University of Rwanda

College of Science and Technology

ACTIVE POWER LOSS MINIMIZATION BASED HYBRID PARTICLE SWARM OPTIMIZATION-ARTIFICIAL BEE COLONY (PSO-ABC) IN LARGE SCALE POWER SYSTEMS (CASE STUDY: 14 IEEE AND 118 IEEE BUS SYSTEMS)

By: **MUTAGANIRA Fidèle**

Reg. **No: 220020674**

A dissertation submitted in partial fulfillment of the requirement for the degree of MASTERS OF SCIENCE IN ELECTRICAL POWER SYSTEM

In the College of Science and Technology; African Center of Excellence in Energy Studies for Sustainable Development (ACE-ESD)

Advisor: **Alexis MUHIRWA (PhD)**

Co-Supervisor: **Maxime BINAMA (PhD)**

October, 2022

Declaration

I, the undersigned, declare that this dissertation is my original work, and has not been presented for a degree in University of Rwanda or any other universities. All sources of materials that will be used for the dissertation work will have been fully acknowledged.

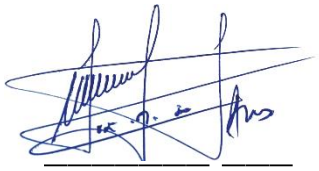
Names: MUTAGANIRA Fidèle

Signature

Approval

Date of Submission: 30th / October/ 2022

This dissertation has been submitted for examination with my approval as a university advisor.



Dr. Alexis MUHIRWA Signature

Abstract

Many decades ago researchers in different fields of science and engineering have been conducting the research on optimization to solve various constrained and unconstrained optimization problems using Particle Swarm Optimization (PSO). This pertinent optimization algorithm has been used in reactive power optimization as well. However, it suffers from premature convergence because it can be easily trapped into local optimum solutions. To improve both the exploration and the exploitation abilities of PSO, this research implements a new algorithm called "Hybrid Particle Swarm Optimization- Artificial Bee colony (PSO-ABC)" which is the hybrid algorithm that combines both Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC) Algorithms. The standard Particle Swarm Optimization (PSO) has better exploration ability but its exploitation ability is very poor. To address these crucial issues, two important modifications will be applied on the standard PSO. The first modification will be to improve the generation of particles in order to allow them to be well spread in search space; then for the second modification all particles which are idle (lazy) during the search process will be replaced by new ones (insertion of scout phase of Artificial Bee Colony Algorithm). The main objective of this research is to minimize the losses within the electric power system by not only satisfying various constraints but also by adjusting control variables (control the voltage at all generator buses, control the tap setting position for all transformers and control the size of shunt capacitors). This new algorithm is applied on both IEEE 14 Bus System and IEEE 118 Bus System to show its better performance in comparison with other cited in the literature review.

The algorithm and simulation were developed in MATLAB 2015a and their have shown that the active power optimization using Hybrid PSO-ABC optimizes better than PSO does and gives accurate results comparing with the results obtained using PSO Algorithm. The algorithm base PSO has been test on IEEE 118-bus test system for reactive power optimization and it reduced the active power losses from 133.694MW up to 112.4116MW while by applying the Hybrid PSO-ABC the active power losses reduced up to 108.5942MW.

Table of content

Contents

List of Acronyms and Abbreviations

Abbreviations

ARCBBO	Adaptive Real Coded Biogeography-Based Optimization
ACO	Ant Colony Optimization
ABC	Artificial Bee Colony
APLM	Active Power Loss minimization
DA	dragon fly algorithm
DE	Differential Evolution
DSA	Differential Search Algorithm
EEA	Efficient Evolutionary Algorithm
FHSA	Differential Search Algorithm
GABC	Gbest-guided Artificial Bee Colony
GA	Genetic Algorithm
GSA	Gravitational Search Algorithm
GWO	Grey Wolf Optimizer
HMPSO-SFLA	Hybrid Modified Particle Swarm Optimization and the Shuffle Frog Leaping algorithms
HSFLA-SA	Hybrid Shuffle Frog Leaping Algorithm and Simulated Annealing
ICBO	Improved Colliding Bodies Optimization
IEM	Improved Electromagnetism-like Mechanism
KHA	Krill Herd Algorithm
LTLBO	Lévy mutation Teaching-Learning-Based Optimization
PSOGSA	Hybrid Particle Swarm Optimization and Gravitational Search Algorithm
IPSO	Improved Particle Swarm Optimization
MDE	Modified Differential Evolution
MICA–TLA	Modified Imperialist Competitive Algorithm and Teaching Learning Algorithm
MGBICA	Modified Gaussian Barebones Imperialist Competitive Algorithm
MO-DEA	Multi-Objective forced initialized Differential Evolution Algorithm
OPF	Optimal Power Flow
PSO	Particle Swarm Optimization
ALC-PSO	Particle Swarm Optimization with an Aging Leader and Challengers

RPLM	Reactive Power Loss minimization
TLBO VAR	Teaching-Learning-Based Optimization

Acronyms

θ_i, θ_j : Voltage angle at ith and jth bus

P_i : Active power at ith bus

Q_i : Reactive power at ith bus

P_{G_i} : Active power generated at ith bus

Q_{G_i} : Reactive power generated at ith bus

P_{D_i} : Active power distributed at ith bus

Q_{D_i} : Reactive power distributed at ith bus

g_k : Conductance of transmission line

G_{ij} : Transfer conductance between line ith and jth bus

B_{ij} : Transfer susceptance between line ith and jth bus

Q_{C_i} : Shunt capacitor connected at ith bus

$P_{\max}$: Minimum and maximum active power

$Q_{\min}, Q_{\max}$: Minimum and maximum Reactive power

$Q_{g_i}^{\min}, Q_{g_i}^{\max}$: Lower and upper reactive power limits at ith generator bus

$V_{L_i}^{\min}, V_{L_i}^{\max}$: Lower and upper voltage limits at ith non-generator bus

$V_{g_i}^{\min}, V_{g_i}^{\max}$: Lower and upper voltage limits at ith generator bus

$T_i^{\min}, T_i^{\max}$: Lower and upper limit of ith branch

N_{bus} : Number of all buses

N_g : Number of generator buses

N_{PQ} : Number of load buses

N_T : Number of transformers

N_C : Number of shunt capacitors

P_{loss} : Active power loss

f_1 : Objective function

F : Augmented objective function

$\lambda_{V_L}, \lambda_{Q_g}$: Penalty factor of voltage at load buses and penalty factor of generator buses

w : Inertia weight factor

$w_{\min}, w_{\max}$: Minimum and maximum inertia weight factor

c_1, c_2, c_3 : Acceleration coefficients

$c_{initial}, c_{final}$: Minimum and maximum acceleration coefficients

r_1, r_2, r_3 : Random number uniformly generated in range [0,1]

$v_{\max}$: Maximum velocity of particle

$x_{id}^{(k)}$: ith particle in d-dimension at kth iteration

$v_{id}^{(k)}$: Velocity of ith particle in d-dimension at kth iteration

$p_{id}^{(k)}$: Personal best of ith particle in d-dimension at kth iteration

$p_g^{(k)}$: Global best particle at kth iteration

$x_{id}^{\max}, x_{id}^{\min}$: Upper and lower limit of ith particle position with d-dimension

$v_{id}^{\max}, v_{id}^{\min}$: Upper and lower limit of ith particle velocity with d-dimension

u_{lower}, u_{upper} : Lower and upper limit of discrete variables

$p_{mean,d}^{(k)}$: Mean value of particle personal best position in d-dimension at kth iteration

$p_{g\_subswarm}^{(k)}$: Global best position of the sub-swarm at kth iteration

$p_{g\_swarm}^{(k)}$: Global best position of the whole swarm at kth iteration

z_{id} : Chaotic vector of ith particle in d-dimension

s_z : Step size

M : Intervals of discrete variables

1. INTRODUCTION

1.1. Background of the research

Within deregulated electric power systems, optimal power flow (OPF) plays a tremendous role in their operation as well as analysis for not only the security but also its economic stabilized operation, i.e. electrical energy supply at minimum cost with improved power quality [2]. Active power loss minimization is very important in an electric power system since it helps to improve the voltage profile of the system, to reduce losses in transmission and distribution lines by satisfying some physical and practical constraints.

In a power system, proper adjustment of control variables (decision variables) such as voltage at generating plant, transformer tap positions, and shunt capacitor banks at proper values contributes to the control of reactive power flows hence minimization of active power losses within the network. Optimal power flow mainly aims at finding proper values for all these control variables aforementioned in order to improve the voltage level of the system. But the voltage profile of the network is the main indicator of poor power factor and hence high losses in the network as well.

Around 1990's, a good number of popular algorithms that are basically nature-inspired have been couched. These include Particle Swarm Optimization (PSO), Genetic Algorithm (GA), Differential Evolution (DE), and Ant Colony Optimization (ACO). There have been modifications of their corresponding operators, strategies and coding in order to be tailored to the application of various science domains [3] Typically, Modified Particle Swarm Optimization (MPSO) has been featured with the velocity reflection and Set-On Boundary strategies [4], and the Modified Differential Evolution (MDE) has benefited from the random uniform distribution in order to tune its mutation and crossover [5].

On the other hand, such nature-inspired techniques of optimization have become a hot spot in power systems to sort out the OPF problem by identifying the optimized parameters. For instance, optimization algorithms of Artificial Bee Colony (ABC) [6], Gbest guided Artificial Bee Colony (GABC) [7], Hybrid Particle Swarm Optimization and Gravitational Search Algorithm (PSOGSA) [8], Particle Swarm Optimization with an Aging Leader and Challengers (ALC-PSO) [9], Hybrid Modified Particle Swarm Optimization and the Shuffle Frog Leaping algorithms (HMPSO-SFLA)

[10], Fuzzy Harmony Search Algorithm (FHSA) [11], Grey Wolf Optimizer (GWO) [12], Efficient Evolutionary Algorithm (EEA) [13], Hybrid Shuffle Frog Leaping Algorithm and Simulated Annealing (HSFLA-SA) [14], Improved Electromagnetism-like Mechanism (IEM) [15], Adaptive Real Coded Biogeography-Based Optimization (ARCBBO) [16], Krill Herd Algorithm (KHA) [17], Differential Search Algorithm (DSA) [18], Modified Gaussian Barebones Imperialist Competitive algorithm (MGBICA) [19], Lévy mutation Teaching-Learning-Based Optimization (LTLBO) [20], Teaching-Learning-Based Optimization (TLBO) [21], Multi-Objective forced initialized Differential Evolution Algorithm (MO-DEA) [22], Improved Colliding Bodies Optimization (ICBO) [23], Gravitational Search Algorithm (GSA) [24], and hybrid of Imperialist Competitive Algorithm and Teaching Learning Algorithm (MICA–TLA) [25].

1.2. Problem statement

With increasing access to electricity and technologies advances, there has been a growing need for an OPF tool able to assess the state and recommend the course of control actions since the first research work publication on OPF in the 1960's. However, the OPF limitations and promise keep posing challenges to electricity dispatching industry [26].

Minimization of Active power losses is a very complex optimization with nonlinear objective function and constraints, wherein there is occurrence of many uncertainties in the large-scale power system. From the consumer side, the demand is very high; this may give rise to instability in voltage, which may eventually lead to the voltage collapse. Voltage control in the power systems mainly goes hand-in-hand with reactive power control, so as to maintain the voltage within acceptable limits, which reduces the occurrence of active power losses. Proper adjustment of the system parameters such as voltage at the generator bus (continuous variables), transformer tap position (continuous variables) and shunt capacitor banks (discrete variables) contribute to good performance of the power system in terms of security, stability, and quality. Presence of both continuous and discrete variables makes the optimization of reactive power becoming a complex combinatorial optimization problem which requires robust, relevant, and competent algorithms to find an optimum solution in order to achieve minimum active power loss and enhance good voltage profile of the entire power system. Although the standard PSO algorithm has better exploration ability and is contributi[1]ng to active power loss minimization, its exploitation ability is very poor. Its contributions do not reach the lower stage of

minimization due to its exploration weakness. The already available literature shows that, for the power systems of IEEE 14 and IEEE 118, the possible lower stage of active power minimization cannot go below 12.9MW. M Vishnu implemented an algorithm based optimization of reactive power and they got aminimum of active power of 13.546MW[27], S Pandya, R Roy implemented an algorithm that optimized IEEE14 bus and IEEE 118 system for improved PSO and the minimum active power loss was 13.101 MW and 133.101 respectively [28]. To address these crucial issues to more reduce active power loss, two important modifications have been applied on the standard PSO. The first modification involves the improvement of particles generation in order to allow them to be well spread in the search space; then for the second modification all particles which were idle (lazy) during the search process have been replaced by new ones (insertion of scout phase of Artificial Bee Colony Algorithm). This research intends to address this issue by reaching the global minimum solutions from local minimum solutions. This has been achieved through the modification of PSO, hybridized with one phase of ABC Algorithm, the results of which were better than those of PSO alone.

1.3 Objectives of the research

1.3.1 Main objective

The main aim of this research is to minimize the active power within large-scale power system by controlling voltage at generation stations, tap setting of transformers, and capacitor bank sizes.

1.3.2 Specific objectives of the research

- To develop an algorithm based PSO which can minimize the active power losses in Large-scale power system, and improve the generation of particles in order to allow them to be well spread in the search space.
- To propose a hybrid algorithm which combines both Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC) algorithms by replacing all particles which are idle (lazy) during the search process with new ones (insertion of scout phase of Artificial Bee Colony Algorithm) to achieve minimum active power loss in large scale power system.
- To apply both algorithms on IEEE 14-bus and IEEE 118-bus systems to find the accurate values of control variables (voltage at generator bus, transformer tap position, and shunt

capacitor), which can minimize the active power loss and improve the voltage profile of the entire system.

1.4 Research Questions

The following research questions are to be asked:

- What are the effects of active power loss in large-scale power system?
- How can electrical power be supplied at minimum cost with improved power quality?
- What are the challenges related to poor regulation of control variables (control the voltage at all generator buses, control the tap setting position for all transformers and control the size of shunt capacitors) in the large-scale power system?

1.5 Scope of the study

This research is strictly delimited on the use of standard PSO that is hybridized by adopting one of ABC stages (Scout Bee Colony) in the active power loss minimization. The power system application aims at voltage at generator bus, transformer tap position, and shunt capacitor issues that arise when the large-scale power system is based on particle swarm algorithm.

1.6 Research Contribution

Most of the countries worldwide are faced with energy efficiency hindrances. Energy optimization would be the solution to this issue. Initialization of PSO algorithm can improve the capability of the algorithm to handle complex multimodal functions and improve the computational ability of the algorithm by shifting from local minimum to global minimum. The output of this research is a useful input in optimization of power systems under construction, thus its great relevance is within developing countries where the power infrastructure development is at the inception stage. Moreover, the present study can be used as reference in energy planning and decision making when it comes to the determination of voltage at generator bus, transformer tap position and shunt capacitor of the power system.

1.7. Research organization

This dissertation is composed of 5 chapters. Chapter 1 offers the Introductory content to the presented research; Chapter 2 concerns itself with the literature review of different related works; Chapter 3 is the methodology, where previously used research methods have been presented as compared to the proposed one; and Chapter 4 presents the study results and associated analysis. Finally, in the last chapter (Chapter 5), general concluding remarks are drawn and future research directions are proposed.

2 LITERATURE REVIEW

2.1 Basic concepts

2.1.1 Power loss

Power loss is the difference between the input and output power of a device, apparatus, pump set, or process. With electrical and electronic devices, as well as pumps, apparatuses and processes, this undesirable loss is converted into heat. Electric power losses are wasteful energy caused by external factors or internal factors, and energy dissipated in the system [6, 8, 10]. They include losses due to resistance, atmospheric conditions, theft, miscalculations, and losses incurred between sources of supply to load centre or consumers. The Nigerian power system network, like all other power systems, waves about the entire country and it is by far the largest interconnection of a dynamic system in existence to date. No matter how carefully the system is designed, losses are present [29] Loss minimization and quantification is very vital in all human endeavors. In power system, it can lead to more economic operation of the system. If we know how the losses occur, we can take steps to limit and minimize the losses. Consequently, this will lead to effective and efficient operation of the system. Therefore, the existing power generation and transmission can be effectively used without having the need to build new installations and at the same time save cost of losses. Basically, losses in electrical power system can be identified as those losses caused by internal factors known as technical losses and those cause by external factors are called non-technical losses.

The Nigerian electricity grid has a large proportion of transmission and distribution losses (whopping 40%). This is attributed to technical losses and non-technical losses. Due to the size of the area the power system serves, the majority of the power systems are dedicated to power transmission. Generally, system losses increase the operating cost of electric utilities and consequently result in high cost of electricity. Therefore, reduction of system losses is of paramount importance because of its financial, economic, and socio-economic values to the utility company, customers, and the host country. However, low losses in transmission system could be achieved by installing generating stations near the load centers power flow, or load flow, is widely used in power system operation and planning. The power flow model of a power system is built using the relevant network, load, and generation data. Outputs of the power flow model include voltages at different buses, line flows in the network, and system losses.

2.1.2 Power-flow techniques

Power-flow or load-flow studies are important for planning future expansion of power systems as well as in determining the best operation of existing systems.

Load flow solves a set of simultaneous non-linear algebraic power equations for the two unknown variables (|V| and ∠δ) at each node in a system. To solve non-linear algebraic equations it is important to have fast, efficient and accurate numerical algorithms[30].

a) ***Gauss-Seidel method for power flow solution***

In this method, first an initial estimate of bus voltages is assumed. By substituting this estimate in the given set of equations, a second estimate, better than the first one, is obtained. This process is progressively repeated, eventually leading to the obtention of better estimates until the difference between two successive estimates become lesser than a prescribed tolerance[1]. When the Gauss-Seidel iterative procedure is used, the voltage at the k^{th} bus during $h+1^{th}$ iteration can be computed as:

$$V_k^{h+1} = \frac{A_k}{V_k^{h*}} - \sum_{m-1}^{k-1} B_{km} V_m^{h+1} - \sum_{m-k+1}^{N} B_{km} V_m^{h} \qquad \text{for k=1,2,..,N} \quad k \neq s \tag{2.1}$$

b) ***Newton Raphson for power flow solution***

Newton Raphson (N.R) method of solving power flow is based on Newton Raphson method of solving a set of non-linear algebraic equations[31]. Newton Raphson (N.R) method of solving power flow problem has very good convergence characteristics. Even for large systems it takes only two to four iterations to converge. Power flow model of NR method is such that the equations describing the performance of the network in the bus admittance form is given by:

$$I = YV \tag{2.2}$$

Matrix and equations:

$$\begin{bmatrix} I_1 \\ I_2 \\ . \\ . \\ . \\ I_N \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} & . & . & . & Y_{1N} \\ Y_{21} & Y_{22} & . & . & . & Y_{2N} \\ . & . & & & & . \\ . & . & & & & . \\ . & . & & & & . \\ Y_{N1} & Y_{N2} & . & . & . & Y_{NN} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ . \\ . \\ . \\ V_N \end{bmatrix}$$

Load flow methods include Gauss-Seidel, Newton Raphson and Fast Decoupled.

Table 2.1 Comparison of the Load Flow Methods

Load Flow Method	**Speed**	**Accuracy**
Gauss-Seidel	Very Slow	Approximate
Newton Raphson	Slow	Accurate
Fast Decoupled	Fast	Accurate

In electrical grid systems, reactive power is the power that flows back from a destination toward the grid in an alternating current scenario.

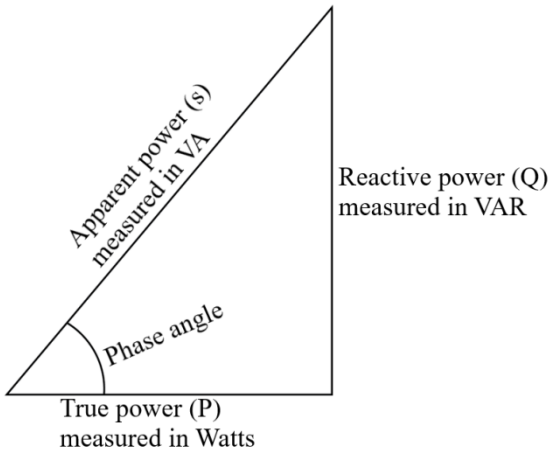


Figure 2.2 The power triangle

Reactive Power is the power which flows back and forth. This means that it moves in both directions in the circuit or react upon itself, hence the name “Reactive Power”. Reactive power is measured in kilovolt ampere reactive (kVAR) or MVAR. Reactive power is a type of power that does no real work

and is generally associated with reactive elements (inductors and capacitors). For example, the inductance of a load such as a motor causes the load current to lag behind the voltage. Power appearing across the inductance sloshes back and forth between the inductance itself and the power sources, thus producing no network. For this reason it is called imaginary or reactive power since no power is dissipated or expended. It is expressed in units of volt-ampere-reactive or var. In the sinusoidal case, the reactive power is simply defined as:

$$Q = S \times \sin\theta_1 = \frac{V_1 I_1}{2} \sin\theta_1 I_{1rms} \sin\theta_1 \qquad (2.3)$$

It is the portion of power in quadrature with the active power and demonstrates the relationship between P, Q, and S in sinusoidal condition. There is some disagreement among harmonics analysts on how to define Q in the presence of harmonic distortion. If it were not for the fact that many utilities, which produce magnetic fields in transmission lines, measure Q and compute demand billing from the power factor computed by Q, it might be a moot point. It is more important to determine P and S; P defines how much active power is being consumed while S defines the capacity of the power system required to deliver P. Q is not actually very useful by itself. However, Q1 the traditional reactive power component at fundamental frequency, may be used to size shunt capacitors.

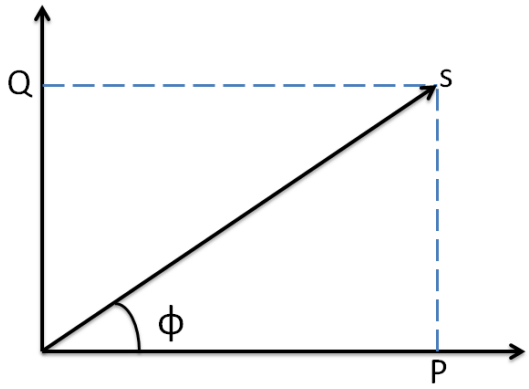


Figure 2.3 Relationship between P, Q, and S in a sinusoidal condition

The reactive power, when distortion is present, has another interesting peculiarity. In fact, it may not be appropriate to call it reactive power. The concept of var flow in the power system is deeply ingrained in the minds of most power engineers[32]. What many do not realize is that this concept is valid only in the sinusoidal steady state. When distortion is present (as in the case of induction

motors), the component of S that remains after P is taken out, is not conserved—that is, it does not sum to zero at a node. Power quantities are presumed to flow around the system in a conservative manner.

This does not imply that P is not conserved or that current is not conserved because the conservation of energy ktnd Kirchhoff's current laws are still applicable for any waveform. The reactive components actually sum in quadrature (square root of the sum of the squares). This has prompted some analysts to propose that Q be used to denote the reactive components that are conserved and introduce a new quantity for the components that are not. Many call this quantity D, for distortion power or, simply, distortion volt amperes. It has units of volt amperes, but it may not be strictly appropriate to refer to this quantity as power, because it does not flow through the system as power is assumed to do. In this concept, Q consists of the sum of the traditional reactive power values at each frequency. D represents all cross products of voltage and current at different frequencies, which yield no average power. P, Q, D, and S are related as follows, using the definitions for S and P above as a starting point:

$$S = \sqrt{P^2 + Q^2 + D^2} \tag{2.4}$$

$$S = \sqrt{P^2 + Q^2 + D^2} \tag{2.5}$$

$$Q = \sum_k V_k I_k \sin\theta_k \tag{2.6}$$

$$Q = \sum_k V_k I_k \sin\theta_k \tag{2.7}$$

Therefore, D can be determined after S,P and Q by $D = \sqrt{(S^2 - P^2 - Q^2)}$. Some prefer to use a three-dimensional vector chart to demonstrate Reactive Power relationships from the components. P and Q contribute the traditional sinusoidal components to S while represents the additional contribution to the apparent power by the harmonics. There are many factors to consider when determining and measuring reactive power in an AC circuit in any power plant:

a) Real power
b) Voltage level
c) Purely resistive true power

d) Active reactive and apparent power
e) Active and reactive powers
f) How to absorb reactive power

2.1.3 Role of Active Power and Reactive Power

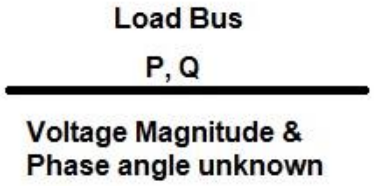
There is an important relationship between active and reactive power and the post below will help to understand that why active power (P) is called true power and reactive power (Q) is called imaginary power. Explanations given in this article are rarely available in the books. First understand what is a coil and inductor. Take an iron rod, wrapped (i.e. winding) it with copper wire. It is a coil or you can say inductor, electromagnet etc. If current passes through the copper wire then iron rod gets magnetized[33]. More will be the current, more the magnetism in the iron rod (i.e. more the flux in iron rod & more the magnetic field around it). Or, it can be said that more the current, more the energy stored by the inductor. (Energy stored by the inductor is given by ½ LI^2, where L is the inductance of the inductor, while I stands for the magnitude of current through the inductor).[34]

2.1.4 Load Bus, Generator Bus and Slack Bus

Power System is nothing but the interconnection of various buses. Each of these buses are associated with four electrical parameters namely voltage with magnitude and phase angle, active power and reactive power. These fours parameters are not completely know but in practical situation only two are known and the remaining two parameters are calculated using Load Flow Analysis. Bus classification is based on the known parameters. On this ground, buses are classified into three types: Load Bus, Generator Bus and Slack Bus.[35]

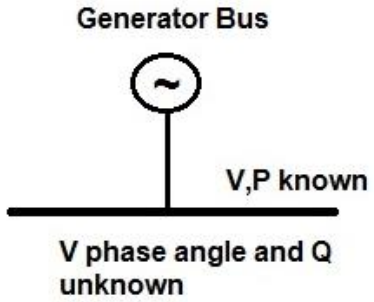
a) ***Load Bus (PQ Bus)***

For load bus real power P and reactive power Q are known but magnitude and phase angle of bus voltage is unknown. It is desired to find bus voltage using load flow analysis. At load bus, voltage is allowed to vary within some specified limit say 5%. This is the reason bus voltage is not important and hence not mentioned.



b) ***Generator Bus (PV Bus)***

Generator Bus is also called voltage controlled bus. Generator is connected to this bus and therefore bus voltage corresponding to generation voltage and active power generation corresponding to generator rating is specified for this bus[36].



c) S***lack Bus (Reference Bus)***

Slack Bus is also known as Swing or Reference Bus. Slack bus does not exist in real life rather it is assumed for the consideration of losses occurring during power transmission. Actually there exist only two buses in power system, Load Bus and Generator Bus for which active power is specified. Since active power delivered by Generator Bus and consumed by Load Bus differ, this means that a power loss equal to the difference between Generator Bus P and Load Bus P is occurring. This loss can only be calculated after the solution of Load Flow[35]. Therefore to supply power loss, an extra generator bus is considered for which bus magnitude and voltage is specified and active power and reactive power are to be calculated. Mind that this active power of slack bus is the equivalent power loss occurring in different system. Generally phase angle of slack bus is taken for reference for the entire load flow solution. Therefore this bus is also called Reference Bus. Table below gives a quick comparison of the three buses.

