Chapter 3

Global Optimization Techniques

This chapter presents the basics of global optimization problems, their types and techniques used to solve these problems. It also describes two nature-inspired optimization algorithms, Particle Swarm Optimization (PSO) and Butterfly Optimizer (BO) which are used in further chapters of this thesis to solve the optimization problems discussed in Chapter 1.

3.1 What is the Optimization ?

Optimization is the process deployed to obtain best possible/feasible solution for a given set of problems. The quality of solution can be represented in terms of a real value and is problem specific. Thus, the main purpose of optimization is to search the best feasible option for a given set of circumstances. In recent decades the area of optimization has matured and is widely employed in numerous engineering, financial and social applications. These applications include and are not limited to the following domains; optimal missile trajectories, optimal petroleum refining, aircrafts designs with minimum weight, profitable business activities, physical, engineering, architecture, economics, biological and chemical sciences and management. There are many algorithms or techniques available to solve optimization problems. The history of optimization goes back to many centuries; Euclid (300 B.C.) and Heron (100 B.C.) solved the problem of light trav-

eling between two points and found the shortest path between them [41]. Leibniz *et al.* [42], in their work, used optimization to solve real valued non-linear and linear constrained problems. Harold Kuhn, Tucker addressed the problem of non-linear constrained optimization in 1951. The authors used the principle of Leibniz-Newton for optimization and therefore the solution was prone to get stuck in local optima. The Monte-Carlo simulation based methods for optimization appeared around 1940 [43]. A search based method similar to evolutionary methods was introduced by George Box in 1957 [44]. The method exploited entire search space for generating random candidate solutions. John Nelder and Roger Mead were first to introduce simplex algorithm in [45]. The method incorporated the ability to learn from previous search and algorithm had adaptation for search surface terrain. Kirkpatrick *et al.* ([47]) introduce simulated annealing (SA) in their work. A breakthrough in evolutionary computation was achieved through the work of John Holland presented in [46]. In this work, genetic algorithm (GA) was first introduce. The method mimics the metallurgical process of annealing of metals. In later years, ant colony optimization (ACO) and particle swarm optimization (PSO) were introduced by Marco Dorigo in 1992 and Kennedy and Eberhart in 1995, respectively. The methods were based on foraging behavior of insects and swarms of birds. An optimization problem can be stated as,

$$\begin{aligned} Search \quad & \vec{x} = (x_1, x_2, x_3....., x_D) \\ Optimizes \quad & f_1(\vec{x}), f_2(\vec{x}), ..., f_k(\vec{x}) \\ subject \quad to: \quad & h_i(\vec{x}) = 0, i = 1, 2, ..., p \\ & g_i(\vec{x}) < 0, i = p+1, p+2, ...p+m \\ where \quad & l_i \leq x_i \leq u_i, i = 1, 2, 3....D \end{aligned} \tag{3.1}$$

Here $\vec{x}$ is a solution vector, D is the dimension of problem, k refers to the

number of objective functions needed to be simultaneously optimized. The $f_i(x)$, $\forall\ i \in [1, k]$, are the objective functions of the problem. l_i and u_i, $\forall\ i \in [1, D]$ are the lower and upper bounds defing the search space. A commonly used method

is Direct search method. This method does not require the gradient information. These methods include iterative methods such as Nelder-Mead [51], Hooke-Jeeves [50],and Heuristic techniques [52]. The Heuristic approaches for optimization widely came into use in the optimization domain quite recently. In Heuristic approach, optima is obtained by use of so-called exploitation and exploration of the search-space. The process of exploitation is search scheme which examines nearby space of a candidate solution. The exploration of search space incorporates searching of faraway spaces to the candidate solution thereby avoiding local minima. Swarm Intelligence (SI) based optimization have evolved and become popular in recent times. One of the SI-based methods, PSO is quite widely used because of simplicity in implementation.

3.1.1 Global Optimization

A multimodal objective functions can have multiple optimal solutions wherein some may be sub-optimal local solutions. Finding global optimum for a multimodal problem is typically considered to be a difficult tasks. The global optimization methods aims to find a global optimum solution in presence of several local sub-optimal solution. In general, global optimization problems are non-linear problems.

3.1.2 Types of Optimization

The global optimization can broadly be categorized into four major domains based on the number of objectives and or type of constraints involved.

1. Single Objective Unconstrained Optimization (SOUO)
2. Single Objective Constrained Optimization (SOCO)
3. Multi Objective Unconstrained Optimization (MOUO)
4. Multi Objective Constrained Optimization (MOCO)

3.1.2.1 Single Objective Unconstrained Optimization

The equation (3.1) states the generic optimization problem. In equation (3.1), if $p = 0$, $m = 0$ and $k = 1$, then this equation reduces to Single Objective Unconstrained Optimization (SOUO). This class of optimization problem involves only one objective function without any constraints. The simplified mathematical equation is given below.

$$\begin{aligned} Search \quad \vec{x} \quad &= \quad (x_1, x_2, x_3..., x_D) \\ optimizes \quad & f(\bar{x}) \\ Where \quad & l_i \leq x_i \leq u_i, i = 1, 2, ..., D \end{aligned} \tag{3.2}$$

