 1975).

Furthermore, the non-linear convex nature of ELD problems has led most researchers such as: (Burns & Gibson, 1975), (Pothiya et al., 2008), (Al-Othman & El-Naggar, 2008), and others to model ELD problem using purely quadratic functions in which the quadratic coefficients are defined at the beginning of the solution search process. Whereas, more realistic models have also been developed in some other research works which includes those of: (Alsumait et al., 2010), (Sinha & Chakrabarti, 2003), (Sun *et al*., 2014), (Mohammadi-Ivatloo & Rabiee, 2013), (Jubril & Komolafe, 2013); etc. These models incorporated the effect of valve point loading and multiple fuel cost functions into ELD problem formulation.

However, the proposed research work will try to address an ELD problem through the development of a more realistic model that will account for the following effects of Valve-Point Loading (VPL), Multiple Fuel Cost Function (MFCF), Ramp Rate (RR) and Prohibited Operating Zone (POZ).

These result into a multi-objective optimization problem that tends to minimize the cost of fueling the generating units during the operation. This kind of complex optimization problem requires the use of robust techniques to achieve a reliable solution. Although, this optimization problem can be solved to some extent, using the heuristic techniques earlier mentioned, but the effectiveness of the solution cannot be guaranteed in the case of large power system.

In order to proffer solution that can be reliable and more efficient, a hybridization of two conventional heuristic techniques (Genetic Algorithm and Artificial Fish Swarm Algorithm) is proposed for solving the complex optimization problem, which give rise to a technique called " Hybrid G-AFS Algorithm" or simply HGAFSA. The choice of GA and AFSA were based on the following reasons:

a. GA is a widely known heuristic technique that has a well-defined set of search equations that have been proven to be effective in solving problems such as: Optimal Location and Sizing of Distributed Generators and Capacitor Banks, (Moradi & Abedinie, 2010), (Atwa *et al*., 2010); Optimal Power Flow, (López-Lezama *et al*., 2012); Optimal Location of Tie and Sectionalizing Switches in Distribution System, (Rao *et al*., 2013); Optimal Network Expansion (Bernardon et al., 2014), A hybrid GA–PS–SQP method to solve power system valve-point economic dispatch problems (Alsumait et al., 2010); etc.
b. AFSA on the other hand is a relatively new heuristic technique that is made up of well refined and sophisticated solution-search equations and has gained large application in areas like: Controller Design, (Fang *et al*., 2014); Optimal PID Tuning, (Amir Ghoreishi *et al*., 2011); Objective Function Minimization/ Maximization, (Wei Guo *et al*., 2011) (Huang *et al*., 2006); etc.

1.2 Motivation/Justification

I derive my motivation from the fact that over the years researchers have tried to address the problem of economic load dispatch with the main focus of how to commit the online generating units economically in order to generate electricity at a minimum cost while addressing generator constraints. The modern power systems encounter numerous technical and economic difficulties under competitive deregulated environment. The generation companies' (GENCOs) aim is to produce electric power at minimum cost therefore; proper allocation of power generation of the existing units may lead to significant savings in cost. This could be achieved by incorporating multiple fuel option into the economic dispatch problems.

1.3 Problem Statement

The modern power system around the world has grown in complexity of interconnection and power demand. The focus has shifted towards the enhanced performance, increased customer focus, low cost, reliable and clean power. In this changed perspective, scarcity of energy resources, increasing power generation cost and environmental concern necessitates economic load dispatch (ELD). In reality power stations, neither are at equal distances from load nor have similar fuel cost functions. Hence for providing cheaper power, load has to be distributed among the various power stations in a way that will result in lowest cost of generation. Practical economic dispatch (ED) problems have highly nonlinear objective function with equality and inequality constraints. Conventional methods such as lambda iteration method, gradient method and non-conventional method such as the heuristic method earlier discussed have been applied to solve the Economic Load Dispatch (ELD) problem. However, these techniques may not give optimal solution because they require incremental fuel cost curves which are piecewise linear and monotonically increasing to find the global optimal solution. In the proposed research work, a hybridization of two heuristic techniques namely: Artificial Fish Swarm Algorithm (AFSA); and Binary Coded Genetic Algorithm (BCGA), will be carried out in order to form a more robust technique called "Hybrid Genetic-Artificial Fish Swarm Algorithm", (HGAFSA). The technique will then be applied to solve a non-linear ELD problem considering the effects of valve-point loading, multiple fuel cost functions, ramp rate and prohibited operating zone. The effectiveness of the proposed approach will be demonstrated using five standard IEEE test systems (13, 40, 110, 140, and 160 generating unit systems); and finally comparing the results with those presented in (Pradhan *et al*., 2017)

1.4 Aim and Objectives

The aim of the proposed research work is to develop a Hybrid Genetic-Artificial Fish Swarm Algorithm (HGAFSA) and use it in solving a non-linear ELD problem considering the effects of: valve-point loading and multiple fuel cost. In achieving this, the following objectives will be met:

1. To hybridize Genetic Algorithm (GA) with Artificial Fish Swarm Algorithm (AFSA) to form a more robust algorithm called HGAFSA.
2. To model a higher order ELD problem while considering the effects of valve-point loading, multiple fuel cost function, ramp rate and prohibited operating zone.
3. To solve the resulting ELD problem in (2) using the developed algorithm in (1) and demonstrate the effectiveness of the proposed approach using five standard test systems (13, 40, 110, 140, and 160 generating unit systems); and finally comparing the results with those presented in (Pradhan *et al*., 2017).

1.5 Scope of Work and Limitation

The following items are the step by step approach that will constitute the scope of the proposed research work:

1. For the ELD, valve point loading, multi fuel cost function, ramp rate and prohibited operating zone effects were considered while emissions were not considered; hence the impact of emissions to ELD objective function was not quantified.
2. GA and AFSA algorithms were formulated using matrix definition method to aid hybridization.

3. A simulation test framework is developed in MATLAB to demonstrate the effectiveness of the formulated multi-objective ELD problem using five standard IEEE test systems. Actual transmission network was not considered.
4. The power demand considered for the networks were based on the reference in Pradhan *et al*., (2017).
5. The proposed ELD uses Encoder and Decoder to serve as an interface between GA and AFSA.

1.6 Dissertation Organization Outline

This chapter describes ELD problem and a brief overview of its solution strategies. However, it forms the introductory chapter. The aim and objectives together with problem statement, methodology and significant contributions are also presented in this chapter. The rest of the chapters are organized as follows. Chapter two presents the literature review of the fundamental concepts and similar works regarding ELD problem formulation and solution approaches. Chapter three presents the methods and materials used for this research work. Chapter four presents the simulation setup, results and analysis. Finally Chapter five presents the conclusion, recommendations and limitations. Quoted references and appendices are also provided at the end of the dissertation.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

In carrying out the research work, some literatures were reviewed, which served as a guide towards achieving the set goals. The review of these relevant literatures is categorized into two parts namely: review of fundamental concepts, and review of similar works, which are further discussed as follows.

2.2 Review of Fundamental Concepts

Some of the fundamental concepts regarding the proposed research work are discussed as follows.

2.2.1 Thermal power plant

A thermal power plant is a power plant in which its prime mover is driven by steam. Water is the working fluid. It is heated at the boiler and circulated with energy to be expanded at the steam turbine. To give work to the rotor shaft of the generator after it passes through the turbine, it is condensed in a condenser and then pumped to feed the boiler where it is heated (Vanita & Thanushkodi, 2011). For simplification, thermal power plants can be modeled as energy conversion from fossil fuel to electricity as described in Figure 2.1.

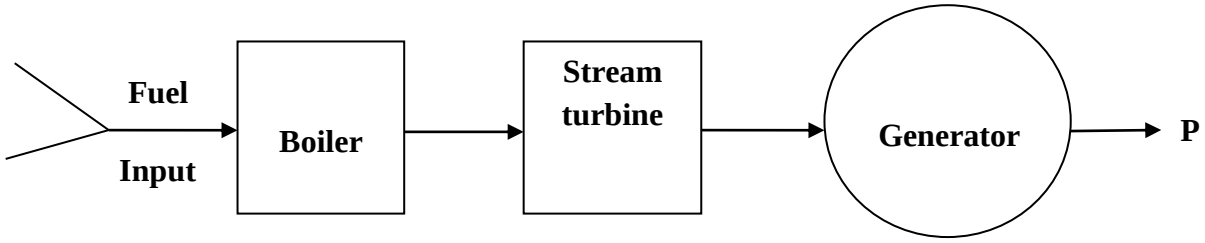


Figure 2.1: Energy conversion in a thermal power plant (Vanita & Thanushkodi, 2011)

The thermal unit system generally consists of the boiler, steam turbine and generator. The input of the boiler is fuel and the output is steam. The relationship between the input and output can be expressed as a convex curve (Vanita & Thanushkodi, 2011). The input of the turbine – generator unit is the volume of steam and the output is electrical power, the overall input-output characteristic of the whole generation unit can be obtained by combining directly the input-output characteristics of the boiler and the input-output characteristic of the turbine-generator unit (Vanita & Thanushkodi, 2011).

2.2.2 Generator operating cost

The total cost of operation includes the fuel cost, cost of labour, supplies and maintenance. Generally, cost of labour, supplies and maintenance are fixed percentages of incoming fuel cost. Other factors influencing power generation are operating efficiencies of generators and transmission losses (Vlachos, 2011). The total cost of generation is a function of the individual generation of the sources which can take values within certain constraints. The problem is to determine the generation of different plants such that total operating cost is minimum. The input of the thermal plant is generally measured in Btu/hr and the output power is the active power in MW. A simplified input – output curve of a thermal unit is known as heat – rate curve and it is

shown in Figure 2.2 (Vlachos, 2011). Where $P_{i,min}$ and $P_{i,max}$ are minimum and maximum power generation by the *ith* generating unit.

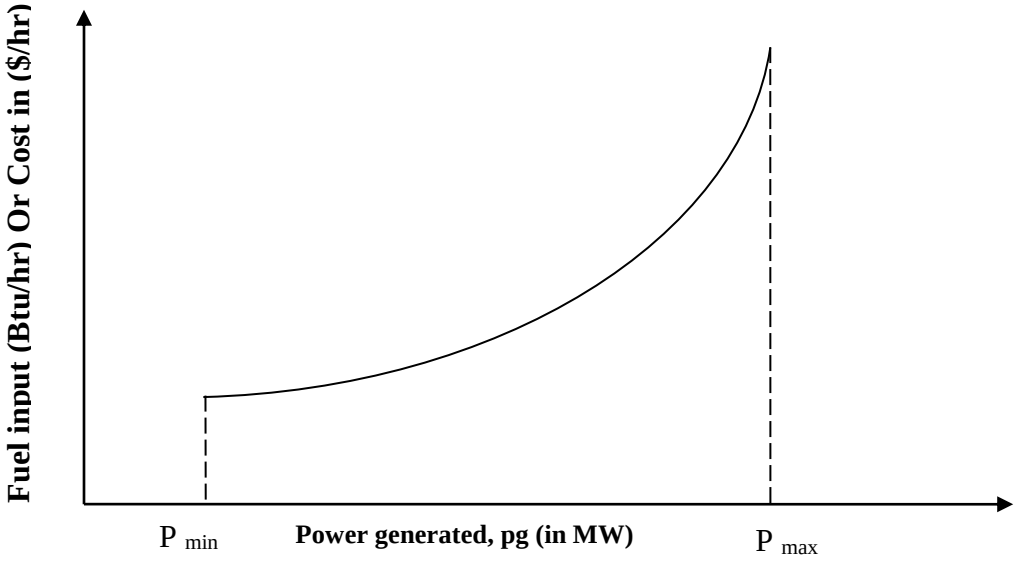


Figure 2.2: Heat rate curve (Vlachos, 2011)

2.2.3 Fuel efficiency

This is the ratio of power output in megawatt (MW) to fuel input in Btu/hr. The criterion of distributing the load between any two units is based on whether increasing the load in one unit as the load is decreased on the other unit by the same amount results in the increase or decrease in total load (Surekha & Sumathi, 2012).

2.2.4 Incremental *cost (*IC*)*

This is the limit of the ratio of increase in cost of fuel input in dollars per hour to corresponding increase in power output in megawatts as the increase in power output approaches zero. Incremental cost is the slope of the fuel cost curve, and the unit of *IC* is in dollars per megawatt hour (MWh). *IC* tells us how much it will cost to run a generator to produce an additional1MW

of power. All units in power plant must operate at the same incremental fuel cost for minimum cost in dollars per hour (Yun *et al*., 2011).

2.2.5 Economic Load Dispatch (ELD) formulation

The objective of an ELD problem is to find the optimal combination of power generations that minimizes the total generation cost while satisfying equality and inequality constraints. The fuel cost curve for any unit is an approximation of segments of quadratic functions of the active power output of the generator by assumption (Balamurugan & Subramanian, 2008).

2.2.6 The cost function

Cost function is a financial term used for expressing how differently costs can behave under a variety of circumstances. It shows how monetary outputs, everything from overhead and operating expenses to charges and fees change as the levels of an activity relating to those outputs change (Balamurugan & Subramanian, 2008). There are three basic types of linear cost functions:

1. Fixed cost functions.
2. Variable cost functions.
3. Mixed cost functions.

In a mixed circumstance, the cost will be fixed to a certain point that can be changed based on related activity. Analysts use these sorts of functions to make important predictions about the market place and to inform a variety of decision making tasks (Balamurugan & Subramanian, 2008).

For a given power system network, the problem may be described as optimization (minimization) of total fuel cost as defined by equation (2.1) under a set of operating constraints (Vlachos, 2011).

$$F_T = \sum_{i=1}^{n} F(P_i) = \sum_{i=1}^{n} a_i + b_i P_i + c_i P_i^2 \quad (2.1)$$

Where F_T represents the total fuel cost of generation in the system ($/hr), a_i, b_i, and c_i are the cost coefficients of the *i*th generator, P_i is the power generated by the *i*th unit and *n* is the number of generators (Vlachos, 2011).

The cost is minimized subject to the following generator capacities (inequality) and active power balance (equality) constraints, as given by equations (2.2) and (2.3) respectively (Vlachos, 2011):

$$P_{i,min} \leq P_i \leq P_{i,max} \quad for \;\; i = 1,2,\ldots,n \quad (2.2)$$

Where $P_{i,min}$ and $P_{i,max}$ are the minimum and maximum power output respectively of the *ith* unit.

$$P_D = \sum_{i=1}^{n} P_i - P_{Loss} \quad (2.3)$$

Where P_D is the total power demand and P_{Loss} is total transmission loss. The transmission loss P_{Loss} is defined by (2.4) as follows (Vlachos, 2011),

$$P_{Loss} = \sum_{i=1}^{n} \sum_{j=1}^{n} P_i B_{ij} P_j + \sum_{i=1}^{n} B_{i0} P_i + B_{00} \quad (2.4)$$

Where the B_{ij}s are the elements of loss coefficient matrix $\boldsymbol{B}$.

The cost function defined by F_T in equation (2.1) assumed a smooth quadratic fuel cost function without valve point (valve-point effects are ignored) loadings of the generating units. Such curve can be represented by the dotted line shown on Figure 2.3. The generating units with multi-valve steam turbines exhibit a greater variation in the fuel-cost functions. Since the valve point results

in ripples, a cost function contains higher order nonlinearity. Therefore, the function $F(P_i)$ in equation (2.1) should be replaced by equation (2.5) when considering the valve-point effects (Vanita & Thanushkodi, 2011).

Sinusoidal functions are thus added to the quadratic cost function to account for the valve-point effect:

$$F(P_i) = a_i + b_i P_i + c_i P_i^2 + |e_i \times \sin(f_i \times (P_{i,min} - P_i))| \quad (2.5)$$

Where e_i and f_i are the fuel cost coefficients of the *ith* unit with valve point effects. In figure 2.3 below, a, b, c, d, e and f are valve points.

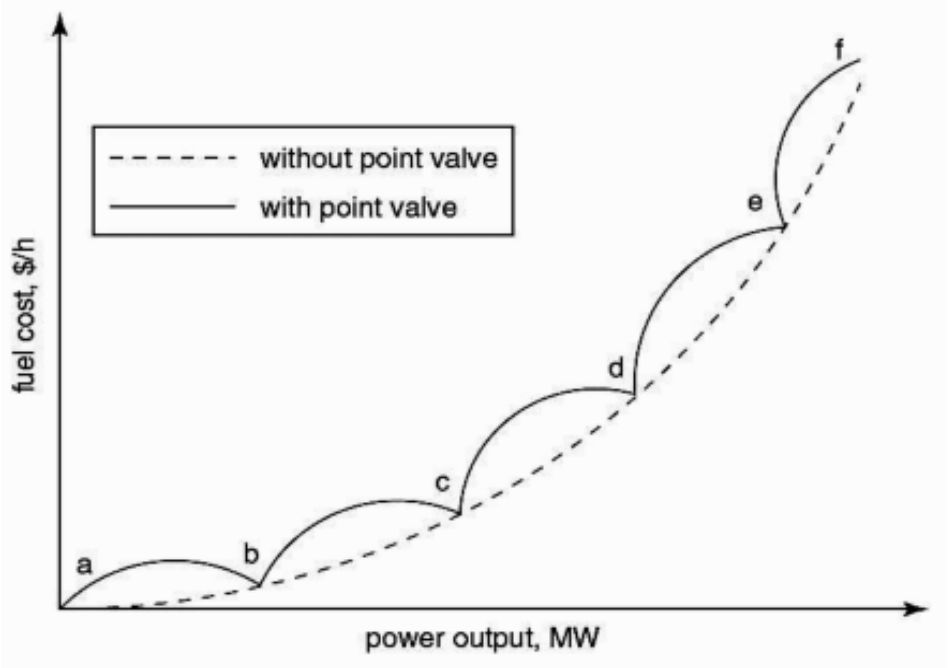


Figure 2.3: Incremental Fuel Cost Curve of a Typical Generating Unit (Vanita & Thanushkodi, 2011)

It is to be noted here that the fuel cost coefficients e_i and f_i are introduced in equation (2.5) to model the valve point loadings.

In practical situations, generating units are made up of subunits. These subunits combine to give rise to the overall installed capacity of the unit. Most units are designed to operate using more than one fuel type (source), particularly in the case where there is a great fluctuation in the price and availability of the dominant fuel types (Balamurugan & Subramanian, 2008). In the case of moderately large units, a combination of the available fuel types may be used to cover the power demand over the specified period of time. This type of scenario introduces a greater non-linearity into the overall fuel cost function. Therefore, the fuel cost function of such system can be modeled using multiple fuel cost function which is only defined for a particular range of power output within the specified maximum and minimum power generation. Considering both the valve point loading effect and multiple fuels, the cost function of the system may be easily represented using equation (2.6) (Balamurugan & Subramanian, 2008).

$$F(P_i) = \begin{cases} a_{i1} + b_{i1}P_i + c_{i1}P_i^2 + |e_{i1} \times \sin(f_{i1} \times (P_{i1,min} - P_{i1}))| & \text{Fuel1: } P_i^{min} \le P_i \le P_{i1} \\ a_{i2} + b_{i2}P_i + c_{i2}P_i^2 + |e_{i2} \times \sin(f_{i2} \times (P_{i2,min} - P_{i2}))| & \text{Fuel2: } P_{i1}^{min} \le P_i \le P_{i2} \\ \vdots & \vdots \\ a_{ik} + b_{ik}P_i + c_{ik}P_i^2 + |e_{ik} \times \sin(f_{ik} \times (P_{ik,min} - P_{ik}))| & \text{Fuelk: } P_{ik-1}^{min} \le P_i \le P_{ik}^{max} \end{cases} \quad (2.6)$$

2.2.7 Solution by Lagrange method

The Lagrange cost minimization function may easily be represented using (2.7) as follows (Sinha & Chakrabarti, 2003):

$$L = F_T + \lambda \emptyset \quad (2.7)$$

In other words, the Lagrange function is defined by cost function F_T plus the constraint function φ multiplied by a penalty coefficient λ. This penalty coefficient is set in order to limit the extent to which a set of predefined constraints are violated. Then to minimize L, its derivative with respect to P_i needs to be set to zero. This will generate a system of simultaneous equations which can be termed as "Coordination Equations", and their solution minimizes the costs, as in equation (2.8) (Sinha & Chakrabarti, 2003):

$$\frac{\partial L}{\partial P_i} = \frac{dF_i}{dP_i} - \lambda\left(1 - \frac{\partial P_L}{\partial P_i}\right) = 0 \tag{2.8}$$

With this, the inequality conditions specified in equation (2.2) expand to the following set of equations (2.9) to (2.11) (Sinha & Chakrabarti, 2003):

$$\frac{dF_i}{dP_i} = \lambda: \qquad P_{i,min} \leq P_i \leq P_{i,max} \tag{2.9}$$

$$\frac{dF_i}{dP_i} < \lambda: \qquad P_i = P_{i,max} \tag{2.10}$$

$$\frac{dF_i}{dP_i} > \lambda: \qquad P_i = P_{i,mim} \tag{2.11}$$

These inequalities signify the fact that any unit with incremental cost higher than λ is "expensive" and should be set to operate at lowest level of production. In this way all equations are solved until all conditions are satisfied. The main point of concern is analyzing and limiting the level of production for each unit. In this Lagrange method, the transmission loss function defined in (2.4) is also applied during loss estimation (Sinha & Chakrabarti, 2003).

2.2.8 Solution by Dynamic programming method

Dynamic programming is a method of solving complex problems by breaking them down into simpler sub-problems. It is applicable to problems exhibiting the properties of overlapping sub-problems and optimal substructure. When applicable the method takes far less time than naïve methods which don't take advantage of the sub-problem overlap. The idea behind DP is quite simple. In general, to solve a given sub-problem, we need to solve different parts of the problem, and then combine the different parts of the solution to get an overall solution. Often when using a more naïve method, many of the sub-problems are generated and solved many times. DP approach seeks to solve each sub-problem only once, thus reducing the number of computations. Once the solution to a given sub-problem has been computed it is stored. The next time the same solution is needed it is simply looked up. The approach is especially useful when the number of repeating sub-problems grow exponentially as a function of the size of the input (Sun *et al*., 2014).

2.2.9 Solution by Quadratic programming method

A linearly constrained optimization problem with a quadratic objective function is called a Quadratic Program (QP). Due to its numerous applications; quadratic programming is often viewed as a discipline in and of itself (Jubril & Komolafe, 2013). Quadratic programming is an efficient optimization technique to trace the global minimum if the objective function is quadratic and the constraints are linear. Quadratic programming is used recursively from the lowest incremental cost regions to highest incremental cost region to find the optimum allocation. Once the limits are obtained and the data are rearranged in such a manner that the incremental cost limits of all the plants are in ascending order. The general quadratic programming can be written as in equation (2.12) and (2.13) (Jubril & Komolafe, 2013):

$$Minimize: \quad f(x) = cx + \frac{1}{2}x^T Q \tag{2.12}$$

$Subject\ to$:

$$Ax \leq b \tag{2.13}$$

and

$$x \geq 0 \tag{2.14}$$

Where $\boldsymbol{c}$ is an n-dimensional row vector describing the coefficients of the linear terms in the objective function, and $\boldsymbol{Q}$ is an ($n \times n$) symmetric matrix describing the coefficients of the quadratic terms. If a constant term exists it is dropped from the model. As in linear programming, the decision variables are denoted by the n-dimensional column vector $\boldsymbol{x}$, and the constraints are defined by an ($m \times n$) **matrix** A and an m-dimensional column vector $\boldsymbol{b}$ of right-hand-side coefficients (Jubril & Komolafe, 2013). We assume that a feasible solution exists and that the constraint region is bounded. When the objective function $f(\mathbf{x})$ is strictly convex for all feasible points the problem has a unique local minimum which is also the global minimum. A sufficient condition to guarantee strictly convexity is for $\boldsymbol{Q}$ to be positively definite. If there are only equality constraints, then the *QP* can be solved by a linear system. Otherwise, a variety of methods for solving the *QP* are commonly used, namely; interior point, active set, conjugate gradient, extensions of the simplex algorithm etc. (Jubril & Komolafe, 2013).

2.2.10 Solution by Non-linear programming method

Power system operation problems are nonlinear. Thus nonlinear programming (NLP) based techniques can easily handle power system operation problems with nonlinear programming problem, the first step in this method is to choose a search direction in the iterative procedure

which is determined by first partial derivatives of the equations (Pothiya *et al*., 2008). Therefore, these methods are referred to as the first order methods, such as generalized reduced gradient method. NLP based methods have higher accuracy than linear programming based approaches and also have global convergence, which means that the convergence can be guaranteed independent of the starting point, but a slow convergence rate may occur because of zig-zagging in the search direction (Pothiya *et al*., 2008).

2.2.11 Solution by Newton's method

Newton's method requires the computation of the second order partial derivatives of the power flow equation and other constraints and is therefore called a second – order method. The necessary conditions of optimality commonly are the Kuhn –Tucker conditions. Newton's method is favored for its quadratic convergence properties (Pothiya *et al*., 2008).

2.2.12 Solution by Heuristic approaches

Nature-inspired meta-heuristics are currently among the most effective tools for optimizing many NP-hard combinatorial problems. These methods are fundamental principles on which existing mechanisms of a biological phenomenon of nature are based. The natural systems are the most interesting inspiration for designing new methods to solve many optimization problems. The ant systems, artificial fish swarm, particle swarm optimization and bee algorithms are the techniques inspired from observing nature. These algorithms use the behavior of swarm intelligence. So they are based on a live insects or simple interactions among individual entities (Holland, 1975).

2.2.12.1 Particle swarm optimization

Particle swarm optimization (PSO) is a population based stochastic optimization technique inspired by the social behavior of flocks of birds or schools of fish. In PSO the potential solutions called particles, fly through the problem space by following the current optimum particles. The particles change their positions by flying around in a multidimensional search space until a relatively unchanged position has been exceeded (Li *et al*., 2013). A particle bases its search not only on its personal experiences but also by the information given by its neighbors in the swarm. Each particle keeps track of its coordinates in the problem space, which is associated with the best solution fitness it has achieved so far. The fitness value is also stored. This value is called pbest. Another best value that is tracked by the particle swarm optimizer is the lbest value obtained thus far by any particles in the neighbors of the particle (Li *et al*., 2013). This location is called lbest. When a particle takes the whole population as its topological neighbors, the best value is a global best and is called gbest. The main advantages of PSO are: easy implementation, single concept, robustness to control the parameters and less computational time compared to other optimization techniques while its disadvantages are: possibility of being trapped in local optimum, problems of dependency on initial point and parameter and difficulty in finding optimal design parameters and stochastic characteristic of the final outputs (Li *et al*., 2013).

2.2.12.2 Artificial fish swarm algorithm

Artificial fish swarm algorithm (AFSA) is a novel method for searching global optimum, which is typical of behaviorism in artificial intelligence. If there are *N* artificial fish in a swarm and that the vector X is the individual state of the artificial fish, $X = (x_1, x_2, \dots, x_n)$ where $x_1 (i = 1, \dots, n)$ is the variable to be optimized of AFSA, $Y=f(X)$ is the food concentration of the

artificial fish at the current position and Y is the objective function of practical problems. $D_{ij}=||X_i-X_j||$ is the distance between the individual artificial fish i and the individual artificial fish j. The other important parameters such as the visual field of the artificial fish, the maximum moving step, the congestion factor and the maximum number of tries in every forage are expressed as *Visual*, *Step*, *d* and *Try-number* (Azad *et al*., 2014). The congestion factor is to limit the fish swarm size of the artificial fish swarm so as to make more artificial fish individuals gather in the region with better state rather than the neighborhood with suboptimal state (Azad *et al*., 2014).