Table 2.2 Comparison of three types of bus

Bus Type	**Known Parameter**	**Unknown Parameter**
Load Bus	P, Q	V, phase angle
Generator Bus	P, V (magnitude)	Q, Voltage phase angle
Slack Bus	Voltage magnitude & phase angle∠	P, Q

2.2 Review of Previous Works

C. Mamandur et al. [25, 37] presented a mathematical formulation of the optimal reactive power control (optimal VAR control) for minimizing the active power losses in the electrical system. Utilizing the dual linear programing and employing the method of linearized sensitivity relationship of power system, they have determined the adjustments of control variables. In R. R Shoults and D. T Sun [38] formulated the decoupled principle well recognized within bulk power transmission system load flow. This principle shows the decomposition of OPF into distinct models: P-problem known as $P-\theta$ real power model, and Q-problem known as $Q-V$ reactive power model. The P-problem was defined as the minimization of hourly production cost by controlling the generator real power output and tap settings on phase shift transformers. On the other hand, the Q-problem was defined as active power losses minimization in transmission lines by controlling generator voltages, transformer tap settings, and shunt capacitors/reactors. The nonlinear minimization technique and first order gradient method were used, and they have reflected the change in control variables thanks to the active penalty functions.M. Olofsson et al.[39] proposed an algorithm based on successive linear programming with consideration of a second order sensitivity approximation of the total active power loss. J. Nanda et al. [14], developed a new algorithm for OPF by using Fletcher's Quadratic programming. They have considered two sub-problems, namely the minimum cost of generation and the minimum transmission losses. They have sequentially solved for: (1) optimal real and reactive generation and transformer tap setting while taking into consideration the system operation constraints on the generation, (2) busbar voltages and, (3) line flows limits. By using the linear decreasing inertia weight, Y. Arane and M. Boudour [40] improved the PSO algorithm for reducing the active power losses in 114-bus Algerian power system. Through IEEE 30, the resultant improved algorithm was compared with other algorithms. P. L. Reddy et al [41] insinuated that due to the fact that many transmission companies have been moving into competitive markets, the problem of reactive power gets more worse and becomes more crucial for both planning and operation of the power system. Hence, they proposed a

PSO-based algorithm and constrained on non-linear programing-interior point. That algorithm has been found with abilities to handle both discrete and continuous variables for APLM. Based on the weak global search ability of the PSO and good global search ability of dragon fly algorithm (DA); S. Khunkitti et al [42] proposed a hybrid setting that can combine the two abilities from two algorithms. This was with a purpose to equilibrate both the exploration and the exploitation abilities for solving the problem of reactive power optimization within power systems. Applying this algorithm on both IEEE 14 and IEEE 118 bus systems, they could compare output results with those obtained by other methods. Due to the complexity of OPF mathematical formulation J. C. Bansal et al. [43] proposed an algorithm based on modified ABC by considering the impact of global and local neighborhood for determining the optimal settings of OPF control variables. X. Zhou et al [44] solved the OPF problem by suggesting a new algorithm based on incorporating the cooperation approach within ABC algorithm to improve its performance. Their method was based on using multiple artificial bee colonies to optimize different components of solution cooperatively for improving the performance of the algorithm in finding optimal solution.

2.3 Research gap

Many of these aforementioned methods and techniques used in reactive power optimization (APLM within electric power systems) were in some cases found not to be effective because they were being trapped into local optimum hence the solutions tend to converge prematurely in local optima. Hybridization technique is one of most strong and effective techniques that can solve this problem. The combination of two or more algorithms has proven to improve the optimization performance better than a lone algorithm. This is due to mutual amalgamation of strong points, where the weaknesses associated with one algorithm becomes compensated by the potential of other algorithms, which mitigates efforts and cost to reach the optimal solution. In general, Optimal Power Flow problem has been solved through many methods such as Linear Programming, quadratic programming, non-linear programming, newton-based techniques and interior point methods, but these methods failed to reach the global optimum, they provided local minimum. They don't consider the amelioration and improvement of voltage profile at the node of network.[19] Li and Al [[1]] have proposed hybrid PSO in combining the binary PSO with the discrete PSO, and the result came up with robust and quickly converging algorithm. With the Usage non-linear power optimization techniques, difficulties may arise like being forced to use partition techniques such as Benders decomposition.

Benders' partition algorithm is a two-level decomposition technique, master and slave, which defines an iterative procedure between two ordered levels for the search for the optimal solution [45]. Therefore, there is a need for developing a more sensitive and strong algorithm which is simple and reliable, as it combines PSO with only one phase of Artificial Bee Colony (Hybrid PSO_ABC) so as to optimize the exploitation and exploration ability of generated particles to solve OPF problems, and this leads to global optimum in solving OPF problem. Power losses are minimized compared to other algorithms such as generic algorithms.

2.4 Problem Formulation

The real power losses of the system are obtained by summing up all the branches' active power losses as presented in Eq. (10). The problem for reactive power optimization was formulated as minimization of the active power losses of the whole electric power system where the Eq. (11) was used as the objective function (f_1) for this problem [12, 46]

$$P_{loss} = \sum_{k=1}^{Nbr} g_k \left(V_i^2 + V_j^2 - 2V_iV_j \cos\left(\theta_i - \theta_j\right)\right) \tag{2.8}$$

$$f_1 = \min\left(P_{loss}\right) \tag{2.9}$$

Where P_{loss} is the total active power loss, V_i, V_j are voltages of ith and jth buses respectively, θ_i, θ_j are voltage angles of ith and jth buses, g_k is the branch conductance. The objective function (f_1) is subjected to different constraints namely equality constrains and inequality constraints.

a) **Equality Constraints**

Equality constraints are power flows, as to Eq. (12) and Eq. (13), and are satisfied in the Newton Raphson load flow computation.

$$P_{G_i} - P_{D_i} - V_i \sum_{j=1}^{N_{bus}} V_j \left(G_{ij} \cos\theta_{ij} + B_{ij} \sin\theta_{ij} \right) = 0, \quad i \in N_{bus} \tag{2.10}$$

$$Q_{G_i} + Q_{c_i} - Q_{D_i} - V_i \sum_{j=1}^{N_{bus}} V_j \left(G_{ij} \sin\theta_{ij} - B_{ij} \cos\theta_{ij} \right) = 0, \quad i \in N_{bus} \tag{2.11}$$

Where P_{G_i} is the active power at i[th] generator bus, Q_{G_i} the reactive power at i[th] generator bus, Q_{c_i} is the reactive power of shunt capacitor at i[th] bus, G_{ij} is the transfer conductance, B_{ij} is the transfer susceptance and N_{bus} is the bus number.

b) **Inequality constraints**

Inequality constraints are of two types namely non-control variables and control variables.

i) Non-control variables.

These variables are voltage (V_L) at all load buses (or PQ buses) and the reactive power at all generator buses (Q_g).

$$V_{L_i}^{\min} < V_{L_i} < V_{L_i}^{\max}, \quad i = 1, 2, 3, ...N_{PQ} \tag{2.12}$$

$$Q_{gi}^{\min} \le Q_{gi} \le Q_{gi}^{\max}, \quad i = 1, 2, 3, ...N_g \tag{2.13}$$

Where $V_{L_i}^{\min}$ is the minimum voltage at i[th] load bus, $V_{L_i}^{\max}$ is the maximum voltage at i[th] load bus, V_{L_i} is the voltage at i[th] load bus, $Q_{g_i}^{\min}$ is the minimum reactive power at ith generator bus, $Q_{g_i}^{\max}$ is the maximum reactive power at ith generator bus, Q_{g_i} is the reactive power at ith generator bus, N_{PQ} is the number of load buses and N_g is the number of generator buses.

ii) Control variables

These variables are voltage (V_g) at all generator buses, all transformer tap position (T) and reactive power (Q_C) at buses where the shunt capacitors are connected.

$$V_{gi}^{\min} < V_{gi} < V_{gi}^{\max}, \quad i = 1, 2, 3, ...N_g \tag{2.14}$$

$$T_i^{\min} < T_i < T_i^{\max}, \quad i = 1, 2, 3, ...N_T \tag{2.15}$$

$$Q_{ci}^{\min} < Q_{ci} < Q_{ci}^{\max}, \quad i = 1, 2, 3, ...N_{Q_C} \tag{2.16}$$

Where V_{g_i} is the voltage at i[th] generator bus, $V_{g_i}^{\min}$ is the minimum voltage at i[th] generator bus, $V_{g_i}^{\max}$ is the maximum voltage at i[th] generator bus, T_i is the transformer tap position at i[th] bus, $T_i^{\min}$ is the minimum value of transformer tap position at i[th] bus, $T_i^{\max}$ is the maximum value of transformer tap position at i[th] bus, Q_{c_i} is the reactive power of shunt capacitor at i[th] bus, $Q_{c_i}^{\min}$ is the minimum reactive power of the shunt capacitor at i[th] bus.

The vector of independent variables (or control variables) u^T is shown in Eq. (2.16).

$$u^T = \left[V_{g_1}, ..., V_{g_{N_g}}, T_1, ...T_{N_T}, Q_{c_1}, ...Q_{c_{N_C}} \right] \tag{2.17}$$

Where the vector of dependent variables (or non-control variables) x^T is represented by Eq. (2.17)

$$x^T = \left[V_{L_1}, ..., V_{L_{N_{PQ}}}, Q_{g_1}, ...Q_{g_{N_g}} \right] \tag{2.18}$$

The reactive power optimization is a constrained optimization problem; therefore, control variables are divided into two types; continuous variables (Voltage at generator buses) and discrete variables (transformer tap position and shunt capacitor) all these control variables are self-constrained they have to be kept within their respective limit during the optimization process [12, 45]

c) **Penalty function**

In optimization problem the "penalty function" is the technique which is mostly used to handle the constraints easily and efficiently. This technique transforms the constraint problem into unconstraint problem by introduction of penalty for the variables violating their limits. If the reactive power at a given generator buses violates its limit, this generator should be brought back within the normal operation reactive power limit and likewise for the loads if the voltage at the PQ buses (load buses) violates its limit, the voltage should be brought back with in the normal operation limit. The load buses and generators buses which have violated their limits are penalized by multiplying them with their respective penalty factors ($\lambda_{V_L}, \lambda_{Q_g}$) and adding them on the main objective function to form the augmented objective function (F), as to Eq. (21). The equality constraints have been satisfied during

the load flow computation by Newton Raphson method. This process is repeated iteratively until the optimal solution is found [16].

$$F = f_1 + \sum_{i=1}^{N_{PQ}^{\lim}} \lambda_{V_L} \left(V_{L_i} - V_{L_i}^{\lim} \right)^2 + \sum_{i=1}^{N_g^{\lim}} \lambda_{Q_g} \left(Q_{gi} - Q_{gi}^{\lim} \right)^2 \tag{2.19}$$

The limit violations at load buses ($V_{L_i}^{\lim}$) and generator buses ($Q_{g_i}^{\lim}$) are given in Eq. (2.19) and Eq. (2.20) respectively.

$$V_{L_i}^{\lim} = \begin{cases} V_{L_i}^{\max} & \text{if} \quad V_{L_i} > V_{L_i}^{\max} \\ V_{L_i}^{\min} & \text{if} \quad V_{L_i} < V_{L_i}^{\min} \end{cases} \tag{2.20}$$

$$Q_{g_i}^{\lim} = \begin{cases} Q_{g_i}^{\max} & \text{if} \quad \mathrm{Q}_{g_i} > Q_{g_i}^{\max} \\ Q_{g_i}^{\min} & \text{if} \quad \mathrm{Q}_{g_i} < Q_{g_i}^{\min} \end{cases} \tag{2.21}$$

Where F is the augmented objective function, λ_{V_L} is the penalty factor for load bus voltage violating its limits, λ_{Q_g} is the penalty factor for generator bus violating its limits, V_{L_i} is the voltage at ith load bus, $V_{L_i}^{\lim}$ is the voltage limit at ith load bus, $V_{L_i}^{\min}$ is the minimum voltage at ith load bus, $V_{L_i}^{\max}$ is the maximum voltage at ith load bus, $Q_{g_i}^{\lim}$ is the reactive power limit at generator bus, $N_{PQ}^{\lim}$ is the number of load buses violating their limits, $N_g^{\lim}$ is the number of generator buses violating its limits.

3. METHODOLOGY

3.1 Review of Previous Methods

3.1.1 Particle Swarm Optimization (PSO) Algorithm

Particle Swarm Optimization is a stochastic population-based algorithm that belongs to the class of metaheuristics. A metaheuristic ("meta" to indicate "higher-level" heuristics) is an algorithm that has been couched to approximately sort out diverse cases of hard optimization problems without going deeper into each problem adaption. Hard optimization problems can be roughly referred to as problems impossible to be solved till a perfect optimality (or until any guaranteed knot is reached) by whatever deterministic (exact) method achievable within a reasonable limit of time. Such problems can be classified according to their ambit of being unconstrained/constrained, discrete/continuous, dynamic/static or multi/mono-objective. Common features of metaheuristics include nature-based inspirations (i.e. learnt from physics, ethology or biology principles), a stochastic element to cater for random variables, a good number of parameters to be tuned to the specific problem and, no objective function gradient or its Hessian matrix [47]

PSO was firstly introduced in 1995 by James Kennedy (Social psychologist) and Russel Eberhart (Electrical engineer) [16]. The PSO development is based on the analogy of social behavior of birds flocking or fish schooling while searching for the food. It mimics the behavioral conduct of a bird flock that flies in alliance within a multi-dimensional space while searching for a certain optimum spot by fine-tuning their motion and intervals for an improved search. For evolution, each PSO particle makes its decision based on experience of its own and the neighbor's experiences. In other words, particles swam gets to approach the optimum thanks to its actual velocity, prior experience and that of its neighbors [48]. This bird-simulation algorithm is an evolutionary computation method that is closely related to GA and, it is applied in various industry and services fields due to its implementation simplicity and efficacy [47].

In the PSO algorithm, the search process is conducted by particles grouped in a swarm where each particle represents a candidate solution. Every particle is assigned a vector position

($x = [x_{1d}, x_{2d}, x_{3d}, ...x_{nd}]$) and vector velocity ($v = [v_{1d}, v_{2d}, x_{3d}, ...v_{nd}]$) and these two parameters change at every iteration of the search.

At the beginning of the search process, particles position and velocity are randomly generated (by means of the uniform random distribution) within their limits. This is done by considering the upper and lower bound and depending on the dimension, as framed in Eq. (1) and Eq. (2).

$$x_{id} = rand() \times \left(x_{id}^{\max} - x_{id}^{\min} \right) \quad \textbf{(3.1)}$$

$$v_{id} = rand() \times \left(v_{id}^{\max} - v_{id}^{\min} \right) \quad \textbf{(3.2)}$$

Particle velocity and position update are represented by Eq. (3) and Eq. (4) respectively.

$$v_{id}^{(k+1)} = wv_{id}^{(k)} + c_1 r_1 \left(p_{id}^{(k)} - x_{id}^{(k)} \right) + c_2 r_2 \left(p_g^{(k)} - x_{id}^{(k)} \right) \quad \textbf{(3.3)}$$

$$x_{id}^{(k+1)} = x_{id}^{k} + v_{id}^{(k+1)} \quad \textbf{(3.4)}$$

Where x_{id} is the position of ith particle in d-dimension, $x_{id}^{\max}$ is the maximum value of ith particle position in d-dimension, $x_{id}^{\min}$ is the minimum value of the ith particle position in d-dimension, v_{id} is the velocity of ith particle in d-dimension, $v_{id}^{\max}$ is the maximum value of ith particle velocity in d-dimension, $v_{id}^{\min}$ is the minimum value of the ith particle velocity in d-dimension, $rand, r_1, r_2$ are random numbers, c_1, c_2 are acceleration coefficients, w is the inertia factor, p_{id} id the personal best position of ith particle in d-dimension, p_g is the global best position of particle.

After updating the velocity and position of each particle, if the value obtained violates the limits (goes beyond the value of the limit set) that value is brought back into the allowed interval. In order to keep particles on good track and stay within the feasible search region during the search process; the usual practice is to set that outlying particle to the minimum or maximum limit depending on the violated bound [12, 16]. Three main components comprise the velocity equation. The first component represents the inertia factor (w), as expressed in Eq. (5).

$$w = w_{\max} - \left(\frac{w_{\max} - w_{\min}}{iter_{\max}} \right) \times iter \quad \textbf{(3.5)}$$

Where $w_{\max}$ is the maximum inertia factor, $w_{\min}$ the minimum inertia factor, and $iter_{\max}$ the current iteration.

The inertia weight intends at controlling the velocity of particles in the search space. The second term stands for the cognitive component that indicates the personal leaning of particle, i.e. the information sharing between personal best particle and the particle itself at each iteration. The third component represents the social component and pertains to the social leaning of particles, i.e. the information sharing between particle and the global best particle of the whole swarm at each iteration [16].

The vector diagram that illustrates the PSO working principle and its corresponding flow chart diagram are depicted through Figure1 and Figure 2, respectively. To ensure the convergence of PSO algorithm a constriction factor "K" (Eq. (6)) was introduced. This factor is multiplied to the velocity update equation to constrict the movement of particles in search space and works interchangeably with inertia weight factor for controlling the convergence of the PSO algorithm [12].

$$K = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, \quad \varphi = c_1 + c_2, \quad \varphi > 4 \qquad \textbf{(3.6)}$$

Where K is the constriction factor and, φ (which is usually greater than four) is the sum of acceleration coefficients.

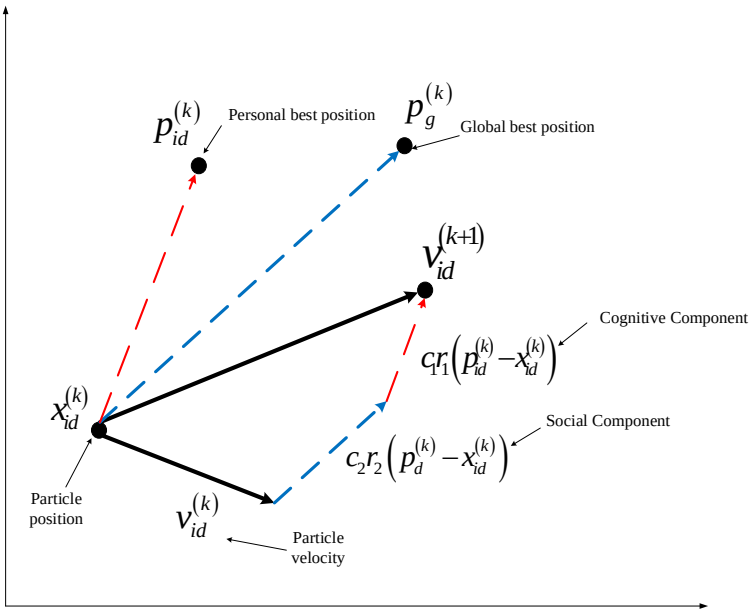


Figure 3.1 Vector diagram of the PSO working principle.

Within the power system community; PSO has stimulated much attention to be used in several optimization problems that seem complex in nature. This is due to its implementation easiness, simple concept, parameters control robustness and computational efficiency [49]. However, some shortfall has been revealed, such as sticking within the local minima, speed of convergence and quality of solution. To that end, a number of PSO variants have been suggested to enhance its performance.

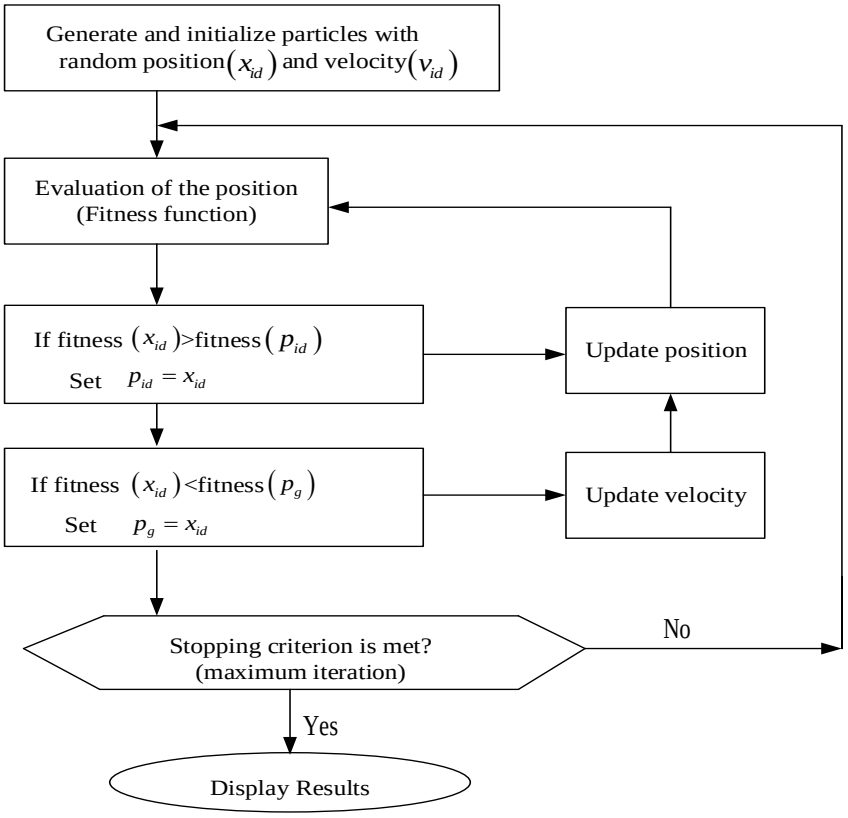


Figure 3.2 PSO algorithm flow chart.

These mostly consist in the improvement of swarm initialization, new parameters introduction (such as the constriction coefficient), different method of defining the inertia weight and introduction of mutation operators from global and local best particles. Another approach has been PSO modification by extending the field searching space, adjusting parameters, and hybrid intervention of another technique [44].

3.1.2 **Artificial Bee Colony (ABC) Algorithm**

Artificial Bee Colony Algorithm belongs to the swarm intelligence of biology-inspired algorithms in the class of metaheuristics. It was proposed by Dervis Karaboga in 2005, with an intent to mimic natural behavior of real honey bees while foraging [12]. Basically, the Artificial Bee Colony Algorithm encompasses four consecutive phases: initialization phase, employee bee phase, onlooker bee phase and the fourth phase is scout bee phase [12, 44, 50].

a) **Initialization phase**

This phase is very important in ABC and in all metaheuristic algorithms in general because the way swarms are generated and spread within the search space determines the diversity of the swarm during the search process [12]. Initial population is randomly generated in similar way like in Eq. (1).

b) **Employee bee phase**

This phase shows the deportment of the employee bees in seeking for better food source within their neighborhoods by means of Eq. (7). The employed bees are deployed to different food sources for seeking the ones having good nectar so that they can be exploited, and each food source is visited by only one employed bee (the food source represents possible solution). After visiting different food sources, employed bees return to the hive to provide information related with food sources [43].

$$v_{ij} = x_{ij} + \phi_{ij}\left(x_{ij} - x_{kj}\right), \quad j \neq k \tag{3.7}$$

Where the parameter ϕ_{ij} is chosen in the interval $[-1,1]$, x_{ij} is the i[th] reference food source, x_{kj} is a random selected food source, v_{ij} is new solution.

$$fit\left(x_i\right) = \begin{cases} \dfrac{1}{\left(1+ f\left(x_i\right)\right)} & if \quad f\left(x_i\right) \geq 0 \\ 1+ abs\left(f\left(x_i\right)\right) & if \quad f\left(x_i\right) < 0 \end{cases} \tag{3.8}$$

Where $fit\left(x_i\right)$ stands for the fitness function, and $f\left(x_i\right)$ the function of i[th] food source.

c) **Onlooker bee**

Back hive, employed bees start the side to side or up and down waggling dance. That waggle dance indicates the direction, position and location of the food sources; then onlooker bees observe well the dance to interpret the information about the location and capacity of the food sources [12]. Based on probability, onlooker bees opt for the best food sources; i.e. the rich food source with a high amount of good nectar. Eq. (9) represents the probability calculation for selection of best food source to be visited [51].

$$p_i = \frac{fit(x_i)}{\sum fit(xi)} \quad \textbf{(3.9)}$$

Where p_i stands for the probability of the food sources.

d) **Scout bee phase**

The food sources whose nectar has been exhausted are abandoned. Employed bees that were deployed to that drained food source became a scout bee, and start to fetch for information about new food sources in the neighborhood. In the ABC Algorithm, a parameter called "limit" is set in order to indicate whether the food source is still having the nectar or not. If the source is exhausted after a certain number of iterations this means the detection of abandonment, an employee bee is immediately changed into scout bee according to Eq. (1) and start to explore new food source [6, 18, 52]. Figure 3 delineates the ABC Algorithm flow.

