CHAPTER - 3

RESEARCH METHODOLOGY

&

OPTIMIZATION TECHNIQUES

3.1 RESEARCH METHODOLOGY

The flow of research work for optimization is given below in the form of steps.

- Governing equations are very much essential (of any method).
- Analyse the input parameters.
- Choose variable (multi / single variable) for maximise or minimise.
- Set the constraints for good results.
- Select algorithms –
  - GA (Genetic algorithm)
  - CS (Cuckoo search algorithm)
  - BA (Bat algorithm)
  - FPA (Flower pollination algorithm)
  - FA (Fire fly algorithm) etc.
- Analyse the algorithm to give the inputs, constraints and limits
- Add plot function to the algorithm to get graph.
- Finally, optimal results are obtained.

The above steps are given in the form of flow chart shown in the Fig. 3.1.
3.2 OBJECTIVES AND SCOPE OF THE RESEARCH WORK

The primary objectives of this research work are of the following:

a) To analyse the different numerical models such as Bell-Delaware, Kern and FEM based pressure drop model and optimize using different meta-heuristic algorithms.

b) The numerical models are optimized using different algorithms like genetic algorithm, cuckoo search algorithm, bat algorithm and flower pollination algorithm and the results are compared.

c) To examine and optimize the new pressure drop model based on FEM method.
d) To present specific recommendations on the type of algorithm, computational efficiency, performance characteristics on the basis of simulation results. The major contribution is to find optimal geometrical values for the heat exchanger using different evolutionary algorithms in order to obtain best thermo hydraulic performances.

Currently, researchers are focusing on exploring the possibilities of applying novel techniques and intelligent algorithms to reach much more accurate solutions and to reduce computational time.

3.3 INTRODUCTION TO OPTIMIZATION

Heat exchangers are used for different applications and there are many criteria's for optimization. The criteria may be cost, volume, heat transfer area, pressure drop, weight and so on. At the time of optimization those criteria’s are subjected to either maximization or minimization quantitatively, it is called an objective function. During optimization constraints or limitations can be imposed on width, length, heat transfer, allowable pressure drop and so on.

In addition to this, mass flow rates and temperatures could also be considered during optimization. Hence, a heat exchanger design involves a large number of design variables. The problem is how to adjust these design variables so that an optimum design can come up meeting the standard of objective function and constraints.

In many engineering applications and industrial equipment’s, we have to find out optimal solution for a given problem under highly complex constraints. Such optimization problems are often highly nonlinear, it is very challenging task to find optimal solution. Most conventional optimization approaches do not work well for problems with nonlinearity and multimodality. Current trend is to use nature-inspired meta-heuristic algorithms to tackle such difficult problems, and it has been shown that meta-heuristics are surprisingly very efficient.
3.4 THERMAL AND HYDRAULIC DESIGN:

It is very much essential to know the technical terms during design and optimization of a shell and tube exchanger.

- **Rating** – determination of heat transfer and pressure drop of a heat exchanger.
- **Sizing** – determination of physical size, flow arrangement, tube material, fins and construction type.

The effects of geometrical variables are illustrated in the form of flow chart Fig. 3.4.1 and Fig. 3.4.2 during optimization of shell and tube heat exchanger. The objective functions may be overall heat transfer rate or pressure drop on both shell and tube side [6]. The main objective of the heat transfer analysis in heat exchanger is to decrease energy consumption through rating analysis and to reduce size through sizing analysis. Rating analysis is to obtain the maximum possible output from the system with minimum possible input. Sizing analysis is to find a better way to attain maximum compactness of the system as well as provides a space for creative design modifications in the system.

![Flowchart](image)

**Fig: 3.4.1** Effects of geometrical variables on heat transfer in STHX.
3.5 CONTRADICTION OF GEOMETRICAL VARIABLES:

The main objective of any shell and tube heat exchangers is to increase heat transfer and decrease pressure drop. From the above Fig. 3.4.1 and 3.4.2, it is revealed that in order to increase heat transfer, the tubes diameter should be decreased. On the other hand, in order to reduce pressure drop, it is found that tube diameter should be increased. This leads to contradiction of the variables in achieving the objective functions. Hence, we need optimization to find the optimal values so as to maximise an objective function and minimize another objective function simultaneously. The design of a shell and tube heat exchanger involves several variables like shell diameter, tube diameter, tube layout angle, baffle cut, tube layout pitch, length of the tube, baffle spacing, number of tubes etc. subjected into contradictory situation. The optimization technique plays a vital role in solving such contradiction problems.