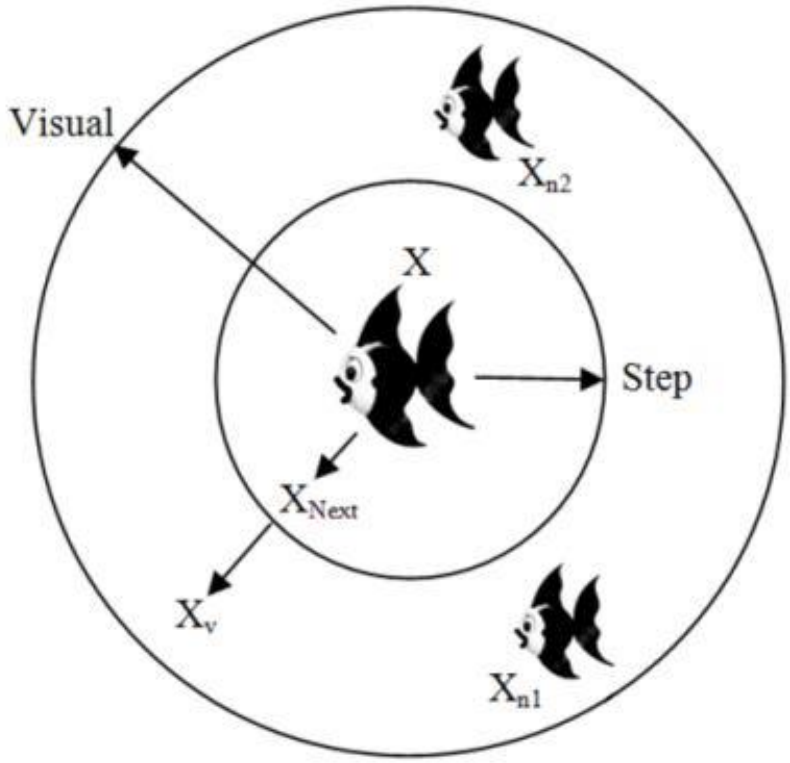


Figure 2.4: The visual and step of artificial fish (Azad *et al*., 2014)

The behaviors of artificial fish include: forage, swarm and follow.

1. **Forage:** If we assume that the current state of artificial fish is X_i and let this artificial fish carries out forage, it will at first select a state X_j randomly within its visual field. In seeking minimum, if $Y_i \geq Y_j$, then forage will be completed if moving one step towards this direction; if $Y_i \leq Y_j$, reselect a state X_j randomly and judge whether it satisfies the condition to move forward. After repeating this for a *Try-number* times, if it does not fulfill the forwarding condition, randomly move one step. This can be represented mathematically as in equation (2.15)(Costa *et al*., 2014):

$$\begin{cases} x_{inextk} = x_{ik} + \frac{x_{jk} - x_{ik}}{\left||x_j - x_i|\right|} . Random\ (step) & Y_j > Y_i \\ x_{inextk} = x_{ik} + Random(step) & Y_j \leq Y_i \end{cases} \quad (2.15)$$

Where: k=1, 2,....., n,

x_{ij} represents the k-th element of the current state vector X_j of artificial fish.

x_{jk} represents the k-th element of the state vector X_j after random movement

x_{inextk} represents the k-th element of the next state vector X_{inext} of artificial fish.

Y_i represents the objective function value of the current state.

Y_j denotes the objective function value after random movement,

Random (step) denotes a random number within [0 step].

2. **Swarm:** While swarming, fish has the natural ability to share the food and avoid any causes of distraction in the way. If the current state of the artificial fish is X_j and the number of companions in its visual domain is n. If n_f= 0, it means that there is no companion in its visual domain and then implement forage, If $n_f \geq 0$, it shows that there

are companions in its visual domain and then search the central position X_c (i.e centre between the fishes) of its companions according to equation (2.15)(Costa *et al*., 2014).

$$X_{ck} = \frac{\left(\sum_{j=1}^{n_f} x_{jk}\right)}{n_f} \tag{2.16}$$

X_c represents the state vector of the central position between fishes;

X_{ck} represents the *k*-th element of the state vector X_c of the central position;

X_{jk} represents the *k*-th element of the *j (j = 1, 2,…., n,*) companion X_j;

Y_c represents the objective function value of the central position.

Calculate the food concentration Y_c of the central position. Satisfying the following condition: $Yc.nf\ /Yi>1$. It indicates that the central position is not very congested and it is quite safe and then move toward this central position according to equation (2.16); otherwise, implement forage (Costa *et al*., 2014).

$$x_{inextk} = x_{ik} + \frac{x_{ck}-x_{ik}}{||X_c-X_i||}.Random.(step) \tag{2.17}$$

3. **Follow:** In the moving process of artificial fish swarm, when a single fish or several ones find food, the neighborhood partners have the natural ability to trail and reach the food quickly. If the current state of the artificial fish is X_i and the number of companions in its visual domain is *n*. If n_f=0, it implies that there is no companion in its visual domain and then implement forage; If $n_f \geq 1$, it is an indication that there are companions in its visual domain and then search the companion with the minimum corresponding function value X_{max} in its visual domain. If it satisfies $Y_{max}.n_f\ /Y_i>1$**.** It shows that the companion has small fitness value and that it is not very congested around here and then implement (2.18); otherwise, implement forage (Costa *et al*., 2014).

$$x_{inextk} = x_{ik} + \frac{x_{max,k} - x_{ik}}{||X_{max} - X_i||}. Random.(step) \quad (2.18)$$

X_{maxk} represents the *k*-th element of the state vector X_{max}.

4. **Bulletin board:** The function of bulletin board is to record the state of the optimal artificial fish. At the optimization iteration, each artificial fish individual examines and makes comparison of its own state with the current state on the bulletin board. If the state on the bulletin board is inferior to its own state, it substitutes the state on the board with its own state; in this way, the historical optimal state can always be recorded on the bulletin board and the final recorded optimal value is the optimal solution (Costa *et al*., 2014).

Based on the behavior description of the above-mentioned artificial fish, every artificial fish searches its environmental conditions and its companions to choose an appropriate behavior in order to move rapidly towards optimal direction. Finally, the artificial fish gathers around several local optimums. The algorithm of implementation flow includes (Fang *et al*., 2014):

I. Initialization: define the population size as *N*; generate randomly *N* individuals within the definition domain of the variable and assign the maximum generation Gen max, the generation Gen, the visual field of the artificial fish Visual, the moving step of the artificial fish Step, the congestion factor *d* and the trials *Try-number*.

II. Assign the value on the bulletin board: calculate and compare the corresponding fitness value to every individual fish; choose the optimal state of the artificial fish and assign its value to the bulletin board.

III. Choose implementation behavior: every artificial fish simulates swarm and follow; implement the optimal behavior by comparing the fitness value; the default behavior is forage and *Gen*=*Gen* +l.

IV. Update the bulletin board: compare the fitness value of every artificial fish and the value on the bulletin board, replace it if it is better than the value on the bulletin board; otherwise, keep the value on the bulletin board unchanged.

V. Judge end condition: when *Gen*>*Gen max*, end the algorithm and output the optimal value; otherwise, turn to step (III).

Figure 2.5 shows the flow chart of the AFSA algorithm

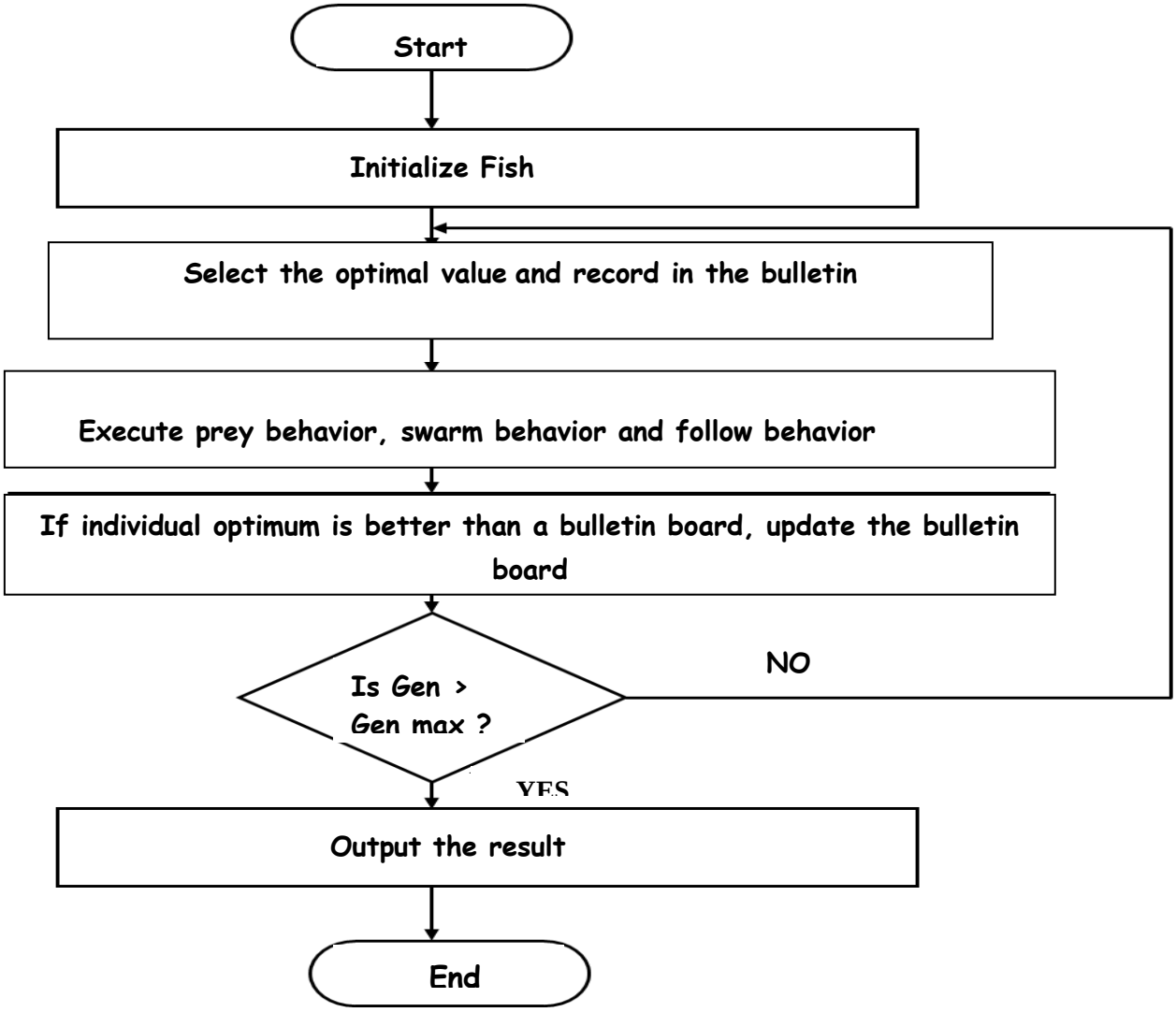


Figure 2.5: Flowchart of Artificial Fish Swarm Algorithm (Fang *et al*., 2014)

2.2.12.3 Genetic algorithm

The idea of evolutionary computing was introduced in 1960 by Rechenberg in his work "Evolutionary strategies". GA is a computerized search and optimization algorithm based on mechanics of natural genetics and natural selection. Prof. John Holland of University of Michigan conceived the concept of these algorithms in the mid-sixties and published (Chatterjee *et al*., 1996).

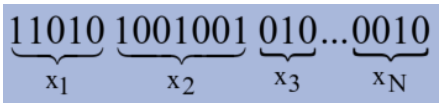
A genetic algorithm (GA) is a search and optimization method which works by mimicking the evolutionary principles and chromosomal processing in natural genetics. A GA begins its search with a random set of solutions usually coded in binary strings. Every solution is assigned a fitness which is directly related to the objective function of the search and optimization problem. Thereafter, the population of solutions is modified to a new population by applying three operators similar to natural genetic operator (Chatterjee *et al*., 1996):

1. Reproduction;
2. Crossover; and
3. Mutation.

It works iteratively by successively applying these three operators in each generation till a termination criterion is satisfied. Over the past decade and more, GA has been successfully applied to a wide variety of problems, because of their simplicity, global perspective, and inherent parallel processing (Chatterjee *et al*., 1996)...

The GA is an iterative optimization procedure which works with a number of solutions (collectively known as population) instead of working with a single solution in each iteration. The steps involved in genetic algorithm are further described as follows (Chatterjee *et al*., 1996):

I. **Representation:** In a binary coded GA, every variable is first coded in a fixed-length of binary string. For example, a string representing *N* problem variables is illustrated below:



String Representation of *N*-problem Variables (x) (Chatterjee *et al*., 1996)

The *ith* problem variable is coded in a binary substring of length l_i, so that total number of alternatives allowed in that variable is 2^{li}. The lower bound solution X_i^{min} is represented by

solution (0, 0… 0) and the upper bound solution X_i^{max} is represented by the solution (1, 1… 1). And other substring s_i decodes to a solution X_i as follows;

$$X_i = X_i^{min} + \frac{X_i^{max} - X_i^{min}}{\left|2^{l_i} - 1\right|} DV(s_i) \quad (2.19)$$

Where $DV(s_i)$ is decoded value of string s_i. The decoded value of a binary substring s_i= (s_{i-1},s_{i-2},….,s_2,s_1,s_0) is calculated as$\sum_{j=0}^{l-1} 2^j s_j$, where$s_j \epsilon [0,1]$. The length of substring is usually decided by precision needed in a variable. For example if three decimal places of accuracy are needed in the *ith* variable, total number of alternatives in the variable must be $\frac{X_i^{max} - X_i^{min}}{0.001}$, which can be set equal to 2^{li} and l_i can be computed as follows:

$$l_i = log_2 \frac{X_i^{max} - X_i^{min}}{\varepsilon_i} \quad (2.20)$$

Here, the parameter ε_i is desired precision in *i*-th variable. The total string length of an *N*-variable solution is then $l = \sum_{i=1}^{N} l_i$ in the population, l-bit strings are created at random (at each of *I* positions, there is an equal probability of creating a 0 or 1). Once such string is created, the first I_i bits can be extracted from the complete string and corresponding value of the variable x_i can be calculated using equation (2.20) and using the chosen lower and upper limits of variable x_I. This process is continued until all N-variables are obtained from complete string. Thus, an l-bit string represents a complete solution specifying all *N* variables uniquely. Once these values are known, the objective function *f*(x_1,x_2, ... x_N) can be computed (Chatterjee *et al*., 1996).

In a GA, each string created either in the initial population or in the subsequent generations must be assigned a fitness value which is related to objective function value. For maximization problem, a string's fitness can be equal to string's objective function value. However, for minimization problems, the goal is to find a solution having minimum objective function value.

Thus, the fitness can be calculated as the negative of the objective function so that solutions with similar objective function value get larger fitness. There are number of advantages of using a string representation to code variables. First, this allows a shielding between working of GA and actual problem. The same GA code can be used for different problems by only changing definition of coding a string. This allows a GA to have widespread applicability. Secondly, a GA can exploit the similarities in string coding to make its search faster, this is important in working of a GA (Chatterjee *et al*., 1996).

II. **Reproduction:** Reproduction (or selection) is usually the first operator applied to a population. Reproduction selects good strings in a population and forms a mating pool. The essential idea is that above-average strings are picked from the current population and duplicates of them are inserted in the mating pool. The commonly used reproduction operator is the proportionate selection operator, where a string in the current population is selected with probability proportional to the string's fitness. Thus, the *i*th string in the population is selected with probability proportional to $\mathcal{E}_i$. Since the population size is usually kept fixed in a simple GA, the cumulative probability for all string in the population must be one.

Therefore, the probability for selecting *i*th string is given in equation (2.21)

$$P(si) = {f_i}\Big/{\sum_{i=1}^{N} f_i} \qquad 2.21$$

Where *N* is the population size and f_i is the fitness of the *i*th string.

One way to achieve this proportionate selection is to use a roulette-wheel with the circumference marked for each string proportionate to the string's fitness (Chatterjee *et al*., 1996).

III. **Crossover:** The crossover operator is applied next to the string of the mating pool. In crossover operator, two strings are picked from the mating pool at random and some portion of the strings is exchanged between the strings. In a single-point crossover operator, both strings are cut at an arbitrary place and right-side portion of both strings are swapped among themselves to create two new strings, as illustrated below:

Parent 1	0	0	0	0	0	$\Rightarrow$	0	0	1	1	1	Child 1
Parent 2	1	1	1	1	1		1	1	0	0	0	Child 2

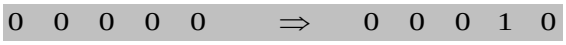
An Illustration of Genetic Crossover (Dhebar and Deb, 2017)

It is interesting to note from the construction that good substrings from either parent string can be combined to form better child string if an appropriate site is chosen. Since the knowledge of an appropriate site is usually not known, a random site is usually chosen (Dhebar & Deb, 2017). However, it is important to realize that the choice of a random site does not make this search operation random. With a single-point crossover on two *l*-bit parent strings, the solution search can only find at most different strings in the search space, whereas there are a total of strings in the search space. With a random site, the children strings produce 2^i which may or may not have a combination of good substrings from parent strings depending on whether the crossing site falls in the appropriate site or not. But we do not worry about this aspect too much, because if good strings are created by crossover, there will be more copies of them in the next mating pool generated by the reproduction operator. But good strings are not created by crossover; they will not survive beyond next generation, because reproduction will not select bad strings for the next mating pool. In a two-point crossover operator, two random sites are chosen. This idea can be extended to create multi-point crossover operator and the extreme of this extension is known as a

uniform crossover operator. In a uniform crossover for binary strings, each bit from either parent is selected with a probability of 0.5 (Dhebar & Deb, 2017).

The main purpose of the crossover operator is to search the parameter space. Other aspect is that the search need to be performed in a way to preserve the information stored in the parent string maximally, because these parent strings are instances of good strings selected using the reproduction operator. In the single-point crossover operator search is not extensive, but the maximum information is preserved from parent to children. On the other hand, in the uniform crossover, the search is very extensive but minimum information is preserved between parent and children strings. If a crossover probability of P_c is used then $100P_c$% strings in the population are used in the crossover operation and $100(1-P_c)$ % of the population are simply copied to the new population (Dhebar & Deb, 2017).

IV. Mutation: Crossover operator is mainly responsible for the search aspect of genetic algorithms, even though the mutation operator is also used for this purpose sparingly. The mutation operator changes a 1 to a 0 and vice versa with a small mutation probability P_m. An example of mutation operation is as illustrated below:



An Illustration of mutation operation (Dhebar and Deb, 2017)

In the above example, fourth gene has changed its value, thereby creating a new solution. The need for mutation is to maintain diversity in population. For instance, if in a particular position along the string length all strings in the population have a value 0, and a 1 is needed in that position to obtain optimum or a near-optimum solution, then mutation operator described above

will be able to create a 1 in that position. The inclusion of mutation introduces some probability of changing that 0 into 1. Furthermore, for local improvement of a solution, mutation is useful (Dhebar & Deb, 2017).

After reproduction, crossover, and mutation are applied to whole population, one generation of GA is completed. The reproduction operator selects good strings and the crossover operator recombines good substrings from two good strings together to hopefully form a better substring. The mutation operator alters a string locally to create a better string. Even though none of these claims is guaranteed and / or tested while creating a new population strings, it is expected that if bad strings are created they will be eliminated by the reproduction operator in next generation and if good strings are created, they will be emphasized. To make a faster convergence of a GA to real-world problems, problem-specific operators are often developed and used, but the above three operators portray fundamental operations of a genetic algorithm and facilitate a comparatively easier mathematical treatment (Dhebar & Deb, 2017).

The flow chart for the GA algorithm is shown in Figure 2.6.

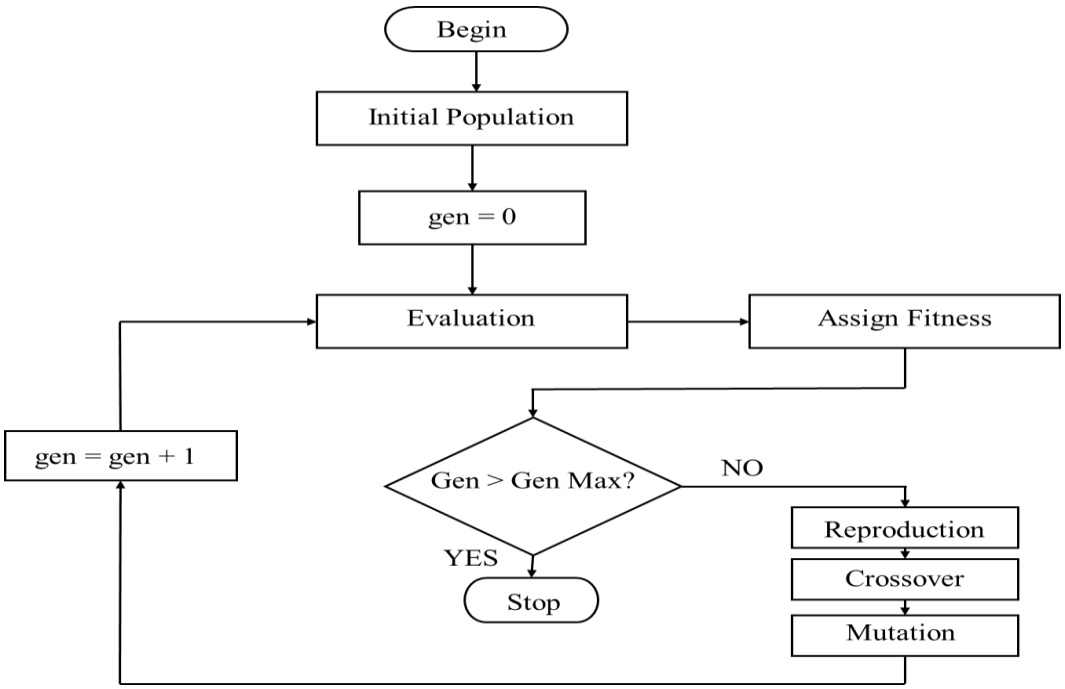


Figure 2.6: Flow chart of Binary Coded GA (Dhebar & Deb, 2017)

2.2.12.4 Hybrid algorithms

Hybrid algorithms try to make use of the merits of different methods to improve the performance of algorithms that are based on a single method. The main aim of proposing an algorithm as a hybrid of two or more methods is to speed up the convergence and to get better quality of solutions than that obtained when applying the individual methods by utilizing their strength and mitigating their weaknesses. A new algorithm integrating Genetic Algorithm (GA) and Artificial Fish Swarm Algorithm (AFSA) can be used to solve an economic dispatch problem. The core of the algorithm is based on GA. AFSA is used to generate new population members in the reproduction phase of the GA. AFSA method is used to accelerate the convergence of the GA by applying the AFSA for all the population members. In other words, GA is used for local search while AFSA is used for global search (Fang *et al*., 2014).

2.3 Review of Similar Works

In the literature, a large number of publications have been made on the various aspects of power system distribution network. Quite a number of such publications have been consulted which served as a guide towards achieving the aim and objectives of the research work. Some of these publications are reviewed below.

Bouktir & Slimani, (2005) presented a solution of optimal power flow (OPF) problem of electrical power system using real type genetic algorithm. The objective was to minimize the total fuel cost of generation and environmental pollution caused by fossil based thermal generating units and also maintain an acceptable system performance in terms of limits on generator real and reactive power outputs, bus voltages, shunt capacitors/reactors, transformers tap-setting and power flow of transmission lines. The algorithm was developed in an Object Oriented environment using C++ programming language. The economic power dispatch was applied to an IEEE 30-bus model system (6-generator, 41-line and 20-load). The numerical results have demonstrated the effectiveness of the stochastic search algorithms. Further analyses indicated that this method can be effective for large-scale power systems. However, the rate of convergence of the proposed method can be improved by hybridization with other heuristic algorithm such as AFSA.

Chiang, C.-L. (2005) presented an improved genetic algorithm with multiplier dating (IGA_MU) for power economic dispatch of units with valve-point effects and multiple fuels. The IGA was equipped with an improved evolutionary direction operator and a migration operation in order to efficiently search and actively explore solutions, while the MU was employed to handle the equality and inequality constraints of the PED problem. In order to demonstrate the effectiveness of the proposed approach, it was applied on four test systems comprising 20, 40, 80

and 160 generating units. Additionally, the proposed algorithm was compared with previous methods and the conventional genetic algorithm (CGA) with the MU (CGA_MU). However, the major setback of this approach is its high computational time. This can be improved by hybridizing GA with AFSA.

DosSantosCoelho, L., & Mariani, V. C. (2006) proposed a new approach for solving economic load dispatch problems with valve-point effect. The proposed method combines the DE algorithm with the generator of chaos sequences and sequential quadratic programming (SQP) technique to optimize the performance of economic dispatch problems. The DE with chaos sequences is the global optimizer, and the SQP is used to fine-tune the DE run in a sequential manner. The combined methodology and its variants are validated for two test systems consisting of 13 and 40 thermal units whose incremental fuel cost function takes into account the valve-point loading effects. The proposed combined method outperforms other state-of-the-art algorithms in solving load dispatch problems with the valve-point effect. However, the solution can be made more robust if hybridized with AFSA.

Vlachogiannis & Lee, (2009a) presented an improved coordinated aggregation-based particle swarm optimization (ICA-PSO) algorithm for solving the optimal economic load dispatch (ELD) problem in power systems. In the ICA-PSO algorithm, the number of search intervals for the particles was selected adaptively and the particles searched the decision space with accuracy up to two decimal places resulting in the improved convergence of the process. The ICA-PSO algorithm was tested on a number of power systems, including the systems with 6, 13, 15, and 40 generating units, the island power system of Crete in Greece and the Hellenic bulk power system, and is compared with other state-of-the-art heuristic optimization techniques (HOTs), demonstrating improved performance over them. However, PSO has the possibility of being

trapped in a local optimum. In addition to that, it always has difficulty in finding optimal design parameters and the stochastic characteristic of the final output. Hence the rate of convergence of the algorithm may be reduced when non linear constraints are considered.

Rahmat Azami, (2011) incorporated two approaches into the EDC and DCOPF problems. One of them was a mathematical optimization technique, Lagrangian Relaxation (LR) and the second was a heuristic one, Particle Swarm Optimization. The LR technique was based on the derivatives while the PSO was a non-derivative technique. The DCOPF methodology has been considered for Locational Marginal Pricing (LMP) calculation in LR, which is not available in PSO techniques. On the other hand, PSO technique may be able to provide the optimal solution, LR usually gets stuck at a local optimum in large scale power system and it takes much time in calculating LMP which makes its convergence difficult when large power systems are involved. This limits its application.