3.1.2.2 Single Objective Constrained Optimization

Considering equation (3.1) for the values of $p > 0$ and/or $m > 0$ and $k = 1$, the resulting equation reduces to Single Objective Constrained Optimization (SOCO). Thus, the problem has one objective function with at least one constraint. The

constraints could be of equality ($p > 0$) or inequality ($m > 0$) type, or both ($m > 0$, $p > 0$). Thus the problem can be simplified as follows.

$$\begin{aligned} Search \quad \vec{x} \quad &= \quad (x_1, x_2, x_3, ..., x_D) \\ optimizes \quad & f(\bar{x}) \\ subject \quad to: \quad & h_i = 0, i = 1, 2, ..., p \\ & g_i < 0, i = p+1, p+2, ...p+m \\ where \quad & l_i \leq x_i \leq u_i, i = 1, 2, 3....D \end{aligned} \tag{3.3}$$

3.1.2.3 Multi Objective Unconstrained Optimization

Consider equation (3.1) with values of $p = 0$, $m = 0$ and $k > 1$. The resulting equation reduced to so-called multi-objective unconstrained optimization (MOUO). The resulting optimization problem has multiple objective functions with no constraints. All objective functions are to be optimized simultaneously. The mathematical of equation (3.1) can be re-written (for $p = 0$, $m = 0$ and $k > 1$) as given below.

$$\begin{aligned} Search \quad \vec{x} \quad &= \quad (x_1, x_2, x_3..., x_D) \\ optimizes \quad & f_1(\vec{x}), f_2(\vec{x}), ...f_k(\vec{x}) \\ Where \quad & l_i \leq x_i \leq u_i, i = 1, 2, ..., D \end{aligned} \tag{3.4}$$

3.1.2.4 Multi Objective Constrained Optimization

The equation (3.1), for the values of $p \geq 1$, $m \geq 1$ and $k > 1$ reduces to Multi Objective Constrained Optimization (MOCO). The optimization problem involves multi-objective function with one or several constraints. The reduced mathematical equation of problem can be written as,

$$Search \quad \vec{x} \quad = \quad (x_1, x_2, x_3, ..., x_D) \tag{3.5}$$

$$optimizes \quad f_1(\vec{x}), f_2(\vec{x}), f_3(\vec{x}), ..., f_k(\vec{x})$$

$$subject \quad to: \quad h_i(\vec{x}) = 0, i = 1, 2, ..., p$$

$$g_i(\vec{x}) < 0, i = p+1, p+2, ...p+m,$$

$$where \quad l_i \leq x_i \leq u_i, i = 1, 2, 3....D.$$

3.1.3 Optimization Problem used in this Work

As discussed in Chapter 1, this thesis deals with the problem of phase balancing in the distribution systems. Phase balancing has been attempted using two approaches, (i) load switching among phases, and (ii) switching of single-phase DGs among phases and sizing of these DGs. Load switching problem and optimal sizing and siting of distributed generation are the types of single objective unconstrained optimization problems. The objective functions of these problems are highly non-linear, discontinuous, multi-modal, and hard to solve using conventional optimization technique. Two nature-inspired optimization algorithms, PSO and BO, have been employed to solve these problems. This phase balancing has been formulated as global optimization problem and basic theory of the algorithms of these two optimization methods has been discussed in the following sections.

3.2 Particle Swarm Optimization

3.2.1 Swarm Intelligence

SI presents an innovative approach towards solving optimization problems. The methods incorporating SI-based approaches mimic naturally occurring swarm behavior such as swarming of insects, bird flocking, fish schooling, and food-foraging of bees herding. the real world problems are characterized by their multi-modality, non-linearity, discontinuities and non-availability of derivatives. The conventional

methods in several cases are not suited for finding solution for such problems show difficulty in optimizing the real world problems. The real world problems often are characterized by noisy, incomplete data or multi- modality due to their inflexible construction. Natural systems have been mimicked for decades to solve the optimization problems. These natural systems often contain many simple elements which, when work together, produce complex emergent behavior. The natural computing paradigms can be used, where conventional computing paradigm perform unsatisfactorily. Swarm Intelligence (SI) belongs to one such natural computing paradigm.

The collective behaviors of animals as swarms have been studied by biologists for animals such insects, fish , bee, bird and mammal. Grasse [53]in their work reported the swarm behavior of African termites for food foraging. The first flocking model was first developed by Craig Reynolds [54] in 1987. Craig Reynolds introduces a flock of entities called Biods in their bio-inspired computational framework for simulating the swarm behavior. Jean-Louis and Simon Goss [55] demonstrated the use of collective patterns and decision making in their work presented in 1991. They investigated the food foraging behavior of Ants and their strategy for finding shortest path between food sources. The term swarm intelligence first introduced by Beni and Wang [56] in their work related to robotics systems. first introduced the term Swarm Intelligence in the context of cellular robotic systems in 1991. In 1992 Marco Dorigo introduced Ant Colony Optimization (ACO) algorithm [48]. The method of PSO presented by Eberhart and Kennedy in [49] mimicked social behavior of flock of birds. Karboga in their work ([57]) introduce Artificial Bee Colony (ABC) algorithm which simulated the food foraging behavior of honey bees. The detailed algorithmic explanations of ABC and ACO algorithms are out of scope of the thesis.