To carry out optimization a large number of techniques are available, most of the techniques are derived from the behaviour of biological systems or physical systems in nature. The traditional powerful methods for solving many tough optimization problems are particle swarm optimization, simulated annealing
optimization, genetic optimization those are all meta-heuristic algorithms. Different authors have used different techniques for example Srinivas et al. [34], Belanger et al. [27], Fettaka et al. [14], Ahmadi et al. [15], Hajabdollahi et al. [16], Goldberg et al. [64], Selbas et al. [10], Tremblay et al. [65], Ozcelik et al. [12] and Ozkol et al. [66] are used genetic algorithm for their investigation. Some authors used swarm optimization Patel et al. [67], Sadegzadeh et al. [68] for investigation. Since the optimization techniques are different due to their behavioural systems optimal solutions are slightly different from one technique to other. To overcome the disadvantages of genetic algorithm some authors carried out the design and optimization using Artificial Bee Colony (ABC) by Sahin et al. [69]. Chadhuri et al. [70] used Simulated annealing (SA), Fesanghary et al. [71] used harmony search algorithm for optimization.

This chapter deals with different optimization techniques and their procedures namely genetic algorithm, cuckoo search algorithm, bat algorithm and flower pollination algorithm. For any optimization of heat exchangers except heat recovery systems, the operating cost, the sum of investment cost and maintenance costs are generally considered as objective criterion in the literature Unuvar et al. [72], Doodman et al. [73], Eryener et al. [21] and Yan et al. [74].

3.6 DIFFERENT OPTIMIZATION TECHNIQUES

3.6.1 FLOWER POLLINATION ALGORITHM (FPA)

3.6.2 BAT ALGORITHM (AB)

3.6.3 GENETIC ALGORITHM (GA)

3.6.4 CUCKOO SEARCH ALGORITHM (CSA)

3.6.1 FLOWER POLLINATION ALGORITHM (FPA)

3.6.1.1 INTRODUCTION

Flower pollination algorithm (FPA) is developed by the idea of flower pollination process. Basically the pollination process carried the pollen from the male parts of a flower to the female part called stigma of flower [76]. From the biological evolution point of view, the objective of the flower pollination is the survival of the fittest and the optimal reproduction of plants in terms
of numbers as well as fittest. This is in fact an optimization process of plant species.

3.6.1.2 CHARACTERISTICS OF FLOWER POLLINATION ALGORITHM

It is estimated that there are over a quarter of a million types of flowering plants in Nature and that about 80% of all plant species are flowering species. It still remains partly a mystery how flowering plants came to dominate the landscape from Cretaceous period [76, 77]. Flowering plant has been evolving for more than 125 million years and flowers have become so influential in evolution, we cannot imagine how the plant world would be without flowers. The main purpose of a flower is ultimately reproduction via pollination. Flower pollination is typically associated with the transfer of pollen, and such transfer is often linked with pollinators such as insects, birds, bats and other animals. In fact, some flowers and insects have co-evolved into a very specialized flower-pollinator partnership. For example, some flowers can only attract and can only depend on a specific species of insects for successful pollination [78].

Pollination can take two major forms: abiotic and biotic. About 90% of flowering plants belong to biotic pollination, that is, pollen is transferred by a pollinator such as insects and animals. About 10% of pollination takes abiotic form which does not require any pollinators. Wind and diffusion in water help pollination of such flowering plants and grass is a good example [78, 79]. Pollinators, or sometimes called pollen vectors, can be very diverse. It is estimated there are at least 200,000 variety of pollinators such as insects, bats and birds.

Honeybees are a good example of pollinator, and they can also develop the so-called flower constancy [80]. That is, these pollinators tend to visit exclusive certain flower species while bypassing other flower species. Such flower constancy may have evolutionary advantages because this will maximize the transfer of flower pollen to the same or conspecific plants, and thus maximizing the reproduction of the same flower species. Such flower constancy may be advantageous for pollinators as well, because they can be sure that nectar supply is available with their limited memory and minimum cost of learning or exploring. Rather than focusing on some unpredictable
but potentially more rewarding new flower species, flower constancy may require minimum investment cost and more likely guaranteed intake of nectar [81].