Gargeya & Pabba, (2013) presented Economic Load Dispatch of real power generation. The work considers valve point loading effects of the generating units. Two intelligent search methods were considered, namely, genetic algorithm and pattern search methods. Equality constraint was satisfied by penalty approach method. Two typical test cases of 5-generator, and 13-generator were carried out. The effectiveness of the approach can be made more realistic by incorporating no linear constraints such as multiple fuel cost function into the objective function.

Azad *et al*., (2014) proposed an improved binary artificial fish swarm algorithm for 0-1 the multidimensional knapsack problem. The algorithm uses population of points in space to represent the position of fish in school; a point is represented by binary string of 0/1 bits. Each bit of a trial point is generated by copying the corresponding bit from the current point or from some other specified point, with equal probability. Some randomly chosen bits of a selected point

were occasionally changed from 0 to 1, or 1 to 0, with a user defined probability. The infeasible solutions were made feasible by a decoding algorithm. The comparison with other methods shows that the proposed method gives a competitive performance. HGAFSA could perform better in terms of execution time.

Fang *et al.*, (2014) presented a hybrid real coded genetic algorithm and artificial fish swarm algorithm for short-term optimal hydrothermal scheduling (SHS). RCGA was used for global search while AFSA was used for local search in order to improve the exploitation capability of the algorithm. The water transport delay between connected reservoirs was taken into account. Moreover, new coarse and fine adjustment methods without any penalty factors and extra parameters were proposed to deal with all equality and inequality constraints. The feasibility and effectiveness of the proposed RCGA–AFSA method was tested on two hydrothermal systems and compared with other method, the simulation results obtained by hybrid RCGA–AFSA showed superiority in fuel cost and computation time. However, HGAFSA still showed lower fuel cost and faster convergence rate.

Gupta & Chawala, (2015) presented a new efficient approach to economic load dispatch (ELD) problem with cost functions using curve fitting, ANN and particle swarm optimization. The ELD problem was modeled using multiple fuel cost function typically for coal power plants with varying quality. Curve fitting technique was used to obtain the coefficients of the cost curve. The same data is used for the training of the artificial neural network. The effectiveness of the algorithm was validated by carrying out extensive test on a power system involving eight (8) thermal generating units. The variation in calorific values of the coal used in different generators results in cost curve coefficients change. This effect was incorporated using curve fitting, ANN and PSO approaches. The ELD problem was then optimized. The comparison showed a better

result. However, the effectiveness of the technique can be improved by considering higher order cost functions.

Khoa *et al*., (2015) proposed a Swarm based Mean-variance mapping optimization (MVMOS) for solving the economic dispatch. The proposed optimization algorithm was the extension of the original single particle mean-variance mapping optimization (MVMO). The novel feature was the special mapping function applied for the mutation based on the mean and variance of n-best population. The MVMOS was investigated on four test power systems, including 3, 13, 20 thermal generating units and large-scale system of 140 units with quadratic cost function and the obtained results were compared with many other known methods in the literature. Test results showed that the method was quite efficient and can be implemented for solving economic dispatch. However, the effectiveness of the technique can be improved by incorporating non linear parameters such as valve point loading.

Sharma *et al*., (2015) presented Grey Wolf Optimization (GWO) to solve convex economic load dispatch (ELD) problem. Grey Wolf Optimization (GWO) is a new meta-heuristic inspired by grey wolves. The leadership hierarchy and hunting mechanism of the grey wolves was mimicked in GWO. The technique was implemented on two different test systems for solving the ELD with various load demands. The results were compared with other existing techniques. However, the ELD problem was modeled based on the quadratic fuel cost function, which is less effective than the higher order cost function (when non linear constraints parameters are considered).

Aref Jalili *et al*., (2015) presented a hybrid method based on Firefly Algorithm (FA) and Fuzzy Mechanism (FM) for solving Economic Load Dispatch (ELD) problem by considering the valve point in power system. Nonlinear constraints of generators, such as ramp rate limits, prohibited

operating zone, generation limits, transmission line loss and non-smooth cost functions were all considered. An attempt was made to find out the minimum cost by using FA using the data of six and forty generating units. Results were also presented. Though the problem definition was quite impressive, but the results failed in demonstrating the effectiveness of the method as compared with other similar methods in the literature.

H Kaur *et al.*, (2015) presented a method for solving Economic Load Dispatch problem in order to operate an electric thermal power station within estimated load demand limits. The ELD objective function was formulated within the limits of equality and inequality constrains. The aim is to minimize fuel cost. The simulation results presented were quite good compared to the results so far in literature, but cannot be relied upon in the case of practical scenario simulation as they were based on quadratic cost function. Therefore, the accuracy of the method can be improved by considering other non linear constraints.

Kumar Sharma & Kasniya, (2015) applied Fuzzy Logic in combination with Genetic Algorithm (GA) to solve various power system problems. The computational results revealed that the proposed algorithm had excellent convergence characteristics and was superior to the GA and LIM. However, the work was only based on comparative analysis. Its effectiveness in solving multi-objective ELD problem can be improved.

Wu *et al.*, (2015) presented an Efficient Population Utilization Strategy for Particle Swarm Optimization (EPUSPSO) to solve the ELD problem of power system. The algorithm was tested on three different ELD cases of power system including 3, 13 and 40 IEEE generating units, and the obtained results were compared with those obtained from other algorithms using the same system parameters. The compared results showed that the algorithm was able to find optimal

solution effectively and accurately. However, a quadratic fuel cost function based ELD model was used, but in practical scenarios, fuel cost functions are highly non-linear and can best be represented by higher order cost functions.

Kasarapu & Sri, (2015) solved economic load dispatch problem by Lagrange Multiplier method for a network of 8 bus system with 4 thermal units under competitive electricity market with inclusion of demand response programs using different strategies for peak load reduction with the application of fuzzy logic principles and independent power producers (IPPs). However, the method presented is not suitable for highly non-linear fuel cost functions (which is one of the typical properties of ELD problems). Therefore, the flexibility of the method can be improved.

Subramanian *et al*., (2015) presented a Novel TANAN's Algorithm (NTA) for solving convex Economic Load Dispatch (ELD) problems considering transmission line losses. The main objective of NTA was to minimize the total fuel cost of the generating units, subject to limits on generator power output. The NTA was a simple numerical random search approach based on a parabolic TANAN function. The work therefore presented an application of NTA to ELD problems for different IEEE standard test systems. The simulation results showed that the simplicity of the proposed algorithm vary widely with increase in the number of generating units. This limits its application in real ELD problems.

Dewangan *et al*., (2015) presented lambda iteration method to solve an ELD problem using MATLAB for three and six generating units with and without transmission line losses. However, the method may not be suitable for application when other non linear constraints such as valve point loading and multiple fuel cost are considered.

Achana N, (2015) proposed a solution for unit commitment and economic load dispatch problem using hybrid Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). The work showed lower operating cost and execution time when compared to several state-of-the-art techniques. The proposed system was tested on the seven unit Neyveli thermal power station system data. The algorithm was developed and executed using C++ and MATLAB 7.1 software. The work looks promising in addressing ELD problem but its guarantee can be made stronger by incorporating multiple fuel cost function into the proposed ELD model.

Elsayed *et al.*, (2016) proposed a Modified Social Spider Algorithm for solving non-convex Economic Load Dispatch problem (MSSA). In their work, the binary mask based random walk used for generating new solution in the social spider algorithm (SSA) was replaced with mutation process after application of selection process. The performance and efficacy of the proposed MSSA were tested using four IEEE test systems having 6, 40, 80 and 140 generating units. The method proved effective in addressing generation fuel cost.

Pradhan *et al.*, (2017) proposed an oppositional grey wolf optimization algorithm (OGWOA) for resolving the optimal economic load dispatch (ELD) problem. Their approach was based on the two common wolf behaviors namely: the hunting and social behavior. The opposition characteristics of grey wolves was mimicked and integrated into the solution strategy in order to ensure optimality and accelerate the rate of convergence of the conventional GWO algorithm. The performance of the proposed algorithm was demonstrated by applying it to a small, medium and large scale hypothetical test systems comprising of 13, 40 and 160 generating units. The results were further compared with that of the conventional GWO and other similar heuristic algorithm based techniques. Its effectiveness could be enhanced if it is hybridized with Artificial Fish Swarm Algorithm.

However, HGAFSA demonstrated superiority in terms of convergence rate, computational expensiveness and objective function minimization. Despite its effectiveness in solving ELD problems, hybrid heuristic algorithms based technique could perform better especially in the case of large number of generators (since ELD problem involves large number of possible solutions).

The reviewed literature so far covered a wide variety of ELD problem formulations and their respective solutions, the accuracy, efficiency, robustness, and flexibility of the approach presented in these literatures can be improved. It is worth noting that, Practical economic dispatch (ED) problems have highly nonlinear objective function with equality and inequality constraints. Conventional methods such as lambda iteration method and gradient method are not suitable for solving such problems. Heuristic techniques on the other hand (if not robust enough) may fail to provide optimal solution to the ELD problems (especially when the fuel cost function is highly non-linear). In the proposed research work, a more robust method was developed by hybridizing AFSA and BCGA to form a method called "HGAFSA". Some of the target features of the HGAFSA algorithm are:

- Speed (able to overcome the computational complexity introduced by the sinusoidal component of the fuel cost function);
- Flexibility (able to optimally solve ELD problems with either quadratic or higher ordered cost function); and
- Accuracy (provide accurate results for five standard IEEE test cases [13, 40, 110, 140 and 160 generating units] and 463 generating units obtained by the combination of the five generating units.

Detailed characteristics of the proposed HGAFSA method will be presented in the subsequent parts of the proposed research work.

CHAPTER THREE

METHODOLOGY

3.1 Introduction

This chapter presents detailed procedure that was used in achieving the set aim and objectives of the proposed research work, some useful findings and a set of pseudo codes describing the algorithm steps that were adopted.

3.2 Formulation of the Proposed HGAFSA

The proposed Hybrid Binary Coded Genetic Algorithm and Artificial Fish Swarm Algorithm (HGAFSA) are designed based on the available parameter dredging steps present in the conventional GA and AFSA. However, each of the separate algorithms (GA and AFSA) is assumed to be composed of three major steps as described in the following.

1. GA
    a. Reproduction;
    b. Crossover;
    c. Mutation.
2. AFSA
    a. Preying;
    b. Swarming;
    c. Chasing.

The mathematical formulation of these steps has been earlier described in Chapter two of this work.

The proposed HGAFSA is a logical combination of the six (6) steps listed above. It is worthy of noting that, the BCGA uses binary operation on a set of binary codes known as chromosomes which further comprises of genes whereas, the AFSA uses real numbers ranging between zero and one as parameters of artificial fish. As such, a Decoder function and an Encoder function are required to serve as converters from GA to AFSA and vice versa. These functions are intended to decode binary code into real numbers and later encode real numbers into binary respectively.

3.3 Decoder Function

The decoder function takes in four parameters as inputs and generates a decoded version of the main parameter as output. Here, the main parameter is the chromosome (*X*), whereas the remaining three parameters are:

i. $x_{\min}$ → Lower boundary of the desired decoded output;

ii. $x_{\max}$ → Higher boundary of the desired decoded output;

iii. N_{bits} → Number of bits per parameter.

The steps involved in decoding a single chromosome can be described using Decoder function as follows:

Algorithm 3.1: The Decoder Function

1. Input: X, $x_{\min}$, $x_{\max}$ and N_{bits};
2. Evaluate: N = number of bits in X;
3. Evaluate: p = $N*\left(N_{bits}\right)^{-1}$ (number of parameters);
4. Evaluate: $q = 0.5^{[1,2,.....N_{bits}]}$ (quantization levels);

5. Evaluate: $q_{norm} = q * \left(\sum_{i=1}^{N_{bits}} q_i \right)^{-1}$ (quantization level normalization);

6. Evaluate:

$$X_{decoded}(i) = [q(1) * X(j+1) \quad q(2) * X(j+2) \quad q(3) * X(j+3) \quad ... \quad q(N_{bits}) * X(j+N_{bits})]$$

for $i = 1 : p$ and $j = (i-1) * N_{bits)}$ (decoding the bits forming the ith parameter)

7. Output:

$$X_{decoded} = \left[X_{decoded}(1) \quad X_{decoded}(2) \quad X_{decoded}(3) \quad .... \quad X_{decoded}(p) \right] * (x_{\max} - x_{\min}) + x_{\min}$$

Using the Decoder function, a vector *X* with $p * N_{bits}$ elements is decoded into a vector $X_{decoded}$ with *p* elements. As an illustration, consider *X= [11011011011101111011011111011]*, if $x_{\min}$ = 0 and $x_{\max}$ =1, let N_{bits}=6. Algorithm 3.1 will yield $X_{decoded}$= *[0.8571 0.8730 0.4762 0.9365 0.9365]*

In the proposed HGAFSA once any of the GA steps is executed, the resulting output/population must be decoded before their respective finesses can be evaluated. Whereas, the resulting output/population from the AFSA steps are directly evaluated using the fitness function without been decoded.

3.4 Encoder Function

The encoder function was also written to counter the effect of the decoder function presented in Algorithm 3.1. However, a reversed procedure was adopted based on Algorithm 3.1 (moving from step 7 to 1). Here, the decoder function was intended to generate chromosome (X) given its

decoded version $X_{decode,}$ x_{min}, x_{max} *and* $N_{bits.}$ However, the Encoder Function formulation was omitted from this manuscript for brevity.

Generally, it can be said that: The Decoder function converts a chromosome into a fish; whereas, the Encoder function converts a fish back to a chromosome. (This can be further illustrated using Figure 3.1).

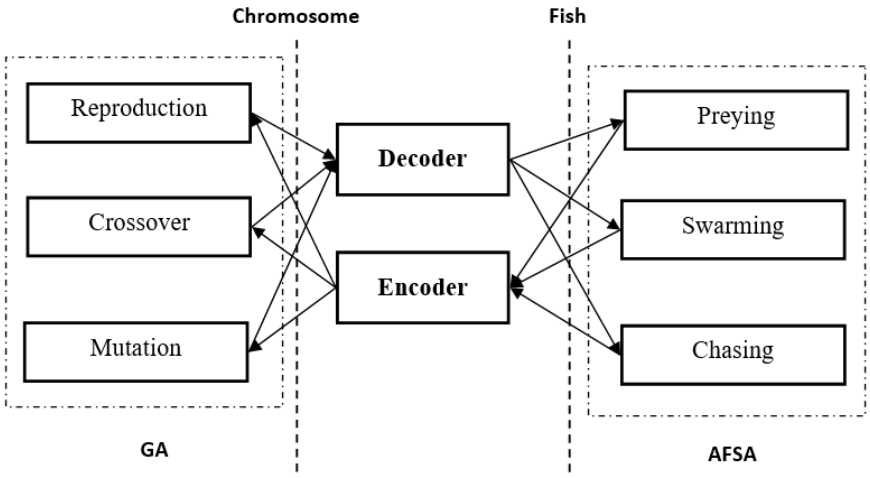


Figure 3.1: Illustration of Chromosome-Fish Conversion and vice versa

Furthermore, it is worthy of noting that, both X and $X_{decoded}$ are kept for reference during the optimization process using the proposed HGAFSA. However, either X or $X_{decoded}$ is later discarded depending on which of GA or AFSA steps (listed above) performs better at a given generation and at a given step in the HGAFSA dredging process. In the proposed HGAFSA, X represents a chromosome whereas $X_{decoded}$ represents a fish. At first, the entire population is stored as chromosomes. However, each chromosome is either left as a chromosome (X) or transformed into a fish $X_{decoded}$ depending on which of the HGAFSA step (GA step or AFSA step) performs better.

3.5 Population Update

The proposed HGAFSA is composed of two unique algorithms (GA and AFSA) with totally different parameter dredging procedure. When one of the GA steps is executed on an encoded fish (chromosome), the resulting chromosome might be of poor fitness than it would have had if an AFSA step was directly performed on the fish itself. However, this consequence might be on the reverse side. Therefore, the population must be carefully updated for optimality and improved convergence rate. Algorithm 3.2 further describes the population update procedure.

Algorithm 3.2: Population Updater

1. Input: S $\in$ (list of steps to be executed in chronological order);

   P $\in$(combined population of fish and chromosomes)
2. Define: $\Omega = \{\ \}$
3. For *k = 1* to *popsize*
4. For s = 1 to N_{steps}
5. If P(k) = fish & S(s)$\in$AFSA
6. Evaluate: $f = F_{obj}(P(k))$
7. Execute: $P(k) \rightarrow S(s) \rightarrow P(k)_{new}$
8. Evaluate: $f_{new} = F_{obj}(P(k)_{new})$
9. If f_{new} is better than f
10. Store : $\left[f_{new} \quad P(k)_{new} \right] \rightarrow \Omega$
11. Else
12. Store : $[f \quad P(k)] \rightarrow \Omega$
13. End

14. Elseif P(k) = fish & S(s)$\in$GA

15. Evaluate: $f = F_{obj}(P(k))$

16. Execute: $P(k) \rightarrow Encoder \rightarrow P(k)_{Encoded}$

17. Execute: $P(k)_{Encoded} \rightarrow S(s) \rightarrow P(k)_{Encoded,new}$

18. Evaluate: $f_{new} = F_{obj}(P(k)_{Encoded,new})$

19. If f_{new} is better than f

20. Store : $[f_{new} \quad P(k)_{Encoded,new}] \rightarrow \Omega$

21. Else

22. Store : $[f \quad P(k)] \rightarrow \Omega$

23. End

24. Elseif P(k) = chromosome & S(s)$\in$AFSA

25. Execute: $P(k) \rightarrow Decoder \rightarrow P(k)_{Decoded}$

26. Evaluate: $f = F_{obj}(P(k)_{Decoded})$

27. Execute: $P(k)_{Decoded} \rightarrow S(s) \rightarrow P(k)_{Decoded,new}$

28. Evaluate: $f_{new} = F_{obj}(P(k)_{Decoded,new})$

29. If f_{new} is better than f

30. Store : $[f_{new} \quad P(k)_{Decoded,new}] \rightarrow \Omega$

31. Else

32. Store : $[f \quad P(k)] \rightarrow \Omega$

33. End

34. Elseif P(k) = chromosome & S(s)$\in$GA

35. Execute: $P(k) \rightarrow Decoder \rightarrow P(k)_{Decoded}$

36. Evaluate: $f = F_{obj}(P(k)_{Decoded})$

37. Execute: $P(k) \rightarrow S(s) \rightarrow P(k)_{new}$

38. Execute: $P(k)_{new} \rightarrow Decoder \rightarrow P(k)_{new,Decode}$

39. Evaluate: $f_{new} = F_{obj}(P(k)_{new,Decode})$

40. If f_{new} is better than f

41. Store : $[f_{new} \quad P(k)_{new,Decode}] \rightarrow \Omega$

42. Else

43. Store : $[f \quad P(k)] \rightarrow \Omega$

44. End

45. End

46. End

47. End

48. Sorting: Rearrage the vectors in Ω (in the order of optimality)

The Flow Chart of the proposed HGAFSA is presented in Figure 3.2

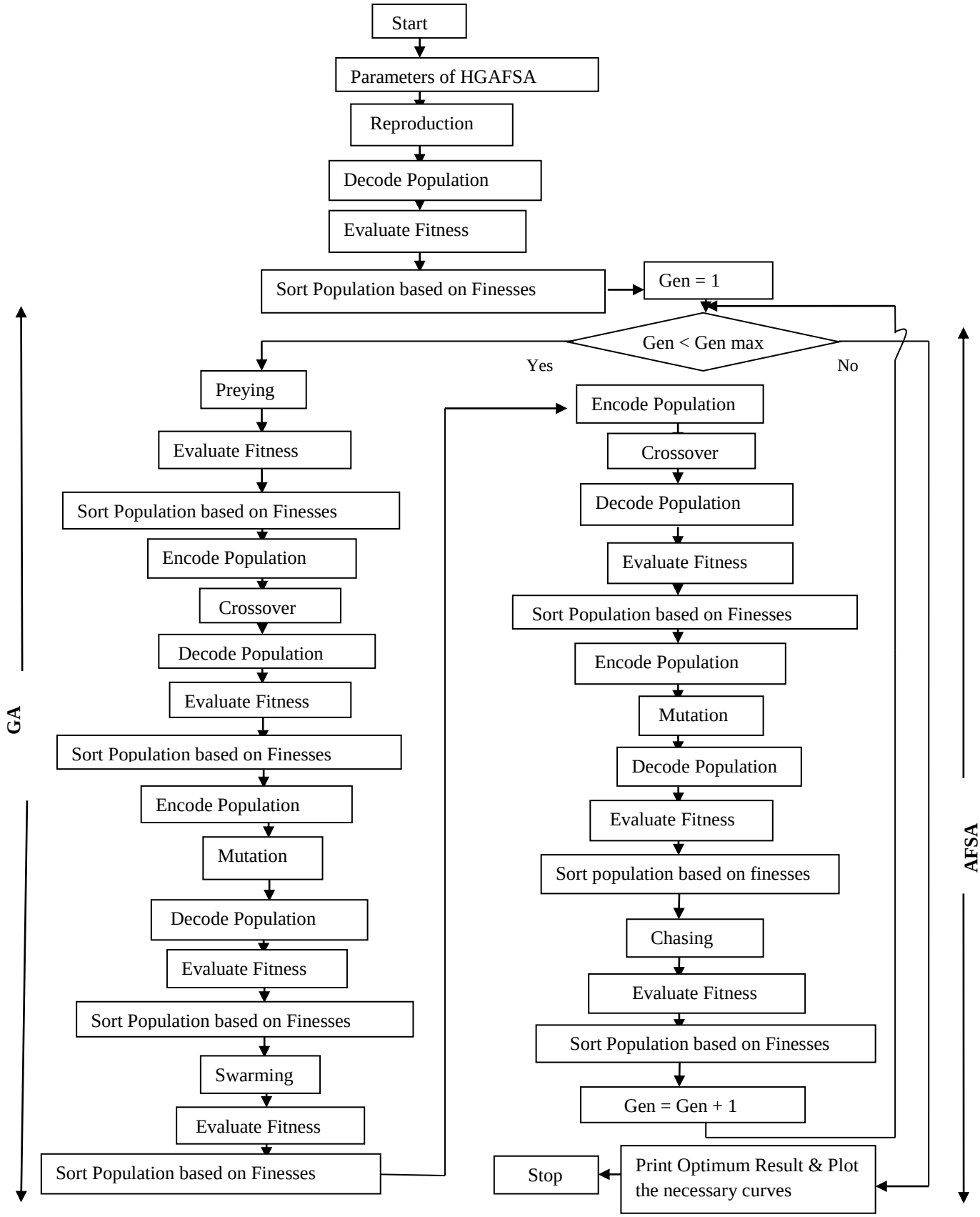


Figure 3.2: Flow Chart of Proposed Hybrid GA and AFSA (HGAFSA)

3.6 Model of the ELD Problem

To solve ELD problem using the proposed HGAFSA, a function is required to convert the random numbers generated by it into electrical power demand scheduled to the set of generating units. Let ***P*** be a set of power to be generated by the available generating units forming the ELD problem. Let $\boldsymbol{P_{max}}$ be the maximum power allowable for each of the units. Let also $\boldsymbol{P_{min}}$ be the minimum power allowable for each of the units. Let N_G be the number of generating units. Then:

$$P = [P_1 \quad P_2 \quad P_3 \quad \ldots \quad P_{N_G-2} \quad P_{N_G-1} \quad P_{N_G}] \tag{3.1}$$

$$P_{\max} = [P_{\max,1} \quad P_{\max,2} \quad P_{\max,3} \quad \ldots \quad P_{\max,N_G-2} \quad P_{\max,N_G-1} \quad P_{\max,N_G}] \tag{3.2}$$

$$\boldsymbol{P}_{\min} = [\boldsymbol{P}_{\min,1} \quad \boldsymbol{P}_{\min,2} \quad \boldsymbol{P}_{\min,3} \quad \ldots \quad \boldsymbol{P}_{\min,N_G-2} \quad \boldsymbol{P}_{\min,N_G-1} \quad \boldsymbol{P}_{\min,N_G}] \tag{3.3}$$

The next important parameter of ELD problem formulation is the total power (P_G) to be generated by the generating units to meet both the power demand and the power losses along the network. This can be described using the equality constraint as in equation (3.4).

$$\boldsymbol{P}_G = \boldsymbol{P}_D + \boldsymbol{P}_L \tag{3.4}$$

Where, P_D is the total power demand by the consumers; and P_L is the total power losses in the network. $\boldsymbol{P}_G$ can be expressed using equation (3.5).

$$P_G = \sum_{i}^{N_G} P_i \tag{3.5}$$

As described earlier, to evaluate the fitness of a population generated by HGAFSA, the population must be decoded into a fish ($X_{decoded}$). However, this fish must be further converted

into real power demand *P*. To achieve this, let $X_{decoded}$ be replaced by χ having *p* elements, such that equation (3.6) holds.

$$\chi = \lfloor x_1 \quad x_2 \quad x_3 \quad \ldots \quad x_{p-2} \quad x_{p-1} \quad x_p \rfloor \tag{3.6}$$

An ELD Encoder function is developed to transform χ into an equivalent *P*. The overall process can be described using Figure 3.3.

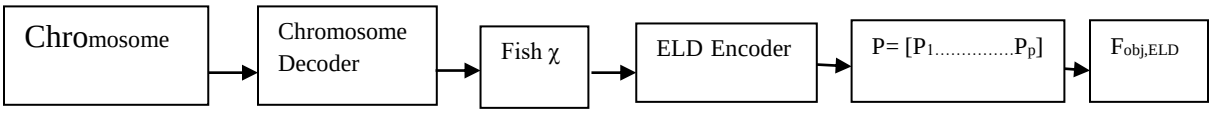


Figure 3.3: Evaluating the Fitness of a Chromosome/Fish

The function $F_{obj,ELD}$ is the objective function of ELD problem described in equation (2.12). The ELD Encoder function is given in Algorithm 3.3.

Algorithm 3.3: ELD Encoder

1. Input: χ ,*p*, N_G , $P_{\max}$, $P_{\min}$, and P_G
2. Create: $\varsigma = [\ ]$
3. For g = [1, 2, 3, …, N_G]
4. $\partial = (P_{\max}(g) - P_{\min}(g)) * \chi / \max(\chi)$
5. $\varsigma = \begin{bmatrix} \varsigma \\ \partial \end{bmatrix}$
6. End
7. Create: $\varepsilon = [\ ]$

8. For *par* = [1, 2, 3, …, *p*]

9. For g = [1, 2, 3, …, N_G]

10. $\rho(g) = \varsigma(g, par)$

11. End

12. For h = [1, 2, 3, …, N_G]

13. $$\gamma(h) = \frac{(P_G - \sum_{g}^{N_G} P_{\min}(g)) * \rho}{\left(\sum_{g}^{N_G} \rho(g)\right)} + P_{\min}(h)$$

14. End

15. While $\max(\gamma - P_{\max}) > 0$ (enforce the upper limit constraint for each generating unit)

16. $\rho = rand(1, N_G).* \rho + rand(1, N_G)$

17. Re-execute steps 12 to 14

18. End

19. $$\varepsilon = \begin{bmatrix} \varepsilon \\ \gamma \end{bmatrix}$$

20. End

21. For *s* = [1, 2, 3, …, *p*]

22. Evaluate: $Fit(s) = F_{obj,ELD}([\varepsilon(s,1) \quad \varepsilon(s,2) \quad \varepsilon(s,3) \quad ... \quad \varepsilon(s,N_G-1) \quad \varepsilon(s,N_G)])$

23. End

24. Determine: $\kappa \leftarrow Fit(\kappa) = \min(Fit)$

25. Assign: $P = [\varepsilon(\kappa,1) \quad \varepsilon(\kappa,2) \quad \varepsilon(\kappa,3) \quad ... \quad \varepsilon(\kappa,N_G-1) \quad \varepsilon(\kappa,N_G)]$

The proposed ELD encoder has the advantage that no generating unit can generate below its minimum allowable generating limit. However, its generation may exceed the allowable maximum. To prevent this, steps 15 to 18 were added to Algorithm 3.3 to enforce the maximum limit constraint.