3.2.2 Particle Swarm Optimization

The particle swarm optimization (PSO) is a parallel evolutionary computation technique developed by Kennedy and Eberhart [49] based on the social behavior

metaphor. Reynolds [54] proposed a behavioral model in which each of the particles follows three rules for finding the food [54].

1. Separation: Each particle tries to move away from its neighbors if they are too close.

2. Alignment: Each particle steers towards a general direction of neighboring particles.

3. Cohesion: Each particle tries to go towards the general position of its neighbors.

The PSO algorithm is initialized with a population of random particles (candidate solution vectors). Each particle is characterized by its position (solution vector) and velocity (direction in which it will move toward a new solution vector in the problem space). For each particle a random velocity is assigned initially. Each particle iteratively moves in the problem space. Using above stated behavioral model, the particle moves towards the position of minimum value of the objective function (or best fitness) achieved so far by the particle itself and towards the position of the best fitness achieved so far by the whole population. For the optimization problem, represented by equation (3.2), the PSO algorithm updates the velocity, v_i, and position, x_i, of i-th particles in the following manner at k^{th} iteration.

$$\vec{v}_i^{k+1} = m_v \otimes \vec{v}_i^{k} + m_{x1} \otimes \bar{r1} \otimes (\vec{p}_{1i} - \vec{x}_i^{k}) + m_{x2} \otimes \bar{r2} \otimes (\vec{p}_2 - \vec{x}_i^{k}) \tag{3.6}$$

$$\vec{x}_i^{k+1} = c_x \otimes \vec{x}_i^{k} + c_v \otimes \bar{r1} \otimes \vec{v}_i^{k+1} \tag{3.7}$$

The symbol $\otimes$ denotes element by element vector multiplication. m_v, m_{x1} , and m_{x2} are momentum values, $\bar{r1}$ and $\bar{r2}$ are the random vectors. $\vec{p}_{1i}$ is vector of local-best solution achieved by particle i, consisting of best fit achieved so far (up to the k^{th} iteration) by the particular particle $\vec{x}_i^{k}$ and $\vec{p}_2$ is called the global best solution. It is a solution vector having best fit achieved so far by particles in the

swarm (up to the k-th iteration). The new solution vector $\vec{x}^{k+1}$ is obtained by linearly combining the last position $\vec{x}^{k}$ and velocity $\vec{v}^{k+1}$.

3.2.3 PSO terminology

The following terminologies are used to describe PSO.

1. **Particle:** A candidate solution is referred as a particle which is analogies to a bird in a flock. All the particles in a flock are independent to each other.

2. **Position:** Every particles is defined by its position in the search space. The position is represented by coordinates of the particle in the N-dimensional search space. An N-dimensional search space is the solution space for the optimization problem, coordinates of which represents a solution to the problem.

3. **pBest location:** This is the information of best location that the particle has found during search process which is reffered as pBest. Each particle has its own local-best location determined by the path that it has flown. At each point along its path, the particle compares the fitness value of its current location to that of pBest location. If the current location has a better fitness, pBest solution is replaced with its current location.

4. **gBest location:** This is the information of best position in the whole swarm. For entire swarm there is one gBest location to which each particle is attracted. At each point along their path every particle compares the fitness of their current location to that of gBest location. If any particle is at a location of better fitness than gBest location is replaced by that particle's current position.

5. **Cognitive factor (m_{x1}):** It is a relative pull on the velocity of the particle by the pBest location of the same particle. This is the major factor that alters the trajectory of the particle flight during due course of search.

6. **Social factor (m_{x2}):** It is a relative pull on the velocity of the particles by the gBest location. This is also the major factor to alters the trajectory of the particle flight.

3.2.4 A Newtonian Mechanical Model

The Newtonian mechanical model explains how iterative (velocity and position) equations of Particle Swarm Optimization are deduced from mechanics. The position and velocity of each particle with mass m can be expressed according to the Newtons second law of motion as

$$\frac{d\vec{x}}{dt} = \vec{v}, \tag{3.8}$$

$$\frac{d\vec{v}}{dt} = \vec{a} = \frac{\vec{F}}{m}, \tag{3.9}$$

where, $\vec{F}$ denotes the force on the particles, $\vec{a}$ and $\vec{v}$ are the acceleration and velocity, respectively. The forward differentiation of equations (3.8) and (3.9) converts it to iterative process as follows.

$$\frac{\vec{x}_t - \vec{x}_{t-1}}{\triangle t} = \vec{v} \Longrightarrow \vec{x}_t = \vec{x}_{t-1} + \vec{v}_t \triangle t, \tag{3.10}$$

$$\frac{\vec{v}_t - \vec{v}_{t-1}}{\triangle t} = \frac{\vec{F}_{t-1}}{m} \Longrightarrow \vec{v}_t = \vec{v}_{t-1} + \frac{\vec{F}_{t-1}}{m}_t \triangle t. \tag{3.11}$$

Considering m and $\triangle t$ to be unity in above equations, (3.10) and (3.11), we get the following.