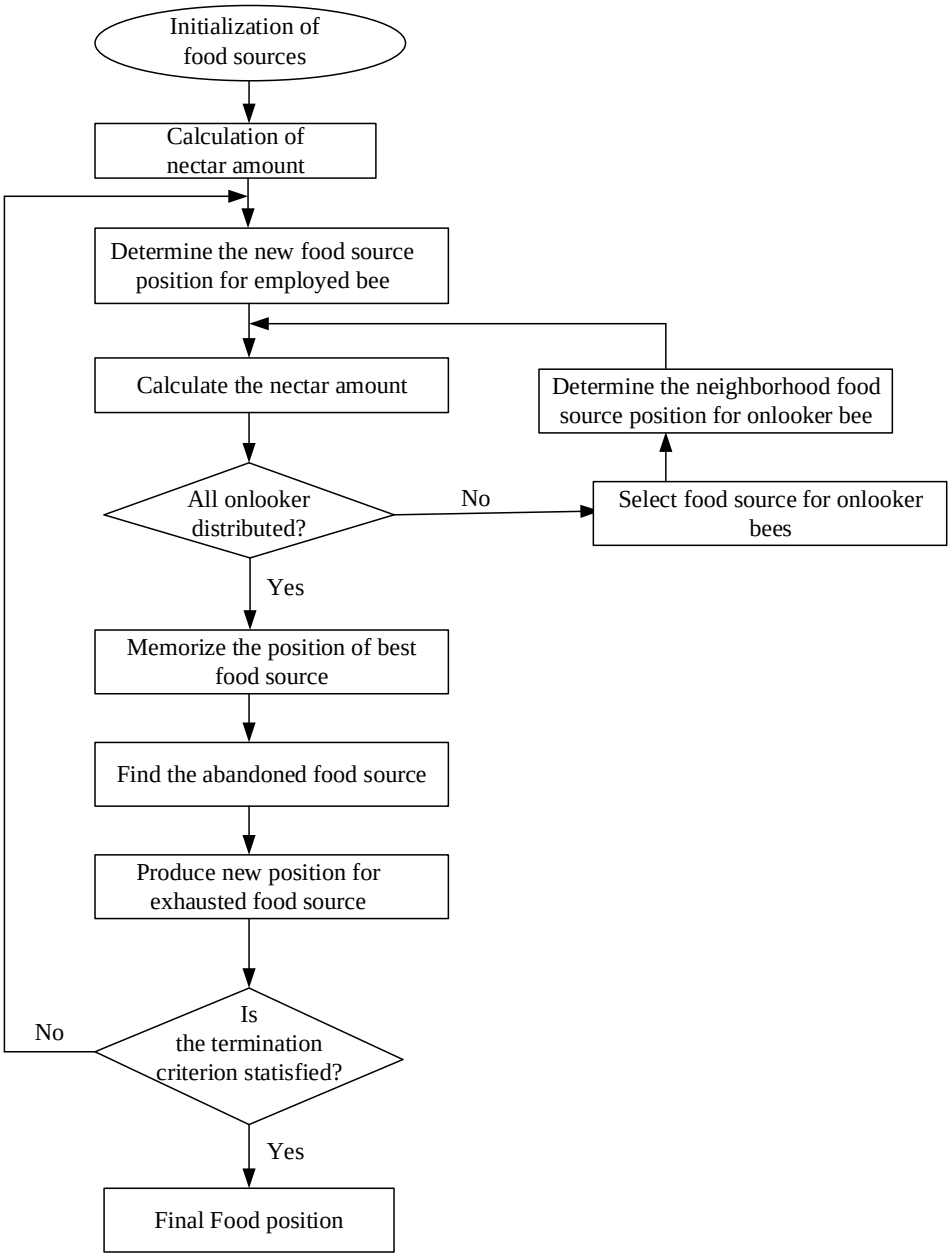


Figure 3.3 Flow chart of ABC Algorithm

3.1.3 **Linear programming (LP)**

Few of recent works in this area have considered the different perspectives of network operation with high amount of constraints and decision criteria [3].

3.1.4 **Quadratic programming (QP)**

Quadratic programming techniques are considered as: nature-swarm-inspired methods, human-inspired algorithms, evolutionary-inspired algorithms, physics-inspired algorithms and ANN.[53]

3.1.5 Newton-based techniques

Different objective functions in power systems can be optimized. These objectives can be the total losses of transmission lines, total generation cost, FACTS cost, voltage deviations, total of power transfer capability, voltage stability, emission of generation units, etc. The control variables of the system can be adjusted during the optimization process. These control variables including the generated active powers, the voltage of generation buses, transformer tap settings [54]. Several classical and modern optimization techniques have been proposed to solve the optimal power flow problem. Most of classical techniques are based on the sensitivity analysis and gradient-based methods. However, different metaheuristic optimization techniques have been proposed to avoid the trapping in local optimum which can be done in the classical technique [55].

3.1.6 **Non-linear programming (NLP)**

These non-linear terms behaves very poorly with commercial mixed integer optimization solvers. In other words for a relatively small problem size, the number of nodes required by the Branch and Bound optimization process coupled with the solution time required by the optimization process could be very large and there exists a possibility of the optimization process never converging [56]. Difficulties encountered in solving nonlinear optimization problems with binary decision variables force the authors to use partition techniques such as Benders decomposition. Benders' partition algorithm is a two-level decomposition technique, master and slave, which defines an iterative procedure between two ordered levels for the search for the optimal solution. The master level in their work represents the decision problem which is defined as a MINLP problem while the slave level deals with the operations problem, being a nonlinear OPF problem. This method allows us an appropriate treatment of the non-convexity associated with the binary variables and divides the total problem in two under problems easy to solve.

3.1.7 **Interior point methods (IPM)**

Interior-point methods (IPMs) are well suited for solving convex non-smooth optimization problems which arise for instance in problems involving plasticity or contact conditions. This work attempts at extending their field of application to optimization problems involving either smooth but non-convex or non-smooth but convex objectives or constraints [57].

3.2 **Proposed Method**

This new optimization algorithm proposed is a hybridization consisting of particle swarm optimization and artificial bee colony algorithms for reactive power optimization to enhance the performance of the existing PSO algorithm. Two particle main major modification will be adopted in order to not only improve the diversity of standard PSO but also to ensure good balance between exploration and exploitation abilities of the algorithm. The first modification will be focusing on improvement of particle generation (Initialization of particles). The commonly used method to generate particles is the "Uniform random generation", this method was found to be somehow week as the particles are not well spread within the search space. This hybridization will adopt chaotic method whereby to methods (Logistic map and tent map) will be intermixed to produce chaos in particle through scout phase of ABC algorithm so that to ensure good dispatch of particles within search space and resurgence of idle particles. The second modification will be focusing on subdivision of swarm into sub-swarms to improve the close communication of particles based on particle best of each sub-swarm and the global best of the entire swarm.

3.1.1 **Initialization of particles**

Initialization phase is very crucial in all metaheuristic algorithms. In PSO algorithm this phase contributes a lot on distribution of particles within search space. In many researches the population has been generated using uniform random distribution, this method has shown a major handicap which is related with weak distribution of particles in the search space. When particles are generated by using normal distribution technique, the particles do not cover the whole area of search space some of them are congested at some areas and letting many gaps appearing in different locations of the search space. During the search process this congestion may constitute a barrier for particles to find the global optimum solution.

To alleviate this scenario, researches have been conducted on initialization of the population (particles) and proven that the chaotic map can bring more improvement in generation of initial population. Chaotic map has the property of generating particles with low discrepancy so that the particles can be well spread in the search space. Two main chaotic maps namely Logistic map and Tent map have been used in this hybridization algorithm for initialization of the population in order to ensure good diversity of the population and to allow the particle exploring well the search space. In Logistic map, the generation of particle follows the equations below:

$$x_{n+1} = \mu x_n (1 - x_n), \quad \mu = 4 \qquad \textbf{(3.10)}$$
$$0 < x_0 < 1$$

While in the Tent map the equation is:

$$x_{n+1} = \begin{cases} \lambda x_n, & 0 \le x_n \le 1/2, \quad \lambda = 2 \\ \lambda (1 - x_n), & 1/2 \le x_n \le 1 \end{cases} \qquad \textbf{(3.11)}$$

Where x_0 is the initial vector, x_n is the vector at nth iteration. The initial vector z_{id} of dimension "d" was generated from logistic map in combination with tent map and the upper and lower bounds of control variables were also considered was used to generate the initial population.

$$x_{id} = x_{\min} + z_{id} \left(x_{\max} - x_{\min} \right) \qquad \textbf{(3.12)}$$

Where z_{id} stands for the vector chao of control variables, $x_{\min}$ is the minimum value of the control variable, and $x_{\max}$ is the maximum value of the control variable.

3.1.2 **Swarm Subdivision**

In the standard PSO algorithm the initial population was mainly formed by unique swarm where whereby there was one personal best particle and one global best particle of the whole swarm at each iteration. If the global best of the swarm is not really the best optimal solution of the swarm all particles may refer to and follow it blindly until they fall into local optimum hence premature converge occurs. In contrary the proposed hybrid algorithm its initial population is formed by multi-swarm, each having its global best ($p^k_{g\_subswam}$), the overall global best ($p^k_{g\_swam}$) and the mean value of

personal best particle ($p^k_{mean,d}$) at each iteration. This subdivision of the swarm into multi-swarms allows the effective communication among particles so that they can explore well the search space and seek for better optimum solution. Velocity update of the particles is shown in the equation below.

$$v_{id}^{(k+1)} = w \times v + c_1 r_1 \left(p^k_{mean,d} - x^k_{id} \right) + c_2 r_2 \left(p^k_{g\_subswarn} - x^k_{id} \right) + c_3 r_3 \left(p^k_{g\_swarm} - x^k_{id} \right), \quad c_2 = c_3 \tag{3.13}$$

Where $p^k_{mean,d}$ is the mean value of the personal best of the subswarm at k[th] iteration, $p^k_{g\_subswarm,d}$ is the global best of the subswarm, $p^k_{g\_swarm,d}$ is the global best of the whole swarm.

The parameters r_1, r_2 and r_3 are also chosen from the matrix generated by logistic map and tent map. The position of particle is updated in similar way as shown in the equation below.

$$x_{id}^{(k+1)} = x_{id}^k + v_{id}^{(k+1)} \tag{3.14}$$

3.1.1 **Adaptive Inertia Weight**

Inertia weight factor contributes much in the convergence of the algorithm because it controls the velocity of the particles. At the early stage of the algorithm the particles need to fly and explore more the search space which means that the inertia weigh must start with high value ($w_{\max}$) the velocity of particle is also controlled in order to not overfly beyond the search space. The velocity of particles is constrained in the interval $[-v_{\max}, v_{\max}]$ because if the maximum velocity of the particle has a significant role in determining the resolution of fitness (the region of search between actual position and targeted position). Later the value of inertial weight reduces progressively up to reach ($w_{\min}$). In this hybridization algorithm the inertia weight has been computed using the following equation.

$$w = w_{\max} - \left(w_{\max} - w_{\min} \right) * \left(\frac{\max_{iter} - iter}{\max_{iter}} \right) \tag{3.15}$$

Where w is the inertia weight factor, $w_{\min}$ is the minimum inertia weight, $w_{\max}$ is the maximum inertia weight, $\max_{iter}$ is the maximum iteration, *iter* is the current iteration.

3.1.2 **Adaptive Acceleration factor**

In standard PSO the constants positive parameters c_1 and c_2 are used for controlling the step size of cognitive and social component of the algorithm. Same parameters have been used in MMSCPSO to control and accelerate the communication and interaction among particles in search space.

c_1 was used to control the self-learning between particles and the mean value of personal best of the sub-swarm, c_2 was used to control the learning between particles and the global best of the sub-swarm and c_3 was used to control the learning between the particles and the global best of the overall swarm. In the algorithm the value of c_2 and c_3 have been set equal and the equations describing the computation of these coefficients are shown in the following two equations.

$$c_1 = c_{final} - \left(c_{final} - c_{initial}\right) * \left(\frac{\max_{iter} - iter}{\max_{iter}}\right) \quad \textbf{(3.16)}$$

$$c_2 = c_{initial} + \left(c_{final} - c_{initial}\right) * \left(\frac{\max_{iter} - iter}{\max_{iter}}\right), \quad c_2 = c_3 \quad \textbf{(3.17)}$$

Where c_{final} stands for the maximum value of acceleration factor, and $c_{initial}$ is the minimum value of acceleration factor.

3.1.3 **Discretization of control variables**

The control variables are of two types; continuous variables (voltage at the generator buses) and discrete variables (transformer tap position and shunt capacitor). During the optimization process the discrete variables may take any value which does not match with practical and commercial value of the components. This problem can be corrected by setting a mechanism which will deal with discretization of these variables. Let u be the vector of discrete variables having in the range $\left[u_{lower}, u_{upper}\right]$, s_z as the step size, $\alpha = \left[0,1,2,3,....M\right]$ be the vector of intervals. Number of intervals M is computed using the equation bellow and the vector of discrete variables is determined as well.

$$M = round\left(\left(u_{upper} - u_{lower}\right) \Big/ s_z\right) \tag{3.18}$$

$$u_i = u_{lower} + \alpha_i \times s_z, \quad i = 1, 2, 3, \ldots n \tag{3.19}$$

The Figure below shows the discrete values and values which need to be discretized because after each iteration some of values may be found in between two consecutive discrete values and eq3.24 is used to find the corresponding discrete value for $\tilde{x}$.

$$u_i = \begin{cases} u_i, & if \quad \left|\tilde{x}_i - u_i\right| \le \left|\tilde{x}_i - u_{i+1}\right| \\ u_{i+1}, & otherwise \end{cases} \tag{3.20}$$

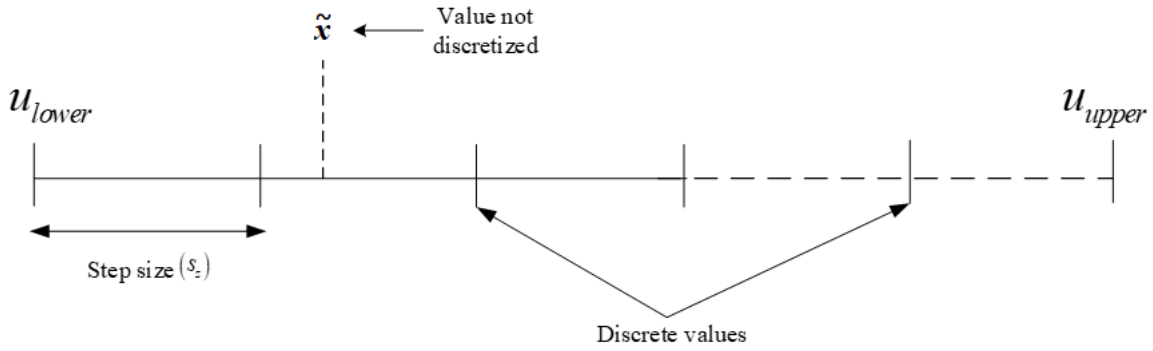


Figure 3.4

Illustration of discretization technique.

Where M stands for the number of intervals, u_{upper} is the upper bound of the discrete variable, u_{lower} is the lower bound of the discrete variable, $\tilde{x}$ is the value to be discretized, u_i is the i[th] discrete variable.

3.1.4 Insertion of Scout phase

The scout phase is normally found in ABC algorithm, in this phase new food source are generated to replace those which have been exhausted in order to improve the fitness. In the normal standard PSO algorithm this mechanism does not exist. The particles which are initialized remain in the search process until the optimal or near optimal solution is found. The particles change their velocities and positions at each iteration but they do not get replaced by new ones for improving their personal best positions, this is one of the handicaps of PSO algorithm because some of particles may stay in the

search space without changing their personal best positions after several iterations and the cognitive component may continue to have same value for a big period of time. To improve the performance of the algorithm this scout phase has been inserted in standard PSO to form new hybridization algorithm. After a given number of iterations a counter (limit) is set for identifying particles which have not been able to improve their personal best positions and the eq.28 is used to generated new particles in the search space which replace the idle particles in order to increase the diversity of the particles and to ensure good balance between exploration and exploitation abilities of the Hybrid PSO-ABC algorithm. Replacement of idle particles by new ones allows the particles in search space to be more competitive and to more communicate among them so that they can optimally find the global solution.

3.1.5 Pseudo code of the Hybrid PSO-ABC algorithm

Step 1: Load the Newton-Raphson (Base case)

Step 2: Generate initial population according to Eq3.10 an Eq3.11

Step 3: Generate initial velocities in range $\left[-v_{\max}, v_{\max}\right]$

Step 4: Set the "**limit**" for idle particles (particles not changing their personal best positions)

Step 5: Evaluate the augmented objective function for the population generated

Step 6: Initialize personal best ($p_{id} = x_{id}$)

Step 7: Choose the best solution having minimum loss ($p_g = \min\left(p_{id}\right)$)

Step 8: Set cycle to 1 and maximum iteration

Step 8: Repeat the process

Step 10: Initialize idle particle counter (Counter=0)

Step 11: Compute the inertia weight by Eq3.15

Step 12: Calculate the acceleration coefficients by Eq3.16 and Eq3.17

Step 13: Compute the velocity by Eq3.13 and control if is within its limit

Step 14: Update the position of particle by equation Eq3.14, conduct discretization by equation Eq3.18, equation Eq.3.19 and equation Eq3.20 then check if the position lies within $[x_{\min}, x_{\max}]$

Step 15: Conduct the evaluation of the augmented objective function

Step 16: If $f(x_{id}^{(k)}) < f(p_{id}^{(k)})$, set $p_{id}^{(k)} = x_{id}^{(k)}$

If $f(p_{id}^{(k)}) < f(p_g^{(k)})$,set $p_g^{(k)} = p_{id}^{(k)}$

Counter=Counter+1

End if

End if

Step 17: Scout phase, Generate new population according to equation…….

Step 18: If *Counter*>= limit

Replace the idle particle by new particle

Set its velocity to zero

Conduct evaluation of the augmented objective function

Counter=0

If $f(x_{id}^{(k)}) < f(p_{id}^{(k)})$, set $p_{id}^{(k)} = x_{id}^{(k)}$

If $f(p_{id}^{(k)}) < f(p_g^{(k)})$,set $p_g^{(k)} = p_{id}^{(k)}$

End if

End if

End if

Step 19: Memorize the global best solution found so far

Step 20: cycle=cycle+1

Step 21: *Until iteration=maximum iteration*

4. RESULTS AND ANALYSIS

This chapter focuses on interpretation and discussion of the results obtained after implementation of both standard PSO and Hybrid PSO-ABC algorithms. The active power loss minimization was conducted using the two aforementioned algorithms and it has been tested on both IEEE 14-bus system and IEEE 118-bus system. The computation of power flow was done by using the Newton Raphson method and implementation of these both algorithms were conducted using MATLAB (2017a) as programming language.

4.1 Considered parameters

All parameters considered during the implementation of both algorithms are shown in Table 4.1. For PSO the population was considered as a whole whereas for the Hybrid PSO-ABC the population was subdivides into three swarms namely swarm 1, swarm 2 and swarm 3 respectively.

Table 4.1 Parameters identification

S/N	Parameter	Value
1	Population size (Number of particles)	50
2	Swarm 1	10
3	Swarm 2	15
4	Swarm 3	25
5	Maximum number of iterations	200
6	Maximum value of inertia weight factor ($w_{\max}$)	0.9
7	Minimum value of inertia weight factor ($w_{\min}$)	0.4
8	Minimum value of acceleration coefficient ($c_{initial}$)	0.5
9	Maximum value of acceleration coefficient (c_{final})	2.5
10	Penalty factors (λ_{V_L}) and (λ_{Q_g})	100000

4.2 IEEE 14-bus test system

The reactive power optimization has been tested on IEEE 14-bus system, the one-line diagram of the system is shown in Fig4.1. The system is composed by 1 slack bus (reference bus), 4 voltage-

controlled buses or PV buses among them there are 1 generator and 3 synchronous condensers, 9 load buses or PQ buses, 20 transmission lines, 3 transformers and 1 shunt capacitor. The system loads total active power and total reactive power are 259MW and 73.5Mvars respectively. The total active power and reactive power losses of the system obtained by applying Newton Raphson are 13.391MW and 55.694Mvars respectively. Total control variables are 9 (which reflects the dimension of the optimization problem) namely voltage generator buses, transformer tap position and shunt capacitor. The step size for transformer tap position is 0.01p.u whereas the step size for switchable shunt capacitor is 5Mvar.

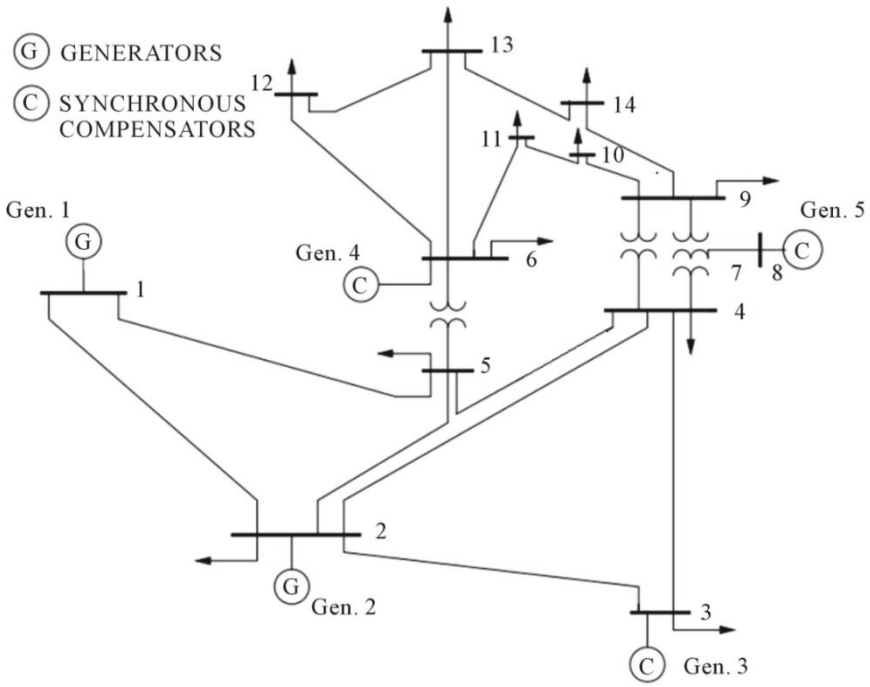


Figure 4. 1 One-line diagram for IEEE 14-bus system

Active power loss minimization based PSO algorithm tested on IEEE 14-bus has reduced the total active power losses of the system from 13.3910MW up to 12.4904MW whereas by using Hybrid PSO-ABC algorithm the losses have been reduced up to 12.2667MW. The Fig4.2 shows the minimization of losses within the system and the Table 4.2 captures the summary of results as comparison of both algorithms after conducting 20 runs. It shows the maximum loss or worse loss among all runs, minimum loss or best loss among all runs and the loss reduction in percentage.

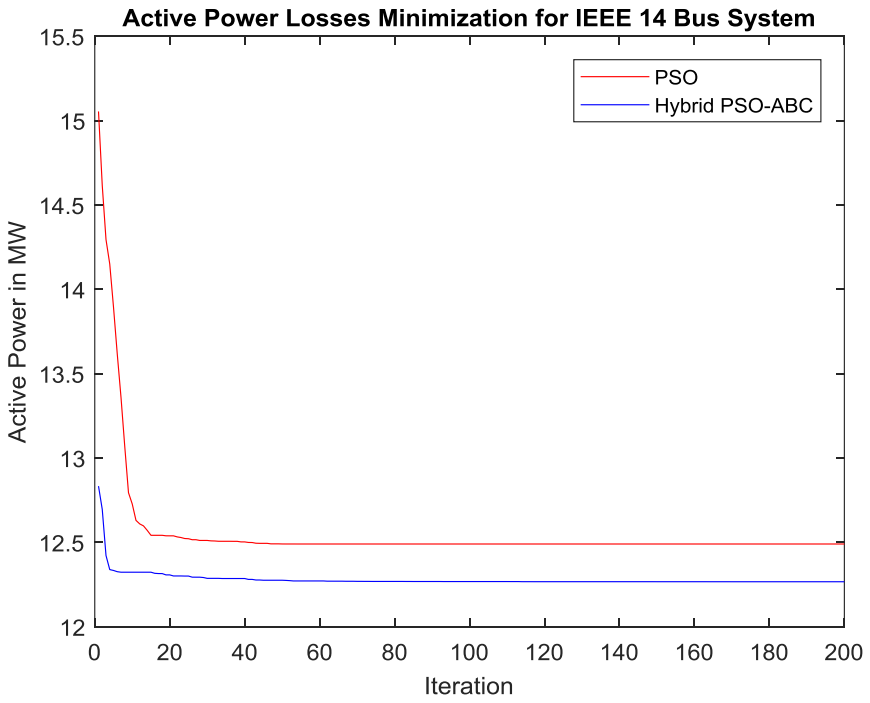


Figure 4. 2 Active power loss minimization in IEEE 14-bus system.