Pollination can be achieved by self-pollination or cross-pollination. Cross-pollination, or allogamy, means pollination can occur from pollen of a flower of a different plant, while self-pollination is the fertilization of one flower, such as peach flowers, from pollen of the same flower or different flowers of the same plant, which often occurs when there is no reliable pollinator available.

Biotic, cross-pollination may occur at long distance, and the pollinators such as bees, bats, birds and flies can fly a long distance, thus they can considered as the global pollination. In addition, bees and birds may behave as Lévy flight behaviour [82], with jump or fly distance steps obey a Lévy distribution. Furthermore, flower constancy can be used an increment step using the similarity or difference of two flowers.

3.6.1.3 THE IDEALIZED RULES OF FLOWER POLLINATION ALGORITHM

Now we can idealize the above characteristics of pollination process, flower constancy and pollinator behaviour as the following rules:
1. Biotic and cross-pollination is considered as global pollination process with pollen-carrying pollinators performing Lévy flights.
2. Abiotic and self-pollination are considered as local pollination.
3. Flower constancy can be considered as the reproduction probability is proportional to the similarity of two flowers involved.
4. Local pollination and global pollination is controlled by a switch probability \( p \in (0, 1) \).

Due to the physical proximity and other factors such as wind, local pollination can have a significant fraction \( p \) in the overall pollination activities.

Obviously, in reality, each plant can have multiple flowers, and each flower patch often release millions and even billions of pollen gametes. However, for
simplicity, we also assume that each plant only has one flower, and each flower only
produce one pollen gamete. Thus, there is no need to distinguish a pollen gamete, a
flower, a plant or solution to a problem. This simplicity means a solution \( x_i \) is
equivalent to a flower and/or a pollen gamete. In future studies, we can easily extend
to multiple pollen gametes for each flower and multiple flowers for multi-objective
optimization problems.

From the above discussions and the idealized characteristics, we can design a
flower-based on algorithm, namely, flower pollination algorithm (FPA). There are
two key steps in this algorithm, they are global pollination and local pollination.

In the global pollination step, flower pollens are carried by pollinators such as
insects, and pollens can travel over a long distance because insects can often fly and
move in a much longer range. This ensures the pollination and reproduction of the
most fittest, and thus we represent the most fittest as \( g^* \). The first rule plus flower
constancy can be represented mathematically as

\[
x_i^{t+1} = x_i^t + L (x_i^t - g^*)
\]  

(3.6.1.1)

where \( x_i^t \) is the pollen i or solution vector \( x_i^t \) at iteration t, and \( g^* \) is the current best
solution found among all solutions at the current generation / iteration. The parameter
L is the strength of the pollination, which essentially is a step size. Since insects may
move over a long distance with various distance steps, we can use a L’evy flight to
mimic this characteristic efficiently [82, 83]. That is, we draw \( L > 0 \) from a Levy
distribution

\[
L \sim \frac{\lambda r(\lambda) \sin(\lambda/2)}{\pi} \frac{1}{s^{1+\lambda}} \quad \text{where} \quad (s \geq s_0 > 0)
\]  

(3.6.1.2)

Here \( \Gamma(\lambda) \) is the standard gamma function, and this distribution is valid for
large steps \( s > 0 \). In all our simulations below, we have used \( \lambda = 1.5 \).

The local pollination (rule 2) and flower constancy can be represented as

\[
x_i^{t+1} = x_i^t + \epsilon (x_j^t - x_k^t)
\]  

(3.6.1.3)

where \( x_j^t \) and \( x_k^t \) are pollens from the different flowers of the same plant
species. This essentially mimic the flower constancy in a limited neighbourhood.
Mathematically, if $x_j^t$ and $x_k^t$ comes from the same species or selected from the same population, this become a local random walk if we $\epsilon$ draw from a uniform distribution in $(0,1)$.

