CHAPTER-2

METHDOLOGIES FOR UNIT COMMITMENT

2.1 INTRODUCTION

Electrical power today plays an exceedingly important role in the life of the community and in the development of various sectors of economy. In fact, the modern economy is totally dependent on the electricity as a basic unit. The technological revolution affects every aspect of modern life. Quickly developing computer technologies are changing the work environment for power engineers employed by utilities and manufacturers. Emerging computer software permits more accurate design of the power system components or networks and facilities, more urbane operational methods that result in high system reliability. Although computational techniques have proved valuable in general function optimization, these appear particularly pertinent for adding non-linearly constrained optimization problems. Constrained optimization problems present the difficulties of potentially non-convex or even disjoint feasible regions. Classical linear as well as non-linear programming methods are often either incompatible or unfeasible when applied to these constrained optimization problems. Difficulties arise because either the amount of computation required quickly becomes riotous as the size of the problem increases or the constraints violate the required supposition, e.g. differentiability or convexity. Unfortunately the real world often poses such problems. In the special case of continuous parameter optimization in which all parameters are real valued, Newton developed the gradient method, which is also known as the method of steepest descent. It is obvious that steepest descent algorithms are applied only to continuously differentiable objective functions. But in case, either the objective function is not continuously differentiable or the function is not completely defined due to limited knowledge, which often occurs in real world applications, the designer has to resort to other methods such as search algorithms [3, 7, 20, 54].

Most real world search and optimization problems naturally involve multiple-objectives. The extremist principle mentioned above cannot be applied to only one objective, when the rest of objectives are also imperative. In an optimization problem involving more than one objective function, the task of finding one or more most favorable solutions is known as multi-objective optimization. Such search and optimization problems are known as multiple criterion decision-making (MCDM) problems in terminology of management. Different solutions may produce conflicting scenarios among different objectives. Optimization is the act of obtaining the best feasible result under given circumstances, which correspond to extreme value of one or more objectives. In design, construction and maintenance of any engineering system, engineers have to take many technological and supervisory decisions at several stages. The ultimate aim of all such decisions is either to minimize the cost incurred or maximize the benefit desired or minimize the effort required. This effort or benefit is generally expressed as a function of certain decision variables. An optimization algorithm is a procedure, which begins with one or more design solutions, supplied by the user and then iteratively generates and checks new design solutions, by comparing them till the optimum or an adequate solution is found [7, 65, 71].

2.2 OPTIMIZATION PROBLEM

An important aspect of the optimal design process is the formulation of the design problem in a mathematical format, which is acceptable to an optimization algorithm. There is no unique way of formulating all the engineering problems. Each problem can be of different type or nature such as: design, decision, control, scheduling, operation, planning or maintenance etc. Different techniques need to be explored in different problems and under different situations. The purpose of the formulation procedure is to create a mathematical model for the optimal problem, which can then be solved using an optimization algorithm. The formulation of an optimization problem begins with identifying the underlying design or decision variables, which are primarily varied during the optimization process. There may be many decision parameters of which some are highly sensitive for the proper working of the strategy and their choice largely depends on the user. These parameters are called design variables. Having chosen the design variables, the next task is to identify the constraints associated with the optimization problem. The constraints represent some functional relationships among the design variables and other design parameters satisfying certain physical phenomenon and certain resource limitations. There are two types of constraints that emerge from most considerations. These are equality constraints and inequality constraints. Equality constraints are usually more difficult to handle and therefore, need to be avoided whenever possible. The ensuing task in the formulation procedure is to find the objective function in terms of the design variables and other problem parameters. The objective function can be of two types, either it is to be maximized or it is to be minimized. Duality principle helps by allowing the same algorithm to be used for maximization or minimization with a minor change in the objective function, instead of a change in the intact algorithm. The final task of the formulation procedure is to set the minimum and maximum bounds on each design variable, in order to confine the search within these bounds. Certain optimization algorithms do not require this information. In these problems, the constraints completely decide the feasible region. On the completion of the above tasks, the optimization problem is mathematically written in a special format, known as non-linear programming (NLP) format. Formulation can represent either a maximization problem or a minimization problem [92, 96, 164, 366].