$$\vec{x}_t = \vec{x}_{t-1} + \vec{v}_t, \tag{3.12}$$

$$\vec{v}_t = \vec{v}_{t-1} + \vec{F}_{t-1}. \tag{3.13}$$

All the terms in equations (3.10) and (3.11) are known except force $\vec{F}_{t-1}$ at $(t-1)^{th}$ iteration. To model this $(\vec{F}_{t-1})$, the swarm characteristics are considered. The important feature of swarm is that any particle is influenced by the relative pull of local-best and global-best locations. These two relative pulls on swarm are equivalent to two spring like attractions and hence can be expressed as

$$\vec{F}_{t-1} = m_{x1} \otimes (\vec{p_1} - \vec{x}^k) + m_{x2} \otimes (\vec{p_2} - \vec{x}^k) \tag{3.14}$$

where m_{x1} and m_{x2} are Hooks constants of two springs. The m_{x1} and m_{x2} are considered as cognitive and social factors respectively. The behavior of the entire swarm can be expressed by writing the above equations in a matrix form. To inject the randomness into the swarm behavior, Hookes constants m_{x1} and m_{x2} are multiplied by random number generators. Thus the final equation can be expressed as below.

$$\vec{v}_i^{k+1} = m_v \otimes \vec{v}_i^k + m_{x1} \otimes \bar{r1} \otimes (\vec{p_{1i}} - \vec{x}_i^k) + m_{x2} \otimes \bar{r2} \otimes (\vec{p_2} - \vec{x}_i^k) \tag{3.15}$$

$$\vec{x}_i^{k+1} = c_x \otimes \vec{x}_i^k + c_v \otimes \bar{r1} \otimes \vec{v}_i^{k+1} \tag{3.16}$$

The equations (3.15) and (3.16) are the position and velocity update equations for PSO. The main steps of PSO are shown in figure 3.1.

3.3 Butterfly Optimizer

BO is a dual population based technique for unconstrained optimization [58]. Brief discussion of the technique is given as follows.

3.3.1 Dual population of BO

BO is based on dual population of positions of male butterflies. Perching and patrolling operations of BO correspond to exploration and exploitation of search space respectively, to look for new solution. For $D-$ dimensional problem, with N butterflies, population P_1 represents current perching positions and, P_2 consists of best perching positions of every male butterfly. P_1 and P_2 are represented as