Most flower pollination activities can occur at both local and global scale. In practice, adjacent flower patches or flowers in the not-so-far-away neighbourhood are more likely to be pollinated by local flower pollens that those far away. For this, we use a switch probability (rule 4) or proximity probability $p$ to switch between common global pollination to intensive local pollination. To start with, we can use $p = 0.5$ as an initially value and then do a parametric study to find the most appropriate parameter range. From our simulations, we found that $p = 0.8$ works better for most applications [84]. The above two key steps plus the switch condition can be summarized the pseudo code shown in Fig. 3.6.1

Flower Pollination Algorithm (or simply Flower Algorithm)

Objective min or max $f(x)$, $x = (x_1, x_2, ..., x_d)$

Initialize a population of $n$ flowers/pollen gametes with random solutions

Find the best solution $g^*$ in the initial population

Define a switch probability $p \in (0,1)$

while ($t < \text{MaxGeneration}$)

for $i = 1 : n$ (all $n$ flowers in the population)

if $\text{rand} < p$,

Draw a (d-dimensional) step vector $L$ which obeys a L’evy distribution

Global pollination via $X_i^{t+1} = X_i^t + L(X_j^t - g_*)$

else

Draw $\epsilon$ from a uniform distribution in $(0,1)$

Randomly choose $j$ and $k$ among all the solutions

Do local pollination via $X_i^{t+1} = X_i^t + \epsilon(X_j^t - X_k^t)$

end if

Evaluate new solutions

If new solutions are better, update them in the population
end for

Find the current best solution $g^*$

end while

Fig. 3.6.1. Pseudo code of the proposed Flower Pollination Algorithm (FPA).

3.6.2 BAT ALGORITHM (BA)

Bats are the only mammals with wings. They are fascinating animals because they have advanced capability of echolocation. It is approximated that 996 different species are there which account 20% of all mammal species [85]. Their size varies from tiny to gaint bats of up to 2 m and weight up to about 1kg. Microbats use ecoolocation extensively while megabats do not [86, 87]. Most microbats are insectivores. They use a type of sonar called echolocation to detect prey, avoid obstacles and locate their roosting crevices in the dark. These bats produce a very loud sound pulse and listen for the echo that bounces back from the surrounding objects. Their pulses differ in properties and can be correlated with their searching strategies, depending on species. Many bats use short, frequency-modulated signals to sweep through about an octave, while others more regularly use constant-frequency signals for echolocation. Their particular signal band width varies is determined by the species, and often increased by using more harmonics.

3.6.2.1 ACOUSTICS OF ECHOLOCATION

Each pulse of microbats is a period of 8 ms to 10 ms and it has frequency range from 25 kHz to 150 kHz. The ultrasonic burst may last 5 to 20 ms and microbats emit these burst 10 to 20 every second. During hunting of prey, the rate of pulse emission can be sped up to about 200 pulses per second when the bats fly near the prey. Many of these short sound bursts indicate fantastic ability of the signal processing of bats. In simple fact, studies shows the integration time of the bat ear is typically about 300 to 400 $\mu$s. As the speed of sound in air is typically $v = 340$ m/s, the wavelength $\lambda$ of the ultrasonic sound bursts with a constant frequency $f$ is given by

$$\lambda = \frac{v}{f}$$  \hspace{1cm} (3.6.2.1)
\( \lambda \) is in the range of 2 mm to 14 mm for the typical frequency range from 25 kHz to 150 kHz. Such wavelengths are in the same order of their prey sizes. Extremely, the emitted pulse could be as loud as 110 dB and they fall in the ultrasonic region. The loudness also varies from the loudest when searching for prey and to a quieter base when homing towards the prey. The travelling range of such short pulses are typically a few metres, depending on the actual frequencies. Microbats can manage to avoid obstacles as small as thin human hairs.

Studies show that microbats use the time delay from the release and detection of the echo, the time big difference between their two ears, and the loudness different versions of the echoes to formulate three dimensional scenario of surrounding. They can discover the distance and direction of the target, the sort of prey, and even the moving speed of the prey such as small insects. Indeed, studies suggested that bats seem to be able to discriminate targets by the variations of the Doppler effect induced by the wing-flutter rates of the target insects.

Obviously, some bats have good eyesight, and most bats also have very sensitive smell sense. In reality, they will use all the senses as a combination to maximize the efficient detection of prey and smooth navigation. However, here we are only interested in the echolocation and the associated behaviour. Such echolocation actions of microbats can be formulated so that it can be associated with the objective function to be optimized, which makes it possible to come up with new optimization algorithms.