However, it can be observed from step 22 of Algorithm 3.3, that the proposed ELD Encoder performs optimization during the encoding process. Therefore, it could be termed as an optimal ELD encoder of order *p*.

To further increase the performance of the proposed ELD Encoder, the dimension of χ can be extended for every given χ generated by the proposed HGAFSA. This extension was modeled as in equation 3.7.

$$\chi_{ext} = [\chi \quad \chi.*\left|\sin(2\pi * \chi)\right| \quad \chi.*\left|\cos(2\pi * \chi)\right|] \tag{3.7}$$

With this extension, a new optimal ELD encoder of order 3**p* can be formed by modifying steps 8 and 21 of the former optimal ELD encoder (order *p*) described earlier. This can be achieved using equation (3.8) and (3.9) respectively.

$$par = [1, 2, 3, \ldots, 3*p] \tag{3.8}$$

$$s = [1, 2, 3, \ldots, 3*p] \tag{3.9}$$

3.7 The Proposed HGAFSA based Higher Order ELD Algorithm

ELD problem is one of the power system analysis problems with large number of possible solutions. However, such solutions form a set of local optimums. As the number of generating units increases, the possible solutions increase exponentially. As such, an algorithm that can deeply search into the solution domain is required to locate the global optimum solution. In this work, HGAFSA optimization algorithm with high computation capability and fast rate of convergence is developed for complex ELD problem solving. The flow chart for the proposed HGAFSA based higher order ELD problem solver is shown in Figure 3.4.

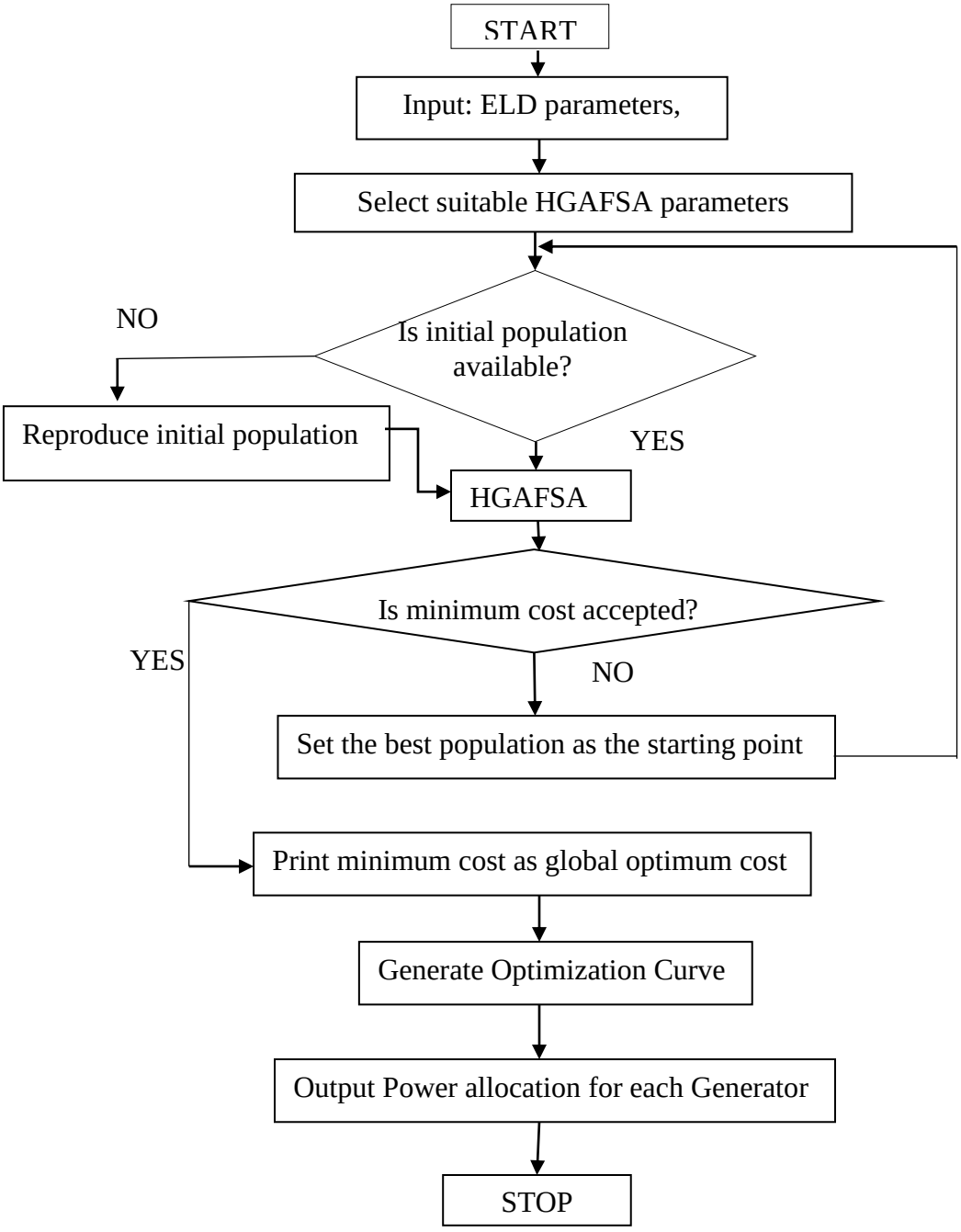


Figure 3 4: HGAFSA based ELD Algorithm

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 Introduction

In this chapter, five-test systems are used in simulation and the results are compared with those presented in (Pradhan *et al*., 2017) and other similar literatures regarding ELD problem solving.

4.2 Simulation Setup

To demonstrate the effectiveness of the developed HGAFSA for ELD problem solving, six test systems were used. The data of the test systems used for the simulation is provided in Appendix A1-A5. The developed algorithm was programmed in MALAB R2016a environment on a HP 8GB-RAM 2.3GHz Core-I3 Computer running Windows 10.1. Some of the M-Files used are also provided in the Appendix section of this work. A set of suitable parameters used for simulation are provided in Table 4.1. To achieve the desired objective, the best solution generated by HGAFSA is set as starting point if it did not meet the desired goal. This process is repeated as far as the optimum cost is higher than the best cost presented in Pradhan *et al*., (2017).

Table 4.1: HGAFSA Simulation Parameter Settings

S/No.	Name	Value
1	Population Size	64
2	Number of Parameters	32
3	Visual Distance AFSA	0.875 to 1
4	Crowding Factor AFSA	0.09 to 0.5
5	Step Size AFSA	0.00125 to 0.1
6	Max. Iteration	100
7	Min Cost	0
8	Mutation Rate GA	0.4 to 0.75
9	Selection Probability GA	0.375 to 0.5
10	Optimization Type	2
11	Min GA par	0
12	Max GA par	1
13	Number of Bits GA	8

4.3 Test system 1

In order to demonstrate the effectiveness of the proposed HGAFSA algorithm compared to the OGWO algorithm (Pradhan et al., 2017), a thirteen-unit test system with valve point effect is used. The system data including fuel cost coefficients and limiting value of active power of various generators are adopted from (dosSantosCoelho & Mariani, 2006). The power demand, in this case is assumed to be 2520 MW. To judge the superiority of OGWO and HGAFSA methods, the test results are compared with the results obtained by other algorithms such as Oppositional

Invasive Weed Optimization (OIWO) (Barisal & Prusty, 2015), Oppositional Real Coded Chemical Reaction Optimization (ORCCRO) (Bhattacharjee, Bhattacharya, & Dey, 2014), Biogeography Based Optimization (BBO) (Bhattacharya & Chattopadhyay, 2010), hybrid Differential Evolution based BBO (DE/BBO) (Bhattacharya & Chattopadhyay, 2010) and Improved Coordinated Aggregation based PSO (ICA-PSO) (Vlachogiannis & Lee, 2009b) available in the literature.

The comparative results of active power generation and fuel cost using the OGWO and HGAFSA methods along with other methods are given in Table 4.2. The results show that the HGAFSA based approach performed better than all other approaches presented in this work. Figure 4.1a shows the HGAFSA based optimization curve. It may be observed from Figure 4.1a that the algorithm converges at the thirteenth generation to a cost of **$24,141.2687/h.** This resulted in an annual savings of $3,253,957.

Figure 4.1b presents the cumulative power generated by the units. In economic load dispatch (ELD), power generated by each unit must lie within its maximum and the minimum allowable power output (Upper and Lower Limits). Therefore, the power allocations must lie within this limit. It is clear that, the optimum power allocated to each unit (P_i) lie within its limits. Finally, the cost functions of the various generating units influence the optimum power allocated to the units and thus, defines the pattern of the 'Optimum P_i' curves for each of the five-test systems in this work. In general, the peak of the cumulative power curves can be described as follows.

I. Optimum P_i Curve: - Defines the total power generated (Demand + Losses) by the system of generating units;
II. Upper Limit P_{max} Curve: - Defines the maximum power that can be generated by the system of generating units;

III. Lower Limit P_{min} Curve: - Defines the minimum power generated by the system of generating units.

In the case of Figure 4.1b, $P_i^{peak} = 2560 \cdot 4, P_{\min}^{peak} = 550$, and $P_{\max}^{peak} = 2960$

Table 4.2: Comparison of Results for 13-Units System

S/No.	HGAFSA	OGWO	GWO	OIWO	SDE	ORCCRO
1	628.320	628.294	628.1678	628.3185	628.32	628.32
2	299.200	299.1803	298.9229	299.1989	299.2	299.2
3	299.300	297.5041	298.2269	299.1991	299.2	299.2
4	159.730	159.7284	159.7232	159.7331	159.73	159.73
5	159.730	159.7325	159.721	159.7331	159.73	159.73
6	159.730	159.7295	159.727	159.7331	159.73	159.73
7	159.730	159.7334	159.7173	159.733	159.73	159.73
8	159.730	159.7323	159.6793	159.7331	159.73	159.73
9	159.730	159.7327	159.6673	159.733	77.4	77.4
10	77.394	77.3963	77.3971	77.3953	113.12	112.14
11	113.307	114.7487	114.6051	113.1079	92.4	92.4
12	92.1300	92.3974	92.3886	92.3594	92.4	92.4
13	92.2300	92.378	92.355	92.3911	92.4	92.4
Fuel Cost $/h	***24,141.2687***	***24,512.725***	***24,514.477***	***24514.83***	***24,514.9***	***24,513.91***
Power Loss MW	***40.2686***	***40.2874***	***40.2983***	***40.3686***	***40.43***	***39.43***

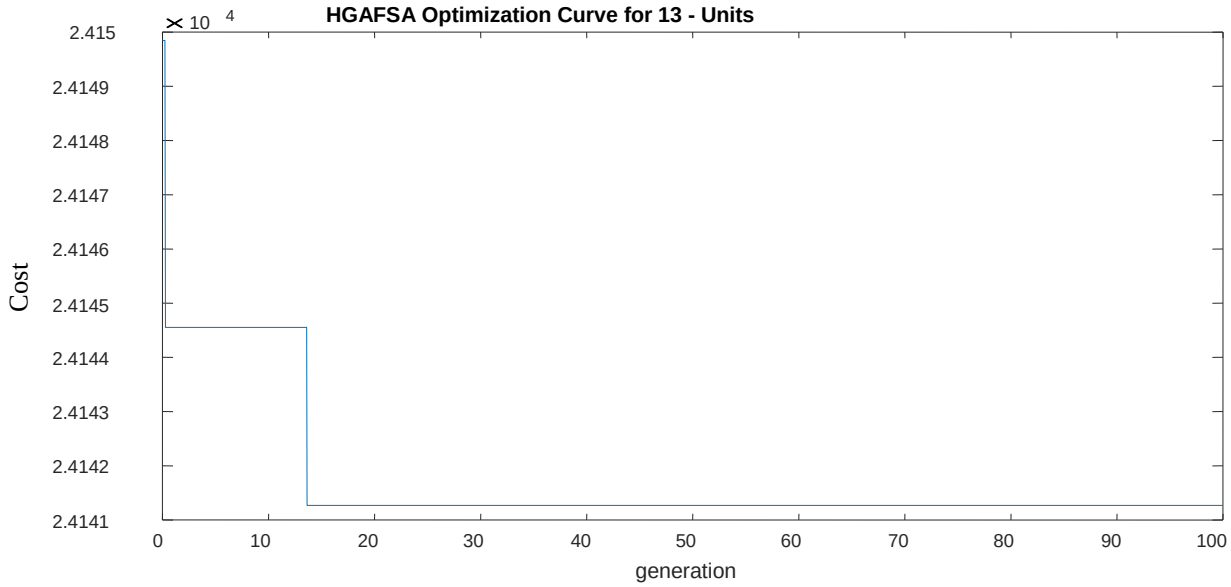


Figure 4.1a: 13-Units: (a) Optimization Curve

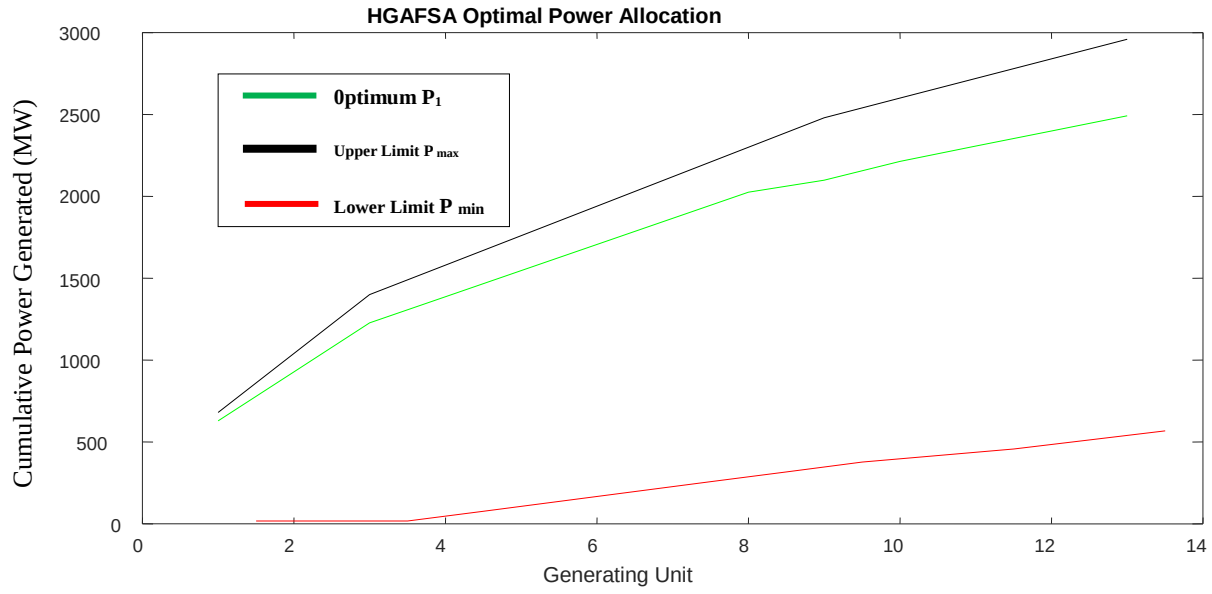


Figure 4.1b: Cumulative Power Generated

4.4 Test system 2

To compare the effectiveness of the OGWO (Pradhan et al., 2017) and HGAFSA approaches, a medium size ELD problem having 40 generating units with valve-point and multiple fuel effects are used. The input data for 40 generating Units system is taken from (Srinivasa Reddy & Vaisakh, 2013). The total power demand is taken as 10,500 MW. The real power generation output and the fuel cost obtained for 40 unit systems using various intelligent techniques like HGAFSA, GWO, OGWO (Pradhan et al., 2017), OIWO (Barisal & Prusty, 2015), Shuffled Differential Evolution (SDE) (Srinivasa Reddy & Vaisakh, 2013), ORCCRO (Bhattacharjee et al., 2014), GAAPI (Ciornei & Kyriakides, 2012), Quasi-Oppositional Teaching-Learning Based Optimization (QOTLBO) (Roy & Mandal, 2014) and Krill Herd Algorithm (KHA) (Roy & Mandal, 2014) are given in Table 4.3. The results show that the HGAFSA based approach

performed better than all other approaches presented in this work. Figure 4.2a shows the HGAFSA based optimization curve. It can be observed from Figure 4.2a that the algorithm converges at the fifty-first generation to a cost of **$136,396.9727/h.** This resulted in an annual savings of $382,350. Furthermore, Figure 4.2b shows that all generators satisfy their inequality constraint.

Table 4.3: Comparison of Results for 40-Units System

S/No.	HGAFSA	OGWO	GWO	OIWO	SDE	ORCCRO	GAAPI	QOTLBO	KHA
1	113.96	114	114	113.9908	110.06	111.68	114	114	114
2	113.69	114	114	114	112.41	112.16	114	114	114
3	120	120	120	119.9977	120	119.98	120	107.8221	120
4	179.74	183.5725	181.049	182.5131	188.72	182.18	190	190	190
5	96.975	87.8151	87.8351	88.4227	85.91	87.28	97	88.3702	88.5944
6	140	140	140	140	140	139.85	140	140	105.5166
7	300	300	300	299.9999	250.19	298.15	300	300	300
8	284.8	300	300	292.0654	290.68	286.89	300	300	300
9	289.02	300	300	299.8817	300	293.38	300	300	300
10	279.65	279.7201	279.9786	279.7073	282.01	279.34	205.25	211.2071	280.6777
11	168.81	243.617	243.6274	168.8149	180.82	162.35	226.3	317.2766	243.5399
12	94	94.1781	94.1436	94	168.74	94.12	204.72	163.7603	168.8017
13	484.04	484.27	484.4562	484.0758	469.96	486.44	346.48	481.5709	484.1198
14	484.05	484.3324	484.2306	484.0477	484.17	487.02	434.32	480.5462	484.1662
15	484.04	484.0484	484.2463	484.0396	487.73	483.39	431.34	483.7683	485.2375
16	484.08	484.0791	484.0333	484.0886	482.3	484.51	440.22	480.2998	485.0698
17	489.28	489.2147	489.6295	489.2813	499.64	494.22	500	489.2488	489.4539
18	489.3	489.2607	489.3228	489.2966	411.32	489.48	500	489.5524	489.3035
19	511.32	511.3341	511.4616	511.3219	510.47	512.2	550	512.5482	510.7127
20	511.33	511.4991	511.4932	511.335	542.04	513.13	550	514.2914	511.304
21	549.94	523.476	523.4767	549.9412	544.81	543.85	550	527.0877	524.4678
22	549.94	546.6445	547.6868	549.9999	550	548	550	530.1025	535.5799
23	523.3	523.3857	523.3738	523.2804	550	521.21	550	524.2912	523.3795
24	523.32	523.3344	523.135	523.3213	528.16	525.01	550	524.6512	523.15527
25	523.27	523.407	523.3472	523.5804	524.16	529.84	550	525.0586	524.1916
26	523.28	523.302	523.3578	523.5847	539.1	540.04	550	524.4654	523.5453
27	10.013	10.0076	10.0678	10.0086	10	12.59	11.44	10.8929	10.1245
28	10.007	10.0104	10.6337	10.0068	10.37	10.06	11.56	17.4312	10.1815
29	10.012	10.0622	10.5181	10.0123	10	10.79	11.42	12.7839	10.0229
30	96.966	87.8011	87.8029	87.8664	96.1	89.7	97	88.8119	87.8154
31	190	190	190	190	185.33	189.59	190	190	190
32	190	190	190	189.9983	189.54	189.96	190	190	190
33	190	190	190	190	189.96	187.61	190	190	190
34	199.99	200	200	199.994	199.9	198.91	200	200	200
35	200	200	200	200	196.25	199.98	200	168.0873	164.9199
36	169.2	164.8986	164.8334	164.8283	185.85	165.68	200	165.5072	164.9787
37	110	110	110	110	109.72	109.98	110	110	110
38	109.99	110	110	109.994	110	109.82	110	110	110
39	110	110	110	110	95.71	109.88	110	110	110
40	550	511.8527	511.5471	550	532.43	548.5	550	511.5313	512.06775
Fuel Cost	***136396.9727***	***136440.62***	***136446.85***	***136452.7***	***138157***	***136855.19***	***139865***	***137329.86***	***136670***

$/h									
Power Loss MW	*957.2965*	*973.1235*	*973.2874*	*957.2965*	*974.43*	*958.75*	*1045.06*	*1008.96*	*978.925*

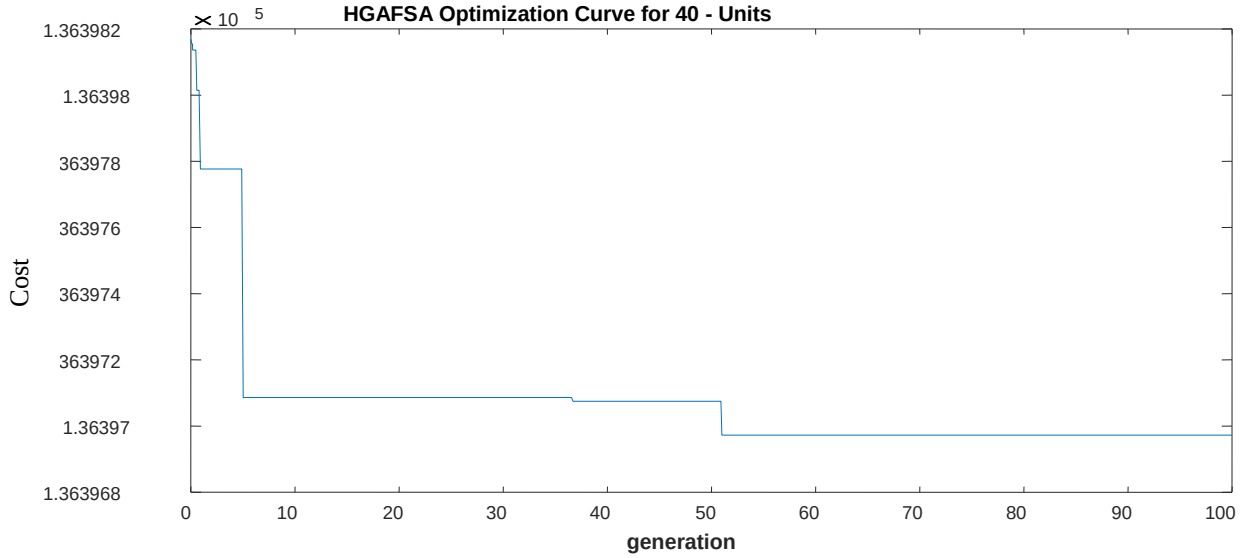


Figure 4.2a: 40-units: (a) Optimization Curve

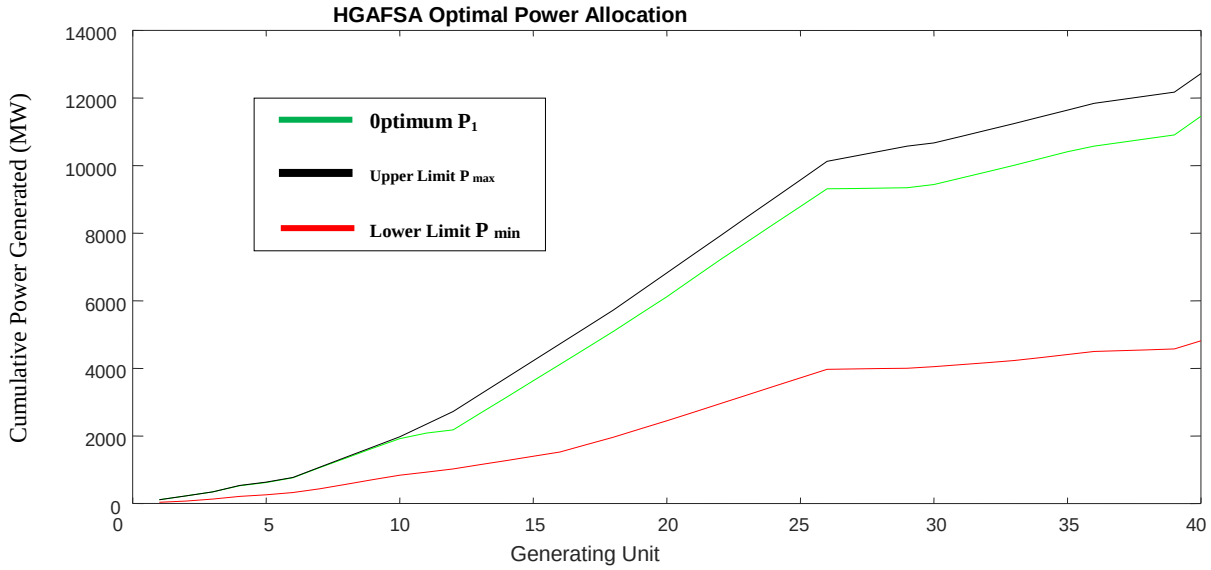


Figure 4.2b: Cumulative Power Generated

4.5 Test system 3

This system comprises 110 generators with quadratic cost characteristic. The input system data were obtained from (Orero, 1997). The load demand is considered as 15, 000MW.The minimum fuel costs and their corresponding generation levels obtained by proposed HGAFSA and other method are provided in Table 4.4. The minimum fuel cost obtained for this system by all the methods are also provided in Table 4.4. From Tables 4.4, it is clear that the proposed method provides cheapest generation schedule of power generation. The results show that the HGAFSA based approach outperformed all other approaches presented in the literature so far. Figure 4.3a shows the HGAFSA based optimization curve. It may be observed from Figure 4.3a that the algorithm was truncated at the hundredth generation and at a cost of **$197,988.892/h.** This resulted in an annual savings of $2,135.69. The saving in cost was less because of the narrow margins for some of the generating units especially units 1 to 9 and the high cost of generation. Furthermore, Figure 4.3b shows that all generators satisfy their inequality constraints.

