CHAPTER – 2
PRELIMINARY CONCEPTS

2.0 INTRODUCTION
This chapter deals with the preliminary concepts on High Density Cluster, Genetic Algorithm, Particle Swarm Optimisation and Interior point method which are fundamental methods to carry out this work.

High Density Cluster formation [7] is useful to group all the data points (solution points) generated with the help of Genetic Algorithm. The High Density Cluster forms the refined data base from which the ultimate optimal power flow solution is obtained.

Genetic Algorithm (GA) [5, 8] has its importance in generation of binary strings representing the data points or solutions around the sub-optimal solutions. For this the GA carries out the operations namely formation of chromosome, parent selection, generation of population, cross over process and mutation process. From this population, GA evaluates the fitness of each string (solution) instead of optimisation of the solution itself. In this way GA explores the search space to find the best optimal solution (where the probability of finding improved performance is high).

Particle Swarm Optimisation (PSO) is one of the modern heuristic algorithms to solve large scale non convex optimisation problem like optimum power flow. PSO is also an optimisation method [6] based on population and it can be used to solve many complex optimisation problems which are non linear, non
differentiable and multi model. The prominent merit of PSO is, it has a fast convergence speed. In addition PSO algorithm can be realized simply for less parameter need adjusting.

Interior point (IP) method [2, 5, 6] is one of the most efficient algorithms, as evident from its salient features like maintaining good accuracy, speed of convergence as much as 12:1, a general starting point (rather than good starting point as in other popular conventional methods).

In this work the basic solution for OPF has been obtained by the IP / PSO method and is used as sub-optimal solution.

2.1 HIGH DENSITY CLUSTER

The details about High Density Cluster [13] are as given below.

2.1.1 Basics on Clusters

Clustering is considered to be an important unsupervised learning problem as every other problem of this kind, it deals with, finds the structure in a collection of unlabelled data. A loose definition of clustering could be “the process of organizing objects in to groups whose members are similar in some way”. A cluster is therefore a collection of objects which are “similar” between them and are “dissimilar” to the objects belonging to other clusters. We can show this with a graphical example as in Fig. 2.1.
In this case we can easily identify the 4 clusters into which the data can be divided; the similarity criterion is distance; two or more objects belong to the same cluster if they are “close” according to a given distance (in this case geometrical distance). This is called distance based clustering.

Another kind of clustering is conceptual clustering: two or more objects belong to the same cluster if this one defines a concept common to all these objects. In other words, objects are grouped according to their fitness to the description concepts, but not according to simple similarity measures.

2.1.2 Distance Measure

An important component of a clustering algorithm is the distance measure between data points. If the components of the data instance vectors are all in the same physical units then it is possible that the simple Euclidian distance metric is sufficient to successfully group similar data instances. However the case of Euclidian distance
can sometimes be misleading. Fig. 2.2 shown below illustrates this with an example of the width and height measurements of an object. Despite both measurements being taken in the same physical units, an inferred decision has to be made with respect to the relative scaling. As the figure shows, different scaling can lead to different clustering.

![Figure: 2.2 Different clusters with scalings](image)

It can be noted that this is not only a graphic issue but the problem arises from the mathematical formula used to combine the distances between the single components of the data feature vectors into a unique distance measure that can be used for clustering purposes; different formulae lead to different clustering. Again domain knowledge must be used to guide the formulation of a suitable distance measure for each parameter application.

**2.1.3 Cluster Analysis**

Cluster Analysis divides data into groups or *Clusters* that are meaningful and useful. The analysis is sometimes useful as it provides a starting point for other purposes, such as data summarization. The
clusters are classified into Well-Separated, Proto-type, Graph-based, Conceptual type and Density based. A *Density based* is a cluster with dense region of objects that is surrounded by a region of data points of similar kind or of with low density difference. In the Fig 2.3, a high density cluster A is shown separated from the low density clusters B and C. In the context of high density clusters, the following definitions are important:

1) *Core Points*: These points are in the interior of a density based cluster. A point is a core point if the number of points within a given neighborhood around the point are as determined by the distance parameter Eps.