4.3 Hybrid PSO-ABC for IEEE 14-bus system

Table 4. 2 Comparative results of PSO and Hybrid PSO-ABC algorithm for IEEE 14-bus system

Algorithm	Active Power Loss (Mw)			Loss Reduction (%)
	Worst Loss	Best Loss	Mean Loss	
Base Case (Nr)	-	-	13.3910	-
Pso	15.3109	12.4904	12.5397	6.72
Hybrid Pso-Abc	13.2528	12.2667	12.2762	8.39

The Table 4.3 presents the status of control variables (voltage at generator buses, transformer tap position and shunt capacitor) before and after optimization using both PSO and Hybrid PSO-ABC algorithms.

Table 4. 3 Control variables before and after optimization case of IEEE 14-bus system

S/N	Control Variables	Voltage in p.u before optimization	Voltage in p.u after optimization by PSO	Voltage in p.u after optimization by Hybrid PSO-ABC	Minimum	Maximum
1	V_{g1}	1.0600 p.u	1.1000 p.u	1.1000 p.u	0.95p.u	1.1p.u
2	V_{g2}	1.0450 p.u	1.0829 p.u	1.0861 p.u	0.95p.u	1.1p.u
3	V_{g3}	1.0100 p.u	1.0510 p.u	1.0570 p.u	0.95p.u	1.1p.u
4	V_{g6}	1.0700 p.u	1.1000 p.u	1.1000 p.u	0.95p.u	1.1p.u
5	V_{g8}	1.0900 p.u	1.1000 p.u	1.1000 p.u	0.95p.u	1.1p.u
6	T_{4-7}	0.9780	1.0500	1.0100	0.90	1.1
7	T_{4-9}	0.9690	0.9000	0.9100	0.90	1.1
8	T_{5-6}	0.9320	0.9000	1.0000	0.90	1.1
9	Q_{c9}	19Mvars	20.0000Mvars	20.0000Mvars	0	20Mvars

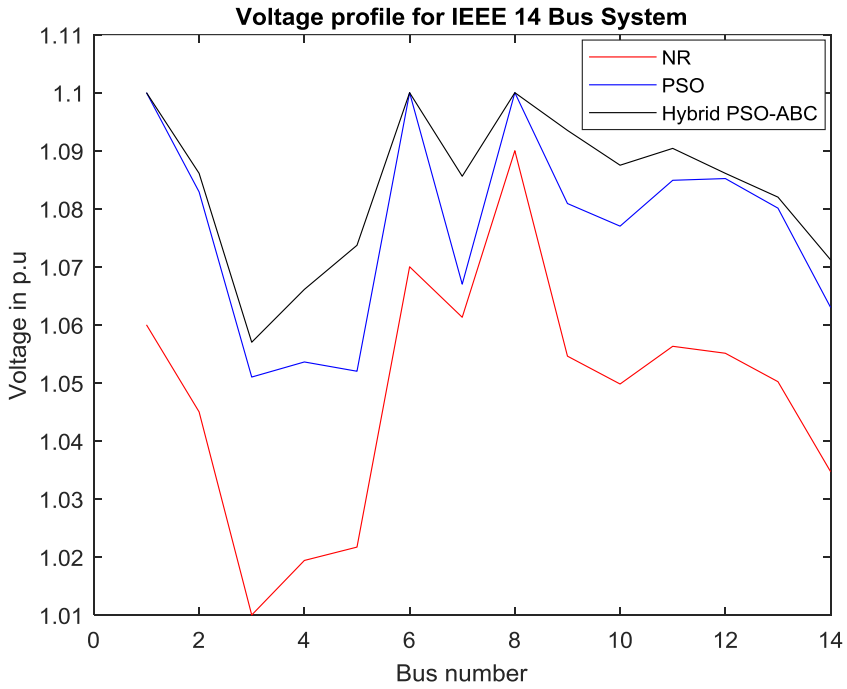


Figure 4. 3 Voltage profile at all buses before and after optimization in IEEE 14-bus system

The Fig 4.3 and Table 4.4 show the improvement of the voltage profile in IEEE 14-bus system. Before optimization the voltage was poor in such way that the active power losses were high. By implementing the algorithms based reactive power optimization and by adjusting all control variables, the voltage level increased hence the active power loss were also reduced.

Table 4. 4 Voltage profile at all buses before and after optimization case of IEEE 14-bus system

Bus Number	Voltage in p.u before optimization	Voltage in p.u after optimization by PSO	Voltage in p.u after optimization by Hybrid PSO-ABC
1	1.0600	1.1000	1.1000
2	1.0450	1.0829	1.0861
3	1.0100	1.0510	1.0570
4	1.0194	1.0536	1.0661
5	1.0217	1.0520	1.0737
6	1.0700	1.1000	1.1000
7	1.0613	1.0670	1.0856
8	1.0900	1.1000	1.1000
9	1.0546	1.0809	1.0935
10	1.0498	1.0770	1.0875
11	1.0563	1.0849	1.0904
12	1.0551	1.0852	1.0861
13	1.0502	1.0801	1.0820
14	1.0347	1.0630	1.0712

4.4 IEEE 118-bus test system

The reactive power optimization has also been tested on IEEE 118-bus system whose one-line diagram is shown in Fig4.4. The system comprises 1 slack bus (reference bus), 53 voltage-controlled buses or PV buses among which there are 18 generators and 35 synchronous condensers, 64 load buses or PQ buses, 177 transmission lines, 9 transformers and 14 shunt capacitors/reactors. The system loads total active power and total reactive power are 4242MW and 1438Mvars respectively. The total active power and reactive power losses of the system obtained by applying Newton Raphson are 133.694 MW and 796.372Mvars. The system has 77dimensions/variables namely voltage generator buses, transformer tap position and shunt capacitors/reactors. The step size for transformer tap position is 0.01p.u whereas the step size for switchable shunt capacitor is 5Mvar.

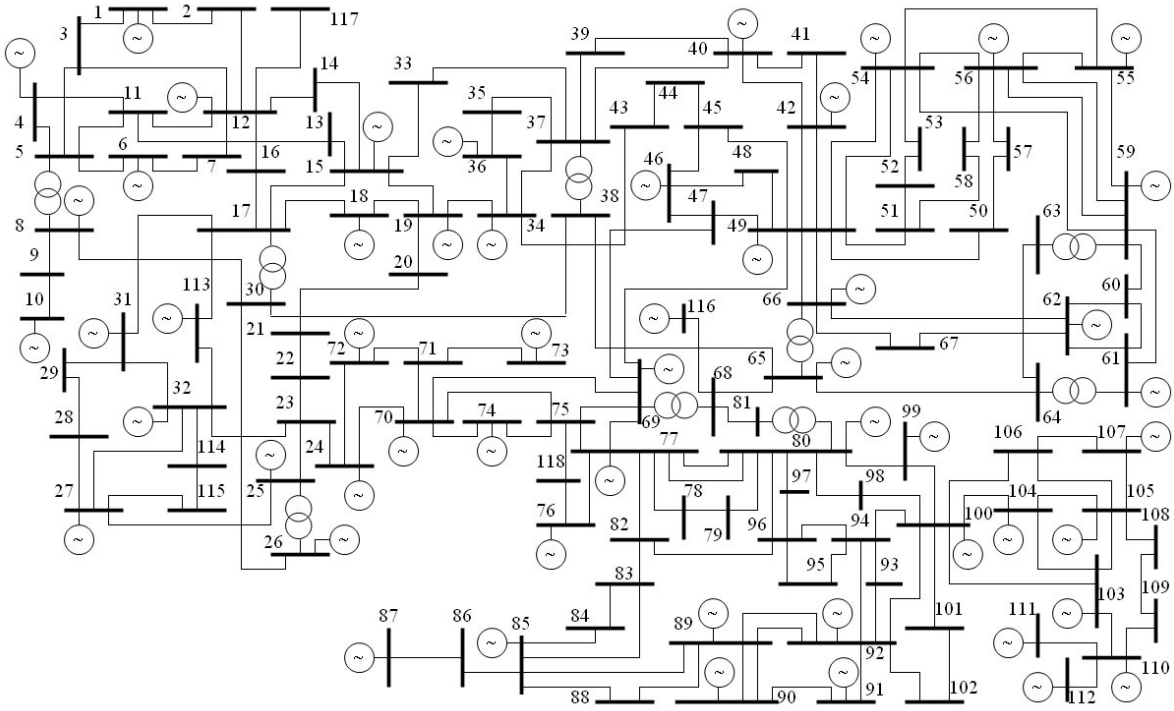


Figure 4. 4 One-line diagram for IEEE 118-bus system

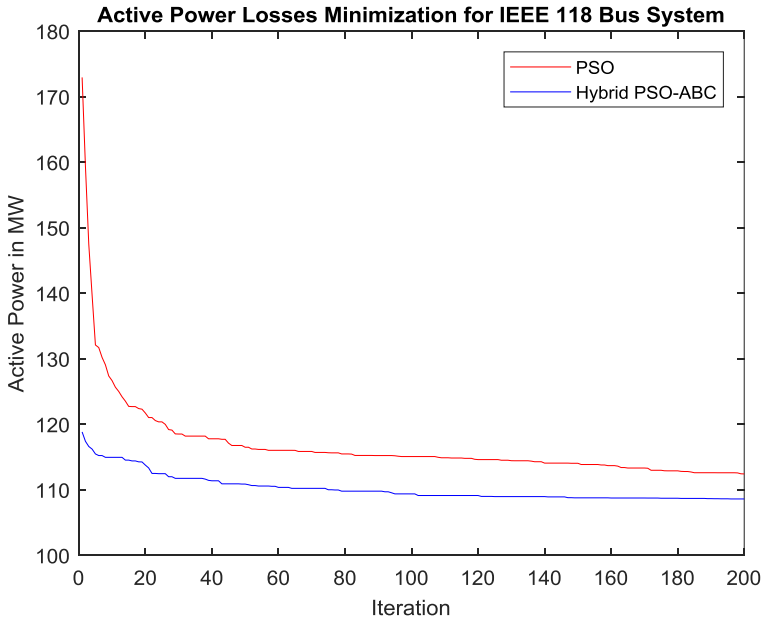


Figure 4. 5 Active power loss minimization in IEEE 118-bus system

The algorithm base PSO has been test on IEEE 118-bus test system for reactive power optimization and it reduced the active power losses from 133.694MW up to 112.4116MW while by applying the Hybrid PSO-ABC the active power losses reduced up to 108.5942MW. The Fig4.5 show the loss minimization by application of

two algorithms and the Table 4.5 shows the summary of results comparison of both algorithms. After conducting 20 runs the maximum loss or worst loss, minimum loss or best loss, mean loss, standard deviation and loss reduction have been obtained and presented.

Table 4. 4: Comparative results of PSO and Hybrid PSO-ABC algorithm for IEEE 118-bus system

Algorithm	**Active Power Loss (Mw)**			**Loss Reduction (%)**
	Worst Loss	**Best Loss**	**Mean Loss**	
Base Case (Nr)	-	-	133.694	-
Pso	179.7413	*112.4116*	113.0280	15.457
Hybrid Pso-Abc	120.4932	*108.5942*	108.7185	18.681

The Table 4.5 presents the value of control variables before and after optimization using both PSO and Hybrid PSO-ABC algorithms. The control variables obtained by Hybrid PSO-ABC was found to be more accurate comparing with those obtained by PSO. Due to the diversity of particles in chaotic initialization, the particles were spread out within the search space so that they facilitated in searching process for the global solution. Not only the resurgence of idle particles by scout phase also contributed in getting good optimal solution and reduction of active power loss of the system.

Table 4. 5 Control variables and their respective maximum and minimum values before and after optimization case of IEEE 118-bus system

S/N	**Control Variables**	**Voltage in p.u before optimization**	**Voltage in p.u after optimization by PSO**	**Voltage in p.u after optimization by Hybrid PSO-ABC**	**Minimum**	**Maximum**
1	V_{g1}	0.9550	1.0405	1.0593	0.95 p.u	1.1 p.u
2	V_{g4}	0.9980	1.0645	1.0847	0.95 p.u	1.1 p.u
3	V_{g6}	0.9900	1.0497	1.0788	0.95 p.u	1.1 p.u
4	V_{g8}	1.0150	1.0604	1.0599	0.95 p.u	1.1 p.u
5	V_{g10}	1.0500	1.1000	1.0697	0.95 p.u	1.1 p.u
6	V_{g12}	0.9900	1.0480	1.0723	0.95 p.u	1.1 p.u
7	V_{g15}	0.9700	1.0450	1.0678	0.95 p.u	1.1 p.u
8	V_{g18}	0.9730	1.0369	1.0719	0.95 p.u	1.1 p.u
9	V_{g19}	0.9630	1.0372	1.0675	0.95 p.u	1.1 p.u
10	V_{g24}	0.9920	1.0565	1.0714	0.95 p.u	1.1 p.u

Table 4. 5 Cont…

S/N	Control Variables	Voltage in p.u before optimization	Voltage in p.u after optimization by PSO	Voltage in p.u after optimization by Hybrid PSO-ABC	Minimum	Maximum
11	V_{g25}	1.0500	1.1000	1.0975	0.95 p.u	1.1 p.u
12	V_{g26}	1.0150	1.1000	1.1000	0.95 p.u	1.1 p.u
13	V_{g27}	0.9680	1.0588	1.0629	0.95 p.u	1.1 p.u
14	V_{g31}	0.9670	1.0488	1.0579	0.95 p.u	1.1 p.u
15	V_{g32}	0.9640	1.0479	1.0600	0.95 p.u	1.1 p.u
16	V_{g34}	0.9860	1.0574	1.0784	0.95 p.u	1.1 p.u
17	V_{g36}	0.9800	1.0585	1.0742	0.95 p.u	1.1 p.u
18	V_{g40}	0.9700	1.0339	1.0493	0.95 p.u	1.1 p.u
19	V_{g42}	0.9850	1.0355	1.0555	0.95 p.u	1.1 p.u
20	V_{g46}	1.0050	1.0663	1.0735	0.95 p.u	1.1 p.u
21	V_{g49}	1.0250	1.0699	1.0901	0.95 p.u	1.1 p.u
22	V_{g54}	0.9550	1.0346	1.0658	0.95 p.u	1.1 p.u
23	V_{g55}	0.9520	1.0326	1.0634	0.95 p.u	1.1 p.u
24	V_{g56}	0.9540	1.0349	1.0649	0.95 p.u	1.1 p.u
25	V_{g59}	0.9850	1.0499	1.0857	0.95 p.u	1.1 p.u
26	V_{g61}	0.9950	1.0598	1.0882	0.95 p.u	1.1 p.u
27	V_{g62}	0.9980	1.0457	1.0827	0.95 p.u	1.1 p.u
28	V_{g65}	1.0050	1.0743	1.0777	0.95 p.u	1.1 p.u
29	V_{g66}	1.0500	1.0700	1.1000	0.95 p.u	1.1 p.u
30	V_{g69}	1.0350	1.0874	1.1000	0.95 p.u	1.1 p.u
31	V_{g70}	0.9840	1.0480	1.0696	0.95 p.u	1.1 p.u
32	V_{g72}	0.9800	1.0334	1.0587	0.95 p.u	1.1 p.u
33	V_{g73}	0.9910	1.0382	1.0629	0.95 p.u	1.1 p.u
34	V_{g74}	0.9580	1.0268	1.0563	0.95 p.u	1.1 p.u
35	V_{g76}	0.9430	1.0405	1.0640	0.95 p.u	1.1 p.u
36	V_{g77}	1.0060	1.0739	1.0820	0.95 p.u	1.1 p.u
37	V_{g80}	1.0400	1.0861	1.0945	0.95 p.u	1.1 p.u
38	V_{g85}	0.9850	1.0977	1.0990	0.95 p.u	1.1 p.u

Table 4. 5 Cont…

S/N	Control Variables	Voltage in p.u before optimization	Voltage in p.u after optimization by PSO	Voltage in p.u after optimization by Hybrid PSO-ABC	Minimum	Maximum
39	V_{g87}	1.0150	1.0266	1.0755	0.95 p.u	1.1 p.u
40	V_{g89}	1.0050	1.1000	1.0997	0.95 p.u	1.1 p.u
41	V_{g90}	0.9850	1.0722	1.0802	0.95 p.u	1.1 p.u
42	V_{g91}	0.9800	1.0730	1.0891	0.95 p.u	1.1 p.u
43	V_{g92}	0.9930	1.0892	1.0993	0.95 p.u	1.1 p.u
44	V_{g99}	1.0100	1.0639	1.0861	0.95 p.u	1.1 p.u
45	V_{g100}	1.0170	1.0756	1.0926	0.95 p.u	1.1 p.u
46	V_{g103}	1.0010	1.0611	1.0795	0.95 p.u	1.1 p.u
47	V_{g104}	0.9710	1.0559	1.0673	0.95 p.u	1.1 p.u
48	V_{g105}	0.9650	1.0553	1.0660	0.95 p.u	1.1 p.u
49	V_{g107}	0.9520	1.0414	1.0583	0.95 p.u	1.1 p.u
50	V_{g110}	0.9730	1.0424	1.0609	0.95 p.u	1.1 p.u
51	V_{g111}	0.9800	1.0397	1.0665	0.95 p.u	1.1 p.u
52	V_{g112}	0.9750	1.0272	1.0471	0.95 p.u	1.1 p.u
53	V_{g113}	0.9930	1.0531	1.0789	0.95 p.u	1.1 p.u
54	V_{g116}	1.0050	1.0674	1.0721	0.95 p.u	1.1 p.u
55	T_{8-5}	0.9850	1.1000	0.9800	0.9	1.1
56	T_{26-25}	0.9600	1.0500	1.0700	0.9	1.1
57	T_{30-17}	0.9600	1.1000	1.0500	0.9	1.1
58	T_{38-37}	0.9350	0.9000	1.0200	0.9	1.1
59	T_{63-59}	0.9600	0.9000	1.0700	0.9	1.1
60	T_{64-61}	0.9850	1.1000	1.0600	0.9	1.1
61	T_{65-66}	0.9350	0.9000	1.0300	0.9	1.1
62	T_{68-69}	0.9350	0.9000	1.0300	0.9	1.1
63	T_{81-80}	0.9350	0.9000	1.0600	0.9	1.1
64	Q_{c5}	-40	25	25	0Mvar	25Mvars
65	Q_{c34}	14	25	25	0Mvar	25Mvars
66	Q_{c37}	-25	15	25	0Mvar	25Mvars
67	Q_{c44}	10	25	15	0Mvar	25Mvars

Table 4. 5 Cont…

S/N	Control Variables	Voltage in p.u before optimization	Voltage in p.u after optimization by PSO	Voltage in p.u after optimization by Hybrid PSO-ABC	Minimum	Maximum
68	Q_{c45}	10	0	0	0Mvar	25Mvars
69	Q_{c46}	10	0	5	0Mvar	25Mvars
70	Q_{c48}	15	25	5	0Mvar	25Mvars
71	Q_{c74}	12	25	25	0Mvar	25Mvars
72	Q_{c79}	20	0	0	0Mvar	25Mvars
73	Q_{c82}	20	10	25	0Mvar	25Mvars
74	Q_{c83}	10	20	25	0Mvar	25Mvars
75	Q_{c105}	20	10	0	0Mvar	25Mvars
76	Q_{c107}	6	0	10	0Mvar	25Mvars
77	Q_{c110}	6	25	10	0Mvar	25Mvars

The Fig 4.6 and Table 4.6 illustrate the voltage profile improvement for IEEE 118-bus system. Whereby the voltage obtained by load flow were poor. But the application of PSO and Hybrid PSO-ABC algorithms showed a great impact in minimizing the active power losses by improving the voltage levels at all buses of the system.

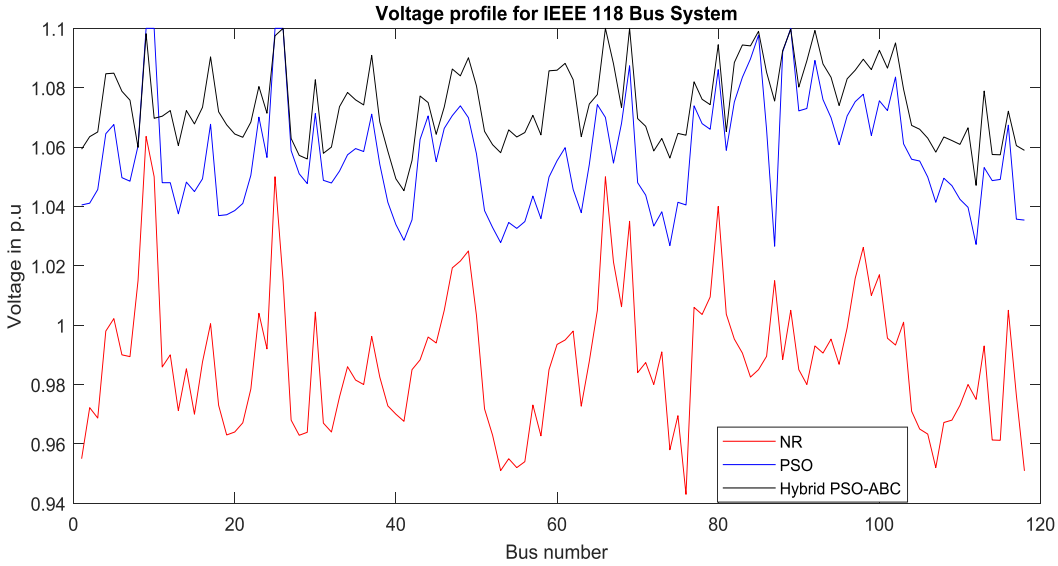


Figure 4. 6 Voltage improvement at all buses before and after optimization for IEEE 118-bus system

Table 4. 6 Voltage profile at all buses before and after optimization case of IEEE 118-bus system

Bus Number	**Voltage in p.u before optimization**	**Voltage in p.u after optimization by PSO**	**Voltage in p.u after optimization by Hybrid PSO-ABC**
1	0.9550	1.0405	1.0593
2	0.9722	1.0411	1.0635
3	0.9687	1.0457	1.0651
4	0.9980	1.0645	1.0847
5	1.0022	1.0676	1.0849
6	0.9900	1.0497	1.0788
7	0.9894	1.0485	1.0758
8	1.0150	1.0604	1.0599
9	1.0636	1.1000	1.0982
10	1.0500	1.1000	1.0697
11	0.9859	1.0480	1.0704
12	0.9900	1.0480	1.0723
13	0.9712	1.0375	1.0605
14	0.9853	1.0482	1.0723
15	0.9700	1.0450	1.0678
16	0.9876	1.0493	1.0734
17	1.0005	1.0677	1.0904
18	0.9730	1.0369	1.0719
19	0.9630	1.0372	1.0675
20	0.9640	1.0386	1.0644
21	0.9671	1.0410	1.0633
22	0.9785	1.0505	1.0684
23	1.0040	1.0701	1.0804
24	0.9920	1.0565	1.0714
25	1.0500	1.1000	1.0975
26	1.0150	1.1000	1.1000
27	0.9680	1.0588	1.0629
28	0.9629	1.0510	1.0572
29	0.9639	1.0477	1.0560
30	1.0044	1.0713	1.0827

Table 4. 6 Cont…

Bus Number	Voltage in p.u before optimization	Voltage in p.u after optimization by PSO	Voltage in p.u after optimization by Hybrid PSO-ABC
31	0.9670	1.0488	1.0579
32	0.9640	1.0479	1.0600
33	0.9758	1.0519	1.0736
34	0.9860	1.0574	1.0784
35	0.9815	1.0595	1.0759
36	0.9800	1.0585	1.0742
37	0.9962	1.0711	1.0909
38	0.9825	1.0542	1.0685
39	0.9728	1.0414	1.0586
40	0.9700	1.0339	1.0493
41	0.9676	1.0286	1.0453
42	0.9850	1.0355	1.0555
43	0.9883	1.0626	1.0772
44	0.9960	1.0705	1.0750
45	0.9940	1.0551	1.0643
46	1.0050	1.0663	1.0735
47	1.0193	1.0706	1.0863
48	1.0216	1.0739	1.0840
49	1.0250	1.0699	1.0901
50	1.0032	1.0577	1.0806
51	0.9718	1.0386	1.0653
52	0.9629	1.0329	1.0608
53	0.9510	1.0278	1.0581
54	0.9550	1.0346	1.0658
55	0.9520	1.0326	1.0634
56	0.9540	1.0349	1.0649
57	0.9731	1.0435	1.0707
58	0.9627	1.0359	1.0641
59	0.9850	1.0499	1.0857
60	0.9935	1.0554	1.0859

Table 4. 6 Cont…

Bus Number	Voltage in p.u before optimization	Voltage in p.u after optimization by PSO	Voltage in p.u after optimization by Hybrid PSO-ABC
61	0.9950	1.0598	1.0882
62	0.9980	1.0457	1.0827
63	0.9727	1.0379	1.0635
64	0.9878	1.0541	1.0745
65	1.0050	1.0743	1.0777
66	1.0500	1.0700	1.1000
67	1.0213	1.0546	1.0881
68	1.0062	1.0680	1.0733
69	1.0350	1.0874	1.1000
70	0.9840	1.0480	1.0696
71	0.9874	1.0438	1.0670
72	0.9800	1.0334	1.0587
73	0.9910	1.0382	1.0629
74	0.9580	1.0268	1.0563
75	0.9695	1.0414	1.0646
76	0.9430	1.0405	1.0640
77	1.0060	1.0739	1.0820
78	1.0036	1.0679	1.0761
79	1.0095	1.0660	1.0743
80	1.0400	1.0861	1.0945
81	1.0037	1.0589	1.0652
82	0.9953	1.0752	1.0885
83	0.9906	1.0835	1.0944
84	0.9825	1.0897	1.0941
85	0.9850	1.0977	1.0990
86	0.9895	1.0659	1.0852
87	1.0150	1.0266	1.0755
88	0.9884	1.0921	1.0925
89	1.0050	1.1000	1.0997
90	0.9850	1.0722	1.0802

Table 4. 6 Cont…

Bus Number	Voltage in p.u before optimization	Voltage in p.u after optimization by PSO	Voltage in p.u after optimization by Hybrid PSO-ABC
91	0.9800	1.0730	1.0891
92	0.9930	1.0892	1.0993
93	0.9906	1.0760	1.0880
94	0.9953	1.0699	1.0835
95	0.9868	1.0608	1.0740
96	0.9991	1.0705	1.0830
97	1.0158	1.0753	1.0859
98	1.0262	1.0778	1.0896
99	1.0100	1.0639	1.0861
100	1.0170	1.0756	1.0926
101	0.9956	1.0723	1.0866
102	0.9933	1.0836	1.0951
103	1.0010	1.0611	1.0795
104	0.9710	1.0559	1.0673
105	0.9650	1.0553	1.0660
106	0.9633	1.0499	1.0630
107	0.9520	1.0414	1.0583
108	0.9672	1.0495	1.0634
109	0.9680	1.0470	1.0622
110	0.9730	1.0424	1.0609
111	0.9800	1.0397	1.0665
112	0.9750	1.0272	1.0471
113	0.9930	1.0531	1.0789
114	0.9613	1.0487	1.0575
115	0.9612	1.0491	1.0574
116	1.0050	1.0674	1.0721
117	0.9763	1.0357	1.0605
118	0.9509	1.0354	1.0589

4.5 **Summary**

This chapter dealt with Implementation of both PSO and Hybrid PSO-ABC for reactive power optimization in large-scale power systems where two main cases were considered namely the IEEE 14-Bus system and IEEE 118-bus system. The algorithm and simulation were developed in MATLAB 2015a and their have shown that the reactive power optimization using Hybrid PSO-ABC optimizes better than PSO does and gives accurate results comparing with the results obtained using PSO Algorithm.