Table 4.4: Comparison of Results for 110-Units System

S/No.	HGAFSA	OIWO		HGAFSA	OIWO		HGAFSA	OIWO
1	2.4	2.4	38	69.9975178	69.981	75	89.99982	89.999
2	2.4004	2.4004	39	99.9999501	99.994	76	49.999	49.999
3	2.4026	2.4026	40	120	120	77	160	160.01
4	2.4	2.4	41	157.18324	156.8	78	295.7606	291.36
5	2.4	2.4	42	220	220	79	175.0589	177
6	4.0011	4.0011	43	440	440	80	98.0084	97.753
7	4	4	44	560	560	81	10.001	10.001
8	4	4	45	660	660	82	12.0124	12.305
9	4	4	46	616.436395	619.53	83	20.0153	20.042
10	64.3953156	63.055	47	5.40003807	5.4004	84	199.9893	199.99
11	62.16223	59.275	48	5.4	5.4	85	324.9917	324.51
12	36.2917819	35.658	49	8.4015	8.4015	86	439.998	439.99
13	56.6260119	57.438	50	8.4	8.4	87	14.42951	18.867
14	25	25	51	8.4	8.4	88	24.32679	23.334
15	25	25	52	12	12	89	82.44312	84.403
16	25	25	53	12	12	90	89.25027	91.9
17	155	155	54	12.001	12.001	91	57.61189	58.29
18	155	155	55	12	12	92	99.99727	98.071
19	155	155	56	25.2	25.2	93	440	440
20	155	155	57	25.2	25.2	94	499.9977	499.97
21	68.9	68.9	58	35	35	95	600	600

22	68.9	68.9	59	35.0004463	35	96	471.479	469.27
23	68.9	68.9	60	45.001	45.001	97	3.6	3.6
24	350	350	61	45.001	45.001	98	3.6	3.6
25	400	400	62	45	45	99	4.4	4.4
26	400	400	63	184.998222	185	100	4.400235	4.4005
27	500	500	64	185	184.99	101	10.00088	10.008
28	500	500	65	185	185	102	10.001	10.001
29	200	199.99	66	184.993084	185	103	20.005	20.005
30	100	100	67	70	70	104	20.00009	20.005
31	10.0005576	10.001	68	70	70	105	40	40
32	19.9992817	19.99	69	70.001	70.001	106	40.001	40.002
33	79.9983153	79.485	70	359.998903	360	107	50	50
34	250	250	71	400	400	108	30	30
35	360	360	72	400	400	109	40	40
36	400	399.99	73	104.96292	107.83	110	20	20
37	39.999	39.999	74	191.49755	188.81	***Fuel Cost $/h***	***197,988.892***	***197,989.1358***

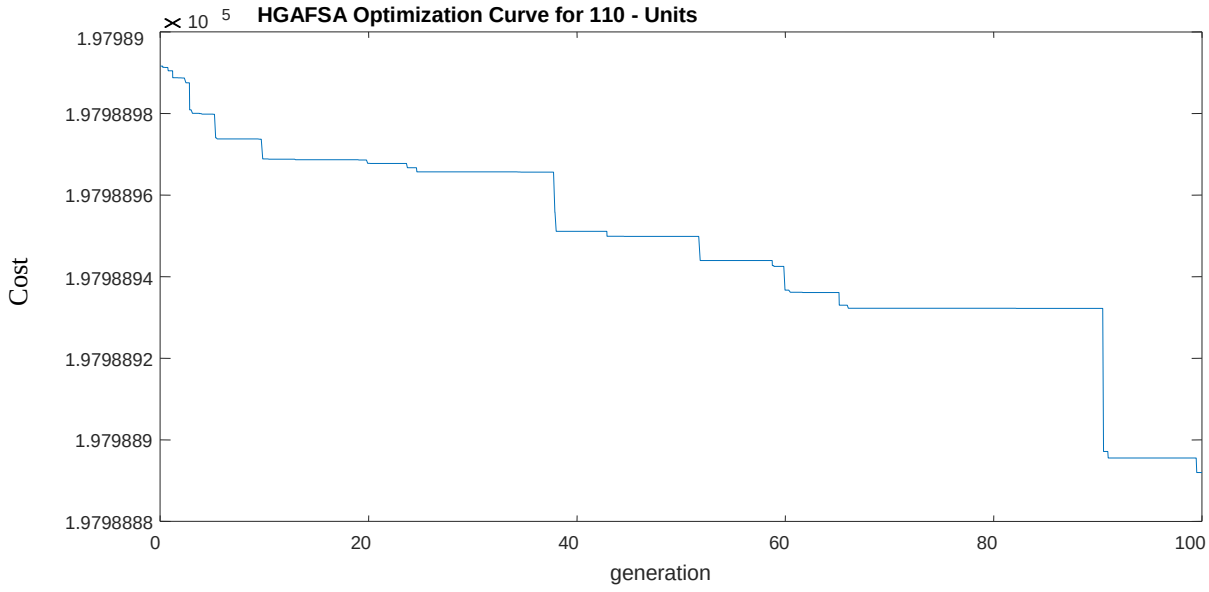


Figure 4.3a: 110-Units: (a) Optimization Curve

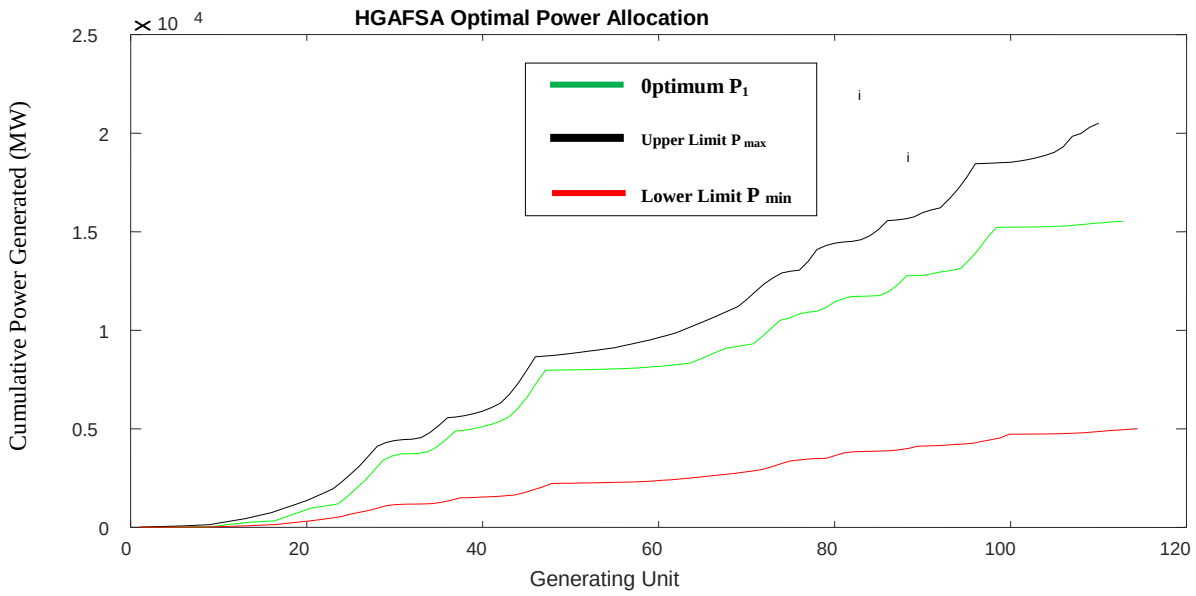


Figure 4.3b: Cumulative Power Generated

4.6 Test system 4

In this case study, a complicated 140-unit test system is considered to verify the effectiveness of the proposed HGAFSA method. The non-linear constraints like valve-point effect prohibited operating zone and ramp rate limits are considered in this case. The load demand is assumed to be 49,342MW. In order to validate the superiority of the proposed methods, the results obtained from the HGAFSA and OGWO are compared with those of GWO, SDE and OIWA reported in the literature. The lowest cost for each of the 50 different trials using HGAFSA, OGWO, GWO, SDE (Srinivasa Reddy & Vaisakh, 2013), and OIWO (Barisal & Prusty, 2015) methods are illustrated in Table 4.5 from which HGAFSA method produces lowest cost compared to the OGWO and the other methods. It clearly suggests that though the performances of both HGAFSA and OGWO are satisfactory but HGAFSA is significantly better than OGWO. Figure

4.4a shows the HGAFSA based optimization curve. It may be observed from Figure 4.4a that the algorithm converges at the sixty-sixth generation to a cost of **$1,558,619.094/h.** This resulted in an annual savings of $2,128,680. Furthermore, Figure 4.4b shows that all generators satisfy their inequality constraints.

Table 4.5: Comparison of Results for 140-Units System

S/No.	HGAFSA	OGWO	GWO		HGAFSA	OGWO	GWO		HGAFSA	OGWO	GWO
1	113.5	114.5	119	48	250	250	250	95	978	978	978
2	189	189	189	49	250	250	250	96	682	682	682
3	190	190	190	50	250	250	250	97	720	720	720
4	190	190	190	51	165	165	165	98	718	718	718
5	168.54	168.54	168.54	52	165	165	165	99	720	720	720
6	190	190	190	53	165	165	165	100	964	964	964
7	490	490	490	54	165	165	165	101	958	958	958
8	490	490	490	55	180	180	180	102	1007	1007	1007
9	496	496	496	56	180	180	180	103	1006	1006	1006
10	496	496	496	57	103	103	103	104	1013	1013	1013
11	496	496	496	58	198	198	198	105	1020	1020	1020
12	496	496	496	59	312	312	312	106	954	954	954
13	506	506	506	60	277.85	280.85	282.89	107	952	952	952
14	509	509	509	61	163	163	163	108	1006	1006	1006
15	506	506	506	62	95	95	95	109	1013	1013	1013
16	505	505	505	63	159.42	160	160.88	110	1021	1021	1021
17	506	506	506	64	160	160	160	111	1015	1015	1015
18	506	506	506	65	490	490	490	112	94	94	94
19	505	505	505	66	196	196	196.26	113	94	94	94
20	505	505	505	67	490	490	490	114	94	94	94
21	505	505	505	68	489	490	489.6	115	244	244	244
22	505	505	505	69	130	130	130	116	244	244	244
23	505	505	505	70	234.71	234.71	234.7	117	244	244	244
24	505	505	505	71	137	137	137	118	95	95	95
25	537	537	537	72	324.5	325.5	325.82	119	95	95	95
26	537	537	537	73	195	195	195	120	116	116	116
27	549	549	549	74	175.39	175.39	175.39	121	175	175	175
28	549	549	549	75	175	175	175	122	2	2	2
29	501	501	501	76	175.99	175.99	175.99	123	4	4	4
30	501	501	501	77	175.41	175.41	175.41	124	15	15	15
31	506	506	506	78	330	330	330	125	9	9	9
32	506	506	506	79	531	531	531	126	12	12	12
33	506	506	506	80	531	531	531	127	10	10	10
34	506	506	506	81	395.1353	398.38	366.4	128	112	112	112
35	500	500	500	82	56	56	56	129	4	4	4
36	500	500	500	83	115	115	115	130	5	5	5
37	241	241	241	84	115	115	115	131	5	5	5
38	241	241	241	85	115	115	115	132	50	50	50
39	774	774	774	86	207	207	207	133	5	5	5
40	769	769	769	87	207	207	207	134	42	42	42
41	3	3	3	88	175	175	175	135	42	42	42
42	3	3	3	89	175	175	175	136	41	41	41
43	250	250	250	90	175	175	175	137	17	17	17
44	246.2765	246.39	250	91	175	175	175	138	15.35825	17	17

45	250	250	250	92	580	580	580	139	7	7	7
46	250	250	250	93	645	645	645	140	26	26	26.13
47	240.92	240.92	249.98	94	984	984	984	***Fuel Cost $/h***	***1558619.0935***	***1559710***	***1559953***

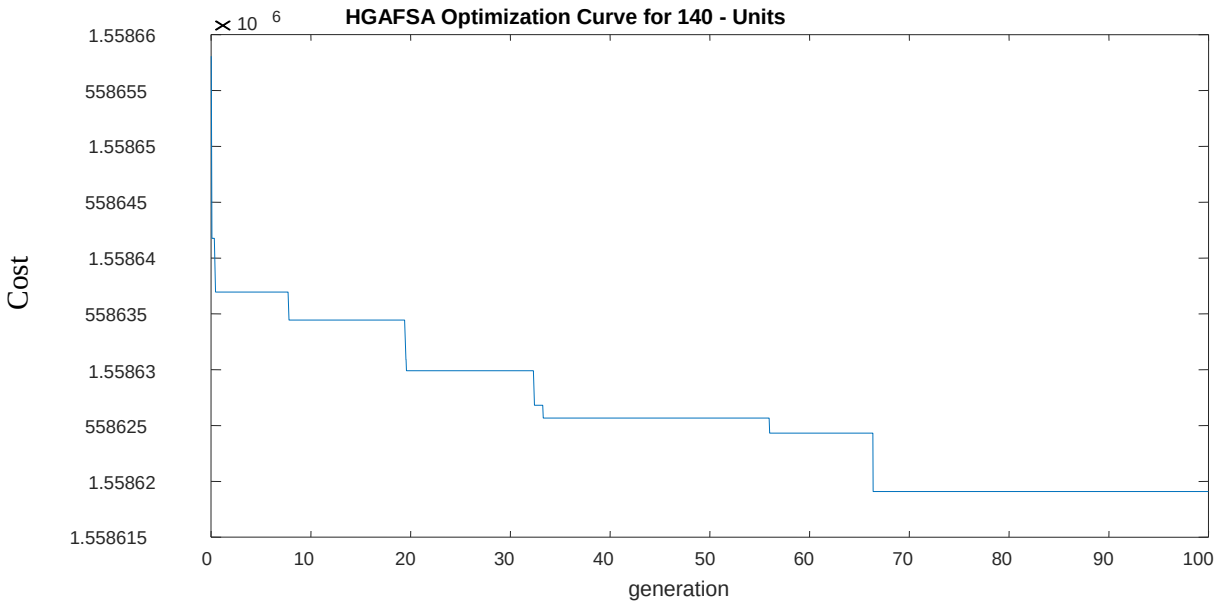


Figure 4.4a: 140-Units: (a) Optimization Curve

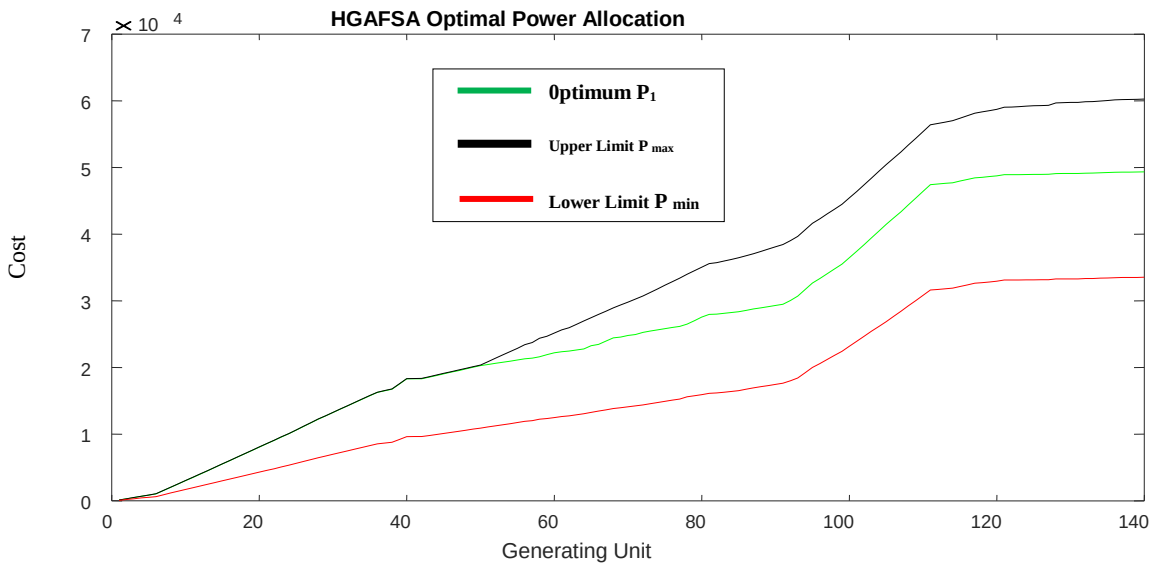


Figure 4.4b: Cumulative Power Generated

4.7 Test system 5

In this case, a 160-unit test system with non-smooth valve point and multiple fuel effects cost function are solved by the proposed GWO and HGAFSA methods. Transmission loss is ignored to verify the feasibility and effectiveness of the proposed algorithms for solving large scale ELD based power system problem. For this 160-units system, the 10-units multiple fuel system data taken from (Chiang, 2005) are duplicated and the demand is multiplied by 16 (i.e. load demand is taken as 43,200 MW). Table 4.6 presents the best cost and generation achieved by the different algorithms for the 160-unit system while satisfying the constraints. From Table 4.6, it can be inferred that the total minimum production cost obtained using HGAFSA technique is comparatively smaller with respect to other algorithms already mentioned in this work. Figure 4.5a shows the HGAFSA based optimization curve. It may be observed from Figure 4.5a that the algorithm converges at the eighth generation to a cost of **$9,612.8295/h.** This resulted in an

annual savings of $1,158,803.46. Furthermore, Figure 4.5b shows that all generators satisfy their inequality constraints.

Table 4.6: Comparison of Results for 160-Units System

S/No.	HGAFSA	OGWO	GWO		HGAFSA	OGWO	GWO		HGAFSA	OGWO	GWO
1	214.486	211.0548	224.4316	55	269.1426	207.4412	265.2612	109	438.9043	402.6279	430.8317
2	211.7117	208.8475	200.24	56	240.9831	254.7693	257.0533	110	275.8686	272.9789	260.2696
3	274.6078	335.1164	355.8632	57	290.0985	294.1646	277.1283	111	214.486	199.7646	217.9593
4	241.2519	243.6651	228.6577	58	240.8488	245.6809	235.9306	112	211.9596	204.8838	199.0124
5	269.1424	266.496	305.8538	59	439.0723	420.2064	394.7028	113	274.6078	351.507	357.91
6	240.58	237.7474	249.9548	60	275.8686	270.5752	278.4325	114	241.2519	249.6972	251.1044
7	290.0985	282.9036	309.7896	61	214.486	198.8766	240.6965	115	272.739	266.0942	267.0206
8	241.2519	239.4979	218.8214	62	211.4642	212.9224	213.3251	116	241.1175	240.0271	252.1589
9	439.0937	408.8918	335.8343	63	273.5996	348.4807	341.0975	117	290.0985	290.287	290.7761
10	275.8686	265.7794	270.5537	64	240.9831	259.8417	248.8005	118	241.2519	242.3206	233.9515
11	215.513	225.2013	187.6995	65	272.7397	292.9373	232.9769	119	439.0246	412.5941	329.9101
12	211.7117	217.9922	195.5662	66	241.2519	221.2448	245.9912	120	275.8686	242.8243	300.1967
13	273.5996	336.6158	353.7443	67	292.4695	286.6376	310.0164	121	215.513	211.0182	237.5402
14	241.3863	232.3276	241.0525	68	241.5206	242.6468	232.721	122	211.7117	210.7714	207.3365
15	269.1425	259.6177	273.3024	69	439.0756	348.3301	353.3462	123	274.6078	345.7508	337.0684
16	240.9831	237.6971	228.1355	70	275.8686	288.0821	281.0286	124	241.1175	227.1918	210.5269
17	292.4695	265.8324	277.3736	71	214.486	227.9132	219.8481	125	269.1424	273.9219	241.1325
18	240.9831	236.9454	224.3982	72	211.9593	215.0451	220.3933	126	241.5206	250.0678	246.6614
19	439.2471	414.7889	404.3422	73	273.5996	339.1038	343.6976	127	290.0985	258.3325	293.9444
20	275.8686	272.9817	314.3855	74	241.3862	244.8685	250.5153	128	240.9831	243.0127	245.9554
21	214.486	222.199	205.5031	75	269.1392	280.3173	273.4252	129	439.185	382.982	431.4487
22	211.9593	204.6515	200.7397	76	240.58	231.957	217.8772	130	275.8686	296.9508	248.3855
23	274.6078	333.0013	374.811	77	292.4695	286.6036	305.1259	131	214.486	200.3463	199.4532
24	241.2519	237.4739	234.0215	78	241.7894	225.9269	243.7327	132	211.7117	214.629	217.2444
25	272.7398	304.7814	284.8751	79	439.0763	382.9488	353.5211	133	274.6148	334.6046	342.2486
26	240.7144	245.5394	221.658	80	275.8686	265.3158	271.8637	134	241.5206	234.0886	229.6668
27	292.4751	265.8384	253.2492	81	215.513	223.9674	223.3807	135	269.1468	286.6132	250.7479
28	241.2519	237.482	223.1936	82	212.2069	200.5557	191.4945	136	241.7894	247.7709	236.0422
29	439.013	381.7295	421.8011	83	273.5996	334.5932	349.8148	137	290.0985	290.5292	305.7284
30	272.7219	267.3033	280.1477	84	241.1175	246.8674	234.8348	138	241.5206	229.8027	230.9169
31	214.486	223.8357	175.7028	85	269.1424	274.2033	275.2954	139	439.1252	391.0035	430.433
32	211.9593	214.3587	203.4868	86	241.1175	249.3916	208.8313	140	275.8686	270.6119	257.5185
33	274.6078	334.3914	348.5865	87	292.4695	297.4366	282.9046	141	214.486	212.7407	228.4485
34	241.5206	247.1993	219.6817	88	240.58	240.7596	255.2069	142	211.2166	226.1497	184.2037
35	269.1425	250.1068	283.914	89	439.0426	340.9873	409.1432	143	274.6078	335.9897	370.6009
36	241.5206	229.7748	244.5662	90	275.8686	291.238	269.0939	144	240.9831	245.3932	219.2088
37	292.4695	265.8045	313.7701	91	214.486	212.3306	192.5265	145	269.1424	261.9323	263.8981
38	240.7144	243.4346	236.54	92	210.969	210.493	201.0805	146	241.9237	244.4414	235.3311
39	439.0891	413.3884	340.5099	93	273.5995	337.77	350.8514	147	290.0985	294.0252	300.9854
40	275.8686	277.7058	333.2419	94	242.0581	236.8209	258.9069	148	241.3863	235.734	242.2109
41	214.486	202.7653	230.1267	95	269.1424	261.4802	290.5386	149	439.0389	390.4426	387.3228
42	211.7117	228.5107	209.9439	96	240.58	235.3425	226.8687	150	275.8686	253.1512	267.7898
43	273.5996	352.7008	341.696	97	292.4695	278.8739	299.7902	151	214.486	193.0014	184.8211
44	241.2519	227.3366	233.436	98	241.3862	237.8441	223.7707	152	212.2078	222.8983	212.4468
45	269.1424	272.1149	299.6092	99	439.1089	420.0762	366.8371	153	274.6078	355.9969	376.6391
46	240.7144	241.8475	263.4945	100	275.8684	268.9686	288.8294	154	240.58	242.056	248.5759
47	287.7275	266.7677	243.0096	101	214.486	224.3774	227.0085	155	269.1424	277.6386	278.2433
48	241.1175	230.7801	212.4306	102	211.9593	209.2698	210.8885	156	241.2519	246.0075	223.6107
49	439.0932	423.1378	362.973	103	273.5996	337.6392	363.4203	157	292.4695	318.3527	264.0864
50	275.8686	254.0386	303.2805	104	241.5206	230.3857	222.0259	158	241.1175	241.0144	234.1904
51	215.513	218.3367	181.5271	105	269.1424	269.3592	234.4937	159	439.097	341.9573	390.9912

52	211.2166	211.9849	211.2522	106	240.7144	235.2646	232.5588	160	275.8686	261.077	286.3951
53	274.6078	337.9485	364.7108	107	290.0985	274.0838	282.2891	***Fuel Cost $/h***	**9,612.8295**	***9,745.113***	***9,813.377***
54	241.7894	238.8923	234.0012	108	242.4612	244.0134	236.214				

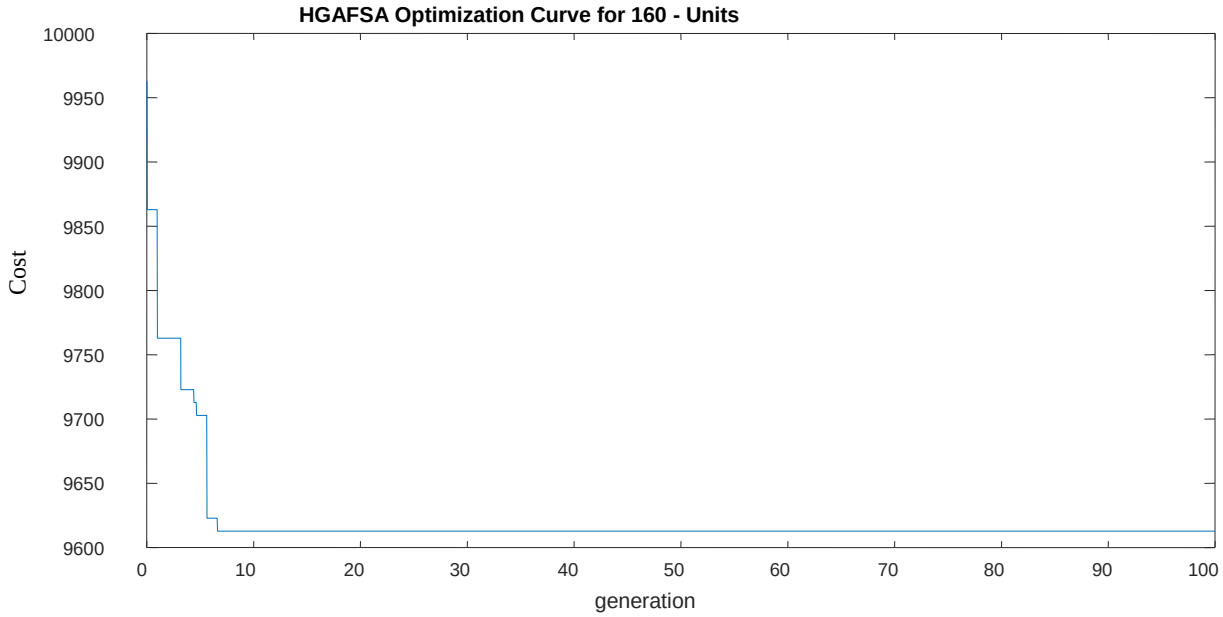


Figure 4.5a: 160-Units: (a) Optimization Curve

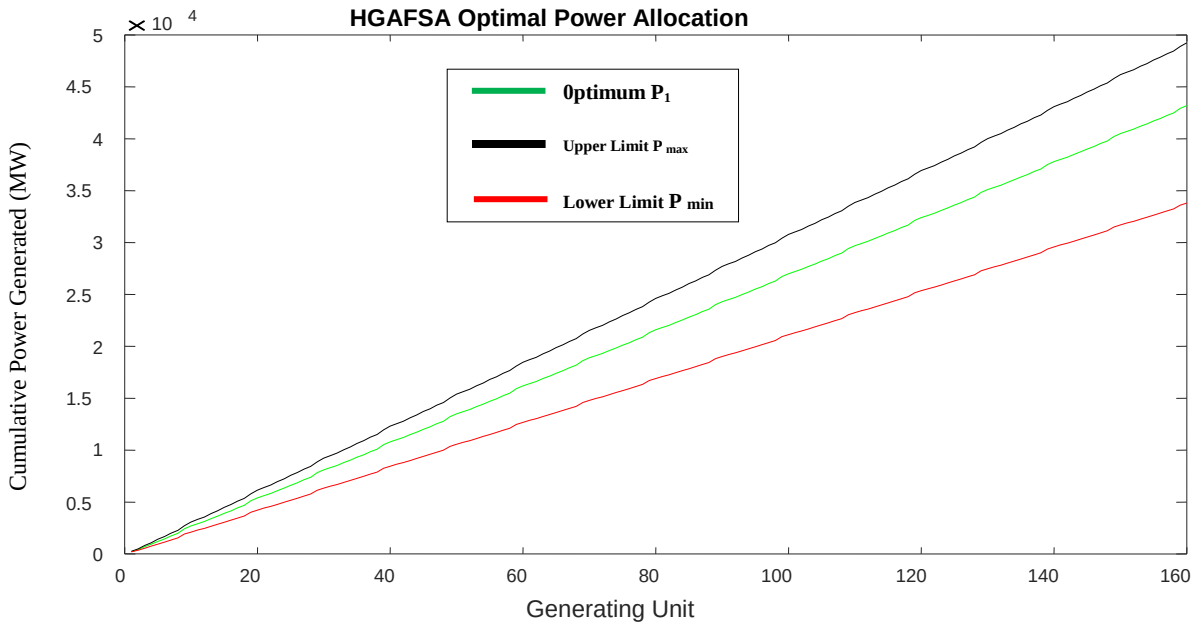


Figure 4.5b: Cumulative Power Generated

In General, the performance of HGAFSA over the best algorithms presented in literature so far and listed in this work can be further summarized as follows. The best cost so far for all the systems in Table 4.7 are obtained by OGWO except for the 110-Units system which was obtained using OIWO. In all the simulated cases, HGAFSA has demonstrated superiority and has yielded a considerable huge amount of annual savings in fuel cost. Figure 4.6 also show the bar charts of annual savings in fuel cost.