3.6.2.2 THE IDEALIZED RULES

The following idealized rules are used:

1. All bats use echolocation to sense distance, to detect prey and to find background barriers.
2. They fly randomly with certain velocity \( v_i \) at position \( x_i \) with minimum frequency \( f_{\text{min}} \) and varying wavelength \( \lambda \) and they can automatically adjust the wavelength depending on the proximity of the target.
3. The loudness ranges from a higher value \( A_0 \) to smaller value \( A_{\text{min}} \).
In addition to these simplified assumptions, we also use the following approximations, for simplicity. In general the frequency $f$ in a range $(f_{\text{min}}, f_{\text{max}})$ corresponds to a range of wavelengths $(\lambda_{\text{min}}, \lambda_{\text{max}})$. For example a frequency range of (20 kHz, 500 kHz) corresponds to a range of wavelengths from 0.7mm to 17mm.

For a given problem, we can also use any wavelength for the ease of implementation. In the actual implementation, we can adjust the range by adjusting the wavelengths (or frequencies), and the detectable range (or the largest wavelength) should be chosen such that it is comparable to the size of the domain of interest, and then toning down to smaller ranges. Furthermore, we do not necessarily have to use the wavelengths themselves, instead, we can also vary the frequency while fixing the wavelength $\lambda$. This is because $\lambda$ and $f$ are related due to the fact $\lambda f$ is constant.

For simplicity, we can assume $f \in (0, f_{\text{max}})$. We know that higher frequencies have short wavelengths and travel a shorter distance. For bats, the typical ranges are a few metres. The rate of pulse can simply be in the range of (0, 1) where 0 means no pulses at all, and 1 means the maximum rate of pulse emission.

Based on these approximations and idealization, the basic steps of the Bat Algorithm (BA) can be summarized in the flow chart shown in Fig. 3.6.2.1 and pseudo code shown in Fig.3.6.2.2.
Fig: 3.6.2.1 Flow chart for bat algorithm.

Objective function $f(x)$, $x = (x_1, ..., x_d)^T$

Initialize the bat population $x_i$ ($i = 1, 2, ..., n$) and $v_i$

Define pulse frequency $f_i$ at $x_i$

Initialize pulse rates $r_i$ and the loudness $A_i$
While \( t < \text{Max number of iterations} \)
Generate new solutions by adjusting frequency, and updating velocities and locations/solutions [equations (3.6.2.2) to (3.6.2.4)]
if (rand > \( r_i \))
Select a solution among the best solutions
Generate a local solution around the selected best solution
End if
Generate a new solution by flying randomly
If (rand \( < A_i \) \& \( f(x_i) < f(x^*) \))
Accept the new solutions
Increase \( r_i \) and reduce \( A_i \)
End if
Rank the bats and find the current best \( x^* \)
End while
Post process results and visualization

Fig: 3.6.2.2 Pseudo code of the bat algorithm (BA).

3.6.2.3 MOVEMENT OF VIRTUAL BATS

In simulations, virtual bats are used. Firstly, the positions \( x_i \) and velocities \( v_i \) are to be indicated in a d-dimensional search space. The new solutions \( x_{i+1} \) and velocities \( v_{i+1} \) at time step \( t \) are given by
\[
\begin{align*}
  f_i &= f_{\text{min}} + (f_{\text{max}} - f_{\text{min}}) \beta \\
  v_{i+1} &= v_i + (x_{i+1} - x_i) f_i \\
  x_{i+1} &= x_i + v_{i+1}
\end{align*}
\]
where \( \beta \in (0, 1) \) is a random vector drawn from a uniform distribution. Here \( x^* \) is the current global best location (solution) which is located after comparing all the solutions among all the \( n \) bats. As the product \( \lambda f_i \) is the velocity increment, we can use either \( f_i \) (or \( \lambda i \)) to adjust the velocity change while fixing the other factor \( \lambda i \) (or \( f_i \)), depending on the type of the problem of interest.
In our implementation, we will use $f_{\text{min}} = 0$ and $f_{\text{max}} = 100$, depending on the domain size of the problem of interest. For the local search part, once a solution is selected among the current best solutions, a new solution for each bat is generated locally using random walk

$$x_{\text{new}} = x_{\text{old}} + \varepsilon A_t$$

where $\varepsilon$ is a random number vector drawn from $(-1, 1)$, while $A_t = \langle A_t \rangle$ is the average loudness of all the bats at this time step. The update of the velocities and positions of bats have some similarity to the procedure in the standard particle swarm optimization as $f_i$ essentially controls the space and range of the movement of the swarming particles. To a degree, BA can be considered as a balanced combination of the standard particle swarm optimization and the intensive local search controlled by the loudness and pulse rate.