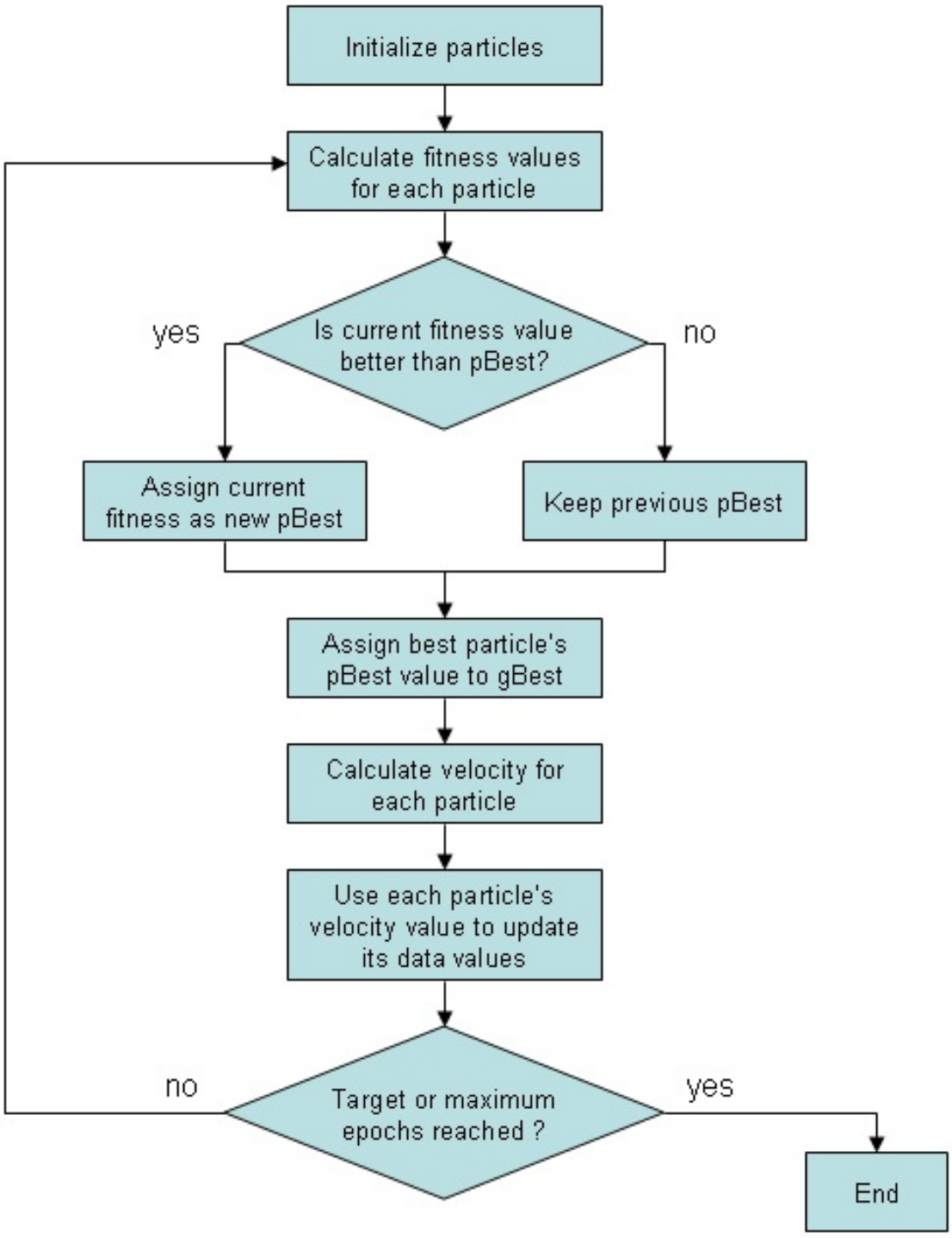

Figure 3.1: Flow chart of Particle Swarm Optimization

$N \times D$ matrix as follows.

$$P_1^k = [\bar{x}_1^k, \bar{x}_2^k ....\bar{x}_N^k] \tag{3.17}$$

where

$$\bar{x}_i^k = [x_{i1}^k, x_{i2}^k, x_{i3}^k....x_{iD}^k]^T, i = 1, 2, ...N \tag{3.18}$$

and

$$P_2^k = [\bar{mx}_1^k, \bar{mx}_2^k....\bar{mx}_N^k] \tag{3.19}$$

where

$$\bar{mx}_i^k = [mx_{i1}^k, mx_{i2}^k, mx_{i3}^k....mx_{iD}^k]^T, i = 1, 2, ...N \tag{3.20}$$

The vector $\bar{x}_i^k$ and $\bar{mx}_i^k$, and thereby P_1 and P_2 are updated using BO algorithm to get newer solutions.

A velocity vector, mv_i is associated with each solution i, which is represented by following equation.

$$\bar{mv}_i^k = [mv_{i1}^k, mv_{i2}^k, mv_{i3}^k.....mx_{iD}^k]^T, i = 1, 2, ...N \tag{3.21}$$

3.3.2 Initialization

Solution process is intialised in search space using equation (3.22).

$$x_{ij}^0 = (U_j - L_j) * rand + L_j \tag{3.22}$$

$$mx_{ij}^0 = (U_j - L_j) * rand + L_j$$

$$mv_{ij}^0 = 0$$

For j^{th} parameter, L_j and U_j are lower and upper bound and $rand$ is uniformly distributed random number between $(0, 1]$

3.3.3 Perching

Perching operation consists of (i) *Crisscross modification* and (ii) I^{st}*-selection.*

Crisscross Modification

Crisscross modification updates the perching position vector $\bar{x}_i^{k+1}$, by modifying one of its randomly selected elements, in the following manner.

$$x_{ij}^{k+1} = \begin{cases} \mathfrak{R}(x_{cc_ij}^k, mx_{cc_ij}^k) + F * (& \\ \mathfrak{R}(x_{q_ij}^k, mx_{q_ij}^k) - \mathfrak{R}(x_{r_ij}^k, mx_{r_ij}^k)), & \text{if } j == d \\ mx_{ij}^k, & \text{otherwise.} \end{cases} \tag{3.23}$$

$\Re(*,*)$, is a random operator, which can pick one of the arguments with equal probability. Crisscross neighbour of i^{th} butterfly is cc_i. Other randomly selected neighbors of i^{th} butterfly, q_i and r_i satisfy the following criteria.

$$i \neq cc_i \neq q_i \neq r_i \tag{3.24}$$

At the beginning of an iteration, $\bar{cc}$ of length N is initialized by randomly shuffling integers from 1 to N.

$$\bar{cc} = [cc_1, cc_2, ...cc_i...cc_N]^T \tag{3.25}$$

Ist-Selection

Ist-selection is given by equation (3.26) which updates $m\bar{i}x_i^k$ of P_2 as follows.

$$m\bar{i}x_i^{k+1} = \begin{cases} \bar{x_i}^{k+1}, & \text{if } f(\bar{x_i}^{k+1}) \leq f(m\bar{i}x_i{}^k) \\ m\bar{i}x_i{}^k, & \text{otherwise.} \end{cases} \tag{3.26}$$

where $f(*)$ is the objective function value at $*$ position.

3.3.4 Patrolling

Male butterflies which remain *un*-updated during perching operation are updated in patrolling operation. Patrolling operation consists of following two sub-operations.

Towards-best modification

This step gives patrolling position vector, $u\bar{i}x_i$

$$u\bar{i}x_i = m\bar{i}x_i^k + s * m\bar{i}v_i^k + F * (m\bar{i}x_{best}^k - m\bar{i}x_i^k) \tag{3.27}$$

where $m\bar{i}x_{best}$ is the best position in the population, F is a random value between $(0, 1]$ and s is a constant between $[0, 1]$.