Table 4.7: Summary of Results for the Comparison of HGAFSA and OGWO/OIWO

System	Number of Units	HGAFSA ($)	OGWO Best Cost so far ($)	Annual Savings ($)	Total Power	OGWO/OIWO Iteration Time (s)	HGAFSA Iteration Time (s)
1	13	24,141.2687	24,512.725	3,253,957.2	2,560.3686	5.16	5.023
2	40	136,396.9727	136,440.62	382,350.35	11,457.297	10.23	10.113
3	110	197,988.892	197,989.1358	2,135.688	15,000	31	104.3
4	140	1,558,619.094	1,559,710	9,556,340.9	49,342	41.77	41.123
5	160	9,612.8295	9,745.113	1,158,803.5	43,200	16.32	10.234

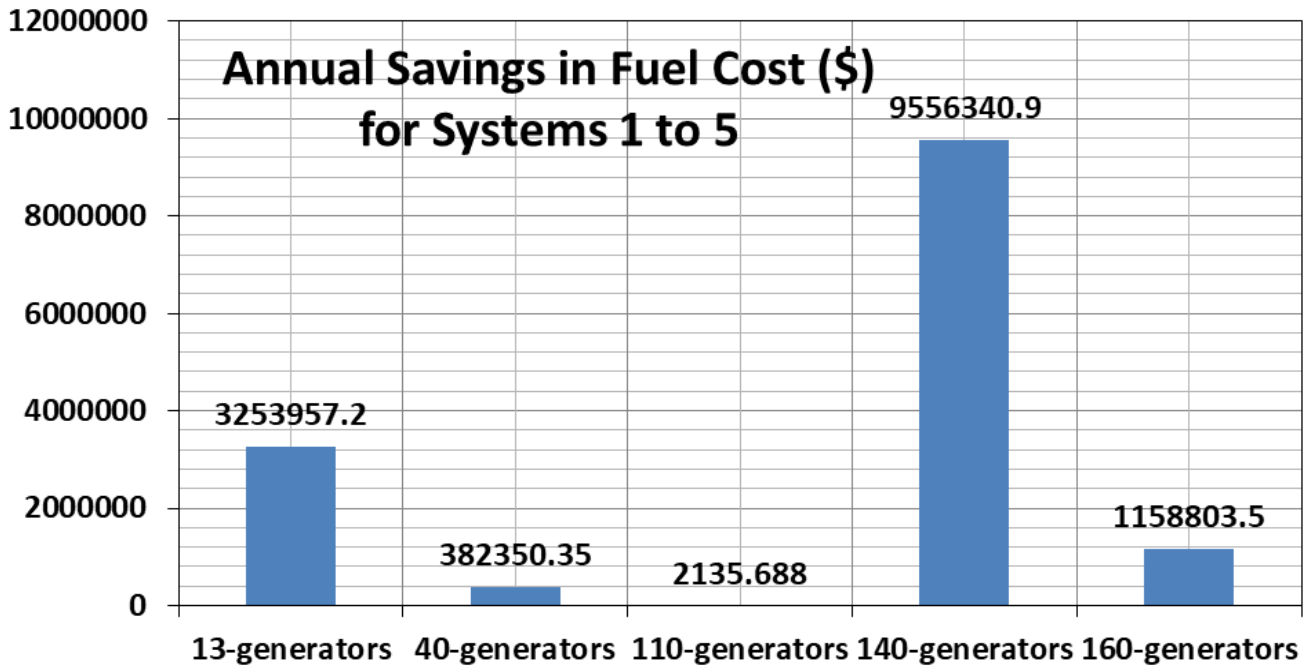


Figure 4.6: Annual Savings in Fuel Cost for Systems 1 to 5

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

Introduction

This chapter presents the general conclusion drawn from the simulation carried out in this work, and itemize a set of recommendations that may be useful in future works regarding ELD problem and its solution strategy.

5.1 Conclusion

In this work, a hybrid Genetic Algorithm and Artificial Fish Swarm Algorithm namely HGAFSA capable of solving a higher order economic load dispatch (ELD) has been developed and used to solve several higher order ELD problems including 13, 40, 110, 140 and 160-generating-Units system. An ELD Encoder algorithm that can be linked to any heuristic algorithm to solve ELD problem has been developed and linked to the developed HGAFSA to form a HGAFSA based ELD algorithm (HGAFSAELDA) that can minimize any ELD cost function lower than every algorithm available in the literature so far. The performance of the developed HGAFSAELDA has been demonstrated over five standard IEEE test systems. Reduction in fuel cost of 1.53%, 0.03%, 0.07%, 0.00012% and 1.37% were recorded on the 13, 40, 110, 140 and 160- generating-units. Annual savings in fuel cost of $3.254e+06, $3.8235e+05, $2135.7, $9.5563e+06, and $1.1588e+06 for the 13, 40, 110, 140, and 160-generating-units respectively were achieved over the existing best costs presented in the (Pradhan *et al*., 2017). It can be generalized that this developed HGAFSAELDA can go a long way in providing solution to any given ELD problem. The simulation environment used was MATLAB R2016a Software running on a Windows 10.1 based 8GB-RAM, 2.3GHz-Core-I3 HP computer.

5.2 Significant Contributions

The major contributions of this work to the existing body of knowledge are summarized in the following.

1. A hybrid Genetic Algorithm and Artificial Fish Swarm Algorithm namely HGAFSA capable of solving a higher order economic load dispatch (ELD) have been developed and used to solve several higher order ELD problems including 13, 40, 110, 140 and 160-generating-Units system.
2. An ELD Encoder algorithm that can be linked to any heuristic algorithm to solve ELD problem has been developed and linked to the developed HGAFSA to form a HGAFSA based ELD algorithm (HGAFSAELDA) that can minimize any ELD cost function lower than every algorithm available in the literature of this work.
3. The performance of the developed HGAFSAELDA has been demonstrated over five ELD systems. Reduction in fuel cost of 1.53%, 0.03%, 0.07%, 0.00012% and 1.37% were recorded on the 13, 40, 110, 140 and 160- generating- units. An annual savings in fuel cost of $3.254e+06, $3.8235e+05, $2135.7, $9.5563e+06, and $1.1588e+06 for the 13, 40, 110, 140, and 160-generating-units respectively were achieved over the existing best costs presented in (Pradhan *et al*., 2017).

5.3 Recommendations

In the case where a researcher wishes to work on ELD problem solving strategies. The following are some suggestions regarding possible research direction.

1. Implementation of the developed Algorithm on a Digital Signal Processor (DSP) chip and testing it using a real power system to ascertain its effectiveness;
2. Development of the proposed algorithm in a suitable java based software package for use by students in their respective course works;
3. Linking the developed ELD encoder with other heuristics and testing their performance compared to the proposed HGAFSA.

5.4 Limitation

The limitation encountered in this work is summarized in the following.

1. The error due to approximation affected the results presented in some test system cases especially the 110-Unit case.

REFERENCES

Al-Othman, A. K., & El-Naggar, K. M. (2008). Application of pattern search method to power system security constrained economic dispatch with non-smooth cost function. *Electric Power Systems Research*, *78*(4), 667–675. https://doi.org/10.1016/j.epsr.2007.05.015

Alsumait, J. S., Sykulski, J. K., & Al-Othman, A. K. (2010). A hybrid GA–PS–SQP method to solve power system valve-point economic dispatch problems. *Applied Energy*, *87*(5), 1773–1781. https://doi.org/10.1016/j.apenergy.2009.10.007

Amir Ghoreishi, S., Ali Nekoui, M., & Omid Basiri, S. (2011). Optimal Design of LQR Weighting Matrices based on Intelligent Optimization Methods. *International Journal of Intelligent Information Processing*, *2*(1). https://doi.org/10.4156/ijiip.vol2.issue1.7

Aref Jalili, Alireza Noruzi, Mohammad Yazdani, and M. M. (2015). *Solving Economic Load Dispatch with Valve Point Effect Based On Firefly Algorithm*.

Atwa, Y., El-Saadany, E., & Salama, M. (2010). Optimal renewable resources mix for distribution system energy loss minimization. *On Power Systems*. Retrieved from http://ieeexplore.ieee.org/abstract/document/5298970/

Azad, M. A. K., Rocha, A. M. A. C., & Fernandes, E. M. G. P. (2014). Improved binary artificial fish swarm algorithm for the 0–1 multidimensional knapsack problems. *Swarm and Evolutionary Computation*, *14*, 66–75. https://doi.org/10.1016/j.swevo.2013.09.002

Balamurugan, R., & Subramanian, S. (2008). An improved dynamic programming approach to economic power dispatch with generator constraints and transmission losses. *Journal of Electrical Engineering &*. Retrieved from

http://www.dbpia.co.kr/Journal/ArticleDetail/NODE01311492

Barisal, A. K., & Prusty, R. C. (2015). Large scale economic dispatch of power systems using oppositional invasive weed optimization. *Applied Soft Computing*, *29*, 122–137. https://doi.org/10.1016/j.asoc.2014.12.014

Bernardon, D. P., Mello, A. P. C., Pfitscher, L. L., Canha, L. N., Abaide, A. R., & Ferreira, A. A. B. (2014). Real-time reconfiguration of distribution network with distributed generation. *Electric Power Systems Research*, *107*, 59–67. https://doi.org/10.1016/j.epsr.2013.09.011

Bhattacharjee, K., Bhattacharya, A., & Dey, S. H. nee. (2014). Oppositional Real Coded Chemical Reaction Optimization for different economic dispatch problems. *International Journal of Electrical Power & Energy Systems*, *55*, 378–391. https://doi.org/10.1016/j.ijepes.2013.09.033

Bhattacharya, A., & Chattopadhyay, P. K. (2010). Biogeography-Based Optimization for Different Economic Load Dispatch Problems. *IEEE Transactions on Power Systems*, *25*(2), 1064–1077. https://doi.org/10.1109/TPWRS.2009.2034525

Bouktir, T., & Slimani, L. (2005). Object-oriented economic power dispatch of electrical power system with minimum pollution using a genetic algorithm. *J. Electrical Systems*. Retrieved from http://journal.esrgroups.org/jes/papers/1_2_2.pdf

Burns, R., & Gibson, C. (1975). Optimization of priority lists for a unit commitment program. *IEEE*. Retrieved from https://scholar.google.com/scholar?hl=en&q=Optimization+of+Priority+Lists+for+a+Unit+Commitment+Program'&btnG=&as_sdt=1%2C5&as_sdtp=

Chatterjee, S., Laudato, M., & Lynch, L. A. (1996). Genetic algorithms and their statistical applications: an introduction. *Computational Statistics and Data Analysis*, *22*(6), 633–654. https://doi.org/10.1016/0167-9473(96)00011-4

Chiang, C.-L. (2005). Improved Genetic Algorithm for Power Economic Dispatch of Units With Valve-Point Effects and Multiple Fuels. *IEEE Transactions on Power Systems*, *20*(4), 1690–1699. https://doi.org/10.1109/TPWRS.2005.857924

Ciornei, I., & Kyriakides, E. (2012). A GA-API Solution for the Economic Dispatch of Generation in Power System Operation. *IEEE Transactions on Power Systems*, *27*(1), 233–242. https://doi.org/10.1109/TPWRS.2011.2168833

Costa, M. F. P., Rocha, A. M. A. C., & Fernandes, E. M. G. P. (2014). An artificial fish swarm algorithm based hyperbolic augmented Lagrangian method. *Journal of Computational and Applied Mathematics*, *259*, 868–876. https://doi.org/10.1016/j.cam.2013.08.017

Dewangan, S. K., Jain, A., & Huddar, A. P. (2015). A Traditional Approach to Solve Economic Load Dispatch Problem Considering the Generator Constraints. *IOSR Journal of Electrical and Electronics Engineering Ver. III*, *10*(2), 2278–1676. https://doi.org/10.9790/1676-10232732

Dhebar, Y., & Deb, K. (2017). A computationally fast multimodal optimization with push enabled genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion on - GECCO '17* (pp. 191–192). New York, New York, USA: ACM Press. https://doi.org/10.1145/3067695.3076023

dosSantosCoelho, L., & Mariani, V. C. (2006). Combining of Chaotic Differential Evolution and

Quadratic Programming for Economic Dispatch Optimization With Valve-Point Effect. *IEEE Transactions on Power Systems*, *21*(2), 989–996. https://doi.org/10.1109/TPWRS.2006.873410

Elsayed, W. T., Hegazy, Y. G., Bendary, F. M., & El-bages, M. S. (2016). Modified social spider algorithm for solving the economic dispatch problem. *Engineering Science and Technology, an International Journal*, *19*(4), 1672–1681. https://doi.org/10.1016/j.jestch.2016.09.002

Fang, N., Zhou, J., Zhang, R., Liu, Y., & Zhang, Y. (2014). A hybrid of real coded genetic algorithm and artificial fish swarm algorithm for short-term optimal hydrothermal scheduling. *International Journal of Electrical Power & Energy Systems*, *62*, 617–629. https://doi.org/10.1016/j.ijepes.2014.05.017

Gargeya, M., & Pabba, S. (2013). Economic load dispatch using genetic algorithm and pattern search methods. *International Journal of*. Retrieved from https://pdfs.semanticscholar.org/8694/29bc9b0769cb348f75e65d824cc7b37e79b8.pdf

Gupta, S. ., & Chawala, P. (2015). Economic Load Dispatch in Thermal Power Plant Considering Additional Constraints Using Curve Fitting and Ann. *Review of Energy Technologies and Policy Research*, *2*(1), 16–28. https://doi.org/10.18488/journal.77/2015.2.1/77.1.16.28

Holland, J. (1975). Adaptation in natural and artificial systems. An introductory analysis with application to biology, control, and artificial intelligence. *Ann Arbor, MI: University of Michigan Press*.

Huang, G., Lu, Q., & Zheng, Y. (2006). An optimization approach to the location of oilfield

multistage stations by fish-swarm algorithm. *XIAN SHIYOU DAXUE XUEBAO ZIRAN KEXUE BAN*.

Jubril, A., & Komolafe, O. (2013). Solving multi-objective economic dispatch problem via semidefinite programming. *IEEE Transactions on*. Retrieved from http://ieeexplore.ieee.org/abstract/document/6466418/

Kaur, H., Singh, M., & Farwaha, H. (2015). ECONOMIC LOAD DISPATCH CONTAINING LOSS COEFFICIENT. Retrieved from https://www.irjet.net/archives/V2/i5/IRJET-V2I583.pdf

Kaur, H., Singh, M., & Singh Farwaha, H. (2015). ECONOMIC LOAD DISPATCH CONTAINING LOSS COEFFICIENT. *International Research Journal of Engineering and Technology*, 2395–56. Retrieved from https://www.irjet.net/archives/V2/i5/IRJET-V2I583.pdf

Khoa, T., Vasant, P., & Singh, B. (2015). Solving Economic Dispatch By Using Swarm Based Mean-Variance Mapping Optimization (MVMOS). *Global J*. Retrieved from https://pdfs.semanticscholar.org/1688/cb604252fcc276b4ac70e0ebb6be6546a121.pdf

Kumar Sharma, V., Kasniya, B., Research Scholar, P., & Professor, A. (2015). Economic Load Dispatch Using Fuzzy Logic Controlled Genetic Algorithms. *International Journal of Scientific & Engineering Research*, *6*(10). Retrieved from http://www.ijser.org

Li, M., Huang, X., Liu, H., Liu, B., Wu, Y., Xiong, A., & Dong, T. (2013). Prediction of gas solubility in polymers by back propagation artificial neural network based on self-adaptive particle swarm optimization algorithm and chaos theory. *Fluid Phase Equilibria*. Retrieved

from http://www.sciencedirect.com/science/article/pii/S0378381213003567

López-Lezama, J. M., Contreras, J., & Padilha-Feltrin, A. (2012). Location and contract pricing of distributed generation using a genetic algorithm. *International Journal of Electrical Power & Energy Systems*, *36*(1), 117–126. https://doi.org/10.1016/j.ijepes.2011.10.032

Mohammadi-Ivatloo, B., & Rabiee, A. (2013). Nonconvex dynamic economic power dispatch problems solution using hybrid immune-genetic algorithm. *IEEE Systems*. Retrieved from http://ieeexplore.ieee.org/abstract/document/6519277/

Moradi, M. H., & Abedinie, M. (2010). A combination of Genetic Algorithm and Particle Swarm Optimization for optimal DG location and sizing in distribution systems. In *2010 Conference Proceedings IPEC* (pp. 858–862). IEEE. https://doi.org/10.1109/IPECON.2010.5697086

N, A. (2015). Unit Commitment and Economic Load Dispatch Using Hybrid Genetic - Particle Swarm Optimization Algorithm. *International Journal of Applied Research*, *1*(6), 109–113. Retrieved from http://www.allresearchjournal.com/archives/?year=2015&vol=1&issue=6&part=C&ArticleId=213

Orero, S. (1997). Large scale unit commitment using a hybrid genetic algorithm. *International Journal of Electrical Power & Energy Systems*, *19*(1), 45–55. https://doi.org/10.1016/S0142-0615(96)00028-2

Pothiya, S., Ngamroo, I., & Kongprawechnon, W. (2008). Application of multiple tabu search algorithm to solve dynamic economic dispatch considering generator constraints. *Energy*

Conversion and Management, *49*(4), 506–516. https://doi.org/10.1016/j.enconman.2007.08.012

Pradhan, M., Roy, P. K., & Pal, T. (2017). Oppositional based grey wolf optimization algorithm for economic dispatch problem of power system. *Ain Shams Engineering Journal*. https://doi.org/10.1016/j.asej.2016.08.023

Rahmat Azami, M. S. J. and G. H. (2011). Economic load Dispatch and DC-Optimal Power Flow Problem-PSO versus LR. *INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY SCIENCES AND ENGINEERING*, *2*(9). Retrieved from http://www.ijmse.org/Volume2/Issue9/paper2.pdf

Rao, R. S., Ravindra, K., Satish, K., & Narasimham, S. V. L. (2013). Power Loss Minimization in Distribution System Using Network Reconfiguration in the Presence of Distributed Generation. *IEEE Transactions on Power Systems*, *28*(1), 317–325. https://doi.org/10.1109/TPWRS.2012.2197227

Roy, P. K., & Mandal, D. (2014). Oppositional biogeography-based optimisation for optimal power flow. *International Journal of Power and Energy Conversion*, *5*(1), 47. https://doi.org/10.1504/IJPEC.2014.059983

Sharma, S., Mehta, S., & Chopra, N. (2015). Economic Load Dispatch Using Grey Wolf Optimization. *Journal of Engineering Research and Applications Www.ijera.com ISSN*, *5*(46). Retrieved from www.ijera.com

Sinha, N., & Chakrabarti, R. (2003). Evolutionary programming techniques for economic load dispatch. *IEEE Transactions on*. Retrieved from

http://ieeexplore.ieee.org/abstract/document/1179910/

Srinivasa Reddy, A., & Vaisakh, K. (2013). Shuffled differential evolution for large scale economic dispatch. *Electric Power Systems Research*, *96*, 237–245. https://doi.org/10.1016/j.epsr.2012.11.010

Subramanian, R., Thanushkodi, K., & Neelakantan, P. (n.d.). A NUMERICAL SOLUTION TO ECONOMIC LOAD DISPATCH PROBLEMS. *Scientificbulletin.upb.ro*. Retrieved from http://www.scientificbulletin.upb.ro/rev_docs_arhiva/full8f2_813967.pdf

Sun, J., Palade, V., Wu, X.-J., Fang, W., & Wang, Z. (2014). Solving the Power Economic Dispatch Problem With Generator Constraints by Random Drift Particle Swarm Optimization. *IEEE Transactions on Industrial Informatics*, *10*(1), 222–232. https://doi.org/10.1109/TII.2013.2267392

Surekha, P., & Sumathi, D. (2012). Solving economic load dispatch problems using differential evolution with opposition based learning. *Wseas Transactions On*. Retrieved from https://pdfs.semanticscholar.org/eb97/7a97770a65ab35bf0960778b4d698f64c910.pdf

Thitithamrongchai, C., & Eua-Arporn, B. (2007). Self-adaptive differential evolution based optimal power flow for units with non-smooth fuel cost functions. *Journal of Electrical Systems*. Retrieved from https://scholar.google.com/scholar?hl=en&q=Self-adaptive+Differential+Evolution+Based+Optimal+Power+Flow+for+Units+with+Non-smooth+Fuel&btnG=&as_sdt=1%2C5&as_sdtp=

Vanita, M., & Thanushkodi, K. (2011). Solution to Economic Dispatch Problem by Differential Evolution Algorithm Considering Linear Equality andInequality Constraints. *International*

Journal of Research and Reviews in Electrical and Computer Engineering, *1*(1), 21–26.

Vlachogiannis, J. G., & Lee, K. Y. (2009b). Economic Load Dispatch—A Comparative Study on Heuristic Optimization Techniques With an Improved Coordinated Aggregation-Based PSO. *IEEE Transactions on Power Systems*, *24*(2), 991–1001. https://doi.org/10.1109/TPWRS.2009.2016524

Vlachos, A. (2011). Simulated annealing algorithm for environmental/economic dispatch problem. *Journal of Information and Computing*. Retrieved from https://pdfs.semanticscholar.org/27ee/18f376c4f66b3ced749e3b9eb320c1c0b81f.pdf

Wei Guo, Guohua Fang, & Xianfeng Huang. (2011). An improved chaotic artificial fish swarm algorithm and its application in optimizing cascade hydropower stations. In *2011 International Conference on Business Management and Electronic Information* (pp. 217–220). IEEE. https://doi.org/10.1109/ICBMEI.2011.5920432

Wu, L., Li, H., Wu, Z., & Wu, C. (2015). Economic Load Dispatch Based on Efficient Population Utilization Strategy for Particle Swarm Optimization. *International Journal of Communications,*. Retrieved from http://file.scirp.org/pdf/IJCNS_2015092914332169.pdf

Yun, L., Ji, K., Zhang, P., & Li, Y. (2011). Research on thermal power plant economic dispatch based on dynamic programming. *Journal of Computational*.