2) *Border Points*: A border point is not a core point, but falls within the neighborhood of a core point. In Fig. 2.3. Cluster B is formed with these points.

3) *Noise Points*: A noise point is any point that is neither a core nor a border point. In Fig. 2.3, the Cluster C is formed with these points.

Figure: 2.3 Density-based clusters: Clusters are regions of high density separated from by the regions of low density.
2.2 GENETIC ALGORITHM

The detailed information regarding Genetic Algorithm [8, 14] is as given below.

Genetic Algorithm (GA) is considered as a global optimization technique due to its flexibility and efficiency for many optimisation applications. GA is a stochastic searching algorithm. It combines an artificial survival of the fittest with genetic operators abstracted from nature to form a robust mechanism for finding optimal solutions to a variety of optimisation problems. One of the advantages of GA is using stochastic operation instead of deterministic rule to search a solution. GA hops randomly from point to point, thus allowing it to escape from the trap of local optimum in which other algorithms might land up. This results in reaching global optimum of the problem with high probability. Another important property of GA is that it searches for many optimum points in parallel.

Like many other optimisation algorithms, GA begins with defining the objective function, optimisation variables and the constraints. In the same manner; GA ends by testing for convergence. However this algorithm is quite different.

GA differs from conventional optimisation and search procedures (applicable algorithms) as mentioned below.

1. GAs work with a coding of solution sets, not the solutions themselves.

2. GAs search from a population of solutions instead of a single solution.
3. GAs use objective function information instead of derivatives or other auxiliary knowledge.

4. GAs use probabilistic transition rules, not deterministic rules.

5. GAs have potential to find solutions in many different areas of the search space simultaneously.

GA has its importance in generation of binary strings representing the solutions. Towards this GA carries out the genetic operations namely formation of strings /chromosomes, parent selection, reproduction, crossover and mutation, as shown in Fig 2.4.

**Figure: 2.4. Block diagram of G A Components**

GA evaluates the fitness of each string (solution), instead of optimisation of the solution. In this way, GA explores the search space to find the best optimal solution, where the probability of finding improved solution is high.
The general method of coding in GA is binary. Each control variable is encoded in to a series of binary bits and each bit simulates a gene. The series of binary bits of all control variables compose a string, which in turn simulates a chromosome. GA is a population search method. A population of strings is kept in each generation. The next generation is produced by the simulation of natural processes of Reproduction, Cross over and Mutation.

The GA belongs to the family of random search algorithms. The information of current population is effectively used to direct the next search. Various procedural steps in GA are briefly given below.

2.2.1 Initialization

The initial population of strings $S_i$, $i=1, 2,.., m$, where $m$ is the population size, is randomly selected in the binary coded domain of control variables. Each $S_i$ will be decoded back to the control variables to compute its fitness function $f_i$.

2.2.2 Reproduction

Reproduction is an operation, where by the strings/chromosomes of the same number as the population size will be copied to the ‘mating pool’, according to their fitness value. The higher the fitness value is (i.e., with better value of objective function) the more number of copies, the string would have in the ‘mating pool’. Copying strings according to their fitness value means those chromosomes with a higher value have a higher probability of contributing one or more offspring in the next generation.
2.2.3 Crossover

The strings in the mating pool are grouped in couples. Each couple of strings swap their bits according to the crossover probability. In this study a new crossover technique known as ‘uniform crossover’ is used, which gives bit in the string the same chance to undergo crossover and hence is better than one point crossover and two point crossover.

The crossover operation happens if a number for crossover, also randomly selected from a uniformly distributed [0, 1] domain, is less than the given crossover probability. A mask string of the same length as one of the couples strings is setup, with randomly selected bits as shown in Fig 2.5. The bits of the couple string corresponding to the bits of the mask string will swap with each other, while others remain unchanged.