IInd-Selection

IInd-Selection is similar to Ist-selection; the only difference is that it also updates velocity vector $m\bar{i}v_i^k$. Equation (3.28) updates velocity vector as follows,

$$m\bar{i}v_i^{k+1} = \begin{cases} u\bar{i}x_i^{k+1} - m\bar{i}x_i^k, & \text{if} f(u\bar{i}x_i{}^{k+1}) \leq f(m\bar{i}x_i{}^k) \\ d * m\bar{i}v_i^k + F* & \\ (m\bar{i}x_{best}^k - m\bar{i}x_i^k), & \text{otherwise.} \end{cases} \tag{3.28}$$

BO algorithm is terminated if specified maximum number of objective function evaluation is reached or solution does not change over a specified number of consecutive iterations.

Algorithms 1 and 3.27 show the main steps of the BO algorithm.

3.4 Performance Analysis of PSO and BO

In this section, BO and PSO are benchmarked on 55 benchmark problems to evaluate their performance. All benchmark problems are taken from [2] paper. These benchmark problems are grouped into four categories according to their property as listed in Tables 3.1-3.4. D indicates the dimension of problem, R is an array representing the upper and lower boundary of variables in search space and $Fmin$ is the global minimum of the problem.
All the benchmark problems are minimization type and fall in any of the following 4 groups, (i) Floating Dimension Unimodal, (ii) Fixed Dimension Unimodal, (iii) Floating Dimension Multimodal, and (iv) Fixed Dimension Multimodal. BO runs 30 times on each benchmark function to reduce the randomness in the error.

In this analysis, following three tests are performed on the all problems (i) Exploitation behavior analysis, (ii) Exploring behavior analysis and (iii) Local optimum avoidance analysis. Results of this experiment are reported in Table 3.5 - 3.8.

3.4.1 Exploitation Behavior Analysis

Benchmark Problems of Table 3.1 and Table 3.2 are suitable for exploitation behavior analysis. These benchmark functions contain only one optimum in search space called global optimum. In this test, all the problem of these table are solved using PSO and BO and the results are reported in Table 3.5-3.6. These tables show that the BO is able to solve all of these problem very well. PSO also could solve all the problems except $F3$, $F4$, $F11$, $F12$, $F14$, and $F16$. These results also show that the performance of BO is far better in terms of exploiting the best solution in search space. Perching behavior of butterfly is the main reason of better

Algorithm 1 Perching

procedure PERCHING()
  $cc_i \leftarrow \text{RandP}(i)$
  $q_i \leftarrow \text{randi}(1, N)$
  while $(p_i == q_i) or (i == q_i)$ **do**
    $q_i \leftarrow \text{randi}(1, N)$
  end while
  $r_i \leftarrow \text{randi}(1, N)$
  while $(p_i == r_i) or (q_i == r_i) or (i == r_i)$ **do**
    $r_i \leftarrow \text{randi}(1, N)$
  end while
  $m_i \leftarrow \text{randi}(1, D)$
  $\text{s} \leftarrow \text{rand}(0, 1)$
  for $j = 1$ to D **do**
    if $j == m$ **then**
      $x_{ij} = \mathfrak{R}(x_{cc_ij}, mx_{cc_ij}) + F \times (\mathfrak{R}(x_{q_ij}, mx_{q_ij}) - \mathfrak{R}(x_{r_ij}, mx_{r_ij}))$
    else
      $x_{ij} = mx_{ij}$
    end if
  end for
  $f(\bar{x}_i) \leftarrow$ function evaluation at $m\bar{x}_i$
  $FEs \leftarrow FEs + 1$
  if $f(\bar{x}_i) \leq f(m\bar{x}_i$ **then**
    $m\bar{x}_i \leftarrow \bar{x}_i$
    $f(m\bar{x}_i) \leftarrow \bar{x}_i$
  end if
end procedure

Algorithm 2 Patrolling

procedure PATROLLING()
  for $j = 1$ to D **do**
    $t \leftarrow$ uniformly distributed number from -1 to 1
    $ux_{ij} \leftarrow x_{ij} + d * v_{ij} + (x_{best} - x_{ij}) * t$
  end for
  $f(\bar{ux}_i) \leftarrow$ function evaluation at $\bar{z}_i$
  $FEs \leftarrow FEs + 1$
  if $f(\bar{ux}_i) \leq f(\bar{x}_i)$ **then**
    $\bar{mv}_i \leftarrow \bar{ux}_i - \bar{mx}_i$
    $\bar{mx}_i \leftarrow \bar{ux}_i$
    $f(\bar{mx}_i) \leftarrow f(\bar{ux}_i)$
  else
    $\bar{mv}_i \leftarrow d * \bar{mv}_i$
  end if
end procedure

Table 3.1: Fixed Dimension Unimodal Functions [2]. (D = Dimension, R = Range of the search-space, $Fmin$ = minimum of the problem)

S.N	Function's Name	D	R	Fmin
F1.	Aluffi-Pentini's Function	2	[-10,10]	-0.3523
F2.	Beale Function	2	[-4.5,4.5]	0
F3.	Colville Function	4	[-10,10]	0
F4.	Easom Function	2	[-100,100]	-1
F5.	Matyas Function	2	[-10,10]	0
F6.	Quadratic Function	2	[-10,10]	-3873.7243
F7.	Three Hump Camel Function	2	[-5,5]	0
F8.	Wolfe Function	3	[0,2]	0
F9.	Zettle Function	2	[-5,10]	-0.003791
F10.	Leon Function	2	[-1.2,1.2]	0