APPENDIX A1

Table 4a:13-Units Economic Load Dispatch Data

Gen. Unit.	Pmin	Pmax	a	B	c	e	f
1	0	680	0.0003	8.1	550	300	0.035
2	0	360	0.0006	8.1	309	200	0.042
3	0	360	0.0006	8.1	307	150	0.042
4	60	180	0.0032	7.74	240	150	0.063
5	60	180	0.0032	7.74	240	150	0.063
6	60	180	0.0032	7.74	240	150	0.063
7	60	180	0.0032	7.74	240	150	0.063
8	60	180	0.0032	7.74	240	150	0.063
9	60	180	0.0032	7.74	240	150	0.063
10	40	120	0.0028	8.6	126	100	0.084
11	40	120	0.0028	8.6	126	100	0.084
12	55	120	0.0028	8.6	126	100	0.084
13	55	120	0.0028	8.6	126	100	0.084

APPENDIX A2

Table 4b: 40-Units Economic Load Dispatch Data

Gen. Unit.	Pmin	Pmax	a	B	c	e	f
1	36	114	0.0069	6.73	94.705	100	0.084
2	36	114	0.0069	6.73	94.705	100	0.084
3	60	120	0.02028	7.07	309.54	100	0.084
4	80	190	0.00942	8.18	369.03	150	0.063
5	47	97	0.0114	5.35	148.89	120	0.077
6	68	140	0.01142	8.05	222.33	100	0.084
7	110	300	0.00357	8.03	278.71	200	0.042
8	135	300	0.00492	6.99	391.98	200	0.042
9	135	300	0.00573	6.6	455.76	200	0.042
10	130	300	0.00605	12.9	722.82	200	0.042
11	94	375	0.00515	12.9	635.2	200	0.042
12	94	375	0.00569	12.8	654.69	200	0.042
13	125	500	0.00421	12.5	913.4	300	0.035
14	125	500	0.00752	8.84	1760.4	300	0.035
15	125	500	0.00708	9.15	1728.3	300	0.035
16	125	500	0.00708	9.15	1728.3	300	0.035
17	220	500	0.00313	7.97	647.85	300	0.035
18	220	500	0.00313	7.95	649.69	300	0.035
19	242	550	0.00313	7.97	647.83	300	0.035
20	242	550	0.00313	7.97	647.81	300	0.035
21	254	550	0.00298	6.63	785.96	300	0.035
22	254	550	0.00298	6.63	785.96	300	0.035
23	254	550	0.00284	6.66	794.53	300	0.035
24	254	550	0.00284	6.66	794.53	300	0.035
25	254	550	0.00277	7.1	801.32	300	0.035
26	254	550	0.00277	7.1	801.32	300	0.035
27	10	150	0.52124	3.33	1055.1	120	0.077
28	10	150	0.52124	3.33	1055.1	120	0.077
29	10	150	0.52124	3.33	1055.1	120	0.077
30	47	97	0.0114	5.35	148.89	120	0.077
31	60	190	0.0016	6.43	222.92	150	0.063
32	60	190	0.0016	6.43	222.92	150	0.063
33	60	190	0.0016	6.43	222.92	150	0.063
34	90	200	0.0001	8.95	107.87	200	0.042
35	90	200	0.0001	8.62	116.58	200	0.042
36	90	200	0.0001	8.62	116.58	200	0.042
37	25	110	0.0161	5.88	307.45	80	0.098
38	25	110	0.0161	5.88	307.45	80	0.098
39	25	110	0.0161	5.88	307.45	80	0.098
40	242	550	0.00313	7.97	647.83	300	0.035

APPENDIX A3

Table 4c1: 110-Units Economic Load Dispatch Data

Gen. Unit.	Pmin	Pmax	a	B	c	e	f
1	2.4	12	0.0253	25.547	24.389	0	0
2	2.4	12	0.0265	25.675	24.411	0	0
3	2.4	12	0.028	25.803	24.638	0	0
4	2.4	12	0.0284	25.932	24.76	0	0
5	2.4	12	0.0286	26.061	24.888	0	0
6	4	20	0.012	37.551	117.76	0	0
7	4	20	0.0126	37.664	118.11	0	0
8	4	20	0.0136	37.777	118.46	0	0
9	4	20	0.0143	37.89	118.82	0	0
10	15.2	76	0.0088	13.327	81.136	0	0
11	15.2	76	0.0089	13.354	81.298	0	0
12	15.2	76	0.0091	13.8	81.464	0	0
13	15.2	76	0.0093	13.407	81.626	0	0
14	25	100	0.0062	18	217.9	0	0
15	25	100	0.0061	18.1	218.34	0	0
16	25	100	0.006	18.2	218.78	0	0
17	54.3	155	0.0046	10.694	142.74	0	0
18	54.3	155	0.0047	10.715	143.03	0	0
19	54.3	155	0.0048	10.737	143.32	0	0
20	54.3	155	0.0049	10.758	143.6	0	0
21	68.9	197	0.0026	23	259.13	0	0
22	68.9	197	0.0026	23.1	259.65	0	0
23	68.9	197	0.0026	23.2	260.18	0	0
24	140	350	0.0015	10.862	177.06	0	0
25	100	400	0.0019	7.492	210	0	0
26	100	400	0.0019	7.503	211.91	0	0
27	140	500	0.0014	12	210	0	0
28	140	500	0.0013	12.1	180	0	0
29	50	200	0.0026	12.2	240	0	0
30	25	100	0.0039	12.5	220	0	0
31	10	50	0.0051	23	60	0	0
32	5	20	0.005	13.5	50	0	0
33	20	80	0.0078	13.2	200	0	0
34	75	250	0.0012	12.4	140	0	0
35	110	360	0.0038	10.3	120	0	0
36	130	400	0.0043	9.9	90	0	0
37	10	40	0.0011	13.4	80	0	0
38	20	70	0.0023	13.3	70	0	0
39	25	100	0.0034	12.9	115	0	0
40	20	120	0.0067	12.8	150	0	0
41	40	180	0.0056	12.7	40	0	0
42	50	220	0.0023	12.6	300	0	0
43	120	440	0.0012	7.4	250	0	0
44	160	560	0.0045	6.6	100	0	0
45	150	660	0.0022	6.5	160	0	0
46	200	700	0.0067	6.2	130	0	0
47	5.4	32	0.0353	26.547	34.389	0	0
48	5.4	32	0.0365	26.675	34.411	0	0
49	8.4	52	0.038	26.803	34.638	0	0
50	8.4	52	0.0384	26.932	34.761	0	0

51	8.4	52	0.0386	17.061	34.888	0	0
52	12	60	0.032	38.551	127.76	0	0
53	12	60	0.0326	36.664	128.11	0	0
54	12	60	0.0236	38.777	128.46	0	0

Table 4c2: 110-Units Economic Load Dispatch Data (Continued)

Gen. Unit.	Pmin	Pmax	a	b	c	e	f
55	12	60	0.0243	38.89	128.82	0	0
56	25.2	96	0.0098	14.327	82.136	0	0
57	25.2	96	0.0099	14.354	82.298	0	0
58	35	100	0.0092	14.38	82.464	0	0
59	35	100	0.0094	14.407	82.626	0	0
60	45	120	0.0072	19	218.9	0	0
61	45	120	0.0071	19.1	219.34	0	0
62	45	120	0.007	19.2	219.78	0	0
63	54.3	185	0.0066	11.694	143.74	0	0
64	54.3	185	0.0057	11.715	144.03	0	0
65	54.3	185	0.0058	11.737	144.32	0	0
66	54.3	185	0.0059	11.758	144.6	0	0
67	70	197	0.0036	24	269.13	0	0
68	70	197	0.0036	24.1	269.65	0	0
69	70	197	0.0036	24.2	270.18	0	0
70	150	360	0.0025	11.862	187.06	0	0
71	160	400	0.0029	8.492	320	0	0
72	160	400	0.003	8.503	321.91	0	0
73	60	300	0.0054	13.327	52.136	0	0
74	50	250	0.0055	12.354	42.298	0	0
75	30	90	0.0099	11.38	32.464	0	0
76	12	50	0.0031	9.407	23.626	0	0
77	160	450	0.0024	14	220	0	0
78	150	600	0.0023	13.1	190	0	0
79	50	200	0.0036	13.2	250	0	0
80	20	120	0.0049	13.5	230	0	0
81	10	55	0.0061	24	70	0	0
82	12	40	0.007	14.5	60	0	0
83	20	80	0.0088	14.2	210	0	0
84	50	200	0.0022	13.4	150	0	0
85	80	325	0.0048	11.3	130	0	0
86	120	440	0.0053	8.9	80	0	0
87	10	35	0.0021	14.4	90	0	0
88	20	55	0.0033	14.3	80	0	0
89	20	100	0.0034	13.9	125	0	0
90	40	220	0.0037	13.8	160	0	0
91	30	140	0.0066	13.7	50	0	0
92	40	100	0.0043	13.6	400	0	0
93	100	440	0.0022	8.4	260	0	0
94	100	500	0.0055	7.6	110	0	0
95	100	600	0.0032	7.5	170	0	0
96	200	700	0.0077	7.2	140	0	0
97	3.6	15	0.0353	26.547	26.389	0	0
98	3.6	15	0.0365	26.675	25.411	0	0
99	4.4	22	0.038	26.803	25.638	0	0
100	4.4	22	0.0384	26.932	25.76	0	0
101	10	60	0.021	15.3	65	0	0
102	10	80	0.023	16	82	0	0
103	20	100	0.024	20.2	86	0	0
104	20	120	0.035	20.2	84	0	0

105	40	150	0.034	25.6	75	0	0
106	40	280	0.037	30.5	56	0	0
107	50	520	0.039	32.5	67	0	0
108	30	150	0.035	26	68	0	0
109	40	320	0.028	25.8	69	0	0
110	20	200	0.026	27	72	0	0

APPENDIX A4

Table 4d1: 140-Units Economic Load Dispatch Data

Gen. Unit.	Pmin	Pmax	a	b	c	e	f
1	71	119	0.032888	61.242	1220.6	0	0
2	120	189	0.00828	41.095	1315.1	0	0
3	125	190	0.003849	46.31	874.29	0	0
4	125	190	0.003849	46.31	874.29	0	0
5	90	190	0.042468	54.242	1976.5	0	0
6	90	190	0.014992	61.215	1338.1	0	0
7	280	490	0.007039	11.791	1818.3	0	0
8	280	490	0.003079	15.055	1134	0	0
9	260	496	0.005063	13.226	1320.6	0	0
10	260	496	0.005063	13.226	1320.6	0	0
11	260	496	0.005063	13.226	1320.6	0	0
12	260	496	0.003552	14.498	1106.5	0	0
13	260	506	0.003901	14.651	1176.5	0	0
14	260	509	0.003901	14.651	1176.5	0	0
15	260	506	0.003901	14.651	1176.5	0	0
16	260	505	0.003901	14.651	1176.5	0	0
17	260	506	0.002393	15.669	1017.4	0	0
18	260	506	0.002393	15.669	1017.4	0	0
19	260	505	0.003684	14.656	1229.1	0	0
20	260	505	0.003684	14.656	1229.1	0	0
21	260	505	0.003684	14.656	1229.1	0	0
22	260	505	0.003684	14.656	1229.1	0	0
23	260	505	0.004004	14.378	1267.9	0	0
24	260	505	0.003684	14.656	1229.1	0	0
25	280	537	0.001619	16.261	975.93	0	0
26	280	537	0.005093	13.362	1532.1	0	0
27	280	549	0.000993	17.203	641.99	0	0
28	280	549	0.000993	17.203	641.99	0	0
29	260	501	0.002473	15.274	911.53	0	0
30	260	501	0.002547	15.212	910.53	0	0
31	260	506	0.003542	15.033	1074.8	0	0
32	260	506	0.003542	15.033	1074.8	0	0
33	260	506	0.003542	15.033	1074.8	0	0
34	260	506	0.003542	15.033	1074.8	0	0
35	260	500	0.003132	13.992	1278.5	0	0
36	260	500	0.001323	15.679	861.74	0	0
37	120	241	0.00295	16.542	408.83	0	0
38	120	241	0.00295	16.542	408.83	0	0
39	423	774	0.000991	16.518	1288.8	0	0
40	423	769	0.001581	15.815	1436.3	0	0
41	3	19	0.90236	75.464	669.99	0	0
42	3	28	0.1103	129.54	134.54	0	0
43	160	250	0.024493	56.613	3427.9	0	0
44	160	250	0.029156	54.451	3751.8	0	0
45	160	250	0.024667	54.736	3918.8	0	0
46	160	250	0.016517	58.034	3379.6	0	0
47	160	250	0.026584	55.981	3345.3	0	0
48	160	250	0.00754	61.52	3138.8	0	0
49	160	250	0.01643	58.635	3453.1	0	0
50	160	250	0.045934	44.647	5119.3	0	0
51	165	504	4.40E-05	71.584	1898.4	0	0
52	165	504	4.40E-05	71.584	1898.4	0	0
53	165	504	4.40E-05	71.584	1898.4	0	0

54	165	504	4.40E-05	71.584	1898.4	0	0

Table 4d2: 140-Units Economic Load Dispatch Data (Continued)

Gen. Unit.	Pmin	Pmax	a	b	c	e	f
55	180	471	0.002528	85.12	2473.4	0	0
56	180	561	0.000131	87.682	2781.7	0	0
57	103	341	0.010372	69.532	5515.5	0	0
58	198	617	0.007627	78.339	3478.3	0	0
59	100	312	0.012464	58.172	6240.9	0	0
60	153	471	0.039441	46.636	9960.1	0	0
61	163	500	0.007278	76.947	3672	0	0
62	95	302	4.40E-05	80.761	1837.4	0	0
63	160	511	4.40E-05	70.136	3108.4	0	0
64	160	511	4.40E-05	70.136	3108.4	0	0
65	196	490	0.018827	49.84	7095.5	0	0
66	196	490	0.010852	65.404	3392.7	0	0
67	196	490	0.018827	49.84	7095.5	0	0
68	196	490	0.018827	49.84	7095.5	0	0
69	130	432	0.03456	66.465	4288.3	0	0
70	130	432	0.08154	22.941	13813	0	0
71	137	455	0.023534	64.314	4435.5	0	0
72	137	455	0.035475	45.017	9750.8	0	0
73	195	541	0.000915	70.644	1042.4	0	0
74	175	536	4.40E-05	70.959	1159.9	0	0
75	175	540	4.40E-05	70.959	1159.9	0	0
76	175	538	0.001307	70.302	1304	0	0
77	175	540	0.000392	70.662	1156.2	0	0
78	330	574	8.70E-05	71.101	2119	0	0
79	160	531	0.000521	37.854	779.52	0	0
80	160	531	0.000498	37.768	829.89	0	0
81	200	542	0.001046	67.983	2333.7	0	0
82	56	132	0.13205	77.838	2029	0	0
83	115	245	0.096968	63.671	4412	0	0
84	115	245	0.054868	79.458	2982.2	0	0
85	115	245	0.054868	79.458	2982.2	0	0
86	207	307	0.014382	93.966	3174.9	0	0
87	207	307	0.013161	94.723	3218.4	0	0
88	175	345	0.016033	66.919	3723.8	0	0
89	175	345	0.013653	68.185	3551.4	0	0
90	175	345	0.028148	60.821	4322.6	0	0
91	175	345	0.01347	68.551	3493.7	0	0
92	360	580	6.40E-05	2.842	226.8	0	0
93	415	645	0.000252	2.946	382.93	0	0
94	795	984	2.20E-05	3.096	156.99	0	0
95	795	978	2.20E-05	3.04	154.48	0	0
96	578	682	0.000203	1.709	332.83	0	0
97	615	720	0.000198	1.668	326.6	0	0
98	612	718	0.000215	1.789	345.31	0	0
99	612	720	0.000218	1.815	350.37	0	0
100	758	964	0.000193	2.726	370.38	0	0
101	755	958	0.000197	2.732	367.07	0	0
102	750	1007	0.000324	2.651	124.88	0	0
103	750	1006	0.000344	2.798	130.79	0	0

104	713	1013	0.00069	1.595	878.75	0	0
105	718	1020	0.00065	1.503	827.96	0	0
106	791	954	0.000233	2.425	432.01	0	0
107	786	952	0.000239	2.499	445.61	0	0
108	795	1006	0.000261	2.674	467.22	0	0

Table 4d3: 140-Units Economic Load Dispatch Data (Continued)

Gen. Unit.	Pmin	Pmax	a	b	c	e	f
109	795	1013	0.000259	2.692	475.94	0	0
110	795	1021	0.000707	1.633	899.46	0	0
111	795	1015	0.000786	1.816	1000.4	0	0
112	94	203	0.014355	89.83	1269.1	0	0
113	94	203	0.014355	89.83	1269.1	0	0
114	94	203	0.014355	89.83	1269.1	0	0
115	244	379	0.030266	64.125	4965.1	0	0
116	244	379	0.030266	64.125	4965.1	0	0
117	244	379	0.030266	64.125	4965.1	0	0
118	95	190	0.024027	76.129	2243.2	0	0
119	95	189	0.00158	81.805	2290.4	0	0
120	116	194	0.022095	81.14	1681.5	0	0
121	175	321	0.07681	46.665	6743.3	0	0
122	2	19	0.95344	78.412	394.4	0	0
123	4	59	4.40E-05	112.09	1243.2	0	0
124	15	83	0.072468	90.871	1454.7	0	0
125	9	53	0.000448	97.116	1011.1	0	0
126	12	37	0.59911	83.244	909.27	0	0
127	10	34	0.24471	95.665	689.38	0	0
128	112	373	4.20E-05	91.202	1443.8	0	0
129	4	20	0.085145	104.5	535.55	0	0
130	5	38	0.52472	83.015	617.73	0	0
131	5	19	0.17652	127.8	90.966	0	0
132	50	98	0.063414	77.929	974.45	0	0
133	5	10	2.7405	92.779	263.81	0	0
134	42	74	0.11244	80.95	1335.6	0	0
135	42	74	0.041529	89.073	1033.9	0	0
136	41	105	0.000911	161.29	1391.3	0	0
137	17	51	0.005245	161.83	4477.1	0	0
138	7	19	0.23479	84.972	57.794	0	0
139	7	19	0.23479	84.972	57.794	0	0
140	26	40	1.1119	16.087	1258.4	0	0

APPENDIX A5

Table 4e1: 160-Units Economic Load Dispatch Data (Continued)

Gen. Unit.	Pmin	Pmax	a	b	c	e	f
1	196	250	0.0019	-0.3059	21.13	0.0211	-3.059
2	157	230	0.0042	-1.269	118.4	0.1184	-12.69
3	200	332	0.0015	-0.3116	39.79	0.0398	-3.116
4	200	265	0.0059	-2.338	266.8	0.2668	-23.38
5	190	338	0.0011	-0.0873	13.92	0.0139	-0.8733
6	200	265	0.0059	-2.338	266.8	0.2668	-23.38
7	200	331	0.0011	-0.1325	18.93	0.0189	-1.325
8	200	265	0.0059	-2.338	266.8	0.2668	-23.38
9	370	440	0.0006	-0.0182	14.23	0.0142	-0.1817
10	200	362	0.0011	-0.0994	13.97	0.014	-0.9938
11	196	250	0.0019	-0.3059	21.13	0.0211	-3.059
12	157	230	0.0042	-1.269	118.4	0.1184	-12.69
13	200	332	0.0015	-0.3116	39.79	0.0398	-3.116
14	200	265	0.0059	-2.338	266.8	0.2668	-23.38
15	190	338	0.0011	-0.0873	13.92	0.0139	-0.8733
16	200	265	0.0059	-2.338	266.8	0.2668	-23.38
17	200	331	0.0011	-0.1325	18.93	0.0189	-1.325
18	200	265	0.0059	-2.338	266.8	0.2668	-23.38
19	370	440	0.0006	-0.0182	14.23	0.0142	-0.1817
20	200	362	0.0011	-0.0994	13.97	0.014	-0.9938
21	196	250	0.0019	-0.3059	21.13	0.0211	-3.059

Table 4e2: 160-Units Economic Load Dispatch Data (Continued)

Gen. Unit.	Pmin	Pmax	a	b	c	e	f
22	157	230	0.0042	-1.269	118.4	0.1184	-12.69
23	200	332	0.0015	-0.3116	39.79	0.0398	-3.116
24	200	265	0.0059	-2.338	266.8	0.2668	-23.38
25	190	338	0.0011	-0.0873	13.92	0.0139	-0.8733
26	200	265	0.0059	-2.338	266.8	0.2668	-23.38
27	200	331	0.0011	-0.1325	18.93	0.0189	-1.325
28	200	265	0.0059	-2.338	266.8	0.2668	-23.38
29	370	440	0.0006	-0.0182	14.23	0.0142	-0.1817
30	200	362	0.0011	-0.0994	13.97	0.014	-0.9938
31	196	250	0.0019	-0.3059	21.13	0.0211	-3.059
32	157	230	0.0042	-1.269	118.4	0.1184	-12.69
33	200	332	0.0015	-0.3116	39.79	0.0398	-3.116
34	200	265	0.0059	-2.338	266.8	0.2668	-23.38
35	190	338	0.0011	-0.0873	13.92	0.0139	-0.8733
36	200	265	0.0059	-2.338	266.8	0.2668	-23.38
37	200	331	0.0011	-0.1325	18.93	0.0189	-1.325
38	200	265	0.0059	-2.338	266.8	0.2668	-23.38
39	370	440	0.0006	-0.0182	14.23	0.0142	-0.1817
40	200	362	0.0011	-0.0994	13.97	0.014	-0.9938
41	196	250	0.0019	-0.3059	21.13	0.0211	-3.059
42	157	230	0.0042	-1.269	118.4	0.1184	-12.69
43	200	332	0.0015	-0.3116	39.79	0.0398	-3.116
44	200	265	0.0059	-2.338	266.8	0.2668	-23.38
45	190	338	0.0011	-0.0873	13.92	0.0139	-0.8733
46	200	265	0.0059	-2.338	266.8	0.2668	-23.38

47	200	331	0.0011	-0.1325	18.93	0.0189	-1.325
48	200	265	0.0059	-2.338	266.8	0.2668	-23.38
49	370	440	0.0006	-0.0182	14.23	0.0142	-0.1817
50	200	362	0.0011	-0.0994	13.97	0.014	-0.9938
51	196	250	0.0019	-0.3059	21.13	0.0211	-3.059
52	157	230	0.0042	-1.269	118.4	0.1184	-12.69
53	200	332	0.0015	-0.3116	39.79	0.0398	-3.116
54	200	265	0.0059	-2.338	266.8	0.2668	-23.38
55	190	338	0.0011	-0.0873	13.92	0.0139	-0.8733
56	200	265	0.0059	-2.338	266.8	0.2668	-23.38
57	200	331	0.0011	-0.1325	18.93	0.0189	-1.325
58	200	265	0.0059	-2.338	266.8	0.2668	-23.38
59	370	440	0.0006	-0.0182	14.23	0.0142	-0.1817
60	200	362	0.0011	-0.0994	13.97	0.014	-0.9938
61	196	250	0.0019	-0.3059	21.13	0.0211	-3.059
62	157	230	0.0042	-1.269	118.4	0.1184	-12.69
63	200	332	0.0015	-0.3116	39.79	0.0398	-3.116
64	200	265	0.0059	-2.338	266.8	0.2668	-23.38
65	190	338	0.0011	-0.0873	13.92	0.0139	-0.8733
66	200	265	0.0059	-2.338	266.8	0.2668	-23.38
67	200	331	0.0011	-0.1325	18.93	0.0189	-1.325
68	200	265	0.0059	-2.338	266.8	0.2668	-23.38
69	370	440	0.0006	-0.0182	14.23	0.0142	-0.1817
70	200	362	0.0011	-0.0994	13.97	0.014	-0.9938
71	196	250	0.0019	-0.3059	21.13	0.0211	-3.059
72	157	230	0.0042	-1.269	118.4	0.1184	-12.69
73	200	332	0.0015	-0.3116	39.79	0.0398	-3.116
74	200	265	0.0059	-2.338	266.8	0.2668	-23.38
75	190	338	0.0011	-0.0873	13.92	0.0139	-0.8733

Table 4e3: 160-Units Economic Load Dispatch Data (Continued)

Gen. Unit.	Pmin	Pmax	a	b	c	e	f
76	200	265	0.0059	-2.338	266.8	0.2668	-23.38
77	200	331	0.0011	-0.1325	18.93	0.0189	-1.325
78	200	265	0.0059	-2.338	266.8	0.2668	-23.38
79	370	440	0.0006	-0.0182	14.23	0.0142	-0.1817
80	200	362	0.0011	-0.0994	13.97	0.014	-0.9938
81	196	250	0.0019	-0.3059	21.13	0.0211	-3.059
82	157	230	0.0042	-1.269	118.4	0.1184	-12.69
83	200	332	0.0015	-0.3116	39.79	0.0398	-3.116
84	200	265	0.0059	-2.338	266.8	0.2668	-23.38
85	190	338	0.0011	-0.0873	13.92	0.0139	-0.8733
86	200	265	0.0059	-2.338	266.8	0.2668	-23.38
87	200	331	0.0011	-0.1325	18.93	0.0189	-1.325
88	200	265	0.0059	-2.338	266.8	0.2668	-23.38
89	370	440	0.0006	-0.0182	14.23	0.0142	-0.1817
90	200	362	0.0011	-0.0994	13.97	0.014	-0.9938
91	196	250	0.0019	-0.3059	21.13	0.0211	-3.059
92	157	230	0.0042	-1.269	118.4	0.1184	-12.69
93	200	332	0.0015	-0.3116	39.79	0.0398	-3.116
94	200	265	0.0059	-2.338	266.8	0.2668	-23.38
95	190	338	0.0011	-0.0873	13.92	0.0139	-0.8733
96	200	265	0.0059	-2.338	266.8	0.2668	-23.38
97	200	331	0.0011	-0.1325	18.93	0.0189	-1.325
98	200	265	0.0059	-2.338	266.8	0.2668	-23.38
99	370	440	0.0006	-0.0182	14.23	0.0142	-0.1817
100	200	362	0.0011	-0.0994	13.97	0.014	-0.9938
101	196	250	0.0019	-0.3059	21.13	0.0211	-3.059
102	157	230	0.0042	-1.269	118.4	0.1184	-12.69
103	200	332	0.0015	-0.3116	39.79	0.0398	-3.116
104	200	265	0.0059	-2.338	266.8	0.2668	-23.38
105	190	338	0.0011	-0.0873	13.92	0.0139	-0.8733
106	200	265	0.0059	-2.338	266.8	0.2668	-23.38

107	200	331	0.0011	-0.1325	18.93	0.0189	-1.325
108	200	265	0.0059	-2.338	266.8	0.2668	-23.38
109	370	440	0.0006	-0.0182	14.23	0.0142	-0.1817
110	200	362	0.0011	-0.0994	13.97	0.014	-0.9938
111	196	250	0.0019	-0.3059	21.13	0.0211	-3.059
112	157	230	0.0042	-1.269	118.4	0.1184	-12.69
113	200	332	0.0015	-0.3116	39.79	0.0398	-3.116
114	200	265	0.0059	-2.338	266.8	0.2668	-23.38
115	190	338	0.0011	-0.0873	13.92	0.0139	-0.8733
116	200	265	0.0059	-2.338	266.8	0.2668	-23.38
117	200	331	0.0011	-0.1325	18.93	0.0189	-1.325
118	200	265	0.0059	-2.338	266.8	0.2668	-23.38
119	370	440	0.0006	-0.0182	14.23	0.0142	-0.1817
120	200	362	0.0011	-0.0994	13.97	0.014	-0.9938
121	196	250	0.0019	-0.3059	21.13	0.0211	-3.059
122	157	230	0.0042	-1.269	118.4	0.1184	-12.69
123	200	332	0.0015	-0.3116	39.79	0.0398	-3.116
124	200	265	0.0059	-2.338	266.8	0.2668	-23.38
125	190	338	0.0011	-0.0873	13.92	0.0139	-0.8733
126	200	265	0.0059	-2.338	266.8	0.2668	-23.38
127	200	331	0.0011	-0.1325	18.93	0.0189	-1.325
128	200	265	0.0059	-2.338	266.8	0.2668	-23.38
129	370	440	0.0006	-0.0182	14.23	0.0142	-0.1817

Table 4e4: 160-Units Economic Load Dispatch Data (Continued)

Gen. Unit.	Pmin	Pmax	a	b	c	e	f
130	200	362	0.0011	-0.0994	13.97	0.014	-0.9938
131	196	250	0.0019	-0.3059	21.13	0.0211	-3.059
132	157	230	0.0042	-1.269	118.4	0.1184	-12.69
133	200	332	0.0015	-0.3116	39.79	0.0398	-3.116
134	200	265	0.0059	-2.338	266.8	0.2668	-23.38
135	190	338	0.0011	-0.0873	13.92	0.0139	-0.8733
136	200	265	0.0059	-2.338	266.8	0.2668	-23.38
137	200	331	0.0011	-0.1325	18.93	0.0189	-1.325
138	200	265	0.0059	-2.338	266.8	0.2668	-23.38
139	370	440	0.0006	-0.0182	14.23	0.0142	-0.1817
140	200	362	0.0011	-0.0994	13.97	0.014	-0.9938
141	196	250	0.0019	-0.3059	21.13	0.0211	-3.059
142	157	230	0.0042	-1.269	118.4	0.1184	-12.69
143	200	332	0.0015	-0.3116	39.79	0.0398	-3.116
144	200	265	0.0059	-2.338	266.8	0.2668	-23.38
145	190	338	0.0011	-0.0873	13.92	0.0139	-0.8733
146	200	265	0.0059	-2.338	266.8	0.2668	-23.38
147	200	331	0.0011	-0.1325	18.93	0.0189	-1.325
148	200	265	0.0059	-2.338	266.8	0.2668	-23.38
149	370	440	0.0006	-0.0182	14.23	0.0142	-0.1817
150	200	362	0.0011	-0.0994	13.97	0.014	-0.9938
151	196	250	0.0019	-0.3059	21.13	0.0211	-3.059
152	157	230	0.0042	-1.269	118.4	0.1184	-12.69
153	200	332	0.0015	-0.3116	39.79	0.0398	-3.116
154	200	265	0.0059	-2.338	266.8	0.2668	-23.38
155	190	338	0.0011	-0.0873	13.92	0.0139	-0.8733
156	200	265	0.0059	-2.338	266.8	0.2668	-23.38
157	200	331	0.0011	-0.1325	18.93	0.0189	-1.325
158	200	265	0.0059	-2.338	266.8	0.2668	-23.38
159	370	440	0.0006	-0.0182	14.23	0.0142	-0.1817
160	200	362	0.0011	-0.0994	13.97	0.014	-0.9938