Parent A: 1 0 1 0 0 1 0
Parent B: 0 1 1 1 0 0 1
A random mask: 0 1 0 1 1 0 0
Child A: 1 1 1 1 0 1 0
Child B: 0 0 1 0 0 0 1

Figure: 2.5 Uniform crossover

2.2.4 Mutation

The third operator in G.A is mutation. Reproduction and crossover effectively search and recombine existing chromosomes, but
they do not create any new genetic material in the population. Mutation is capable of overcoming this shortcoming. Every string in the ‘mating pool’ may be mutated with a given mutation probability. For each bit in the string, undergoing the mutation, a number will be randomly selected from a uniformly distributed \([0, 1]\) domain. If the number is less than or equal to the mutation probability, this bit will be changed from 1 to 0 or vice versa. If the number is larger than the mutation probability, this bit will retain its original value. The mutation operator produces a new string as shown in Fig 2.6. This provides a background variation and also occasionally introduces beneficial materials into the population.

Child A : 1 1 1 1 0 1 0

New Child A : 1 1 0 1 0 1 0

Figure: 2.6. The binary mutation

When all the strings finish crossover and mutation, a new generation is reproduced. Each string is decoded back to the control variable to compute its fitness value. The statistics will compute the new maximum fitness, minimum fitness, sum of fitnesses and average fitness. At the end the convergence condition is checked the program will stop if the condition is met, otherwise a new cycle of reproduction, crossover and mutation is started.
2.2.5 Salient Features

**GAs** differs from conventional optimisation and search procedures as mentioned below.

1. GAs work with a coding of solution sets, not the solutions themselves.
2. GAs search from a population of solutions instead of a single solution.
3. GAs use objective function information instead of derivatives or other auxiliary knowledge.
4. GAs use probabilistic transition rules, not deterministic rules.
5. GAs have potential to find solutions in many different areas of the search space simultaneously.

2.3 **PARTICLE SWARM OPTIMISATION**

The details about Particle Swarm Optimisation (PSO) are given below.

2.3.1 Overview

A new evolutionary computation technique, namely PSO [15 – 17], is described below. This technique is a combination of social psychology, observed in socio cognition human beings and evolutionary computations. PSO is inspired by the behaviour of organisms such as fish schooling and bird flocking. PSO is simple in concept, easy to implement and computationally efficient. When compared to other heuristic techniques, PSO has a flexible and well-
balanced mechanism to enhance and adapt to the global and local exploration abilities.

Like GA, PSO is also an optimisation method based on population. It conducts a search of a population of particles corresponding to individuals. Each particle is represented as a solution to the problem under study. In the context of social psychology it is observed that PSO system combines a social only model and a cognition only model. The social only indicates model that the individuals adjust their behaviour according to the successful beliefs in the neighborhood, but the cognition only treats individuals as isolated beings. However a particle changes its position using these models.

In PSO, a number of particles form a swarm that evolves or flies throughout the problem hyperspace to search for optimal or near-optimal solution. The coordinates of each particle represent a possible solution, with two vectors associated with it, the position $\mathbf{X}$ and velocity $\mathbf{V}$ vectors. To optimise the search experience, the particles interact with each others in a certain way, during their search. A particle moves towards the optimum based on its present velocity, its previous experience and the experience of others. There are different variants of the particle paradigms, but the most effective one is the $P_{gb}$ model (explained later), where in, the entire population is treated as a single neighborhood throughout the optimisation process. In each iteration, the particle with the best solution shares its position coordinates ($P_{gb}$) information with the rest of the swarm.
2.3.2 PSO Methodology

Description of basic elements required to understand the development of PSO technique is given below.

2.3.2.1 Description of basic elements in PSO

- **Particle, \( X(t) \):** It is a candidate solution represented by an \( m \)-dimensional vector, where \( m \) is the number of optimised parameters. At time \( t \), the \( j \)th particle \( X_j(t) \) can be described as
  \[
  X_j(t) = [x_j,1(t), \ldots, x_j,m(t)],
  \]
  where \( x_k \) are the optimised parameters and \( x_j,k(t) \) is the position of the \( j \)th particle with respect to the \( k \)th dimension, i.e. the value of the \( k \)th optimised parameter in the \( j \)th candidate solution.

- **Population, \( \text{pop}(t) \):** It is a set of \( n \) particles at time \( t \), i.e. \( \text{pop}(t) = [X_1(t), \ldots, X_n(t)]^T \).