Table 3.2: Floating Dimension Unimodal Functions [2]. (D = Dimension, R = Range of the search-space, $Fmin$ = minimum of the problem)

S.N	**Function's Name**	**D**	**R**	**Fmin**
F11.	Dixon and Price Function	30	[-10,10]	0
F12.	Rosenbrock Function	30	[-30,30]	0
F13.	Schwefel 1.2 Function	30	[-100,100]	0
F14.	Schwefel 2.22 Function	30	[-10,10]	0
F15.	Step Function	30	[-100,100]	0
F16.	Stepint Function	30	[-5.12,5.12]	0
F17.	Sphere Function	30	[-5.12,5.12]	0
F18.	Sumsquare Function	30	[-10,10]	0
F19.	Trid Function	30	[-100,100]	-4930
F20.	Zakharov Function	30	[-5,5]	0

exploiting performance.

3.4.2 Exploration Behavior Analysis

Benchmark functions of Table 3.3 and 3.4 are having multiple minimum in search space. So, these functions are suitable for exploration analysis. In this test, all the problems of these tables are solved by PSO and BO and the results are reported in Table 3.7-3.8. The results in the tables show that the BO is able to solve all of these problems very well except F37 and F54. PSO also could solve all the problems satisfactory except a few such as F37, F40, F46, F47, F48, F51, F53, and F55. BO algorithm provide better result on the majority of multimodal functions as compared to PSO. Therefore, BO algorithm has very competitive exploring behavior. Patrolling behavior of butterfly is the main reason of better exploring performance.

Table 3.3: Fixed Dimension Multimodal Functions [2]. (D = Dimension, R = Range of the search-space, $Fmin$ = minimum of the problem)

S.N	**Function's Name**	**D**	**R**	**Fmin**
F21.	Beckar-Lago Function	2	[-10,10]	0
F22.	Bohachevsky 1 Function	2	[-100,100]	0
F23.	Bohachevsky 2 Function	2	[-100,100]	0
F24.	Bohachevsky 3 Function	2	[-100,100]	0
F25.	Booth Function	2	[-10,10]	0
F26.	Branin Function	2	[-5,15]	0.398
F27.	Butterfly Function	2	[-2,2]	-1
F28.	Carrom Table Function	2	[-10,10]	-24.157
F29.	Chichinadze Function	2	[-30,30]	-42.315
F30.	Egg Crate Function	2	[-5,5]	0
F31.	Freudenstein Roth Function	2	[-10,10]	0
F32.	Giunta Function	2	[-1,1]	0.060447
F33.	Goldstein-Price Function	2	[-2,2]	3
F34.	Hartman 3 Function	3	[0,1]	-3.863
F35.	Hartman 6 Function	6	[0,1]	-3.322
F36.	Hosaki Function	2	[-10,10]	-2.3458
F37.	Kowalik Function	4	[-5,5]	0.00030784
F38.	Penholder Function	2	[-11,11]	-0.96354
F39.	Periodic Function	2	[-10,10]	0.9
F40.	Shekel 5 Function	4	[0,10]	-10.15
F41.	Shekel 7 Function	4	[0,10]	-10.40
F42.	Shekel 10 Function	4	[0,10]	-10.53
F43.	Six Hump Camel Function	2	[-5,5]	-1.0316
F44.	Testtube Holder Function	2	[-10,10]	-10.8723
F45.	Trecanni Function	2	[-5,5]	0

Table 3.4: Floating Dimension Multimodal Functions [2]. (D = Dimension, R = Range of the search-space, $Fmin$ = minimum of the problem)

S.N	**Function's Name**	**D**	**R**	**Fmin**
F46.	Ackley Function	30	[-35,35]	0
F47.	Alpine Function	30	[-10,10]	0
F48.	Cosine Function	30	[-1,1]	3
F49.	Csendas Function	30	[-1,1]	0
F50.	Exponential Function	30	[-1,1]	-1
F51.	Griewank Function	30	[-100,100]	0
F52.	Quintic Function	30	[-10,10]	0
F53.	Schwefel Function	30	[-500,500]	-12569.48
F54.	Trigonometric Function	30	[0,pi]	0
F55.	Wavy's Function	30	[-100,100]	0

Table 3.5: Result of Fixed Unimodal Functions

S.N	**Min**	**BO**	**PSO**
F1	-3.5239E-01	-3.5239E-01	-3.5239E-01
F2	0.0000E+00	0.0000E+00	0.0000E+00
F3	0.0000E+00	0.0000E+00	4.2771E-02
F4	-1.0000E+00	-1.0000E+00	-9.8882E-01
F5	0.0000E+00	0.0000E+00	0.0000E+00
F6	-3.8737E+03	-3.8737E+03	-3.8737E+03
F7	0.0000E+00	0.0000E+00	0.0000E+00
F8	0.0000E+00	0.0000E+00	0.0000E+00
F9	-3.7912E-03	-3.7912E-03	-3.7912E-03
F10	0.0000E+00	0.0000E+00	0.0000E+00