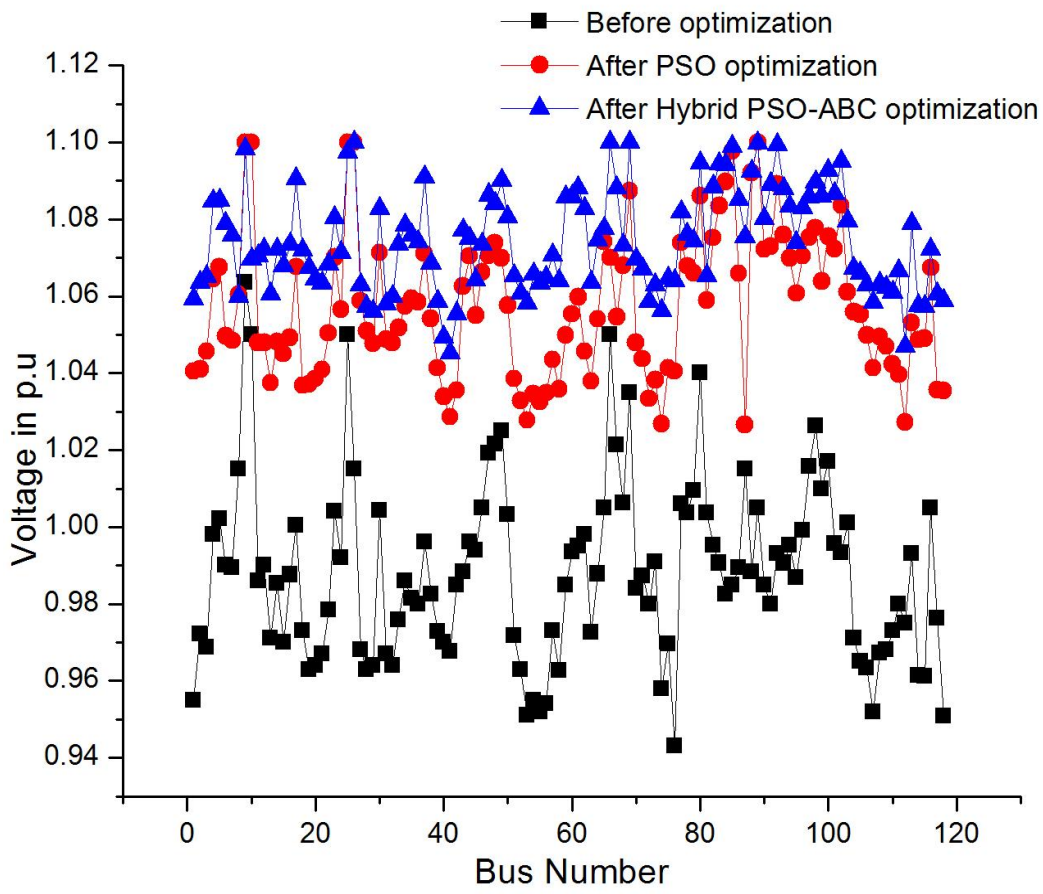


Fig. 4.7: Voltage profile at all buses before and after optimization case of IEEE 118-bus system

5 CONCLUSION AND RECOMMENDATION

5.1 Conclusion

The main objective function in this Thesis was the minimization of Active power losses within large-scales electric power systems. Various constraints were also considered, the equality constraints were satisfied by Newton Raphson load flow computation whereas inequality constraints were transformed into equality constraints and were being added to the main objective function in form of penalty function so as to penalize those violating their limits. The objectives of the research were achieved because both algorithms PSO and Hybrid PSO-ABC were successfully developed, implemented in MATLAB 2017(a) and were applied on both IEEE 14-bus system and IEEE 118-bus system to ensure the minimization of active power by not only fine-tuning the control variables (voltage at generator bus, transformer tap position and shunt capacitor), reducing the entire active power losses of the system but also improving the voltage profile of all buses. Using the PSO algorithm for the case of IEEE 14-bus system the losses have reduced from 13.3910MW to 12.4904MW corresponding to 6.72% of loss reduction whereas using Hybrid PSO-ABC the active power losses were reduced up to 12.2667MW corresponding to 8.39% of loss reduction. For the case of IEEE 118-bus system, using PSO algorithm the active power losses have reduced for 133.694MW up to 112.4116MW which mean 15.457% of loss reduction whereas using Hybrid PSO-ABC the active power losses were reduced up to 108.5942MW which means 18.681% of loss reduction. As the results have been shown the Hybrid PSO-ABC algorithm out performed well comparing to the PSO algorithm this was due to two main modifications made. Initialization of initial population by using chaotic map (logistic map and tent map) have had a great impact in letting particles being well spread all over the search space compared to uniform random distribution as to increase not only the diversity but also the chance of find good optimal solution. The insertion of scout phase has also brought big contribution because it has alleviated the handicap of PSO by sustaining the resurgence of idle particles (particles unable to improve their personal best position) and this has contributed in balancing the exploration and exploitation abilities of the algorithm.

5.2 Recommendations

Nowadays the researchers are more attracted by the advancement of evolutionary algorithms and metaheuristic algorithms in general for solving various optimization problems in our everyday life.

The PSO algorithm is one of them which has many drawbacks and some disadvantages and it needs more improvement. Associating the PSO algorithm with other natural inspired algorithm which have good exploration ability can contribute in suppressing some handicaps and drawbacks of PSO. For further researches, people should take note of this scenario so that they can develop much stronger hybrid algorithms which can easily optimize the reactive power of the electric systems with accuracy and high convergence rate. This research has only focused on fine-tuning of three control variables (voltage at generator bus, transformer tap position and shunt capacitor) for minimizing the active power losses of the system. For further research, as the electric power system is getting more complex, other advanced parameters (such optimally adding renewable energies and new reactive power compensators) should also be considered during the reactive power optimization process within large scale electric power systems. And also in this research the objective function used is single one (focusing on one parameter to be minimized). For further research, optimization by considering multi-objective functions can be done to improve the minimization of active power systems.

References

[1] H. Abaali, T. Talbi, and R. Skouri, "Comparison of Newton Raphson and Gauss Seidel methods for power flow analysis," *International Journal of Energy and Power Engineering,* vol. 12, pp. 627-633, 2018.

[2] A.-A. A. Mohamed, Y. S. Mohamed, A. A. M. El-Gaafary, and A. M. Hemeida, "Optimal power flow using moth swarm algorithm," *Electric Power Systems Research,* vol. 142, pp. 190-206, 2017/01/01/ 2017.

[3] I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Information Sciences,* vol. 237, pp. 82-117, 2013/07/10/ 2013.

[4] A. Karami, "A fuzzy anomaly detection system based on hybrid pso-kmeans algorithm in content-centric networks," pp. 1253–1269, 2015.

[5] L. d. S. Coelho, V. C. Mariani, and J. V. Leite, "Solution of Jiles–Atherton vector hysteresis parameters estimation by modified Differential Evolution approaches," *Expert Systems with Applications,* vol. 39, pp. 2021-2025, 2012/02/01/ 2012.

[6] M. R. Adaryani and A. Karami, "Artificial bee colony algorithm for solving multi-objective optimal power flow problem," *International Journal of Electrical Power & Energy Systems,* vol. 53, pp. 219-230, 2013.

[7] R. Roy and H. T. Jadhav, "Optimal power flow solution of power system incorporating stochastic wind power using Gbest guided artificial bee colony algorithm," *International Journal of Electrical Power & Energy Systems,* vol. 64, pp. 562-578, 2015/01/01/ 2015.

[8] J. Radosavljević, D. Klimenta, M. Jevtić, and N. Arsić, "Optimal Power Flow Using a Hybrid Optimization Algorithm of Particle Swarm Optimization and Gravitational Search Algorithm," *Electric Power Components and Systems,* vol. 43, pp. 1958-1970, 2015/10/21 2015.

[9] R. P. Singh, V. Mukherjee, and S. P. Ghoshal, "Particle swarm optimization with an aging leader and challengers algorithm for the solution of optimal power flow problem," *Applied Soft Computing,* vol. 40, pp. 161-177, 2016/03/01/ 2016.

[10] M. R. Narimani, R. Azizipanah-Abarghooee, B. Zoghdar-Moghadam-Shahrekohne, and K. Gholami, "A novel approach to multi-objective optimal power flow by a new hybrid optimization algorithm considering generator constraints and multi-fuel type," *Energy*, vol. 49, pp. 119-136, 2013/01/01/ 2013.

[11] K. Pandiarajan and C. K. Babulal, "Fuzzy harmony search algorithm based optimal power flow for power system security enhancement," *International Journal of Electrical Power & Energy Systems*, vol. 78, pp. 72-79, 2016/06/01/ 2016.

[12] A. A. El-Fergany and H. M. Hasanien, "Single and Multi-objective Optimal Power Flow Using Grey Wolf Optimizer and Differential Evolution Algorithms," *Electric Power Components and Systems*, vol. 43, pp. 1548-1559, 2015/08/09 2015.

[13] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Applied Soft Computing*, vol. 8, pp. 687-697, 2008/01/01/ 2008.

[14] J. Nanda, D. P. Kothari, and S. C. Srivastava. (1989, New optimal power-dispatch algorithm using Fletcher's quadratic programming method. *IEE Proceedings C (Generation, Transmission and Distribution) 136(3)*, 153-161. Available: https://digital-library.theiet.org/content/journals/10.1049/ip-c.1989.0022

[15] H. R. El-Hana Bouchekara, M. A. Abido, and A. E. Chaib, "Optimal Power Flow Using an Improved Electromagnetism-like Mechanism Method," *Electric Power Components and Systems*, vol. 44, pp. 434-449, 2016/02/25 2016.

[16] A. Ramesh Kumar and L. Premalatha, "Optimal power flow for a deregulated power system using adaptive real coded biogeography-based optimization," *International Journal of Electrical Power & Energy Systems*, vol. 73, pp. 393-399, 2015/12/01/ 2015.

[17] P. K. Roy and C. Paul, "Optimal power flow using krill herd algorithm," *International Transactions on Electrical Energy Systems*, vol. 25, pp. 1397-1419, 2015/08/01 2015.

[18] K. Abaci and V. Yamacli, "Differential search algorithm for solving multi-objective optimal power flow problem," *International Journal of Electrical Power & Energy Systems*, vol. 79, pp. 1-10, 2016/07/01/ 2016.

[19] M. Ghasemi, S. Ghavidel, M. M. Ghanbarian, and M. Gitizadeh, "Multi-objective optimal electric power planning in the power system using Gaussian bare-bones imperialist competitive algorithm," *Information Sciences,* vol. 294, pp. 286-304, 2015/02/10/ 2015.

[20] M. Ghasemi, S. Ghavidel, M. Gitizadeh, and E. Akbari, "An improved teaching–learning-based optimization algorithm using Lévy mutation strategy for non-smooth optimal power flow," *International Journal of Electrical Power & Energy Systems,* vol. 65, pp. 375-384, 2015/02/01/ 2015.

[21] H. R. E. H. Bouchekara, M. A. Abido, and M. Boucherma, "Optimal power flow using Teaching-Learning-Based Optimization technique," *Electric Power Systems Research,* vol. 114, pp. 49-59, 2014/09/01/ 2014.

[22] A. M. Shaheen, R. A. El-Sehiemy, and S. M. Farrag, "Solving multi-objective optimal power flow problem via forced initialised differential evolution algorithm," *IET Generation, Transmission & Distribution,* vol. 10, pp. 1634-1647, 2016/05/01 2016.

[23] H. R. E. H. Bouchekara, A. E. Chaib, M. A. Abido, and R. A. El-Sehiemy, "Optimal power flow using an Improved Colliding Bodies Optimization algorithm," *Applied Soft Computing,* vol. 42, pp. 119-131, 2016/05/01/ 2016.

[24] S. Duman, U. Güvenç, Y. Sönmez, and N. Yörükeren, "Optimal power flow using gravitational search algorithm," *Energy Conversion and Management,* vol. 59, pp. 86-95, 2012/07/01/ 2012.

[25] M. Ghasemi, S. Ghavidel, S. Rahmani, A. Roosta, and H. Falah, "A novel hybrid algorithm of imperialist competitive algorithm and teaching learning algorithm for optimal power flow problem with non-smooth cost functions," *Engineering Applications of Artificial Intelligence,* vol. 29, pp. 54-69, 2014/03/01/ 2014.

[26] J. A. Momoh, R. J. Koessler, M. S. Bond, B. Stott, S. Di, A. Papalexopoulos, *et al.*, "Challenges to optimal power flow," *Power Systems, IEEE Transactions on,* vol. 12, pp. 444-455, 03/01 1997.

[27] M. Vishnu and S. K. TK, "An improved solution for reactive power dispatch problem using diversity-enhanced particle swarm optimization," *Energies,* vol. 13, p. 2862, 2020.

[28] S. Pandya and R. Roy, "Particle swarm optimization based optimal reactive power dispatch," in *2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, 2015, pp. 1-5.

[29] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, pp. 459-471, 2007/11/01 2007.

[30] T. Niknam, M. R. Narimani, and R. Azizipanah-Abarghooee, "A new hybrid algorithm for optimal power flow considering prohibited zones and valve point effect," *Energy Conversion and Management*, vol. 58, pp. 197-206, 2012/06/01/ 2012.

[31] A. Ahmadi, M. C. Smith, E. R. Collins, V. Dargahi, and S. Jin, "Fast Newton-Raphson Power Flow Analysis Based on Sparse Techniques and Parallel Processing," *IEEE Transactions on Power Systems*, vol. 37, pp. 1695-1705, 2021.

[32] M. Shakil, Z. Rashid, G. Hussain, and F. Umer, "Power flow analysis and optimization in ring distribution network of Bahawalpur using Newton Raphson method," *Indian J Sci Technol*, vol. 13, pp. 2720-2732, 2020.

[33] A. Ramamoorthy and R. Ramachandran, "Reactive power optimization using GSA," in *2014 6th IEEE Power India International Conference (PIICON)*, 2014, pp. 1-4.

[34] H. Wang, H. Jiang, K. Xu, and G. Li, "Reactive power optimization of power system based on improved particle swarm optimization," in *2011 4th International Conference on Electric Utility Deregulation and Restructuring and Power Technologies (DRPT)*, 2011, pp. 606-609.

[35] S. A. Salgar and C. Mallareddy, "Load flow analysis for a 220 kV line-case study," *Int. J. Innov. Eng. Res. Technol*, vol. 2, pp. 1-12, 2015.

[36] V. Suresh, "Comparison of solvers performance for load flow analysis," *Transactions on Environment and Electrical Engineering*, vol. 3, 2019.

[37] K. Mamandur and R. Chenoweth, "Optimal control of reactive power flow for improvements in voltage profiles and for real power loss minimization," *IEEE transactions on power apparatus and systems*, pp. 3185-3194, 1981.

[38] R. R. Shoults and D. T. Sun, "Optimal Power Flow Based Upon P-Q Decomposition," *IEEE Transactions on Power Apparatus and Systems,* vol. PAS-101, pp. 397-405, 1982.

[39] M. Olofsson, G. Andersson, and L. Soder, "Linear programming based optimal power flow using second order sensitivities," *IEEE Transactions on Power Systems,* vol. 10, pp. 1691-1697, 1995.

[40] Y. Amrane and M. Boudour, "Optimal reactive power dispatch based on particle swarm optimization approach applied to the Algerian electric power system," in *2014 IEEE 11th International Multi-Conference on Systems, Signals & Devices (SSD14)*, 2014, pp. 1-6.

[41] P. L. Reddy, S. S. Prashant, and G. Yesuratnam, "Reactive Power Optimization of Power System based on Particle Swarm Optimization and Non Linear Programming," 2014.

[42] S. Khunkitti, A. Siritaratiwat, S. Premrudeepreechacharn, R. Chatthaworn, and N. R. Watson, "A Hybrid DA-PSO Optimization Algorithm for Multiobjective Optimal Power Flow Problems," *Energies,* vol. 11, 2018.

[43] J. C. Bansal, S. S. Jadon, R. Tiwari, D. Kiran, and B. K. Panigrahi, "Optimal power flow using artificial bee colony algorithm with global and local neighborhoods," *International Journal of System Assurance Engineering and Management,* vol. 8, pp. 2158-2169, 2017/12/01 2017.

[44] X. Zhou, A. Su, A. Liu, W. Cui, and W. Liu, "Cooperative approach to artificial bee colony algorithm for optimal power flow," *Cluster Computing,* vol. 22, pp. 8059-8067, 2019/07/01 2019.

[45] S. G. M. Ghasemi, M.M. Ghanbarian, M. Gitizadeh, "Multi-objective optimal electric power planning in the power system using Gaussian bare-bones imperialist competitive algorithm," pp. 286–304, 2015.

[46] F. R. C. Soldevilla and F. A. C. Huerta, "Minimization of losses in power systems by reactive power dispatch using particle swarm optimization," in *2019 54th International Universities Power Engineering Conference (UPEC)*, 2019, pp. 1-5.

[47] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, 1995, pp. 1942-1948 vol.4.

[48] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 1995, pp. 39-43.

[49] M. Imran, R. Hashim, and N. E. A. Khalid, "An Overview of Particle Swarm Optimization Variants," *Procedia Engineering*, vol. 53, pp. 491-496, 2013/01/01/ 2013.

[50] M. Al-Kaabi and L. Al-Bahrani, "Modified artificial bee colony optimization technique with different objective function of constraints optimal power flow," *International Journal of Intelligent Engineering and Systems*, vol. 13, pp. 378-388, 2020.

[51] W.-f. Gao and S.-y. Liu, "A modified artificial bee colony algorithm," *Computers & Operations Research*, vol. 39, pp. 687-697, 2012/03/01/ 2012.

[52] L. Le Dinh, D. Vo Ngoc, and P. Vasant, "Artificial Bee Colony Algorithm for Solving Optimal Power Flow Problem," *The Scientific World Journal*, vol. 2013, p. 159040, 2013/12/29 2013.

[53] S. Mhanna and P. Mancarella, "An exact sequential linear programming algorithm for the optimal power flow problem," *IEEE Transactions on Power Systems*, vol. 37, pp. 666-679, 2021.

[54] W. Rohouma, R. S. Balog, A. A. Peerzada, and M. M. Begovic, "Reactive power compensation of time-varying load using capacitor-less D-STATCOM," in *2019 10th International Conference on Power Electronics and ECCE Asia (ICPE 2019-ECCE Asia)*, 2019, pp. 2296-2301.

[55] K. S. Sambaiah and T. Jayabarathi, "Loss minimization techniques for optimal operation and planning of distribution systems: A review of different methodologies," *International Transactions on Electrical Energy Systems*, vol. 30, p. e12230, 2020.

[56] N. Majumdar, M. Sarstedt, L. Kluß, and L. Hofmann, "Linear Optimization Based Distribution Grid Flexibility Aggregation Augmented With OLTC Operational Flexibilities," *IEEE Access*, vol. 10, pp. 77510-77521, 2022.

[57] L. Ogunwolu, O. Ero, and O. Ibidapo-Obe, "Modeling and optimization of an electric power distribution network planning system using mixed binary integer programming," *Nigerian Journal of Technology*, vol. 36, pp. 552-562, 2017.

APPENDICES

Appendix 1: Newton Raphson codes for IEEE 14 Bus System

```
%=========================================================================%
%                  Computation of the load flow using the Newton-Raphson algorithm                  %
%                                          IEEE 14 Bus System                                       %
%=========================================================================%

clc;
clear;
close all;
Sbase=100;              % The base of the system

load line_dat.txt;      % Loading the line data
frombus=line_dat(:,1);  % From the bus number
tobus=line_dat(:,2);    % To the bus number
r=line_dat(:,3);        % The line data of resistance
x=line_dat(:,4);        % The line data of reactance
b=line_dat(:,5);        % The line da of the line charging admittance
a=line_dat(:,9);        % Transformer tap ratio
z=r+1i*x;               % Impedanxe calculation
y=1./z;                 % Admittance calculation
b=1i*b;                 % The charging admittance in compex notation

nbranch=length(frombus); % Number of branches in the Network

load bus_dat.txt;       % loading the bus data
bus_num=bus_dat(:,1);   % Bus number
type=bus_dat(:,2);      % The type of the bus
Pd=bus_dat(:,3);        % The real power injected into the load
Qd=bus_dat(:,4);        % The reactive power injected into the load
Gs=bus_dat(:,5);        % The conductance
Q_sh=bus_dat(:,6);      % Shunt reactance (Bs)
V=bus_dat(:,8);         % The voltage magnitude at the bus(Va)
Delta=bus_dat(:,9);     % The bus voltage angle
```

```
Vmax=bus_dat(:,12);    % Voltage upper limit
Vmin=bus_dat(:,13);    % Voltage lower limit

load gen_dat.txt;      % Loading generator data
bus_number=gen_dat(:,1);  % Bus number
Pg=gen_dat(:,2);       % The real power injected into the bus
Qg=gen_dat(:,3);       % The reactive power injected into the bus
Q_max=gen_dat(:,4);    % The maximum reactive power at generator bus
Q_min=gen_dat(:,5);    % The minimum reactive power at generator bus
Pg_max=gen_dat(:,9);   % The maximum Active power at generator bus
Pg_min=gen_dat(:,10); % The minimum Active power at generator bus
V_set=gen_dat(:,6);    % Voltage set at generator bus


% Specifying the type and number of buses
Slack=length(find(type==3));   % The number of slack bus
PV_bus=length(find(type==2));  % The number of PV_bus
PQ_bus=length(find(type==1));  % The number of PQ_bus
nbus=Slack+PV_bus+PQ_bus;      % The total number of buses in the system
ng=Slack+PV_bus;               % The number of generator buses

% Defining the Admitance matrix,Impedance matrix and their components
Y=Ymatrix(line_dat);       % Calling the function of admittance matrix
G=real(Y);             % Real part of admittance matrix
g=real(y);             % Real part of the branch admittance
B=imag(Y);             % imaginary part of admittance matrix
Theta=angle(Y);        % The angle
Z=inv(Y);              % The impedance matrix
R=real(Z);             % The real part of elements of Z

%---Calling the function which computes some quantities of generator bus--%
[Pg_all,Qg_all,V_set_all,Pg_max_all,Pg_min_all,Q_max_all,Q_min_all]=...
   gen_quantities(Pg,Qg,V_set,Pg_max,Pg_min,Q_max,Q_min,type,nbus);

%------------Changing Real and Reactive power in per unit system----------%
[Pg_all_pu,Pd_pu,Qg_all_pu,Qd_pu,Q_sh_pu]=Power_per_unit(nbus,Pg_all,Qg_all,Pd,Qd,Q_sh,Sbase);
```

```
%----Computing the Real power and Reactive power injections at buses------%
[P_inj,Q_inj]=power_inject(nbus,Pg_all_pu,Pd_pu,Qg_all_pu,Qd_pu,Q_sh_pu);


%=========================================================================%
%         Starting the Newton Raphson iterations                          %
%=========================================================================%

iter=1;
precision=1;
epsilone=10^-4; % Smallest value to check the stopping criterion
maxiter=100;

while precision>epsilone && iter<=maxiter

%------------To calculate Active and Reactive power at all buses----------%
[P_calc,Q_calc]=Active_Reactive_power(nbus,Delta,V,G,B);

P_calc_tot=P_calc+Pd_pu; % Sum of P_calculated and P_injected to the load
Q_calc_tot=Q_calc+Qd_pu; % Sum of Q_calculated and Q_injeced to the load


%---------Computation of active and reactive power Mismatches-------------%

Active_mis=(P_inj)-(P_calc);    % Real power mismatches including the slack bus
Reactive_mis=(Q_inj)-(Q_calc);

% Reactive_mis=(Q_inj)-(Q_calc);  % Reactive power mismatches including the slack bus and PV bus

P_mis=Active_mis(type~=3);      % Real power mismatches excluding the slack bus
Q_mis=Reactive_mis(type==1);    % Reactive power mismatches excluding the slack bus and PV bus


delPQ=vertcat(P_mis,Q_mis);
precision=max(abs(delPQ)); % Checking the iteration
```

```
%-----------------------Computation of Jacobian Matrix--------------------%

Jacob=Jacobian_new(nbus,type,V,Delta,G,B);

%------------Determining Corrections (Delta and Voltage)------------------%

Correct=(Jacob)\(delPQ);
Correct_Delta=Correct(1:nbus-1,1);
Correct_Volt=Correct(nbus:end,1);

%-------------------------Updating the Corrections-------------------------%
% Updating the voltage angle (Delta)
k=1;
for x=1:nbus
   if type(x)~=3
      Delta(x)=Delta(x)+Correct_Delta(k);
      k=k+1;
   end
end

% Updating the voltage (V)
d=1;
for n=1:nbus
   if type(n)==1
      V(n)=V(n)+Correct_Volt(d);
      d=d+1;
   end
end

iter=iter+1; % Iteration counter
end
V;

Delta_deg=rad2deg(Delta);              % Converting radians to degrees

V_comp=zeros(nbus,1);
for bs=1:nbus
   V_comp(bs)=abs(V(bs))*exp(1i*Delta(bs)); % Rectangular form of Voltages
```

```
end



%-----------------To compute the line losses and total losses-------------%

[Current_line,P_loss_line,Q_loss_line,P_loss_total,Q_loss_total]=losses(nbranch,frombus,tobus,V_comp,Sbase,y,z,b); % calling function losses

%------------To compute the Real and Reactive power at the Slack bus------%

[S,P_slack,Q_slack,num_from,y1,b1]=Slack_bus_power(V_comp,frombus,tobus,y,b,Sbase); % Calling the function Slack_power

% Compute the voltage stability Index
V_stab_index=zeros(PQ_bus,1);
r=1;
for d=1:nbus
%     if type(d)==1
      V_stab_index(d)=abs(conj(P_calc(d)+1i*(Q_calc(d)))/(Y(d,d)*V(d)^2));
%       r=r+1;
%     end
end

%--------------To display the results obtained after computation----------%

figure;
plot(bus_num,V,'--*r');
title('Voltage at buses for IEEE 14 Bus System before OPF');
xlabel('Bus number');
ylabel('Voltage in p.u')
grid on

figure;

Q_calc_tot_act=Q_calc_tot*Sbase;
plot(bus_num(type~=1),Q_calc_tot_act(type~=1),'--*b');
title('Reactive power(MVar) at generator bus before OPF');
```

```
xlabel('Bus number of generator bus');
ylabel('Reactive power(MVar) at generator bus')
grid on



fprintf('                                                \n');
disp('              FINAL RESULTS FOR IEEE 14 BUS SYSTEM          ');
fprintf('============================================================\n');
fprintf('               iter                              \n');
fprintf('============================================================\n');
fprintf(' %18.0f                                    \n',(iter'));
fprintf('                                                \n');
fprintf('============================================================\n');
fprintf('               Voltage magnitude and angle at bus       \n');
fprintf('      bus_num       Angle(degree)   V(pu_volt)         \n');
fprintf('============================================================\n');
fprintf('%13.0f  %13.5f  %13.5f              \n',[bus_num,Delta_deg,V]');
fprintf('                                                \n');
fprintf('                                                \n');
fprintf('============================================================\n');
fprintf('               Current flow in branches                \n');
fprintf('  frombus   tobus     Current_line(pu)               \n');
fprintf('============================================================\n');
fprintf('%7.0f  %7.0f   %14.3f      \n',[frombus,tobus,Current_line]');
fprintf('                                                \n');
fprintf('                                                \n');
fprintf('============================================================\n');
fprintf('                   Line flows                     \n');

fprintf('  frombus   tobus   P_loss_line(MW)   Q_loss_line(Mvar)\n');
fprintf('============================================================\n');
fprintf('%7.0f  %7.0f   %14.3f  %14.3f  \n',[frombus,tobus,P_loss_line,Q_loss_line]');
fprintf('                                                \n');
fprintf('                                                \n');

fprintf('============================================================\n');
fprintf('                   Total losses                   \n');
```

```
fprintf('       P_loss_total(MW)    Q_loss_total(Mvar)          \n');
fprintf('=============================================================\n');
fprintf('%14.3f         %14.3f       \n',[P_loss_total,Q_loss_total]');
fprintf('                                          \n');
fprintf('                                          \n');

fprintf('=============================================================\n');
fprintf('          Real and Reactive power at slack bus         \n');

fprintf('       P_slack(MW)         Q_slack(Mvar)              \n');
fprintf('=============================================================\n');
fprintf('%14.3f         %14.3f               \n',[P_slack,Q_slack]');
```