The heat exchanger variables (or design variables) are baffle spacing, baffle cut, tube pitch and tube length. These variables ranges have to be specified in the lower bounds and upper bounds which are provided in the bat algorithm code. The lower and upper bounds are then taken inside to calculate the different velocity ($v_i$), position ($x_i$) and frequency ($f_i$). Hence numbers of local solutions are formed. For each local solution a fitness value is calculated. The minimum fitness value is the survival of the fittest. This is how the heat exchanger variables are taken into consideration during bat movements. Further, the number of variables must be specified in the bat algorithm code.

### 3.6.2.4 LOUDNESS AND PULSE EMISSION

Furthermore, the loudness $A_i$ and the rate $r_i$ of pulse emission have to be updated accordingly as the iterations proceed. As the loudness usually decreases once a bat has found its prey, while the rate of pulse emission increases, the loudness can be chosen as any value of convenience. For example, we can use $A_0 = 100$ and $A_{\text{min}} = 1$. For simplicity, we can also use $A_0 = 1$ and $A_{\text{min}} = 0$, assuming $A_{\text{min}} = 0$ means that a bat has just found the prey and temporarily stop emitting any sound. Now we have

$$A_i^{t+1} = \alpha A_i^t, \quad r_i^{t+1} = r_i^0 [1 - \exp(-\theta t)],$$

(3.6.2.6)
where $\alpha$ and $\vartheta$ are constants. In fact, $\alpha$ is similar to the cooling factor of a cooling schedule in the simulated annealing. For any $0 < \alpha < 1$ and $\vartheta > 0$, we have

$$A_t^t \to 0, \; r_i^t \to r_i^0, \; \text{as} \; t \to \infty. \quad (3.6.2.7)$$

In the simplicity case, we can use $\alpha = \vartheta$, and we have used $\alpha = \vartheta = 0.9$ in our simulations. The choice of parameters requires some experimenting. Initially, each bat should have different values of loudness and pulse emission rate, and this can be achieved by randomization. For example, the initial loudness $A_i^0$ can typically be $(1, 2)$, while the initial emission rate $r_i^0$ can be around zero, or any value $r_i^0 \epsilon (0, 1)$ if using equation (3.6.2.6). Their loudness and emission rates will be updated only if the new solutions are improved, which means that these bats are moving towards the optimal solution.

### 3.6.3 GENETIC ALGORITHM (GA)

Genetic algorithms (GAs) are stochastic search and enhancement methods that are propelled by Darwin's hypothesis of advancement and the regular law of survival of the fittest. Genetic algorithms have a place with the vast class of developmental calculations, which imitates nature's transformative standards to produce solutions for enhancement issues. Genetic Algorithms were presented first by John Holland in 1970 at University of Michigan. GAs use hereditary administrators to execute the standard of development. The genetic operators utilized as a part of the genetic algorithms are selection operator, crossover operator and mutation operator. Selection operator (additionally called as multiplication administrator) stresses the survival of the fittest in GA. Crossover operator is utilized to mate two chromosomes by sharing the qualities and bringing about formation of better offspring. Mutation operator protects the differences among the population. The essential procedure of genetic algorithm is shown in Fig.3.6.3.1.
A Genetic Algorithm starts with expected candidate solutions and is advanced towards better solution through the procedure of iteration. In every one of the iterations, fitness of each of the hopeful arrangement is assessed.

A fitness function permits every potential solution to be quantitatively assessed. After an arbitrary initial population in the ranges of design variables is produced, the calculation makes a succession of new generations iteratively until the ceasing foundation is met. In this procedure, the choice of parents depends on their fitness children are created by random out arbitrary improvements to a single parent (change) or by joining the vector passages of a couple of parents (hybrid), and after that replace the present current population with the children to form the next generation. The calculation chooses individuals with better fitness values as parents, and eliminates the inferior. This promises the calculation joins to a best person, which most likely represents to the best solution of the given issue [89-92]. The flow chart of a genetic algorithm is presented in Fig. 3.6.3.2.