Table 3.6: Result of Unfixed Unimodal Functions

S.N	**Min**	**BO**	**PSO**
F11	0.0000E+00	0.0000E+00	6.6667E-01
F12	0.0000E+00	0.0000E+00	9.3021E-01
F13	0.0000E+00	0.0000E+00	0.0000E+00
F14	0.0000E+00	0.0000E+00	4.7446E-02
F15	0.0000E+00	0.0000E+00	0.0000E+00
F16	0.0000E+00	0.0000E+00	3.7000E+00
F17	0.0000E+00	0.0000E+00	0.0000E+00
F18	0.0000E+00	0.0000E+00	0.0000E+00
F19	-4.9300E+03	-4.9300E+03	-4.9300E+03
F20	0.0000E+00	0.0000E+00	0.0000E+00

3.4.3 Local Optimum Avoidance

In case of multimodal problems, algorithms some time get stuck on local optimum of problems. This must be avoided by the algorithm while searching of global optimum. The result of some problem of Table 3.7-3.8 also provide a comparison of local optimum avoidance property of PSO and BO. These results show that BO avoids local minimum very well in all of the multimodal benchmark function as compared to PSO. Other meta-heuristics get stuck on local optima of highly modal problems, whereas BO easily avoid these local optimum even in high dimensional multimodal problem. Therefore, BO algorithm is able to avoid local optimum of the complex, high modal and high dimensional mulitimodal functions. Previous velocity term of velocity update equation is the key factor of local optimum avoidance of BO algorithm.

The results presented in this chapter show the performance of PSO and BO on well known problems, but as we are not sure about the nature of the resulting objective function corresponding to the phase balancing problem, PSO (widely accepted in the literature) and recently proposed BO (which outperforms on the conventional benchmark problems) both have been attempted to see their perfor-

Table 3.7: Result of Fixed Multimodal Functions

S.N	MIN	BO	PSO
F21	0	0.000000e+00	0.000000e+00
F22	0	0.000000e+00	0.000000e+00
F23	0	0.000000e+00	0.000000e+00
F24	0	0.000000e+00	0.000000e+00
F25	0	0.000000e+00	0.000000e+00
F26	0.397887	3.978874e-01	3.978874e-01
F27	-1	-1.000000e+00	-1.000000e+00
F28	-24.156815	-2.415682e+01	-2.415682e+01
F29	-42.944387	-4.294439e+01	-4.294439e+01
F30	0	0.000000e+00	0.000000e+00
F31	0	0.000000e+00	0.000000e+00
F32	0.064470	6.447042e-02	6.447042e-02
F33	3	3.000000e+00	3.000000e+00
F34	-3.9273827	-3.927383e+00	-3.927383e+00
F35	-3.306605	-3.306605e+00	-3.292826e+00
F36	-2.345812	-2.345812e+00	-2.345812e+00
F37	0.0003075	3.176639e-04	3.706802e-04
F38	-0.963534	-9.635348e-01	-9.635348e-01
F39	0.9	9.000000e-01	9.000000e-01
F40	-10.1532	-1.015320e+01	-9.565726e+00
F41	-10.4029	-1.040294e+01	-1.040294e+01
F42	-10.5364	-1.053641e+01	-1.053641e+01
F43	-1.031628	-1.031628e+00	-1.031628e+00
F44	-10.8723	-1.087230e+01	-1.087164e+01
F45	0	0.000000e+00	0.000000e+00

mance on phase balancing problems.

Table 3.8: Result of Unfixed Multimodal Functions

S.N	**Min**	**BO**	**PSO**
F46	0.0000E+00	1.0629E-14	5.0058E-02
F47	0.0000E+00	0.0000E+00	8.1955E-03
F48	3.0000E+00	3.0000E+00	3.3216E+00
F49	0.0000E+00	0.0000E+00	0.0000E+00
F50	-1.0000E+00	-1.0000E+00	-1.0000E+00
F51	0.0000E+00	0.0000E+00	4.5971E-03
F52	0.0000E+00	0.0000E+00	8.7886E+00
F53	-1.2569E+04	-1.2569E+04	-8.3947E+03
F54	0.0000E+00	1.8810E-06	0.0000E+00
F55	0.0000E+00	0.0000E+00	3.1799E+01

3.5 Conclusion

In this chapter, the basics of global optimization with their types and techniques used to solve global optimization problems are briefly described. Furthermore, two nature-inspired optimization algorithms, PSO and BO, are described which are used in this thesis to solve the mix-integer global optimization problems, phase balancing problem of an unbalanced distribution system using load switching and phase balancing problem of an unbalanced distribution system using optimal sizing and siting of single-phase distributed generation. A comparative analysis between PSO and BO has also been done on the conventional benchmark problems. Outcomes of the comparative analysis show that both algorithms perform well in all the benchmark problems, however, BO outperforms the PSO some of the benchmark problems. From the outcomes of the PSO and BO on benchmark problems, it can be concluded that these algorithms can be applied to the real world complex global optimization problems such as phase balancing problem being studied in the following chapters.