In the earlier days of power system operation, the *unit commitment problem* (UCP) was defined in terms of the conventional dispatching, whose objective was to schedule the generators to meet the load, maintain voltages and frequency within prescribed tolerances and minimize operating cost. Today, the UCP is used to solve a constrained combinatorial, mixed integer and non-linear optimization problem whose development has closely followed the advancement in numerical optimization techniques and computer technology. An optimization problem generally requires solving a set of linear/non-linear equations, subjected to various equality and inequality constraints, to get a global optimal solution and can be expressed as:

Minimize $F(X,U)$ (2.1)

Subject to $g_i(X,U) \geq 0 \qquad (i=1,2,...,q)$ (2.2)

$$h_j(X,U) = 0 \qquad (j=1,2,...,p) \tag{2.3}$$

$$x_i^{\min} \leq x_i \leq x_i^{\max} \qquad (i=1,2,...,n) \tag{2.4}$$

$$u_j^{\min} \leq u_j \leq u_j^{\max} \qquad (j=1,2,...,m) \tag{2.5}$$

where,

$g_i(X,U)$ Set of inequality constraints of vector arguments X and U;

$h_j(X,U)$ Set of non-linear equality constraints**;**

U Vector of dependent variables; $[u_1,u_2,...,u_m]^T$

X Vector of control variables; $[x_1,x_2,...,x_n]^T$

$x_i^{\min}, x_i^{\max}$ Lower and upper bounds on control variable x_i;

$u_j^{\min}, u_j^{\max}$ Lower and upper bounds on dependent variable u_j;

When load forecasts and generators available for power generation are given, then there is a need to decide when each generator would be started up and shut down as fixed costs are involved in starting and stopping generators. So, the main objective is to minimize the operating cost while having enough capacity online to track the load changes and cover for random generator failures. Common objective in power system may include: active power generation cost minimization, pollutant gaseous emission minimization, active power loss minimization, risk minimization, multi area commitment and generation schedule, minimum number of controls scheduled and so on. Equality and in-equality constraints include limits on all control variables, load dispatch equations, load/power balance, spinning reserve limits, minimum up/down time limits, crew limits, initial operating status of generating units, power limits and thermal limits [311].

In recent years, several major modern optimization techniques have been applied to power systems. Evolutionary algorithms and their related computations have emerged as the latest solver systems that use computational models inspired by Darwinian natural selection processes as key elements in algorithm design and implementation. In general, evolutionary computation is applied to solve optimization problems as well as multi objective optimization problems that cannot be solved with other tools because of their intrinsic difficulty, high dimensionality, or incomplete definition. Evolutionary algorithms mimic natural evolutionary principles to constitute random search and optimization procedures. There are two distinct types of optimization algorithms, which are vastly in use. Evolutionary algorithms are different from direct search and optimization procedures in a variety of ways. Evolutionary optimization techniques find and maintain multiple solutions in one single simulation run. However, direct search optimization algorithms use a single solution update during iterations and use a deterministic transition rule. Deterministic algorithms search with specified rule for moving for one solution to the other. These algorithms have been in use for quite some time and have been successfully applied to many engineering design problems. These are called conventional optimization methods. There is another type of algorithms, which is stochastic/random in nature, with probabilistic transition rules and is called non-conventional optimization algorithms. These are gaining popularity due to certain properties like efficiency, conversion rate, solution accuracy, which the deterministic algorithms some times do not have. The designer must know the advantages and limitations of various methods and choose the one that is more efficient to the problem at hand [1, 2, 4, 7, 11].

2.3 OPTIMIZATION METHODOLOGIES

Optimization algorithms are becoming gradually more popular in engineering design activities, because of the accessibility and affordability of high-speed computation. Over the preceding three decades, several algorithms have been developed to solve various engineering and scientific optimization problems. The majority of these algorithms are based on linear and nonlinear programming techniques that require extensive gradient information and usually try to find an improved solution in the vicinity of a starting point. These numerical optimization algorithms provide an useful strategy to obtain the global optimum in simple and ideal models. However, many real-world engineering and scientific optimization problems are very complex and difficult to solve, using these algorithms. If there is more than one local optimum in the problem, the result may depend on the selection of an initial point, and the obtained optimal solution may not necessarily be the global optimum. Furthermore, the gradient search may become difficult and unstable when the objective function and constraints have multiple or sharp peaks. The computational drawbacks of existing numerical linear and non linear methods have forced researchers to rely on meta-heuristic algorithms based on simulations to solve engineering and scientific optimization problems.