- **Swarm:** It is an apparently disorganized population of moving particles that tend to cluster together, while each particle seems to be moving in a random direction.

- **Particle velocity, \( V(t) \):** It is the velocity of the moving particles represented by an \( m \) dimensional vector. At time \( t \), the \( j \)th particle velocity \( V_j(t) \) can be described as
  \[
  V_j(t) = [v_j,1(t), \ldots, v_j,m(t)],
  \]
  where \( v_j,k(t) \) is the velocity component of the \( j \)th particle with respect to the \( k \)th dimension.

- **Inertia weight, \( w(t) \):** It is a control parameter to control the impact of the previous velocities on the present velocity. Thus it influences the trade off, between the global and local exploration abilities of the particles, large inertia weight to enhance the
global exploration, is recommended at the initial stages where as for final stages, the inertia weight is reduced for better local exploration.

- **Individual best** $X^* (t)$: During the search process, the particle compares its fitness value at the current position, to the best fitness value it has ever attained at any time up to the current time. The best position that is associated with the best fitness encountered so far is called the individual best, $X^* (t)$. In this way, the best position $X^* (t)$ for each particle in the swarm, can be determined and updated during the search. For example, in a minimisation problem with objective function $J$, the individual best of the $j$th particle $X^*_j (t)$ is determined such that $J[X^*_j (t)] \leq J[X^*_j (\tau)]$, $\tau \leq t$. For simplicity it is assumed that $J^* = J[X^*_j (t)]$. For the $j$th particle, individual best can be expressed as $X^*_j (t) = [x^*_{j,1}(t) \ldots \ldots \ldots x^*_{j,m}(t)]$.

- **Global best** $X^{**} (t)$: It is the best position among all individual best positions (i.e. the best of all) achieved so far. Therefore, the global best can be determined as such that $J(X^{**}(t)) \leq J(X^*_j (\tau))$, $j=1,\ldots,n$. For simplicity, assume that $J^{**} = J(X^{**}(t))$.

- **Stopping criteria**: the conditions under which the search process will terminate. In the present case, the search will terminate if one of the following conditions is met.

  a) The number of iterations since, the last change of the best solution is greater than a prespecified number.

  or
b) The number of iterations reaches the maximum allowable number.

With the description of basic elements as above, the basic steps in PSO are described below.

2.3.2.2 PSO Implementation

Step-1: In order to make uniform search in the initial stages and very local search in later stages, an annealing procedure is followed. A decrement function for decreasing the inertia weight given as $w(t) = \alpha w(t-1)$, $\alpha$ is a decrement constant smaller than but close to 1, is considered here.

Step-2: Feasibility checks, for imposition of procedure of the particle positions, after the position updating to prevent the particles from flying outside the feasible search space.

Step-3: The particle velocity in the $k$th dimension is limited by some maximum value, $v_{k \text{ max}}$. With this limit, enhancement of local exploration space is achieved and it realistically simulates the incremental changes of human learning. In order to ensure uniform velocity through all dimensions, the maximum velocity in the $k$th dimension is given as:

$$v_{k \text{ max}} = (X_{k \text{ max}} - X_{k \text{ min}}) / N$$

(2.1)

where $N$ is chosen number of intervals

In PSO, each particle $X(t)$ represents a solution in the search space. All particles pop(t) have fitness value, which are evaluated by the objective function and have velocities, $V(t)$ which direct the flying of the particles (Swarm). The particles fly throughout the problem
hyperspace to search for optimal or near-optimal solution by following the current optimum particles.

To begin with, PSO is initialized with a group of random particles (solutions) and search for optimal by updating present values. Each particle keeps track of its coordinates in the problem space, which are associated with the best solution (fitness) that it has achieved so far. This implies that each particle has a memory, which allows it to remember the best position on the feasible search space that it has ever visited. This value is commonly called Individual best $X^* (t)$. Another best value that is tracked by the particle swarm optimizer is the best value obtained so far by any particle in the neighborhood of the particle. This location is commonly called Global best $X^{**} (t)$. The basic concept behind the PSO technique consists of changing the velocity $V (t)$ (or accelerating) of each particle toward its $X^* (t)$ and the $X^{**} (t)$ positions at each time step. This means that each particle tries to modify its current position and velocity according to the distance between its current position and $X^* (t)$, and the distance between its current position and $X^{**} (t)$.