Appendix 2: Newton Raphson codes for IEEE 118 Bus System

```
%=========================================================================%
%                    Computation of the load flow using the Newton-Raphson algorithm                    %
%                                              IEEE 118 Bus System                                        %
%=========================================================================%

clc;
clear;
close all;
Sbase=100;              % The base of the system

load line_dat_118.txt;      % Loading the line data
frombus=line_dat_118(:,1);   % From the bus number
tobus=line_dat_118(:,2);     % To the bus number
r=line_dat_118(:,3);        % The line data of resistance
x=line_dat_118(:,4);        % The line data of reactance
b=line_dat_118(:,5);        % The line da of the line charging admittance
a=line_dat_118(:,9);        % Transformer tap ratio
z=r+1i*x;                  % Impedanxe calculation
y=1./z;                   % Admittance calculation
b=1i*b;                    % The charging admittance in compex notation
```

```
nbranch=length(frombus); % Number of branches in the Network

load bus_dat_118.txt;      % loading the bus data
bus_num=bus_dat_118(:,1);  % Bus number
type=bus_dat_118(:,2);     % The type of the bus
Pd=bus_dat_118(:,3);       % The real power injected into the load
Qd=bus_dat_118(:,4);       % The reactive power injected into the load
Gs=bus_dat_118(:,5);       % The conductance
Q_sh=bus_dat_118(:,6);     % Shunt reactance (Bs)
V=bus_dat_118(:,8);        % The voltage magnitude at the bus(Va)
Delta=bus_dat_118(:,9);    % The bus voltage angle
Vmax=bus_dat_118(:,12);    % Voltage upper limit
Vmin=bus_dat_118(:,13);    % Voltage lower limit

load gen_dat_118.txt;      % Loading generator data
bus_number=gen_dat_118(:,1);  % Bus number
Pg=gen_dat_118(:,2);       % The real power injected into the bus
Qg=gen_dat_118(:,3);       % The reactive power injected into the bus
Q_max=gen_dat_118(:,4);    % The maximum reactive power at PV_bus
Q_min=gen_dat_118(:,5);    % The minimum reactive power at PV_bus
Pg_max=gen_dat_118(:,9);   % The maximum Active power at generator bus
Pg_min=gen_dat_118(:,10);  % The minimum Active power at generator bus
V_set=gen_dat_118(:,6);    % Voltage set at generator bus

% Specifying the type and number of buses
Slack=length(find(type==3));   % The number of slack bus
PV_bus=length(find(type==2));  % The number of PV_bus
PQ_bus=length(find(type==1));  % The number of PQ_bus
nbus=Slack+PV_bus+PQ_bus;      % The total number of buses in the system
ng=Slack+PV_bus;               % The number of generator buses

% Defining the Admitance matrix,Impedance matrix and their components
Y=Ymatrix(line_dat_118);       % Calling the function of admittance matrix
G=real(Y);            % Real part of admittance matrix
g=real(y);            % Real part of the branch admittance
B=imag(Y);            % imaginary part of admittance matrix
Theta=angle(Y);       % The angle
Z=inv(Y);             % The impedance matrix
```

```
R=real(Z);            % The real part of elements of Z

%---Calling the function which computes some quantities of generator bus--%
[Pg_all,Qg_all,V_set_all,Pg_max_all,Pg_min_all,Q_max_all,Q_min_all]=...
   gen_quantities(Pg,Qg,V_set,Pg_max,Pg_min,Q_max,Q_min,type,nbus);
%------------Changing Real and Reactive power in per unit system----------%
[Pg_all_pu,Pd_pu,Qg_all_pu,Qd_pu,Q_sh_pu]=...
   Power_per_unit(nbus,Pg_all,Qg_all,Pd,Qd,Q_sh,Sbase);
%----Computing the Real power and Reactive power injections at buses------%
[P_inj,Q_inj]=power_inject(nbus,Pg_all_pu,Pd_pu,Qg_all_pu,Qd_pu,Q_sh_pu);
%=========================================================================%
%          Starting the Newton Raphson iterations                         %
%=========================================================================%
iter=1;
precision=1;
epsilone=10^-4; % Smallest value to check the stopping criterion
maxiter=100;

while precision>epsilone && iter<=maxiter

%------------To calculate Active and Reactive power at all buses----------%
[P_calc,Q_calc]=Active_Reactive_power(nbus,Delta,V,G,B);

P_calc_tot=P_calc+Pd_pu; % Sum of P_calculated and P_injected to the load
Q_calc_tot=Q_calc+Qd_pu; % Sum of Q_calculated and Q_injeced to the load



% %-------To check the reactive power limit violation at generator bus----------%
%
[Q_calc_tot,Qg_all,V,type]=reactive_power_violation(Pg_all,Qg_all,Q_calc_tot,nbus,Q_min_all,Q_max_all,V_set_all,V,t
ype,Sbase);
%
% Qg_all_pu_new=Qg_all/Sbase; % Updated Q generated in per unit
% Q_inj=Qg_all_pu_new-Qd_pu;  % Updated Qg-Qd in per unit
%
% % Updating the type and number of buses
% Slack=length(find(type==3));   % The number of slack bus
% PV_bus=length(find(type==2));  % The number of PV_bus
```

```
% PQ_bus=length(find(type==1));  % The number of PQ_bus
% nbus=Slack+PV_bus+PQ_bus;      % The total number of buses in the system
% ng=Slack+PV_bus;               % The number of generator buses

%---------Computation of active and reactive power Mismatches----------%

Active_mis=(P_inj)-(P_calc);    % Real power mismatches including the slack bus
Reactive_mis=(Q_inj)-(Q_calc);

% Reactive_mis=(Q_inj)-(Q_calc);  % Reactive power mismatches including the slack bus and PV bus

P_mis=Active_mis(type~=3);      % Real power mismatches excluding the slack bus
Q_mis=Reactive_mis(type==1);    % Reactive power mismatches excluding the slack bus and PV bus


delPQ=vertcat(P_mis,Q_mis);
precision=max(abs(delPQ)); % Checking the iteration

%----------------------Computation of Jacobian Matrix--------------------%

Jacob=Jacobian_new(nbus,type,V,Delta,G,B);

%------------Determining Corrections (Delta and Voltage)------------------%

Correct=(Jacob)\(delPQ);
Correct_Delta=Correct(1:nbus-1,1);
Correct_Volt=Correct(nbus:end,1);

%-------------------------Updating the Corrections-----------------------%
% Updating the voltage angle (Delta)
k=1;
for x=1:nbus
   if type(x)~=3
      Delta(x)=Delta(x)+Correct_Delta(k);
      k=k+1;
   end
end
```

```
% Updating the voltage (V)
d=1;
for n=1:nbus
   if type(n)==1
      V(n)=V(n)+Correct_Volt(d);
      d=d+1;
   end
end

iter=iter+1; % Iteration counter

end
V;

Delta_deg=rad2deg(Delta);                % Converting radians to degrees

V_comp=zeros(nbus,1);
for bs=1:nbus
   V_comp(bs)=abs(V(bs))*exp(1i*Delta(bs));  % Rectangular form of Voltages
end



%-----------------To compute the line losses and total losses-------------%

[Current_line,P_loss_line,Q_loss_line,P_loss_total,Q_loss_total]=losses(nbranch,frombus,tobus,V_comp,Sbase,y,z,b); %
calling function losses

%------------To compute the Real and Reactive power at the Slack bus------%

[P_slack,Q_slack]=Slack_power(V_comp,Y,nbus,type,Sbase); % Calling the function Slack_power

% Compute the voltage stability Index
V_stab_index=zeros(PQ_bus,1);
r=1;
for d=1:nbus
%     if type(d)==1
      V_stab_index(d)=abs(conj(P_calc(d)+1i*(Q_calc(d)))/(Y(d,d)*V(d)^2));
```

```
%       r=r+1;
%    end
end

%--------------To display the results obtained after computation----------%

figure;
plot(bus_num,V,'--*r');
title('Voltage at buses for IEEE 118 Bus System before OPF');
xlabel('Bus number');
ylabel('Voltage in p.u')
grid on

figure;

Q_calc_tot_act=Q_calc_tot*Sbase;
plot(bus_num(type~=1),Q_calc_tot_act(type~=1),'--*b');
title('Reactive power(MVar) at generator bus before OPF');
xlabel('Bus number of generator bus');
ylabel('Reactive power(MVar) at generator bus')
grid on




fprintf('                                                \n');
disp('              FINAL RESULTS FOR IEEE 118 BUS SYSTEM          ');
fprintf('============================================================\n');
fprintf('               iter                              \n');
fprintf('============================================================\n');
fprintf(' %18.0f                                    \n',(iter'));
fprintf('                                                \n');
fprintf('============================================================\n');
fprintf('               Voltage magnitude and angle at bus       \n');
fprintf('      bus_num       Angle(degree)   V(pu_volt)          \n');
fprintf('============================================================\n');
fprintf('%13.0f  %13.5f  %13.5f               \n',[bus_num,Delta_deg,V]');
```

```
fprintf('                                                    \n');
fprintf('                                                    \n');
fprintf('============================================================\n');
fprintf('              Current flow in branches              \n');
fprintf('  frombus   tobus     Current_line(pu)               \n');
fprintf('============================================================\n');
fprintf('%7.0f  %7.0f   %14.3f      \n',[frombus,tobus,Current_line]');
fprintf('                                                    \n');
fprintf('                                                    \n');
fprintf('============================================================\n');
fprintf('                   Line flows                       \n');

fprintf('  frombus   tobus   P_loss_line(MW)   Q_loss_line(Mvar)\n');
fprintf('============================================================\n');
fprintf('%7.0f  %7.0f   %14.3f  %14.3f  \n',[frombus,tobus,P_loss_line,Q_loss_line]');
fprintf('                                                    \n');
fprintf('                                                    \n');

fprintf('============================================================\n');
fprintf('                   Total losses                     \n');

fprintf('       P_loss_total(MW)    Q_loss_total(Mvar)          \n');
fprintf('============================================================\n');
fprintf('%14.3f         %14.3f       \n',[P_loss_total,Q_loss_total]');
fprintf('                                                    \n');
fprintf('                                                    \n');

fprintf('============================================================\n');
fprintf('           Real and Reactive power at slack bus         \n');

fprintf('       P_slack(MW)         Q_slack(Mvar)               \n');
fprintf('============================================================\n');
fprintf('%14.3f         %14.3f               \n',[P_slack,Q_slack]');
```

Appendix 3: PSO codes for IEEE 14 Bus system

```
%=========================================================================%
%                  Implementation of Particle Swarm Optimization (PSO)    %
%                        Based Active Power Loss Minimization             %
%                              IEEE 14 Bus System                         %
%=========================================================================%

clc;
clear;

%%  Loading the case IEEE 14 bus system
load line_dat.txt;     % Loading the line data
load bus_dat.txt;      % loading the bus data
load gen_dat.txt;      % Loading generator data

%% Defining the problem

members=9;   % All control Variables (Vg,T,Qc)
members_size=[1 members]; % The size of vector of all control variables
member1=5;   % Control variable Vg (Voltage at generator buses)
member2=3;   % Control variable T (Tap setting of transformer)
member3=1;   % Control variable Qc (Reactive power of shunt capacitor)
size_V=[1 member1]; % The size of control variable Vg
size_T=[1 member2]; % The size of control variable T
size_Qc=[1 member3]; % The size of control variable Qc
```

```
%% Defining the limits of voltage, tap setting and shunt capasitor
Vg_min=0.95;  % Minimun voltage
Vg_max=1.10;  % Maximum voltage
T_min=0.90;   % Minimum tap position
T_max=1.10;   % Maximum tap position
Qc_min=0;     % Minimum value of capacitor connected
Qc_max=20;    % Maximum value of capacitor connected

%% Defining all PSO parameters
num_particle=100;   % Number of particles (Swarm size)
w_min=0.4;          % Minimum value of inertia weight coefficient
w_max=0.9;          % Maximum value of inertia weight coefficient
c1=2.05;            % Acceleration coefficient (Cognitive/Personal)
c2=2.05;            % Acceleration coefficient (Social/Global)
H=c1+c2;            % Coefficient facilitating the exproration

%% Initialization of the algorithm/Generating initial population

Vg_control=(Vg_max-Vg_min)*rand(num_particle,member1)+Vg_min; % For Voltage at generator buses
tap_control=(T_max-T_min)*rand(num_particle,member2)+T_min;   % For Tap position
Qc_control=(Qc_max-Qc_min)*rand(num_particle,member3)+Qc_min; % For shunt capacitor

particle_position=zeros(num_particle,members); % Initializing the particle position
particle_loss=zeros(num_particle,1); % Initializing losses for each particle
particle_velocity=zeros(num_particle,members); % Initializing the velocity
penalty_bus_violation1=zeros(num_particle,1);   % Storing the penalty of PQ-bus violating voltage limits
penalty_gen_violation1=zeros(num_particle,1);   % Storing the penalty of generator violating reactive powerlimits
Power_loss=zeros(num_particle,1); % Storing the losses

particle_best_position=zeros(num_particle,members); % Initializing the best position
particle_best_loss=zeros(num_particle,1); % Initializing the best loss

for k=1:num_particle
   % Concatenating the matrices to form particle.position
   particle_position(k,:)=[Vg_control(k,:) tap_control(k,:) Qc_control(k,:)]; % Concatenation of matrix

   step_size_tap=0.01; % Tap position step size
```

```
step_size_Qc=5;     % Shunt capacitor step size
M_tap=round((T_max-T_min)/step_size_tap); % Tap intevals
M_Qc=round((Qc_max-Qc_min)/step_size_Qc); % Qc intervals

discrete_tap=T_min:step_size_tap:T_max; % Discretizing values for tap setting
discrete_Qc=Qc_min:step_size_Qc:Qc_max; % Discretizing values for capacitor

% Finding discrete value for tap position
for d=member1+1:member1+member2
   for n=1:length(discrete_tap)-1
      if particle_position(k,d)>discrete_tap(n)&&particle_position(k,d)<discrete_tap(n+1)
         if abs(particle_position(k,d)-discrete_tap(n))<=abs(particle_position(k,d)-discrete_tap(n+1))
            particle_position(k,d)=discrete_tap(n);
         else
            particle_position(k,d)=discrete_tap(n+1);
         end
      end
   end
end

% Finding discrete value for shunt capacitor
for d=member1+member2+1:members
   for m=1:length(discrete_Qc)-1
       if particle_position(k,d)>discrete_Qc(m)&&particle_position(k,d)<discrete_Qc(m+1)
          if abs(particle_position(k,d)-discrete_Qc(m))<=abs(particle_position(k,d)-discrete_Qc(m+1))
             particle_position(k,d)=discrete_Qc(m);
          else
             particle_position(k,d)=discrete_Qc(m+1);
          end
      end
   end
end

% Defining the position of voltage at generator bus in bus_dat matrix

v1=particle_position(k,1); bus_dat(1,8)=v1;
v2=particle_position(k,2); bus_dat(2,8)=v2;
v3=particle_position(k,3); bus_dat(3,8)=v3;
```

```
v6=particle_position(k,4); bus_dat(6,8)=v6;
v8=particle_position(k,5); bus_dat(8,8)=v8;

% Defining the position of voltage set generator bus in gen_dat matrix
v1=particle_position(k,1); gen_dat(1,6)=v1;
v2=particle_position(k,2); gen_dat(2,6)=v2;
v3=particle_position(k,3); gen_dat(3,6)=v3;
v6=particle_position(k,4); gen_dat(4,6)=v6;
v8=particle_position(k,5); gen_dat(5,6)=v8;

% Defining the position of tap ratio
% tr1(4-7), tr2(4-9), tr3(5-6)

tr1=particle_position(k,6); line_dat(8,9)=tr1;
tr2=particle_position(k,7); line_dat(9,9)=tr2;
tr3=particle_position(k,8); line_dat(10,9)=tr3;

% Defining the position of the shunt capacitor
qsh9=particle_position(k,9); bus_dat(9,6)=qsh9;

% Calling the function Function which computes the power loss
[P_calc_tot,Q_calc_tot,Pg_max_all,Pg_min_all,Q_max_all,Q_min_all,PV_bus,PQ_bus,V,type,nbus,...
nbranch,Delta,g,Sbase,frombus,tobus]=...
newton_2(bus_dat,line_dat,gen_dat);

% Penalty of the bus voltage violation
penalty_bus1=zeros(PQ_bus,1);

% Checking the voltage limits violations for non-generator buses(PQ)
m=1;
for n=1:nbus
   if type(n)==1
      if V(n)>Vg_max
         penalty_bus1(m)=100000*(V(n)-Vg_max)^2;
         m=m+1;
      elseif V(n)<Vg_min
         penalty_bus1(m)=100000*(Vg_min-V(n))^2;
         m=m+1;
```

```
        else
            penalty_bus1(m)=0;
            m=m+1;
        end
    end
end
penalty_bus_violation1(k)=sum(penalty_bus1);

% Penality of the generator violating the limits
penalty_gen1=zeros(PV_bus,1);
r=1;
for d=1:nbus
    if type(d)==2
        if Q_calc_tot(d)>Q_max_all(d)/Sbase
            penalty_gen1(r)=100000*(Q_calc_tot(d)-(Q_max_all(d)/Sbase))^2;
            r=r+1;
        elseif Q_calc_tot(d)<Q_min_all(d)/Sbase
             penalty_gen1(r)=100000*((Q_min_all(d)/Sbase)-Q_calc_tot(d))^2;
             r=r+1;
        else
            penalty_gen1(r)=0;
            r=r+1;
        end
    end
end
penalty_gen_violation1(k)=sum(penalty_gen1);

% Calling newton raphson fuction after replacing parameters

[P_loss]=object_function(nbranch,Delta,V,g,Sbase,frombus,tobus);
Power_loss(k)=P_loss;

% Evaluation of the augmented function (fitness function)
particle_loss(k)=Power_loss(k)+penalty_bus_violation1(k)+penalty_gen_violation1(k);

% Initialization of the Personal best postion and loss
particle_best_position(k,:)=particle_position(k,:);
particle_best_loss(k)=particle_loss(k);
```

```
end

% Initialization of the velocity of particles
Vg_maximum=Vg_max*0.01;
T_maximum=(T_max-T_min)/M_tap;
Qc_maximum=(Qc_max-Qc_min)/M_Qc;

% Defining the matrix of maximum velocities for all particles
Velocity_max=[Vg_maximum*ones(k,member1) T_maximum*ones(k,member2) Qc_maximum*ones(k,member3)];

% Defining the matrix of maximum velocities for all particles
Velocity_min=[-Vg_maximum*ones(k,member1) -T_maximum*ones(k,member2) -Qc_maximum*ones(k,member3)];

velocity_gen=zeros(num_particle,member1);
velocity_tap=zeros(num_particle,member2);
velocity_shunt=zeros(num_particle,member3);

random_val=rand(num_particle,members);

for k=1:num_particle
   % For generator buses
   for d=1:member1
      velocity_gen(k,d)=-Vg_maximum+(random_val(k,d)*(Vg_maximum-(-Vg_maximum)));
   end
   % For tap position
   r=1;
   for d=member1+1:member1+member2
      velocity_tap(k,r)=-T_maximum+(random_val(k,d)*(T_maximum-(-T_maximum)));
      r=r+1;
   end
   % For shunt capacitor
   r=1;
   for d=member1+member2+1:members
      velocity_shunt(k,r)=-Qc_maximum+(random_val(k,d)*(Qc_maximum-(-Qc_maximum)));
      r=r+1;
   end
   % Concatenation of particle velocity
```

```
    particle_velocity(k,:)=[velocity_gen(k,:) velocity_tap(k,:) velocity_shunt(k,:)];

%     % Storing best position of each particle
%     for d=1:members
%         best_position(k,d)=particle_best_position(k,d);
%     end

end
% Computing the Global best position and loss
Globalbest_loss=min(particle_loss); % The minimum value of losses
location= find(particle_loss==min(particle_loss)); % Locate the position of the minimum loss
Globalbest_position=particle_position(location,:);

%% Implementation of the main loop of Multimean_PSO

max_iter=200;
% Setting a mechanism that will inform us the Globalbest loss at every iteration
Best_loss=zeros(max_iter,1);
it=1;

for it=1:max_iter
    w=zeros(num_particle,1); % Initialize the weight factor


    penalty_bus_violation2=zeros(num_particle,1);   % Storing the penalty of PQ-bus violating voltage limits
    penalty_gen_violation2=zeros(num_particle,1);   % Storing the penalty of generator violating reactive powerlimits
    Power_loss2=zeros(num_particle,1); % Initialize the power loss

    for k=1:num_particle
        % Calculation of weight factor for particles of the swarm
        w(k)=w_max-(((w_max-w_min)/max_iter)*it);

        % Computing the constriction factor
        Constr_fact=2/abs(2-H-sqrt(H^2-4*H));

        % Updating the velocity of the particles

        particle_velocity(k,:)=Constr_fact*((w(k)*particle_velocity(k,:)) ...
```

```
    +(c1*random_val(k,:).*(particle_best_position(k,:)-particle_position(k,:))) ...
    +(c2*random_val(k,:).*(Globalbest_position-particle_position(k,:))));

%% Controlling the velocity of voltage at generator bus
for d=1:member1
    if particle_velocity(k,d)>Vg_maximum
        particle_velocity(k,d)=Vg_maximum;
    elseif particle_velocity(k,d)<-Vg_maximum
        particle_velocity(k,d)=-Vg_maximum;
    end
end

%% Controlling the velocity of tap position

for d=member1+1:member1+member2
    if particle_velocity(k,d)>T_maximum
        particle_velocity(k,d)=T_maximum;
    elseif particle_velocity(k,d)<-T_maximum
        particle_velocity(k,d)=-T_maximum;
    end
end

%% Controlling the velocity of shunt capacitor

for d=member1+member2+1:members
    if particle_velocity(k,d)>Qc_maximum
        particle_velocity(k,d)=Qc_maximum;
    elseif particle_velocity(k,d)<-Qc_maximum
        particle_velocity(k,d)=-Qc_maximum;
    end
end
%% Updating and Controlling boundaries of position of generator bus
for d=1:member1
    particle_position(k,d)=particle_position(k,d)+particle_velocity(k,d);
    if particle_position(k,d)>Vg_max
        particle_position(k,d)=Vg_max;
    elseif particle_position(k,d)<Vg_min
        particle_position(k,d)=Vg_min;
```

```
    end
end
%% Updating and Controlling boundaries of tap setting

for d=member1+1:member1+member2
    particle_position(k,d)=particle_position(k,d)+particle_velocity(k,d);
    if particle_position(k,d)>T_max
        particle_position(k,d)=T_max;
    elseif particle_position(k,d)<T_min
        particle_position(k,d)=T_min;
    end
end
%% Updating and Controlling boundaries of Shunt capacitor

for d=member1+member2+1:members
    particle_position(k,d)=particle_position(k,d)+particle_velocity(k,d);
    if particle_position(k,d)>Qc_max
        particle_position(k,d)=Qc_max;
    elseif particle_position(k,d)<Qc_min
        particle_position(k,d)=Qc_min;
        r=r+1;
    end
end

step_size_tap=0.01; % Tap position step size
step_size_Qc=5;     % Shunt capacitor step size
M_tap=round((T_max-T_min)/step_size_tap); % Tap intevals
M_Qc=round((Qc_max-Qc_min)/step_size_Qc); % Qc intervals

discrete_tap=T_min:step_size_tap:T_max; % Discretizing values for tap setting
discrete_Qc=Qc_min:step_size_Qc:Qc_max; % Discretizing values for capacitor

% Finding discrete value for tap position
for d=member1+1:member1+member2
    for n=1:length(discrete_tap)-1
        if particle_position(k,d)>discrete_tap(n)&&particle_position(k,d)<discrete_tap(n+1)
            if abs(particle_position(k,d)-discrete_tap(n))<=abs(particle_position(k,d)-discrete_tap(n+1))
                particle_position(k,d)=discrete_tap(n);
```

```
            else
                particle_position(k,d)=discrete_tap(n+1);
            end
        end
    end
end

% Finding discrete value for shunt capacitor
for d=member1+member2+1:members
    for m=1:length(discrete_Qc)-1
        if particle_position(k,d)>discrete_Qc(m)&&particle_position(k,d)<discrete_Qc(m+1)
            if abs(particle_position(k,d)-discrete_Qc(m))<=abs(particle_position(k,d)-discrete_Qc(m+1))
                particle_position(k,d)=discrete_Qc(m);
            else
                particle_position(k,d)=discrete_Qc(m+1);
            end
        end
    end
end

% Defining the position of voltage at generator bus in bus_dat matrix

v1=particle_position(k,1); bus_dat(1,8)=v1;
v2=particle_position(k,2); bus_dat(2,8)=v2;
v3=particle_position(k,3); bus_dat(3,8)=v3;
v6=particle_position(k,4); bus_dat(6,8)=v6;
v8=particle_position(k,5); bus_dat(8,8)=v8;

% Defining the position of voltage set generator bus in gen_dat matrix
v1=particle_position(k,1); gen_dat(1,6)=v1;
v2=particle_position(k,2); gen_dat(2,6)=v2;
v3=particle_position(k,3); gen_dat(3,6)=v3;
v6=particle_position(k,4); gen_dat(4,6)=v6;
v8=particle_position(k,5); gen_dat(5,6)=v8;

% Defining the position of tap ratio
% tr1(4-7), tr2(4-9), tr3(5-6)
```

```
tr1=particle_position(k,6); line_dat(8,9)=tr1;
tr2=particle_position(k,7); line_dat(9,9)=tr2;
tr3=particle_position(k,8); line_dat(10,9)=tr3;

% Defining the position of the shunt capacitor
qsh9=particle_position(k,9); bus_dat(9,6)=qsh9;

% Calling the function Function which computes the power loss
[P_calc_tot,Q_calc_tot,Pg_max_all,Pg_min_all,Q_max_all,Q_min_all,PV_bus,PQ_bus,V,type,nbus,...
nbranch,Delta,g,Sbase,frombus,tobus]=...
newton_2(bus_dat,line_dat,gen_dat);

% Penalty of the bus voltage violation
penalty_bus2=zeros(PQ_bus,1);

% Checking the voltage limits violations for non-generator buses(PQ)
m=1;
for n=1:nbus
   if type(n)==1
      if V(n)>Vg_max
         penalty_bus2(m)=100000*(V(n)-Vg_max)^2;
         m=m+1;
      elseif V(n)<Vg_min
         penalty_bus2(m)=100000*(Vg_min-V(n))^2;
         m=m+1;
      else
         penalty_bus2(m)=0;
         m=m+1;
      end
   end
end
penalty_bus_violation2(k)=sum(penalty_bus2);

% Penality of the generator violating the limits
penalty_gen2=zeros(PV_bus,1);
r=1;
for d=1:nbus
   if type(d)==2
```

```
            if Q_calc_tot(d)>Q_max_all(d)/Sbase
               penalty_gen2(r)=100000*(Q_calc_tot(d)-(Q_max_all(d)/Sbase))^2;
               r=r+1;
            elseif Q_calc_tot(d)<Q_min_all(d)/Sbase
                penalty_gen2(r)=100000*((Q_min_all(d)/Sbase)-Q_calc_tot(d))^2;
                r=r+1;
            else
               penalty_gen2(r)=0;
               r=r+1;
            end
         end
      end
      penalty_gen_violation2(k)=sum(penalty_gen2);

      % Calling newton raphson fuction after replacing parameters

      [P_loss]=object_function(nbranch,Delta,V,g,Sbase,frombus,tobus);
      Power_loss2(k)=P_loss;

      % Evaluation of the augmented function (fitness function)
      particle_loss(k)=Power_loss2(k)+penalty_bus_violation2(k)+penalty_gen_violation2(k);

      % Updating the personal best
      if particle_loss(k)<=particle_best_loss(k)
        particle_best_position(k,:)=particle_position(k,:);
        particle_best_loss(k)=particle_loss(k);

        % Updating the global best
        if particle_best_loss(k)<=Globalbest_loss
          Globalbest_position=particle_best_position(k,:);
          Globalbest_loss=particle_best_loss(k);
        end
      end
   end

      % Storing the best loss value at each iteration
      Best_loss(it)=Globalbest_loss;
      disp(['Iteration', num2str(it) ': Best loss= ', num2str(Best_loss(it))]);
```

```
end

bus_num=bus_dat(:,1);  % Bus number

%% Results of the PSO

% Plotting the loss iterations
figure;
plot(Best_loss,'LineWidth',1);
title('Reactive power optimization based PSO (14 Bus IEEE)');
xlabel('Iteration');
ylabel('Active power loss in MW ');
grid on;

% Plotting the voltage after optimization with PSO
figure;
plot(bus_num,V,'--*r');
title('Voltage at buses for IEEE 14 Bus System after Optimization based PSO');
xlabel('Bus number');
ylabel('Voltage in p.u')
grid on
```