**Fig : 3.6.3.1** Basic process of genetic algorithm.
In the genetic algorithm, each individual of the population may be represented either as a string of binary coded or real coded variables that correspond to the chromosomes in natural genetics. The efficiency of the genetic algorithm improves by using real coding as

1) It does not need the translation of chromosomes to binary type,
2) Less memory needed for storing,
3) No loss of precision since the values are not converted into binary type.

The utility of the genetic algorithm is achieved by exploiting the best solution from the available search space. To enhance their performance, all the components of the genetic algorithm must be examined thoroughly and additional heuristics should be included in the algorithm. In genetic algorithm, accumulated information is exploited by the selection mechanism, whereas new regions of search space are explored by means of genetic operators. Therefore genetic operators play a key
role in arriving at the global optima. Hence it is necessary to investigate the suitability of genetic operators in solving real world problems.

An algorithm based on non-dominated sorting was proposed by Srinivas and Deb et al. [34] and called non-dominated sorting genetic algorithm (NSGA). A more detailed description of GA can be found in the text written by Goldberg (1989) [64] and Lee et al. [90]. Multi-objective optimization using GA was carried out by Hilbert et al. [91], Belanger et al. [92].

3.6.4 CUCKOO SEARCH ALGORITHM (CSA)

Cuckoo search is an evolutionary optimization algorithm that is developed by Yang and Deb (2009) et al. [93]. The cuckoo search theory was based on the species of bird called cuckoo. They are fascinating birds because of their beautiful sounds and aggressive reproduction strategy. The matured cuckoo lay their eggs in the nests of other host species or birds. This is known as obligate brood parasitism. Every egg in a nest represents a solution and especially a cuckoo egg represents a new solution. In case, if a host bird finds the eggs are not its own, it will either discard these outsider eggs or just surrender its nest and builds another nest somewhere else. The chances of more cuckoo eggs survive is the profit gained. So, the optimization is the survival of more eggs and become mature cuckoo.

3.6.4.1 CUCKOO SEARCH METHODOLOGY

The search is based on three important assumptions:

i) Each cuckoo lays one egg at a time and dumps its egg in a randomly chosen nest.

ii) The high quality eggs in the best nest will carry over to the next generation.

iii) The number of available nests is fixed and the probability of finding the cuckoo egg by the host bird is $p_a \in (0, 1)$.

Based on the above rules, the host bird can either throw the egg away or destroy the entire nest and build a new nest completely [94]. The algorithm is
summarized in the flow chart in Fig.3.6.4.1 and the pseudo code for cuckoo search is shown in Fig. 3.6.4.2.

**Fig: 3.6.4.1** Flowchart for cuckoo optimization algorithm [58].

- **Objective function** \( f(x) = (x_1, \ldots, x_d)^T \)
- **Generate initial population** of \( n \) host nests \( x_i \) \( (i = 1, 2, \ldots, n) \)
- **While** \( t < \text{MaxGeneration} \) or \( \text{(stop criterion)} \)
- **Get a cuckoo randomly** or by Lévy flights and then evaluate its quality/fitness \( F_i \)
Choose a nest among \( n \) (say, \( j \)) randomly
if \( F_i > F_j \),
replace \( j \) by the new solution;
end
A fraction \( (p_a) \) of worse nests is abandoned and new ones are built;
Keep the best solutions (or nests with quality solutions);
Rank the solutions and find the current best
End while
Post process results and visualization
end

**Fig: 3.6.4.2** Pseudo code of the Cuckoo search.

### 3.6.4.2 GENERATING INITIAL CUCKOO HABITAT

To solve an optimization problem, it is necessary that the values of the problem parameters should be formed as an array. In this algorithm, this array is called a “habitat”. In \( N_{\text{var}} \) dimensional optimization, a habitat is an array of \( 1 \times N_{\text{var}}, \) representing current living position of cuckoo. The following formulas are used for generating initial cuckoo habitat.

\[
\text{Habitat} = (x_1,x_2,\ldots,x_{N\text{var}}) \quad (3.6.4.1)
\]

\[
\text{Profit} = f_p(\text{habitat})=f_p(x_1,x_2,\ldots,x_{N\text{var}}) \quad (3.6.4.2)
\]