APPENDIX B

m.File MATLAB Codes for the proposed ELD algorithm

Appendix B1: Hybrid Genetic-Artificial Fish Swarm Algorithm

```
1. function HGAFSA(Ngen)
2. addpath('AFSA');
3. addpath('BCGA');
4. addpath('Integrators');
5. [
   ~,maxit,mincost,popsize,mutrate,selection,nbits,opt_type,lo
   ,hi] = parameters_bcga;
6. [nfish,npar,Visual_distance,Crowdness_factor,S,max_iga,ntry
   ]=Parameters_afsa;
7. [repparam,crosparam,mutatparam]=gengaparams;
8. [par,pop]=reproduction(repparam);
9. [Value,L]=objF(par);
10.     % Vv=Value(1);
11.     % Ll=L(1);
12.     [cost,id]=sortrows([Value,L],1);
13.     pop=pop(id,:);
14.     if opt_type==2
15.     minc=cost(1,:);
16.     else
17.     minc=cost(end,:);
```

```
18.     end
19.     minc1=[];
20.     mH=[];
21.     if opt_type==2
    a. minc1=sort([minc1,minc(:,1)'],'descend');
    b. mH=sort([mH,minc(:,2:end)'],'descend');
22.     else
    a. minc1=sort([minc1,minc(:,1)']);
    b. mH=sort([mH,minc(:,2:end)']);
23.     end
24.     iga=1;
25.     while iga<max_iga
26.     %@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
  @@@@@@@@@@@@@@@
27.     New_Fish=[];
28.     Xi=par;
29.     for Fish=1:size(par,1)
30.     %
  %%%%%%%%%%%%%%%%%%%¬¬¬¬¬¬¬¬PREYING¬¬¬¬¬¬¬¬%%%%%%%%%%%%%%%
  %%%%
31.     X_new=feval('prey',Xi(Fish,:),Visual_distance,S,iga,ma
  x_iga,ntry,opt_type);
```

```
32.     %
   %%%%%%%%%%%%%%%%%%%%¬¬¬¬¬¬¬¬£££££££¬¬¬¬¬¬¬¬%%%%%%%%%%%%%%%%
   %%%%
33.     if Fish==1
     a. if iga==1
34.     X_best=X_new;
     a. end
35.     else
     a. Y_best=objF(X_best);
     b. Y_new=objF(X_new);
     c. if opt_type==1
          i. if Y_new>Y_best
         ii. X_best=X_new;
        iii. end
     d. end
     e. if opt_type==2
          i. if Y_new<Y_best
         ii. X_best=X_new;
        iii. end
     f. end
36.     end
37.     New_Fish=[New_Fish;X_new];
38.     end
39.     new_par=New_Fish;
```

```
40.     new_pop = transform_ga_afsa( New_Fish,size(pop,2) );

41.     [par,pop,Minc] = compare_ga_afsa(
  par,new_par,pop,new_pop,opt_type);
42.     if opt_type==2
     a. minc=flipud(sortrows([minc;Minc],1));
43.     else
     a. minc=sortrows([minc;Minc],1);
44.     end

45.     if opt_type==2
     a. minc1=sort([minc1,minc(:,1)'],'descend');
     b. mH=sort([mH,minc(:,2:end)'],'descend');
46.     else
     a. minc1=sort([minc1,minc(:,1)']);
     b. mH=sort([mH,minc(:,2:end)']);
47.     end
48.     [new_par,new_pop]=crossover(pop,crosparam);
49.     [par,pop,Minc] = compare_ga_afsa(
  par,new_par,pop,new_pop,opt_type);
50.     if opt_type==2
     a. minc=flipud(sortrows([minc;Minc],1));
51.     else
     a. minc=sortrows([minc;Minc],1);
```

```
52.     end
53.     if opt_type==2
     a. minc1=sort([minc1,minc(:,1)'],'descend');
     b. mH=sort([mH,minc(:,2:end)'],'descend');
54.     else
     a. minc1=sort([minc1,minc(:,1)']);
     b. mH=sort([mH,minc(:,2:end)']);
55.     end
56.     [new_par,new_pop]=mutation(pop,mutatparam);
57.     [par,pop,Minc] = compare_ga_afsa(
  par,new_par,pop,new_pop,opt_type);
58.     if opt_type==2
     a. minc=flipud(sortrows([minc;Minc],1));
59.     else
     a. minc=sortrows([minc;Minc],1);
60.     end
61.     if opt_type==2
     a. minc1=sort([minc1,minc(:,1)'],'descend');
     b. mH=sort([mH,minc(:,2:end)'],'descend');
62.     else
     a. minc1=sort([minc1,minc(:,1)']);
     b. mH=sort([mH,minc(:,2:end)']);
63.     end
```

```
%@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
%@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
New_Fish=[];
Xi=par;
for Fish=1:size(par,1)
%%%%%%%%%%%%%%%%%%%%%¬¬¬¬¬¬¬¬PREYING¬¬¬¬¬¬¬¬%%%%%%%%%%%%%%%%%%%%
X_new=feval('swarm',Xi(Fish,:),Visual_distance,Crowdness_factor,S,iga,max_iga,ntry,opt_type);
%%%%%%%%%%%%%%%%%%%%%¬¬¬¬¬¬¬¬£££££££¬¬¬¬¬¬¬¬%%%%%%%%%%%%%%%%%%%%
if Fish==1
if iga==1
X_best=X_new;
end
else
Y_best=objF(X_best);
Y_new=objF(X_new);
if opt_type==1
if Y_new>Y_best
```

```
        ii. X_best=X_new;
       iii. end
    d. end
    e. if opt_type==2
        i. if Y_new<Y_best
        ii. X_best=X_new;
       iii. end
    f. end
75.    end
76.    New_Fish=[New_Fish;X_new];
77.    end
78.    new_par=New_Fish;
79.    new_pop = transform_ga_afsa( New_Fish,size(pop,2) );
80.    [par,pop,Minc] = compare_ga_afsa(
  par,new_par,pop,new_pop,opt_type);
81.    if opt_type==2
    a. minc=flipud(sortrows([minc;Minc],1));
82.    else
    a. minc=sortrows([minc;Minc],1);
83.    end
84.    if opt_type==2
    a. minc1=sort([minc1,minc(:,1)'],'descend');
    b. mH=sort([mH,minc(:,2:end)'],'descend');
85.    else
```

```
    a. minc1=sort([minc1,minc(:,1)']);
    b. mH=sort([mH,minc(:,2:end)']);
86.    end
87.    [new_par,new_pop]=crossover(pop,crosparam);
88.    [par,pop,Minc] = compare_ga_afsa(
  par,new_par,pop,new_pop,opt_type);
89.    if opt_type==2
    a. minc=flipud(sortrows([minc;Minc],1));
90.    else
    a. minc=sortrows([minc;Minc],1);
91.    end
92.    if opt_type==2
    a. minc1=sort([minc1,minc(:,1)'],'descend');
    b. mH=sort([mH,minc(:,2:end)'],'descend');
93.    else
    a. minc1=sort([minc1,minc(:,1)']);
    b. mH=sort([mH,minc(:,2:end)']);
94.    end
95.    [new_par,new_pop]=mutation(pop,mutatparam);
96.    [par,pop,Minc] = compare_ga_afsa(
  par,new_par,pop,new_pop,opt_type);
97.    if opt_type==2
    a. minc=flipud(sortrows([minc;Minc],1));
98.    else
```

```
    a. minc=sortrows([minc;Minc],1);
99.     end
100.    if opt_type==2
    a. minc1=sort([minc1,minc(:,1)'],'descend');
    b. mH=sort([mH,minc(:,2:end)'],'descend');
101.    else
    a. minc1=sort([minc1,minc(:,1)']);
    b. mH=sort([mH,minc(:,2:end)']);
102.    end
103.    %@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
  @@@@@@@@@@@@@@@@
104.    %@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
  @@@@@@@@@@@@@@@
105.    New_Fish=[];
106.    Xi=par;
107.    for Fish=1:size(par,1)
108.    %
  %%%%%%%%%%%%%%%%%%%%¬¬¬¬¬¬¬¬PREYING¬¬¬¬¬¬¬¬%%%%%%%%%%%%%%%%
  %%%%
109.    X_new=feval('chase',Xi(Fish,:),Visual_distance,Crowdne
  ss_factor,S,iga,max_iga,ntry,opt_type);
110.    %
  %%%%%%%%%%%%%%%%%%%%¬¬¬¬¬¬¬¬£££££££¬¬¬¬¬¬¬¬%%%%%%%%%%%%%%%%
  %%%%
```

```
111.    if Fish==1
     a. if iga==1
112.    X_best=X_new;
     a. end
113.    else
     a. Y_best=objF(X_best);
     b. Y_new=objF(X_new);
     c. if opt_type==1
          i. if Y_new>Y_best
         ii. X_best=X_new;
        iii. end
     d. end
     e. if opt_type==2
          i. if Y_new<Y_best
         ii. X_best=X_new;
        iii. end
     f. end
114.    end
115.    New_Fish=[New_Fish;X_new];
116.    end
117.    new_par=New_Fish;
118.    new_pop = transform_ga_afsa( New_Fish,size(pop,2) );
119.    [par,pop,Minc] = compare_ga_afsa(
  par,new_par,pop,new_pop,opt_type);
```

```
120.    if opt_type==2
     a. minc=flipud(sortrows([minc;Minc],1));
121.    else
     a. minc=sortrows([minc;Minc],1);
122.    end
123.    if opt_type==2
     a. minc1=sort([minc1,minc(:,1)'],'descend');
     b. mH=sort([mH,minc(:,2:end)'],'descend');
124.    else
     a. minc1=sort([minc1,minc(:,1)']);
     b. mH=sort([mH,minc(:,2:end)']);
125.    end
126.    [new_par,new_pop]=crossover(pop,crosparam);
127.    [par,pop,Minc] = compare_ga_afsa(
  par,new_par,pop,new_pop,opt_type);
128.    if opt_type==2
     a. minc=flipud(sortrows([minc;Minc],1));
129.    else
     a. minc=sortrows([minc;Minc],1);
130.    end
131.    if opt_type==2
     a. minc1=sort([minc1,minc(:,1)'],'descend');
     b. mH=sort([mH,minc(:,2:end)'],'descend');
132.    else
```

```
    a. minc1=sort([minc1,minc(:,1)']);
    b. mH=sort([mH,minc(:,2:end)']);
end
[new_par,new_pop]=mutation(pop,mutatparam);

[par,pop,Minc] = compare_ga_afsa(
  par,new_par,pop,new_pop,opt_type);
if opt_type==2
    a. minc=flipud(sortrows([minc;Minc],1));
else
    a. minc=sortrows([minc;Minc],1);
end
if opt_type==2
    a. minc1=sort([minc1,minc(:,1)'],'descend');
    b. mH=sort([mH,minc(:,2:end)'],'descend');
else
    a. minc1=sort([minc1,minc(:,1)']);
    b. mH=sort([mH,minc(:,2:end)']);
end
%@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
  @@@@@@@@@@@@@@@@
iga=iga+1;
end
day=clock;
```

```
146.    disp(datestr(datenum(day(1),day(2),day(3),day(4),day(5
  ),day(6)),0))
147.    format shortg
148.    disp(['popsize = ' num2str(popsize) ' mutrate = '
  num2str(mutrate) ' # par = ' num2str(npar)])
149.    disp(['#generations=' num2str(iga) ' best cost='
  num2str(cost(1))])
150.    disp('best solution')
151.    disp(num2str(par(1,:)))
152.    disp('Hybrid Genetic Algorithm and Artificial Fish
  Swarm Algorithm')
153.    disp(['each parameter represented by ' num2str(nbits)
  ' bits'])
154.    % % minc=minc(end-popsize+1:end,:);
155.    % minc1=minc(:,1)';
156.    % mH=minc(:,2:end)';
157.    % % minc1(1)=Vv;
158.    % % mH(1)=Ll;
159.    for y=1:length(minc1)
160.    if isnan(minc1(y))==0
     a. z=minc1(y);
     b. break
161.    end
162.    end
```

```
163.    figure
164.    % plot(1:Ngen,minc1(end-Ngen+1:end));
165.    plot(0:Ngen,[z,minc1(end-Ngen+1:end)],'LineWidth',2);
166.    xlabel('generation');ylabel('cost');
167.    title(['HGAFSA Optimization Curve for
  ',num2str(size(mH,1)),' - Units'])
168.    % figure
169.    % for k=1:size(mH,1)
170.    % plot(1:size(mH,2),mH(k,:));
171.    % hold on
172.    % end
173.    % xlabel('generation');ylabel('parameter(X)');
174.    % title('HGAFSA')
175.    disp(['Optimum Cost =  ',num2str(minc(end,1))])
176.    disp(['Best Value of X =  ',num2str(minc(end,2:end))])
177.    %text(0,minc(1),'best');text(1,minc(2),'population
  average')
178.    rmpath('AFSA');
179.    rmpath('BCGA');
180.    rmpath('Integrators');
181.    end
```

APPENDIX B2

m.File: Sub-Function "ELD encoder algorithm"

```
1. function Xeld=eldencoder(X)
2. [ Data,Ptotal] = generation_data;
3. Xi=X;
4. X=abs([X,X.*sin(2*pi*X),X.*cos(2*pi*X),sin(2*pi*X),cos(2*pi
   *X),log(X),cumsum(X,2),cumsum(X,1)]);
5. [a,b]=size(X);
6. % X=[X,rand(size(X)).X];
7. % Xeld=[];
8. Ptotal=Ptotal-sum(Data(:,2));
9. % while size(Xeld,1)<size(X,1)
10.     Xl=[];
11.     for loc=1:size(Data,1)
12.     for kk=1:size(X,1)
13.     Xo(kk,:)=(Data(loc,3)-Data(loc,2))*X(kk,:);
14.     end
15.     Xl=[Xl,{Xo}];
16.     X=X(randperm(a),randperm(b));
17.     end
18.     X0=[];
19.     for i=1:size(X,1)
```

```
20.     for j=1:size(X,2)
     a. for loc=1:size(Data,1)
     b. Xx(1,loc)=Xl{loc}(i,j);
     c. end
     d. X0=[X0;Xx];
21.     end
22.     end
23.     % Xeld=[];
24.     % while size(Xeld,1)<size(Xi,1)
25.     Xmax=[];
26.     for k=1:size(X0,1)
27.     X0(k,:)=X0(k,:)*Ptotal/sum(X0(k,:))+(Data(:,2))';
28.     Xmax=[Xmax;Data(:,3)'];
29.     %     X0(k,:)=cenforce(X0(k,:),(Data(:,3)-
  Data(:,2))');
30.     end
31.     while isempty(find(sum((X0-Xmax)>0,2)'>0, 1))==0
32.     for i=find(sum((X0-Xmax)>0,2)'>0)
33.     for j=1:size(X0,2)
     a. if X0(i,j)>Data(j,3)
          i. Xs=X0(i,j);
         ii. X0(i,j)=rand*(Data(j,3)-Data(j,2))+Data(j,2);
        iii. d=Xs-X0(i,j);
         iv. while d~=0
```

```
          v.  k=randi(size(X0,2));
34.       %                   k=k+1;
35.       %                   if k>size(X0,2)
36.       %                       k=1;
37.       %                   end
          i.  if X0(i,k)<Data(k,3)
                1. h=rand*(-X0(i,k)+Data(k,3));
                2. if d<h
                     a. X0(i,k)=X0(i,k)+d;
                     b. d=0;
                3. else
                     a. d=d-h;
                     b. X0(i,k)=X0(i,k)+h;
                4. end
         ii.  end
        iii.  end
      b. end
38.     end
39.     end
40.     clc
41.     X0
42.     end
43.     Xeld=X0;
```

APPENDIX B3

m. File: Sub-Function "ObjF_ELD_fuel(X)"

```
1. function [Cost,Xeld] = ObjF_ELD_fuel(X)
2. %OBJF_ELD Summary of this function goes here
3. %   Detailed explanation goes here
4. [ Data] = generation_data;
5. Xeld=eldencoder(X);
6. Cost=0;
7. for i=1:size(Xeld,2)
8. Cost=Cost+Data(i,6)+Xeld(:,i)*Data(i,5)+(Xeld(:,i).^2)*Data
   (i,4)+abs(Data(i,7)*sin(Data(i,8)*(Data(i,2)-Xeld(:,i))));
9. end
10.     [Cost,idc]=sort(Cost);
11.     Xeld=Xeld(idc,:);
12.     Cost=Cost(1:size(X,1),:);
13.     Xeld=Xeld(1:size(X,1),:);
```

APPENDIX B4

m. File: Sub-Function

"compare_ga_afsa(par,new_par,pop,new_pop,opt_type)"

```
1. function [Par,Pop,Minc] = compare_ga_afsa(
   par,new_par,pop,new_pop,opt_type)
2. %COMPARE_GA_AFSA Summary of this function goes here
3. %   Detailed explanation goes here
4. [Value,L]=objF(par);
5. [cost,id]=sortrows([Value,L],1);
6. pop=pop(id,:);
7. par=par(id,:);
8. if opt_type==2
9. minc=cost(1,:);
10.     else
11.     minc=cost(end,:);
12.     end
13.     % NP=size(new_par)
14.     % P=size(par)
15.     if size(new_par,1)<size(par,1)
16.     new_par=[new_par;par(end,:)];
17.     end
18.     % NP1=size(new_par)
19.     % P1=size(par)
20.     %
```

```
21.     % NP=size(new_pop)
22.     % P=size(pop)
23.     if size(new_pop,1)<size(pop,1)
24.     new_pop=[new_pop;pop(end,:)];
25.     end
26.     % NP1=size(new_pop)
27.     % P1=size(pop)
28.     [Value1,L]=objF(new_par);
29.     [cost1,id]=sortrows([Value1,L],1);
30.     new_pop=new_pop(id,:);
31.     new_par=new_par(id,:);
32.     for r=1:size(cost,1)
33.     if opt_type==2
34.     if cost(r,1)<cost1(r,1)
     a. Cost(r,:)=cost(r,:);
     b. Par(r,:)=par(r,:);
     c. Pop(r,:)=pop(r,:);
35.     else
     a. Cost(r,:)=cost1(r,:);
     b. Par(r,:)=new_par(r,:);
     c. Pop(r,:)=new_pop(r,:);
36.     end
37.     else
38.     if cost(r,1)>cost1(r,1)
```

```
    a. Cost(r,:)=cost(r,:);
    b. Par(r,:)=par(r,:);
    c. Pop(r,:)=pop(r,:);
39.    else
    a. Cost(r,:)=cost1(r,:);
    b. Par(r,:)=new_par(r,:);
    c. Pop(r,:)=new_pop(r,:);
40.    end
41.    end
42.    end
43.    [Cost,id]=sortrows(Cost,1);
44.    Pop=Pop(id,:);
45.    Par=Par(id,:);
46.    if opt_type==2
47.    Minc=Cost(1,:);
48.    else
49.    Minc=Cost(end,:);
50.    end
51.    end
```

APPENDIX B5

m. File: Sub-Function "reproduction (repparam)"

```
1. function [par,pop]=reproduction(repparam)
2. lo=repparam{2}(1);
3. hi=repparam{2}(2);
4. nbits=repparam{2}(3);
5. pop=round(rand(repparam{1})); % random population of
6. % 1s and 0s
7. par=gadecode(pop,lo,hi,nbits); % convert binary to
```

APPENDIX B6

m. File: Sub-Function "mutation (pop, mutatparam)"

```
1. function [par,pop]=mutation(pop,mutatparam)
2. mrow=mutatparam{1};
3. mcol=mutatparam{2};
4. nmut=mutatparam{3};
5. lo=mutatparam{4};
6. hi=mutatparam{5};
7. nbits=mutatparam{6};
8. for ii=1:nmut
9. pop(mrow(ii),mcol(ii))=abs(pop(mrow(ii),mcol(ii))-1);
10.     % toggles bits
11.     end% ii
12.     par=gadecode(pop,lo,hi,nbits); % convert binary to
```

APPENDIX B7

m. File: Sub-Function "crossover(pop,crossparam)"

```
1. function [par,pop]=crossover(pop,crosparam)
2. keep=crosparam{1};
3. ix=crosparam{2};
4. ma=crosparam{3};
5. xp=crosparam{4};
6. pa=crosparam{5};
7. Nt=crosparam{6};
8. lo=crosparam{7};
9. hi=crosparam{8};
10.     nbits=crosparam{9};
11.     pop(keep+ix,:)=[pop(ma,1:xp) pop(pa,xp+1:Nt)];
12.     pop(keep+ix+1,:)=[pop(pa,1:xp) pop(ma,xp+1:Nt)];
13.     pop(end,:)=[];
14.     par=gadecode(pop,lo,hi,nbits); % convert binary to
```

APPENDIX B8

m. File: Sub-Function "gadecode(chrom,lo,hi,bits)"

```
1. function f=gadecode(chrom,lo,hi,bits)
2. %   gadecode.m
3. %     Decodes binary encripted parameters
4. %
5. %           f=gadecode(chrom,lo,hi,bits,gray)
6. %           chrom = population
7. %           lo = minimum parameter value
8. %           hi = maximum parameter value
9. %           bits = number of bits/parameter
10.     % Haupt & Haupt
11.     % 2003
12.     [M,N]=size(chrom);
13.     npar=N/bits; % number of variables
14.     quant=(0.5.^(1:bits)'); % quantization levels
15.     quant=quant/sum(quant); % quantization levels
  normalized
16.     ct=reshape(chrom',bits,npar*M)';% each column contains
17.     % one variable
18.     par=((ct*quant)*(hi-lo)+lo); % DA conversion and
19.     % unnormalize varaibles
20.     f=reshape(par,npar,M)';% reassemble population
```

APPENDIX B9

m. File: Sub-Function

"prey(Xi,visual_distance,S,itr,max_itr,ntry,opt_type)"

```
1. function
   X_new=prey(Xi,Visual_distance,S,itr,max_itr,ntry,opt_type)
2. % Execute the preying behaviour in an AFSA
3. % opt_type is:
4. % 1 for maximization; and
5. % 2 for minimization.
6. V=Visual_distance;
7. T=0; % tested trials;
8. X_new=feval('move_fish',Xi,Visual_distance,itr,max_itr);
9. %     for u=1:length(X_new)
10.     %         if X_new(u)>1
11.     %             X_new(u)=X_new(u)-floor(X_new(u));
12.     %         end
13.     %     end
14.     while T<ntry
15.     Xj=feval('move_fish',Xi,V,itr,max_itr);
16.     Yi=objF(Xi);
17.     Yj=objF(Xj);
18.     if opt_type==1
19.     if Yj>Yi
```

```
20.     X_new=feval('new_fish',Xi,Xj,S);
21.     %     for u=1:length(X_new)
22.     %         if X_new(u)>1
23.     %             X_new(u)=X_new(u)-floor(X_new(u));
24.     %         end
25.     %     end
26.     break
27.     end
28.     end
29.     if opt_type==2
30.     if Yj<Yi
31.     X_new=feval('new_fish',Xi,Xj,S);
32.     %     for u=1:length(X_new)
33.     %         if X_new(u)>1
34.     %             X_new(u)=X_new(u)-floor(X_new(u));
35.     %         end
36.     %     end
37.     break
38.     end
39.     end
40.     T=T+1;
41.     End
```

APPENDIX B10

m. File: Sub-Function

"swarm(Xi,Visual_distance,Crowdness_factor,S,itr,max_itr,ntry,opt_type) "

```
1. function
   X_new=swarm(Xi,Visual_distance,Crowdness_factor,S,itr,max_i
   tr,ntry,opt_type)
2. % Execute the swarmming behaviour in an AFSA
3. % opt_type is:
4. % 1 for maximization; and
5. % 2 for minimization.
6. V=Visual_distance;
7. C=Crowdness_factor;
8. Xj=feval('move_fish',Xi,Visual_distance,itr,max_itr);
9. D=feval('distance',Xi,Xj);
10.     [n,~]=size(Xi);
11.     nf=sum(D<V*ones(size(D)));
12.     Xc=(Xi+Xj)/2;
13.     Yi=objF(Xi);
14.     Yc=objF(Xc);
15.     if opt_type==1
```

```
16.     X_new=feval('prey',Xi,Visual_distance,S,itr,max_itr,nt
  ry,opt_type);
17.     %     for u=1:length(X_new)
18.     %         if X_new(u)>1
19.     %             X_new(u)=X_new(u)-floor(X_new(u));
20.     %         end
21.     %     end
22.     if Yc>Yi
23.     if (nf/n)<C
24.     X_new=feval('new_fish',Xi,Xc,S);
25.     %     for u=1:length(X_new)
26.     %         if X_new(u)>1
27.     %             X_new(u)=X_new(u)-floor(X_new(u));
28.     %         end
29.     %     end
30.     end
31.     end
32.     end
33.     if opt_type==2
34.     X_new=feval('prey',Xi,Visual_distance,S,itr,max_itr,nt
  ry,opt_type);
35.     %     for u=1:length(X_new)
36.     %         if X_new(u)>1
37.     %             X_new(u)=X_new(u)-floor(X_new(u));
```

```
38.     %         end
39.     %     end
40.     if Yc<Yi
41.     if (nf/n)<C
42.     X_new=feval('new_fish',Xi,Xc,S);
43.     %     for u=1:length(X_new)
44.     %         if X_new(u)>1
45.     %             X_new(u)=X_new(u)-floor(X_new(u));
46.     %         end
47.     %     end
48.     else
49.     end
50.     end
51.     end
```

APPENDIX B11

m. File: Sub-Function

"chase(Xi,Visual_distance,Crowdness_factor,S,itr,max_itr,ntry,opt_type)"

```
1. function
   X_new=chase(Xi,Visual_distance,Crowdness_factor,S,itr,max_i
   tr,ntry,opt_type)
2. % Execute the chasing behaviour in an AFSA
3. % opt_type is:
4. % 1 for maximization; and
5. % 2 for minimization.
6. V=Visual_distance;
7. C=Crowdness_factor;
8. Xj=feval('move_fish',Xi,Visual_distance,itr,max_itr);
9. D=feval('distance',Xi,Xj);
10.     [n,~]=size(Xi);
11.     nf=sum(D<V*ones(size(D)));
12.     Yi=objF(Xi);
13.     Yc=objF(Xj);
14.     if opt_type==1
15.     X_new=feval('prey',Xi,Visual_distance,S,itr,max_itr,nt
   ry,opt_type);
16.     %     for u=1:length(X_new)
```

```
17.     %         if X_new(u)>1
18.     %             X_new(u)=X_new(u)-floor(X_new(u));
19.     %         end
20.     %     end
21.     if Yc>Yi
22.     if (nf/n)<C
23.     X_new=feval('new_fish',Xi,Xj,S);
24.     %     for u=1:length(X_new)
25.     %         if X_new(u)>1
26.     %             X_new(u)=X_new(u)-floor(X_new(u));
27.     %         end
28.     %     end
29.     end
30.     end
31.     end
32.     if opt_type==2
33.     X_new=feval('prey',Xi,Visual_distance,S,itr,max_itr,nt
  ry,opt_type);
34.     %     for u=1:length(X_new)
35.     %         if X_new(u)>1
36.     %             X_new(u)=X_new(u)-floor(X_new(u));
37.     %         end
38.     %     end
39.     if Yc<Yi
```

```
40.     if (nf/n)<C
41.     X_new=feval('new_fish',Xi,Xj,S);
42.     %     for u=1:length(X_new)
43.     %         if X_new(u)>1
44.     %             X_new(u)=X_new(u)-floor(X_new(u));
45.     %         end
46.     %     end
47.     end
48.     end
49.     end
```