Appendix4: Hybrid PSO-ABC codes for IEEE 14 Bus system

```
%=========================================================================%
%                        Implementation of Hybrid PSO-ABC                 %
%                        Based Active Power Loss Minimization             %
%                              IEEE 14 Bus System                         %
%=========================================================================%

clc;
clear;

%%  The case 14_bus IEEE
load line_dat.txt;     % Loading the line data
load bus_dat.txt;      % loading the bus data
```

```
load gen_dat.txt;      % Loading generator data
%% Defining the problem

members=9;   % All control Variables (Vg,T,Qc)
members_size=[1 members]; % The size of vector of all control variables
member1=5;   % Control variable Vg (Voltage at generator buses)
member2=3;   % Control variable T (Tap setting of transformer)
member3=1;   % Control variable Qc (Reactive power of shunt capacitor)
size_V=[1 member1]; % The size of control variable Vg
size_T=[1 member2]; % The size of control variable T
size_Qc=[1 member3]; % The size of control variable Qc

% Defining the limits of voltage, tap setting and shunt capasitor
Vg_min=0.95;  % Minimun voltage
Vg_max=1.10;  % Maximum voltage
T_min=0.90; % Minimum tap position
T_max=1.10; % Maximum tap position
Qc_min=0;    % Minimum value of capacitor connected
Qc_max=20;   % Maximum value of capacitor connected

% Defining all PSO parameters
num_particle=50;   % Number of particles (Swarm size)
w_min=0.4;         % Minimum value of inertia weight coefficient
w_max=0.9;         % Maximum value of inertia weight coefficient
% c1=2.05;          % Acceleration coefficient (Cognitive/Personal)
% c2=2.05;          % Acceleration coefficient (Social/Global)
% H=c1+c2;          % Coefficient facilitating the exproration
limit=5;
%% Initialization of the algorithm

% The particle template in form of structures
void_particle.position=[];       % To initialize the particle position
void_particle.velocity=[];       % To initialize the particle velocity
void_particle.loss=[];           % To initialize measurement of each particle
void_particle.best.position=[];   % To initialize the particle personal best ever experienced
void_particle.best.loss=[];      % To initialise the best measurement ever experienced

void_particle1.position=[];       % To initialize the particle position
```

```
void_particle1.velocity=[];        % To initialize the particle velocity
void_particle1.loss=[];            % To initialize measurement of each particle
void_particle1.best.position=[];   % To initialize the particle personal best ever experienced
void_particle1.best.loss=[];

% Creation of the swarm array
% Control vatiable are:
% - V (Voltage at generator bus)
% - T (Transformer tap setting)
% - Qc (Reacative power at shunt capacitors)
% For this case the total control variables are 9
particle=repmat(void_particle,num_particle,1); % To proliferate the size of all particles
particle1=repmat(void_particle1,num_particle,1); % To proliferate the size of all particles

%-------------------------------------------------------------------------%
% Generating Chaotic random numbers based Logistic and Tent map
lamda1=2;
lamda2=4;
random_val=zeros(num_particle,members); % Storing all logistic vectors
r=1;
for k=1:num_particle
   for m=1:members
      initial_rand=rand(1,members); % Initial random vector
      if k==1
        random_val(r,m)=initial_rand(1,m);
      else
         if random_val(k-1,m)<=0.2
            random_val(k,m)=lamda2*random_val(k-1,m);
         elseif random_val(k-1,m)>0.2 && random_val(k-1,m)<0.5
            random_val(k,m)=lamda1*random_val(k-1,m);
         elseif random_val(k-1,m)>=0.5 && random_val(k-1,m)<=0.7
            random_val(k,m)=lamda1*(1-random_val(k-1,m));
         elseif random_val(k-1,m)>0.7
            random_val(k,m)=lamda2*random_val(k-1,m)*(1-random_val(k-1,m));
         end
      end
   end
```

```
end

% Generating initial population/solution based logistic map
Vg_control=zeros(num_particle,member1);
tap_control=zeros(num_particle,member2);
Qc_control=zeros(num_particle,member3);
for k=1:num_particle
   % For Voltage at generator buses
   for d=1:member1
      Vg_control(k,d)=Vg_min+(random_val(k,d)*(Vg_max-Vg_min));
   end
   % For Tap position
   r=1;
   for d=member1+1:member1+member2
     tap_control(k,r)=T_min+(random_val(k,d)*(T_max-T_min));
     r=r+1;
   end
   % For shunt capacitor
   r=1;
   for d=member1+member2+1:members
     Qc_control(k,r)=Qc_min+(random_val(k,d)*(Qc_max-Qc_min));
     r=r+1;
   end
end
% Opposition based
% Matrices of maximum and minimum
maximum_Vg=ones(num_particle,member1)*Vg_max;
minimum_Vg=ones(num_particle,member1)*Vg_min;
maximum_T=ones(num_particle,member2)*T_max;
minimum_T=ones(num_particle,member2)*T_min;
maximum_Qc=ones(num_particle,member3)*Qc_max;
minimum_Qc=ones(num_particle,member3)*Qc_min;

Vg_control1=zeros(num_particle,member1);
tap_control1=zeros(num_particle,member2);
Qc_control1=zeros(num_particle,member3);
for k=1:num_particle
   % For generator buses
```

```
    for d=1:member1
        Vg_control1(k,d)=minimum_Vg(k,d)+maximum_Vg(k,d)-Vg_control(k,d);
    end
    % For tap position
    for d=1:member2
        tap_control1(k,d)=minimum_T(k,d)+maximum_T(k,d)-tap_control(k,d);
    end
    % For shunt capacitor
    for d=1:member3
        Qc_control1(k,d)=minimum_Qc(k,d)+maximum_Qc(k,d)-Qc_control(k,d);
    end
end
% Replacing values in matrices depending on the highest value
for k=1:num_particle
    % For generator buses
    for d=1:member1
        if Vg_control(k,d)>Vg_control1(k,d)
            Vg_control1(k,d)=Vg_control(k,d);
        end
    end
    % For tap position
    for d=1:member2
        if tap_control(k,d)>tap_control1(k,d)
            tap_control1(k,d)=tap_control(k,d);
        end
    end
    % For shunt capacitor
    for d=1:member3
        if Qc_control(k,d)>Qc_control1(k,d)
            Qc_control1(k,d)=Qc_control(k,d);
        end

    end
end

control_variable=zeros(num_particle,members);   % Initialization of matrix of control variables
penalty_bus_violation1=zeros(num_particle,1);   % Storing the penalty of PQ-bus violating voltage limits
penalty_gen_violation1=zeros(num_particle,1);   % Storing the penalty of generator violating reactive powerlimits
```

```
penalty_Slack_violation1=zeros(num_particle,1); % Storing the penality of Slack bus when active power limits are violated
Power_loss1=zeros(num_particle,1); % Storing the losses

velocity_initial=zeros(num_particle,members);
best_position=zeros(num_particle,members);

for k=1:num_particle

   % Concatenating the matrices to form particle.position
   particle(k).position=[Vg_control1(k,:) tap_control1(k,:) Qc_control1(k,:)];
   %+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++%
   step_size_tap=0.01; % Tap position step size
   step_size_Qc=5;     % Shunt capacitor step size
   M_tap=round((T_max-T_min)/step_size_tap); % Tap intevals
   M_Qc=round((Qc_max-Qc_min)/step_size_Qc); % Qc intervals


   discrete_tap=T_min:step_size_tap:T_max;
   discrete_Qc=Qc_min:step_size_Qc:Qc_max;

   % Finding discrete value for tap position
   for d=member1+1:member1+member2
      for n=1:length(discrete_tap)-1
         if particle(k).position(d)>discrete_tap(n)&&particle(k).position(d)<discrete_tap(n+1)
            if abs(particle(k).position(d)-discrete_tap(n))<=abs(particle(k).position(d)-discrete_tap(n+1))
               particle(k).position(d)=discrete_tap(n);
            else
               particle(k).position(d)=discrete_tap(n+1);
            end
         end
      end
   end

   % Finding discrete value for shunt capacitor
   for d=member1+member2+1:members
      for m=1:length(discrete_Qc)-1
          if particle(k).position(d)>discrete_Qc(m)&&particle(k).position(d)<discrete_Qc(m+1)
```

```
            if abs(particle(k).position(d)-discrete_Qc(m))<=abs(particle(k).position(d)-discrete_Qc(m+1))
               particle(k).position(d)=discrete_Qc(m);
            else
               particle(k).position(d)=discrete_Qc(m+1);
            end
         end
      end
end

% Defining the parameters'positions in the vector of control variable
% Defining the position of voltage at generator bus in bus_dat matrix

v1=particle(k).position(1); bus_dat(1,8)=v1;
v2=particle(k).position(2); bus_dat(2,8)=v2;
v3=particle(k).position(3); bus_dat(3,8)=v3;
v6=particle(k).position(4); bus_dat(6,8)=v6;
v8=particle(k).position(5); bus_dat(8,8)=v8;

% Defining the position of voltage set generator bus in gen_dat matrix
v1=particle(k).position(1); gen_dat(1,6)=v1;
v2=particle(k).position(2); gen_dat(2,6)=v2;
v3=particle(k).position(3); gen_dat(3,6)=v3;
v6=particle(k).position(4); gen_dat(4,6)=v6;
v8=particle(k).position(5); gen_dat(5,6)=v8;

% Defining the position of tap ratio
% tr1(4-7), tr2(4-9), tr3(5-6)

tr1=particle(k).position(6); line_dat(8,9)=tr1;
tr2=particle(k).position(7); line_dat(9,9)=tr2;
tr3=particle(k).position(8); line_dat(10,9)=tr3;

% Defining the position of the shunt capacitor
qsh9=particle(k).position(9); bus_dat(9,6)=qsh9;

% Calling the function Function which computes the power loss
[P_calc_tot,Q_calc_tot,Pg_max_all,Pg_min_all,Q_max_all,Q_min_all,PV_bus,PQ_bus,V,type,nbus,...
nbranch,Delta,g,Sbase,frombus,tobus]=...
```

```
newton_2(bus_dat,line_dat,gen_dat);

% Penalty of the bus voltage violation
penalty_bus1=zeros(PQ_bus,1);

% Checking the voltage limits violations for non-generator buses(PQ)
m=1;
for n=1:nbus
   if type(n)==1
      if V(n)>Vg_max
         penalty_bus1(m)=100000*(V(n)-Vg_max)^2;
         m=m+1;
      elseif V(n)<Vg_min
         penalty_bus1(m)=100000*(Vg_min-V(n))^2;
         m=m+1;
      else
         penalty_bus1(m)=0;
         m=m+1;
      end
   end
end
penalty_bus_violation1(k)=sum(penalty_bus1);

% Penality of the generator violating the limits
penalty_gen1=zeros(PV_bus,1);
r=1;
for d=1:nbus
   if type(d)==2
      if Q_calc_tot(d)>Q_max_all(d)/Sbase
         penalty_gen1(r)=100000*(Q_calc_tot(d)-(Q_max_all(d)/Sbase))^2;
         r=r+1;
      elseif Q_calc_tot(d)<Q_min_all(d)/Sbase
          penalty_gen1(r)=100000*((Q_min_all(d)/Sbase)-Q_calc_tot(d))^2;
          r=r+1;
      else
         penalty_gen1(r)=0;
         r=r+1;
      end
```

```
        end
    end
    penalty_gen_violation1(k)=sum(penalty_gen1);

    %=====================================================================%
    % Checking violations of active power at Slack bus
    penalty_Slack1=zeros(1,1);
    r=1;
    for d=1:nbus
        if type(d)==3
            if P_calc_tot(d)>Pg_max_all(d)/Sbase
                penalty_Slack1(r)=100000*(P_calc_tot(d)-(Pg_max_all(d)/Sbase))^2;
                r=r+1;
            elseif P_calc_tot(d)<Pg_min_all(d)/Sbase
                 penalty_Slack1(r)=100000*((Pg_min_all(d)/Sbase)-P_calc_tot(d))^2;
                 r=r+1;
            else
                penalty_Slack1(r)=0;
                r=r+1;
            end
        end
    end
    penalty_Slack_violation1(k)=sum(penalty_Slack1);

    %=====================================================================%

    % Calling newton raphson fuction after replacing parameters

    [P_loss]=object_function(nbranch,Delta,V,g,Sbase,frombus,tobus);
    Power_loss1(k)=P_loss;

    % Evaluation of the augmented function (fitness function)
    particle(k).loss=Power_loss1(k)+penalty_bus_violation1(k)+penalty_gen_violation1(k)+penalty_Slack_violation1(k);

    % Initialization of the Personal best postion and loss
    particle(k).best.position=particle(k).position;
    particle(k).best.loss=particle(k).loss;
end
```

```
% Initialization of the velocity of particles
Vg_maximum=Vg_max*0.01;
T_maximum=(T_max-T_min)/M_tap;
Qc_maximum=(Qc_max-Qc_min)/M_Qc;

% Defining the matrix of maximum velocities for all particles
Velocity_max=[Vg_maximum*ones(k,member1) T_maximum*ones(k,member2) Qc_maximum*ones(k,member3)];

% Defining the matrix of maximum velocities for all particles
Velocity_min=[-Vg_maximum*ones(k,member1) -T_maximum*ones(k,member2) -Qc_maximum*ones(k,member3)];

velocity_gen=zeros(num_particle,member1);
velocity_tap=zeros(num_particle,member2);
velocity_shunt=zeros(num_particle,member3);
for k=1:num_particle
   % For generator buses
   for d=1:member1
      velocity_gen(k,d)=-Vg_maximum+(random_val(k,d)*(Vg_maximum-(-Vg_maximum)));
   end
   % For tap position
   r=1;
   for d=member1+1:member1+member2
      velocity_tap(k,r)=-T_maximum+(random_val(k,d)*(T_maximum-(-T_maximum)));
      r=r+1;
   end
   % For shunt capacitor
   r=1;
   for d=member1+member2+1:members
      velocity_shunt(k,r)=-Qc_maximum+(random_val(k,d)*(Qc_maximum-(-Qc_maximum)));
      r=r+1;
   end
   % Concatenation of particle velocity
   particle(k).velocity=[velocity_gen(k,:) velocity_tap(k,:) velocity_shunt(k,:)];

   % Storing best position of each particle
   for d=1:members
      best_position(k,d)=particle(k).best.position(d);
   end
```

```
end
% Computing the Global best position and loss
Globalbest.loss=min([particle(:).loss]); % The minimum value of losses
location= find([particle.loss]==min([particle(:).loss])); % Locate the position of the minimum loss
Globalbest.position=[particle(location).position];

% Defining Swarms
swarm1=10;
swarm2=15;
swarm3=25;

% Identifying the first swarm
loss_swarm1=zeros(swarm1,1);
best_position_swarm1=zeros(swarm1,members);
for n=1:swarm1
   loss_swarm1(n)=particle(n).loss;
   for d=1:members
     best_position_swarm1(n,d)=best_position(n,d);
   end
end
mean_best_swarm1=sum(best_position_swarm1)/swarm1;

Globalbest1.loss=min(loss_swarm1); % The minimum value of losses
location1= find([particle.loss]==Globalbest1.loss); % Locate the position of the minimum loss
Globalbest1.position=[particle(location1).position];

% Identifying the second swarm
loss_swarm2=zeros(swarm2,1);
best_position_swarm2=zeros(swarm2,members);
r=1;
for m=swarm1+1:swarm1+swarm2
   loss_swarm2(r)=particle(m).loss;
   best_position_swarm2(r,:)=best_position(m,:);
   r=r+1;
end
mean_best_swarm2=sum(best_position_swarm2)/swarm2;
```

```
Globalbest2.loss=min(loss_swarm2); % The minimum value of losses
location2= find([particle.loss]==Globalbest2.loss); % Locate the position of the minimum loss
Globalbest2.position=[particle(location2).position];

% Identifying the third swarm
loss_swarm3=zeros(swarm3,1);
best_position_swarm3=zeros(swarm3,members);
r=1;
for t=swarm1+swarm2+1:swarm1+swarm2+swarm3
   loss_swarm3(r)=particle(t).loss;
   best_position_swarm3(r,:)=best_position(t,:);
   r=r+1;
end
mean_best_swarm3=sum(best_position_swarm3)/swarm3;
Globalbest3.loss=min(loss_swarm3); % The minimum value of losses
location3= find([particle.loss]==Globalbest3.loss); % Locate the position of the minimum loss
Globalbest3.position=[particle(location3).position];

%% Implementation of the main loop of Multimean_PSO

max_iter=200;
% Setting a mechanism that will inform us the Globalbest loss at every iteration
Best_loss=zeros(max_iter,1);
it=1;
abandon=zeros(num_particle,1); % store the pbest which do not improve
for it=1:max_iter

   c_max=2.5;
   c_min=0.5;
   %=========================================================================%
   % Updating the velocity and position of swarm1
   w1=zeros(swarm1,1); % Initialize the weight factor
   c1=zeros(swarm1,1); % Acceleration factor for cognitive
   c2=zeros(swarm1,1); % Acceleration factor for social in local swarm
   c3=zeros(swarm1,1); % Acceleration factor for social in all swarm

   % random vectors r1,r2, r3 by permutation of vectors
   size_Initial=size(random_val); % Size of random_val matrix
```

```
permute1=randperm(size_Initial(1,1),1);
permute2=randperm(size_Initial(1,1),1);
permute3=randperm(size_Initial(1,1),1);
r1=random_val(permute1,:);
r2=random_val(permute2,:);
r3=random_val(permute3,:);

for n=1:swarm1
   % Calculation of weight factor for particles of the swarm
   w1(n)=w_min+((w_max-w_min)*(max_iter-n))/(max_iter-1);
   c2(n)=c_min+((c_max-c_min)*((max_iter-it)/max_iter));
   c1(n)=c_max-((c_max-c_min)*((max_iter-it)/max_iter));
   c3(n)=c2(n);

   % Updating the velocity of the particles
   particle(n).velocity=(w1(n)*particle(n).velocity) ...
    +((c1(n)*r1).*(mean_best_swarm1(1,:)-particle(n).position)) ...
    +((c2(n)*r2).*(Globalbest1.position-particle(n).position))...
    +((c3(n)*r3).*(Globalbest.position-particle(n).position));

   % Controlling the velocity boundaries
   for d=1:members
      if particle(n).velocity(d)>Velocity_max(n,d)
         particle(n).velocity(d)=Velocity_max(n,d);
      elseif particle(n).velocity(d)<Velocity_min(n,d)
         particle(n).velocity(d)=Velocity_min(n,d);
      end
   end

   % Updating the positions of swarm1
   particle(n).position=particle(n).position+particle(n).velocity;
end
%=========================================================================%
% Updating the velocity and position of swarm2
w2=zeros(swarm2,1); % Initialize the weight factor
c4=zeros(swarm2,1); % Acceleration factor for cognitive
c5=zeros(swarm2,1); % Acceleration factor for social in local swarm
c6=zeros(swarm2,1); % Acceleration factor for social in all swarm
```

```
% random vectors r4,r5,r6 by permutation of vectors
size_Initial=size(random_val); % Size of random_val matrix
permute4=randperm(size_Initial(1,1),1);
permute5=randperm(size_Initial(1,1),1);
permute6=randperm(size_Initial(1,1),1);
r4=random_val(permute4,:);
r5=random_val(permute5,:);
r6=random_val(permute6,:);

r=1;
for m=swarm1+1:swarm1+swarm2
   % Calculation of weight factor for particles of the swarm
   w2(r)=w_min+((w_max-w_min)*(max_iter-m))/(max_iter-1);
   c5(r)=c_min+((c_max-c_min)*((max_iter-it)/max_iter));
   c4(r)=c_max-((c_max-c_min)*((max_iter-it)/max_iter));
   c6(r)=c5(r);

   % Updating the velocity of the particles
   particle(m).velocity=(w2(r)*particle(m).velocity) ...
    +((c4(r)*r4).*(mean_best_swarm2(1,:)-particle(m).position)) ...
    +((c5(r)*r5).*(Globalbest2.position-particle(m).position))...
    +((c6(r)*r6).*(Globalbest.position-particle(m).position));
   r=r+1;

   % Controlling the velocity boundaries
   for d=1:members
      if particle(m).velocity(d)>Velocity_max(m,d)
         particle(m).velocity(d)=Velocity_max(m,d);
      elseif particle(m).velocity(d)<Velocity_min(m,d)
         particle(m).velocity(d)=Velocity_min(m,d);
      end
   end

   % Updating the positions of swarm2
   particle(m).position=particle(m).position+particle(m).velocity;
end
%=================================================================%
```

```
 % Updating the velocity and position of swarm3
w3=zeros(swarm3,1); % Initialize the weight factor
c7=zeros(swarm3,1); % Acceleration factor for cognitive
c8=zeros(swarm3,1); % Acceleration factor for social in local swarm
c9=zeros(swarm3,1); % Acceleration factor for social in all swarm
r=1;
for t=swarm1+swarm2+1:swarm1+swarm2+swarm3
   % Calculation of weight factor for particles of the swarm
   w3(r)=w_min+((w_max-w_min)*(max_iter-t))/(max_iter-1);
   c8(r)=c_min+((c_max-c_min)*((max_iter-it)/max_iter));
   c7(r)=c_max-((c_max-c_min)*((max_iter-it)/max_iter));
   c9(r)=c8(r);

   % random vectors r7,r8,r9 by permutation of vectors
   size_Initial=size(random_val); % Size of random_val matrix
   permute7=randperm(size_Initial(1,1),1);
   permute8=randperm(size_Initial(1,1),1);
   permute9=randperm(size_Initial(1,1),1);
   r7=random_val(permute7,:);
   r8=random_val(permute8,:);
   r9=random_val(permute9,:);

   % Updating the velocity of the particles
   particle(t).velocity=(w3(r)*particle(t).velocity) ...
    +((c7(r)*r7).*(mean_best_swarm3(1,:)-particle(t).position)) ...
    +((c8(r)*r8).*(Globalbest3.position-particle(t).position))...
    +((c9(r)*r9).*(Globalbest.position-particle(t).position));