Modification of the particle position is realized by the current position and velocity information. Each particle knows its best value so far $X^* (t)$, its velocity $(t)$ and the best value in the group among $X^{**} (t)$. Each agent tries to modify its position using the current velocity and the distance from $X^* (t)$ and $X^{**} (t)$. Velocity of each agent can be modified by the following equation.
\[ v_{j,k}(t) = v_{j,k}(t-1) + c_1 r_1 (x^*_j(t-1) - x_{j,k}(t-1)) + c_2 r_2 (x^{**}_j(t-1) - x_{j,k}(t-1)) \]  
\[ x_{j,k}(t) = v_{j,k}(t) + x_{j,k}(t-1) \]

Where

\[ v_{j,k}(t) \] : Velocity of \( j \)th particle with respect to \( k \)th dimension at time \( t \).

\[ v_{j,k}(t-1) \] : Velocity of \( j \)th particle with respect to \( k \)th dimension at time \( t - 1 \).

\[ x_{j,k}(t) \] : Position of \( j \)th particle with respect to \( k \)th dimension at time \( t \).

\[ x_{j,k}(t-1) \] : Position of \( j \)th particle with respect to \( k \)th dimension at time \( t - 1 \).

\( C_1 \) : Positive constant related to \( X^*_j \) (pbest)

\( C_2 \) : Positive constant related to \( X^{**} \) (gbest)

\( r_1 \) : Random number between 0 and 1

\( r_2 \) : Random number between 0 and 1

\( x^*_j(t-1) \) : pbest Position of particle \( j \) at \( t - 1 \)

\( x^{**}(t-1) \) : gbest Position of the swarm at \( t - 1 \)

Eq. (2.2) and Eq. (2.3) describe the velocity and position update, respectively. Eq. (2.1) calculates a new velocity for each particle based on the particle's previous velocity, the particle's location at which the
best fitness has been achieved so far, and the population global location at which the best fitness has been achieved so far.

The updations of particle position and velocity as explained above are shown in the form of a flow chart given below.

---

**Figure: 2.7 Flow chart depicting the General PSO Algorithm**
Fig. 2.8 as given below shows the concept of updation in the position and velocity of particles (searching points).

![Diagram showing the concept of modification of a searching point.]

**Figure: 2.8 Concept of modification of a searching point.**

- $X(t)$: Current Position
- $X(t-1)$: Previous Position
- $V(t)$: Current Velocity
- $V(t-1)$: Previous Velocity
- $V_{pbest}$: Velocity based on $pbest$
- $V_{gbest}$: Velocity based on $gbest$

There are two key steps when applying PSO to optimization problems, as mentioned below,

1. The representation of the solution and
2. The fitness function.
One of the advantages of PSO is that it takes real numbers as particles, unlike GA, which needs to change the variables to binary encoding or special genetic operators have to be used. For example, when we try to find the solution for \( f(x) = x_1^2 + x_2^2 + x_3^2 \), the particle can be set as \((x_1, x_2, x_3)\), and fitness function is \( f(x) \). Then we can use the standard procedure to find the optimum. The searching is a repetitive process and the stopping criteria are that the maximum iteration number is reached or the minimum error condition is satisfied.

2.3.2.3 Parameter Tuning

There are not many parameters which need to be tuned in PSO. A list of the parameters and their typical values is mentioned below.

1. The number of particles: The typical range is 20 - 40. Actually for most of the problems 10 particles is large enough to get good results. For some difficult or special problems, one can try 100 or 200 particle as well.

2. Dimension of particles: It is determined by the problem to be optimized.

3. Range of particles: It is also determined by the problem to be optimized. Different ranges can be specified for different dimension of particles.