The profit is obtained by evaluation of the above profit function.

\[
\text{Profit} = - \text{cost (habitat)} = - f_c(x_1, x_2, x_3,\ldots, x_{N\text{var}}) \quad (3.6.4.3)
\]

The above equation is find maximum profit of habitat. To start the optimization algorithm, a candidate habitat matrix of size \( N_{\text{pop}} \times N_{\text{var}} \) is generated. Another important thing is the egg laying radius (ELR), it is the maximum distance in which cuckoo lay eggs from their habitat as shown in Fig. 3.3.4.3. The ELR equation is given below

\[
\text{ELR} = \alpha \times \frac{\text{Number of current cuckoo's eggs}}{\text{Total number of eggs}} \times (\text{var}_{hi} - \text{var}_{low}) \quad (3.6.4.4)
\]

where \( \alpha \) is an integer, \( \text{var}_{hi} \) and \( \text{var}_{low} \) are the upper and the lower limit of variables respectively.
3.6.4.3 CUCKOOS’ STYLE FOR EGG LAYING

Each cuckoo starts laying eggs randomly in some other host bird’s nest within her ELR, as shown in Fig. 3.7. After that, some cuckoo eggs that are less similar to host birds’ own eggs are detected by the host birds and are thrown out of the nests. So, after the egg-laying process, p% of all eggs (usually 10%), with less profit values, will be killed. These eggs have no chance to grow. The rest of the eggs grow in host nests, hatch, and are fed by host birds.

Also, only one egg has the chance to grow in a nest. This happens because when a cuckoo egg hatches and the chick comes out, it throws the host bird’s own eggs out of the nest. In the case that the host bird’s eggs hatch earlier and the cuckoo egg hatches later, the cuckoo’s chick eats most of the food the host bird brings to the nest (because of the three-times-bigger body, the cuckoo chick pushes other chicks and eats more). After a couple of days, the host bird’s own chicks die from hunger and only the cuckoo chick remains in the nest.

3.6.4.4 IMMIGRATION OF CUCKOOOS

After cuckoos have laid their eggs, the next stage is the immigration of the cuckoos. Young cuckoos will grow and become mature and they will live in their own area and society for some time. But when the time for egg laying approaches, they immigrate to new and better habitats where their eggs are more similar.
to eggs of the host birds and also where there is more food for new youngsters. This situation will make the cuckoo group form in a different area. The society with the best profit is the goal point for cuckoos to immigrate.

When mature cuckoos live all over the environment, it is difficult to recognize which cuckoo belongs to which group. To solve this problem, the grouping of cuckoos is done with the k-means clustering method (‘k’ of 3–5 seems to be sufficient in simulations). The clustering method means to group cuckoos in a cluster and find the best group and selects the goal habitat. For that, the cuckoo groups are constituted and their mean profit value is calculated. The maximum values of these mean profit values determine the goal group and, consequently, the best-group habitat is the new destination habitat for immigrant cuckoos. When moving toward a goal point, the cuckoos do not fly all the way to the destination habitat. They fly only a part of the way and also have a deviation [95, 96]. This movement is clearly shown in Fig. 3.6.4.4. As seen in Fig. 3.6.4.4. each cuckoo flies only λ % of the entire distance toward the goal habitat and also has a deviation of Ψ radians. These two parameters, λ and Ψ, help cuckoos search for more positions in all environments. For each cuckoo, λ and Ψ are defined as follows:

![Cuckoos' Living Area](image)

**Fig: 3.6.4.4.** Immigration of a sample cuckoo toward goal habitat.
\[ \lambda = U(0, 1) \quad (3.6.4.5) \]
\[ \Psi = U(-\omega, \omega) \quad (3.6.4.6) \]

where \( \omega \) is a parameter which constrains the deviation from goal habitat. \( \lambda \) is a random number uniformly distributed between 0 and 1. \( \omega \) is a parameter that confines the deviation from the goal habitat. An \( \omega \) of \( \pi/6 \) (rad) seems to be enough for good convergence of the cuckoo population to global maximum profit. When all cuckoos have immigrated toward the goal point and new habitats are specified, each mature cuckoo is given some eggs. Then, considering the number of eggs dedicated to each bird, an ELR is calculated for each cuckoo. Afterward, the new-egg-laying process restarts [97].