   r=r+1;

   % Controlling the velocity boundaries
   for d=1:members
      if particle(t).velocity(d)>Velocity_max(t,d)
         particle(t).velocity(d)=Velocity_max(t,d);
      elseif particle(t).velocity(d)<Velocity_min(t,d)
         particle(t).velocity(d)=Velocity_min(t,d);
      end
   end
```

```
    % Updating the positions of swarm3
    particle(t).position=particle(t).position+particle(t).velocity;
end

penalty_bus_violation2=zeros(num_particle,1);
penalty_gen_violation2=zeros(num_particle,1);
penalty_Slack_violation2=zeros(num_particle,1);
Power_loss2=zeros(num_particle,1); % Initialize the power loss
for k=1:num_particle
    % Controlling the violation of boundaies of control variables
    % Controlling the boundaries of generator bus
    for d=1:member1
        if particle(k).position(d)>Vg_max
            particle(k).position(d)=Vg_max;
        elseif particle(k).position(d)<Vg_min
            particle(k).position(d)=Vg_min;
        end
    end

    % Controlling the boundaries for tap position
    for d=member1+1:member1+member2
        if particle(k).position(d)>T_max
            particle(k).position(d)=T_max;
        elseif particle(k).position(d)<T_min
            particle(k).position(d)=T_min;
        end
        range_tap=T_min:step_size_tap:T_max;
        for n=1:length(range_tap)-1
            if particle(k).position(d)>range_tap(n)&&particle(k).position(d)<range_tap(n+1)
                if abs(particle(k).position(d)-range_tap(n))<=abs(particle(k).position(d)-(range_tap(n+1)))
                    particle(k).position(d)=range_tap(n);
                else
                    particle(k).position(d)=range_tap(n+1);
                end
            end
        end
    end
```

```
% Controlling the boundaries for shunt capacitor
for d=member1+member2+1:members
   if particle(k).position(d)>Qc_max
      particle(k).position(d)=Qc_max;
   elseif particle(k).position(d)<Qc_min
      particle(k).position(d)=Qc_min;
   end
   range_Qc=0:5:Qc_max;
   for m=1:length(range_Qc)-1
       if particle(k).position(d)>range_Qc(m)&&particle(k).position(d)<range_Qc(m+1)
          if abs(particle(k).position(d)-range_Qc(m))<=abs(particle(k).position(d)-(range_Qc(m+1)))
             particle(k).position(d)=range_Qc(m);
           else
             particle(k).position(d)=range_Qc(m+1);
          end
       end
   end
end

% Defining the parameters'positions in the vector of control variable
% Defining the position of voltage at generator bus in bus_dat matrix

v1=particle(k).position(1); bus_dat(1,8)=v1;
v2=particle(k).position(2); bus_dat(2,8)=v2;
v3=particle(k).position(3); bus_dat(3,8)=v3;
v6=particle(k).position(4); bus_dat(6,8)=v6;
v8=particle(k).position(5); bus_dat(8,8)=v8;

% Defining the position of voltage set generator bus in gen_dat matrix
v1=particle(k).position(1); gen_dat(1,6)=v1;
v2=particle(k).position(2); gen_dat(2,6)=v2;
v3=particle(k).position(3); gen_dat(3,6)=v3;
v6=particle(k).position(4); gen_dat(4,6)=v6;
v8=particle(k).position(5); gen_dat(5,6)=v8;

% Defining the position of tap ratio
% tr1(4-7), tr2(4-9), tr3(5-6)
```

```
    tr1=particle(k).position(6); line_dat(8,9)=tr1;
    tr2=particle(k).position(7); line_dat(9,9)=tr2;
    tr3=particle(k).position(8); line_dat(10,9)=tr3;

    % Defining the position of the shunt capacitor
    qsh9=particle(k).position(9); bus_dat(9,6)=qsh9;

    % Calling the function which computes the powe loss
    [P_calc_tot,Q_calc_tot,Pg_max_all,Pg_min_all,Q_max_all,Q_min_all,PV_bus,PQ_bus,V,type,nbus,...
nbranch,Delta,g,Sbase,frombus,tobus]=...
newton_2(bus_dat,line_dat,gen_dat);

    % Penalty of the bus voltage violation
    penalty_bus2=zeros(PQ_bus,1);

    % Checking the voltage limits violations for non-generator buses(PQ)
    r=1;
    for x=1:nbus
       if type(x)==1
          if V(x)>Vg_max
             penalty_bus2(r)=100000*(V(x)-Vg_max)^2;
             r=r+1;
          elseif V(x)<Vg_min
             penalty_bus2(r)=100000*(Vg_min-V(x))^2;
             r=r+1;
          else
             penalty_bus2(r)=0;
             r=r+1;
          end
       end
    end
    penalty_bus_violation2(k)=sum(penalty_bus2);

    % Penality of the generator violating the limits
    penalty_gen2=zeros(PV_bus,1);
    r=1;
    for d=1:nbus
       if type(d)==2
```

```
        if Q_calc_tot(d)>Q_max_all(d)/Sbase
            penalty_gen2(r)=100000*(Q_calc_tot(d)-(Q_max_all(d)/Sbase))^2;
            r=r+1;
        elseif Q_calc_tot(d)<Q_min_all(d)/Sbase
             penalty_gen2(r)=100000*((Q_min_all(d)/Sbase)-Q_calc_tot(d))^2;
             r=r+1;
        else
            penalty_gen2(r)=0;
            r=r+1;
        end
    end
end
penalty_gen_violation2(k)=sum(penalty_gen2);

%=====================================================================%
% Checking violations of active power at Slack bus
penalty_Slack2=zeros(1,1);
r=1;
for d=1:nbus
    if type(d)==3
        if P_calc_tot(d)>Pg_max_all(d)/Sbase
            penalty_Slack2(r)=100000*(P_calc_tot(d)-(Pg_max_all(d)/Sbase))^2;
            r=r+1;
        elseif P_calc_tot(d)<Pg_min_all(d)/Sbase
             penalty_Slack2(r)=100000*((Pg_min_all(d)/Sbase)-P_calc_tot(d))^2;
             r=r+1;
        else
            penalty_Slack2(r)=0;
            r=r+1;
        end
    end
end
penalty_Slack_violation2(k)=sum(penalty_Slack2);

%=====================================================================%

% Calculating power losses after replacing parameters
```

```
      [P_loss]=object_function(nbranch,Delta,V,g,Sbase,frombus,tobus);
      Power_loss2(k)=P_loss;

      % To conduct the evaluation of members in terms of loss
      particle(k).loss=Power_loss2(k)+penalty_bus_violation2(k)+penalty_gen_violation2(k)+penalty_Slack_violation2(k);

      % Updating the personal best
      if particle(k).loss<=particle(k).best.loss
        particle(k).best.position=particle(k).position;
        particle(k).best.loss=particle(k).loss;
        abandon(k)=abandon(k)+1;
      end

      % Storing the best position of each particle at each iteration
      for d=1:members
         best_position(k,d)=particle(k).best.position(d);
      end

   end

   Globalbest4.loss=min([particle(:).loss]); % The minimum value of losses
   locate= find([particle.loss]==min([particle(:).loss])); % Locate the position of the minimum loss
   Globalbest4.position=[particle(locate).position];

   if Globalbest4.loss<Globalbest.loss
      Globalbest=Globalbest4;
   end
%-------------------------------------------------------------------------%
%           Scout phase (generation of new population)                    %
%-------------------------------------------------------------------------%

% Generating Chaotic random numbers based Logistic map
lamda1=2;
lamda2=4;
random_val2=zeros(num_particle,members); % Storing all logistic vectors
r=1;
for k=1:num_particle
   for m=1:members
```

```
        initial_rand2=rand(1,members); % Initial random vector
        if k==1
          random_val2(r,m)=initial_rand2(1,m);
        else
          if random_val2(k-1,m)<=0.2
            random_val2(k,m)=lamda2*random_val2(k-1,m);
          elseif random_val2(k-1,m)>0.2 && random_val2(k-1,m)<0.5
            random_val2(k,m)=lamda1*random_val2(k-1,m);
          elseif random_val2(k-1,m)>=0.5 && random_val2(k-1,m)<=0.7
            random_val2(k,m)=lamda1*(1-random_val2(k-1,m));
          elseif random_val2(k-1,m)>0.7
            random_val2(k,m)=lamda2*random_val2(k-1,m)*...
              (1-random_val2(k-1,m));
          end
        end
    end
end
% Generating new population/solution based logistic map
Vg_control_new=zeros(num_particle,member1);
tap_control_new=zeros(num_particle,member2);
Qc_control_new=zeros(num_particle,member3);
for k=1:num_particle
    % For Voltage at generator buses
    for d=1:member1
        Vg_control_new(k,d)=Vg_min+(random_val2(k,d)*(Vg_max-Vg_min));
    end
    % For Tap position
    r=1;
    for d=member1+1:member1+member2
      tap_control_new(k,r)=T_min+(random_val2(k,d)*(T_max-T_min));
      r=r+1;
    end
    % For shunt capacitor
    r=1;
    for d=member1+member2+1:members
      Qc_control_new(k,r)=Qc_min+(random_val2(k,d)*(Qc_max-Qc_min));
      r=r+1;
    end
```

```
end

% Opposition based learning
Vg_control1_new=zeros(num_particle,member1);
tap_control1_new=zeros(num_particle,member2);
Qc_control1_new=zeros(num_particle,member3);
for k=1:num_particle
   % For generator buses
   for d=1:member1
      Vg_control1_new(k,d)=minimum_Vg(k,d)+maximum_Vg(k,d)-Vg_control_new(k,d);
   end
   % For tap position
   for d=1:member2
      tap_control1_new(k,d)=minimum_T(k,d)+maximum_T(k,d)-tap_control_new(k,d);
   end
   % For shunt capacitor
   for d=1:member3
      Qc_control1_new(k,d)=minimum_Qc(k,d)+maximum_Qc(k,d)-Qc_control_new(k,d);
   end
end
% Replacing values in matrices depending on the highest value
for k=1:num_particle
   % For generator buses
   for d=1:member1
      if Vg_control_new(k,d)>Vg_control_new(k,d)
         Vg_control1_new(k,d)=Vg_control_new(k,d);
      end
   end
   % For tap position
   for d=1:member2
      if tap_control_new(k,d)>tap_control1_new(k,d)
         tap_control1_new(k,d)=tap_control_new(k,d);
      end
   end
   % For shunt capacitor
   for d=1:member3
      if Qc_control_new(k,d)>Qc_control1_new(k,d)
         Qc_control1_new(k,d)=Qc_control_new(k,d);
```

```
        end

    end
end

swarm=[Vg_control1_new tap_control1_new Qc_control1_new];
%---------------------------------------------------------------------%

penalty_bus_violation3=zeros(num_particle,1);
penalty_gen_violation3=zeros(num_particle,1);
penalty_Slack_violation3=zeros(num_particle,1);
Power_loss3=zeros(num_particle,1); % Initialize the power loss

for k=1:num_particle
    if abandon(k)>=limit
        particle(k).position=swarm(k,:);
%         particle(k).best.position=swarm(k,:);
        particle(k).velocity=velocity_initial(k,:);

        %+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++%
        step_size_tap=0.01; % Tap position step size
        step_size_Qc=5;     % Shunt capacitor step size
        M_tap=round((T_max-T_min)/step_size_tap); % Tap intevals
        M_Qc=round((Qc_max-Qc_min)/step_size_Qc); % Qc intervals

        discrete_tap=T_min:step_size_tap:T_max;
        discrete_Qc=Qc_min:step_size_Qc:Qc_max;

        % Finding discrete value for tap position
        for d=member1+1:member1+member2
            for n=1:length(discrete_tap)-1
                if particle(k).position(d)>discrete_tap(n)&&particle(k).position(d)<discrete_tap(n+1)
                    if abs(particle(k).position(d)-discrete_tap(n))<=abs(particle(k).position(d)-discrete_tap(n+1))
                        particle(k).position(d)=discrete_tap(n);
                    else
                        particle(k).position(d)=discrete_tap(n+1);
                    end
                end
```

```
    end
end

% Finding discrete value for shunt capacitor
for d=member1+member2+1:members
    for m=1:length(discrete_Qc)-1
        if particle(k).position(d)>discrete_Qc(m)&&particle(k).position(d)<discrete_Qc(m+1)
            if abs(particle(k).position(d)-discrete_Qc(m))<=abs(particle(k).position(d)-discrete_Qc(m+1))
                particle(k).position(d)=discrete_Qc(m);
            else
                particle(k).position(d)=discrete_Qc(m+1);
            end
        end
    end
end
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++%
% Defining the parameters'positions in the vector of control variable
% Defining the position of voltage at generator bus in bus_dat matrix

v1=particle(k).position(1); bus_dat(1,8)=v1;
v2=particle(k).position(2); bus_dat(2,8)=v2;
v3=particle(k).position(3); bus_dat(3,8)=v3;
v6=particle(k).position(4); bus_dat(6,8)=v6;
v8=particle(k).position(5); bus_dat(8,8)=v8;

% Defining the position of voltage set generator bus in gen_dat matrix
 v1=particle(k).position(1); gen_dat(1,6)=v1;
 v2=particle(k).position(2); gen_dat(2,6)=v2;
 v3=particle(k).position(3); gen_dat(3,6)=v3;
 v6=particle(k).position(4); gen_dat(4,6)=v6;
 v8=particle(k).position(5); gen_dat(5,6)=v8;

 % Defining the position of tap ratio
 % tr1(4-7), tr2(4-9), tr3(5-6)

 tr1=particle(k).position(6); line_dat(8,9)=tr1;
 tr2=particle(k).position(7); line_dat(9,9)=tr2;
 tr3=particle(k).position(8); line_dat(10,9)=tr3;
```

```
 % Defining the position of the shunt capacitor
 qsh9=particle(k).position(9); bus_dat(9,6)=qsh9;

% Calling the function which computes the powe loss
[P_calc_tot,Q_calc_tot,Pg_max_all,Pg_min_all,Q_max_all,Q_min_all,PV_bus,PQ_bus,V,type,nbus,...
nbranch,Delta,g,Sbase,frombus,tobus]=...
newton_2(bus_dat,line_dat,gen_dat);

% Penalty of the bus voltage violation
 penalty_bus3=zeros(PQ_bus,1);

% Checking the voltage limits violations for non-generator buses(PQ)
m=1;
for n=1:nbus
   if type(n)==1
      if V(n)>Vg_max
         penalty_bus3(m)=100000*(V(n)-Vg_max)^2;
         m=m+1;
      elseif V(n)<Vg_min
         penalty_bus3(m)=100000*(Vg_min-V(n))^2;
         m=m+1;
      else
         penalty_bus1(m)=0;
         m=m+1;
      end
  end
end
penalty_bus_violation3(k)=sum(penalty_bus3);

% Penality of the generator violating the limits
penalty_gen3=zeros(PV_bus,1);
r=1;
for d=1:nbus
   if type(d)==2
      if Q_calc_tot(d)>Q_max_all(d)/Sbase
         penalty_gen3(r)=100000*(Q_calc_tot(d)-(Q_max_all(d)/Sbase))^2;
         r=r+1;
```

```
        elseif Q_calc_tot(d)<Q_min_all(d)/Sbase
            penalty_gen3(r)=100000*((Q_min_all(d)/Sbase)-Q_calc_tot(d))^2;
            r=r+1;
        else
            penalty_gen3(r)=0;
            r=r+1;
        end
    end
end
penalty_gen_violation3(k)=sum(penalty_gen3);

%=====================================================================%
% Checking violations of active power at Slack bus
penalty_Slack3=zeros(1,1);
r=1;
for d=1:nbus
   if type(d)==3
      if P_calc_tot(d)>Pg_max_all(d)/Sbase
         penalty_Slack3(r)=100000*(P_calc_tot(d)-(Pg_max_all(d)/Sbase))^2;
         r=r+1;
      elseif P_calc_tot(d)<Pg_min_all(d)/Sbase
          penalty_Slack3(r)=100000*((Pg_min_all(d)/Sbase)-P_calc_tot(d))^2;
          r=r+1;
      else
         penalty_Slack3(r)=0;
         r=r+1;
      end
   end
end
penalty_Slack_violation3(k)=sum(penalty_Slack3);

%=====================================================================%

 % Calling newton raphson fuction after replacing parameters
 [P_loss]=object_function(nbranch,Delta,V,g,Sbase,frombus,tobus);
 Power_loss3(k)=P_loss;

 % Evaluation of the augmented function (fitness function)
```

```
particle(k).loss=Power_loss3(k)+penalty_bus_violation3(k)+penalty_gen_violation3(k)+penalty_Slack_violation3(k);

        abandon(k)=0;

        % Initializing new personal best
        if particle(k).loss<=particle(k).best.loss
            particle(k).best.position=particle(k).position;
            particle(k).best.loss=particle(k).loss;
        end
    end
end
for m=1:num_particle
    particle1(m).position=particle(m).position;
    particle1(m).loss=particle(m).loss;
    particle1(m).velocity=particle(m).velocity;
end
    Globalbest5.loss=min([particle1(:).loss]); % The minimum value of losses
    loca= find([particle1.loss]==min([particle1(:).loss])); % Locate the position of the minimum loss
    Globalbest5.position=[particle1(loca).position];

if Globalbest5.loss<Globalbest4.loss
    Globalbest4=Globalbest5;
elseif Globalbest5.loss<Globalbest.loss
    Globalbest=Globalbest5;
end

%=========================================================================%
                % Updating the swarms %
%=========================================================================%
for t=1:num_particle
    particle(t).position=particle1(t).position;
    particle(t).velocity=particle1(t).velocity;
end
% Defining Swarms
swarm1=10;
swarm2=15;
swarm3=25;
```

```
% Identifying the first swarm
loss_swarm1=zeros(swarm1,1);
best_position_swarm1=zeros(swarm1,members);
for n=1:swarm1
   loss_swarm1(n)=particle(n).loss;
   for d=1:members
     best_position_swarm1(n,d)=best_position(n,d);
   end
end
mean_best_swarm1=sum(best_position_swarm1)/swarm1;

Globalbest1.loss=min(loss_swarm1); % The minimum value of losses
location1= find([particle.loss]==Globalbest1.loss); % Locate the position of the minimum loss
Globalbest1.position=[particle(location1).position];

% Identifying the second swarm
loss_swarm2=zeros(swarm2,1);
best_position_swarm2=zeros(swarm2,members);
r=1;
for m=swarm1+1:swarm1+swarm2
   loss_swarm2(r)=particle(m).loss;
   best_position_swarm2(r,:)=best_position(m,:);
   r=r+1;
end
mean_best_swarm2=sum(best_position_swarm2)/swarm2;

Globalbest2.loss=min(loss_swarm2); % The minimum value of losses
location2= find([particle.loss]==Globalbest2.loss); % Locate the position of the minimum loss
Globalbest2.position=[particle(location2).position];

% Identifying the third swarm
loss_swarm3=zeros(swarm3,1);
best_position_swarm3=zeros(swarm3,members);
r=1;
for t=swarm1+swarm2+1:swarm1+swarm2+swarm3
   loss_swarm3(r)=particle(t).loss;
   best_position_swarm3(r,:)=best_position(t,:);
```

```
    r=r+1;
end
mean_best_swarm3=sum(best_position_swarm3)/swarm3;
Globalbest3.loss=min(loss_swarm3); % The minimum value of losses
location3= find([particle.loss]==Globalbest3.loss); % Locate the position of the minimum loss
Globalbest3.position=[particle(location3).position];

% Storing the best loss value at each iteration
Best_loss(it)=Globalbest.loss;
disp(['Iteration', num2str(it) ': Best loss= ', num2str(Best_loss(it))]);

end

% Defining the parameters'positions in the vector of control variable
% Defining the position of voltage at generator bus in bus_dat matrix

v1=Globalbest.position(1); bus_dat(1,8)=v1;
v2=Globalbest.position(2); bus_dat(2,8)=v2;
v3=Globalbest.position(3); bus_dat(3,8)=v3;
v6=Globalbest.position(4); bus_dat(6,8)=v6;
v8=Globalbest.position(5); bus_dat(8,8)=v8;

% Defining the position of voltage set generator bus in gen_dat matrix
v1=Globalbest.position(1); gen_dat(1,6)=v1;
v2=Globalbest.position(2); gen_dat(2,6)=v2;
v3=Globalbest.position(3); gen_dat(3,6)=v3;
v6=Globalbest.position(4); gen_dat(4,6)=v6;
v8=Globalbest.position(5); gen_dat(5,6)=v8;

% Defining the position of tap ratio
% tr1(4-7), tr2(4-9), tr3(5-6)

tr1=Globalbest.position(6); line_dat(8,9)=tr1;
tr2=Globalbest.position(7); line_dat(9,9)=tr2;
tr3=Globalbest.position(8); line_dat(10,9)=tr3;

% Defining the position of the shunt capacitor
qsh9=Globalbest.position(9); bus_dat(9,6)=qsh9;
```

```
% Calling the Newton Raphson function
[P_calc_tot,Q_calc_tot,Pg_max_all,Pg_min_all,Q_max_all,Q_min_all,PV_bus,PQ_bus,V,type,nbus,...
   nbranch,Delta,g,Sbase,frombus,tobus]=...
   newton_2(bus_dat,line_dat,gen_dat);

Q_calc_tot_act=Q_calc_tot*Sbase;
for d=1:nbus
   if type(d)==2
      if Q_calc_tot_act(d)>Q_max_all(d)
         Q_calc_tot_act(d)=Q_max_all(d);
      elseif Q_calc_tot_act(d)<Q_min_all(d)
         Q_calc_tot_act(d)=Q_min_all(d);
      end
   end
end

% Computing loss
[P_loss]=object_function(nbranch,Delta,V,g,Sbase,frombus,tobus);

load bus_dat.txt;      % loading the bus data
bus_num=bus_dat(:,1);  % Bus number

%% Results of the Hybrid PSO-ABC
figure;
plot(Best_loss,'LineWidth',1);
title('Reactive power optimization based Hybrid PSO-ABC (14 Bus IEEE)');
xlabel('Iteration');
ylabel('Active power loss in MW ');
grid on;

figure;
plot(bus_num,V,'--*r');
title('Voltage at buses for IEEE 14 Bus System after Optimization by Hybrid PSO-ABC ');
xlabel('Bus number');
ylabel('Voltage in p.u')
grid on
```

```
figure;


plot(bus_num(type~=1),Q_calc_tot_act(type~=1),'--*b');
title('Reactive power(MVar) at generator bus after Optimization by Hybrid PSO-ABC');
xlabel('Bus number of generator bus');
ylabel('Reactive power(MVar) at generator bus')
grid on
```