4. \( V_{max} \): It determines the maximum change one particle can take during one iteration. Usually the range of the particle is set as \( V_{max} \). For example, the particle \((x_1, x_2, x_3)\). \( x_1 \) belongs \([-10, 10]\), then \( V_{max} = 20 \)
5. Learning factors: $c_1$ and $c_2$ usually equal to 2. However, other settings can also be used. But usually $c_1$ equals to $c_2$ and ranges from $[0, 4]$.

6. The stop condition: This refers to the maximum number of iterations the PSO to execute. The minimum error requirement or the maximum number of iterations depends on the problem to be optimized.

Later, in order to improve PSO performance, a linear decreasing weight (LDW) parameter called inertia weight is developed. This is brought in for balancing the global and local search and Eq. (2.2) is changed to:

$$v_{j,k}(t) = \omega(t)v_{j,k}(t-1) + c_1 r_1(x^*_j(t-1) - x_{j,k}(t-1)) + c_2 r_2(x_{j,k}^{**}(t-1) - x_{j,k}(t-1))$$

(2.4)

$$x_{j,k}(t) = v_{j,k}(t) + x_{j,k}(t-1)$$

(2.5)

Where, $\omega$ is the inertia weight. Suitable selection of inertia weight ‘$\omega$’ provides a balance between global and local explorations, thus requiring less iteration on average to find a sufficiently optimal solution. As originally developed, $\omega$ often decreases linearly from about 0.9 to 0.4 during a run. In general, the inertia weight $\omega$ is set according to the following equation,

$$\omega(t) = \omega_{max} - \frac{(\omega_{max} - \omega_{min})}{t_{max}} . t$$

(2.6)

Where $t$ : current iteration number

$t_{max}$ : maximum number of iterations

$\omega_{max}$ : upper limit of inertia weighing factor

$\omega_{min}$ : lower limit of inertia weighing factor
In the iteration process the particle velocity is limited by some maximum value $v_{\text{max}}$. The parameter $v_{\text{max}}$ determines the resolution, or fitness, with which regions are to be searched between the present position and the target position. This limit enhances the local exploration of the problem space and it realistically simulates the incremental changes of human learning. If $v_{\text{max}}$ is too high, particles might fly past good solutions. If $v_{\text{max}}$ is too small, particles may not explore sufficiently beyond local solutions. In many experiences with PSO, $v_{\text{max}}$ was often set at 10\%–20\% of the dynamic range of the variable on each dimension.

### 2.3.3 SALIENT FEATURES

The silent features of PSO method are summarised and given below.

1. PSO is a swarm algorithm, inspired by the social dynamics and the noted behaviour observed in socially organized colonies.
2. PSO is a population based algorithm, i.e. it has implicit parallelism. This ensures PSO to be less susceptible to getting trapped to local minima.
3. PSO can easily deal with non differentiable and non convex objective functions, as it uses their information for guidance in the search space.
4. PSO is more flexible and robust than conventional methods, as it uses probabilistic rules instead of deterministic rules for particle movements.
5. Unlike GA, PSO can easily control the balance between the global and local exploration of search space, there by avoiding premature convergence of the search process.
6. The solution quality does not rely on the initial population, as PSO can start from anywhere in the search space, as the algorithm ensures the convergence to the optimal solution.

2.4 CONCLUSIONS

In this Chapter we have briefly presented heuristic methods like High Density cluster Algorithm, Genetic Algorithm and Particle Swarm Optimisation techniques. These are covered as part of preliminary concepts for better understanding of these techniques during their actual implementation, described in chapter 5. It starts with High Density cluster, covering the Basics on Clusters, Cluster Analysis, definitions of core point, border point and noise point and the formation of High Density Cluster. A brief description of Genetic Algorithm, which includes the procedure and important operations like Reproduction, Crossover and Mutation is given. Further, an evolutionary computation technique, Particle Swarm optimisation giving the brief coverage on description of basic elements like Particle, Population, Swarm, Position, particle Velocity, Inertia weight, Individual Best, Global Best and formation of PSO has been given. At the end Primal Dual Interior Point method is explained mentioning important aspects like over view, methodology and salient features.

With the description given as above, an attempt has been made to explain preliminary concepts on High Density Cluster, Genetic Algorithm and Particle Swarm Optimisation which are fundamental methods to carry out this work.