In the present study of unit commitment problem of electric power system with multiple objectives and for multiple areas, two global search algorithms i.e. Harmony

Search algorithm and Differential Evolution algorithm are discussed in details and further attempt is made to combine with local search method of random search to obtain better performing hybrid methods. Further, two global search algorithms i.e. Harmony Search and Differential Evolution are integrated to enhance the performance by acquiring their good quality. Underlying theory and corresponding algorithms for these optimization methods are discussed below.

2.3.1 Harmony Search Algorithm

Harmony search (HS) algorithm developed by Geem et al. [185, 247] is a meta-heuristic algorithm search algorithm, which was based on the musical process of searching for a perfect condition of harmony. The harmony in melody is equivalent to the optimization solution vector, and the musician's improvisations are similar to local and global search schemes in optimization techniques. The HS algorithm does not require initial values for the decision variables and uses a stochastic random search rather than gradient search. The mathematical formulation of HS algorithm is based on the harmony memory consideration rate and the pitch adjustment rate; therefore the derivative information is not required. The pitch of each musical instrument determines the aesthetic quality of music just as the set of values assigned to each decision variable decide the objective function value. Compared to others meta-heuristic optimization algorithms, the HS algorithm requires less mathematical operations and can be easily adopted to solve various types of discrete and continuous variables engineering and scientific optimization problems[284].

Harmony search algorithm has ability to overcome the drawback of GA's building block theory and explicitly considers the relationship using ensemble operation [261]. Geem et al. [242] proposed a multi-pitch adjusting rate (multiple P_{AR}) for generalized orienteering problem. They proposed three P_{AR}'s that are the rates of moving to nearest, second nearest, and third nearest cities, respectively. Geem [261] presented the use of fixed parameter values, such as H_{MS}, H_{MCR}, P_{AR}, and $iter_{max}$, while bandwidth (bw) was set to a range from 1 to 10% of the total value of data range. Mahdavi et al. [274] proposed improved harmony search (IHS) algorithm, which includes dynamic adaptation for both pitch adjustment rate (P_{AR}) and bandwidth (bw) values. But it faces the difficulty of determining the lower and upper bound of automatic bandwidth, which was overcome by global-best harmony search (GHS) algorithm proposed by Omran and Mahdavi [297]. GHS algorithm incorporate the PSO concept, global best particle, by replacing the bw parameter altogether and adding a randomly selected decision variables from the best harmony vector in HM. Mukhopadhyay et al.[298] suggested that bw will be the standard deviation of the current population when H_{MCR} is close to unity. Degertekin[299] proposed a HM initialization technique that generated two times of HMS initial harmonies but placed only the best HMS of these into the initial HM. Chakraborty et al. [332] proposed Differential Harmony Search algorithm, a new improvement to HS through the Differential Evolution (DE) mutation operator, which replaces the pitch adjustment operation in classical HS with a mutation strategy borrowed from the DE (DE/rand/1/bin class) algorithm. Hasancebi et al. [333] and Saka and Hasancebi [334] proposed a new adaptation for HS by making both HMCR and PAR change dynamically during the improvisation process of HS. This step is to make the selection of these parameter values problem independent, therefore, improves the performance of HS in finding an optimal solution. Kattan et al. [351] used HS as a new training technique for feed-forward artificial neural networks (ANN). Wang and Huang [352] proposed a new variation of HS algorithm that focuses on the dynamic selection of bw and PAR parameters. Al-Betar et al. [353] also proposed a Multi-pitch Adjusting Rate strategy for enhancing the performance of HS in solving course timetabling problem. They proposed eight procedures instead of using one PAR value, each of which is controlled by its PAR value range. Each pitch adjustment procedure is responsible for a particular local change in the new harmony. Furthermore, the acceptance rule for each pitch adjustment procedure is changed to accept the adjustment that leads to a better or equal objective function. Even though, HS has the ability to escape from local minima, does not require differential gradients and initial value setting for the variables and free from divergence and has strong ability to explore the regions of solution space in a reasonable time. However, it has lower exploitation ability in later period and it easily gets trapped into local optima and converges very slowly. To improve the exploitation ability of HS algorithm in later stage and provide global optimal solution, a novel and hybrid version of harmony search combined with random search algorithm is presented in the present research to solve single and multi-area unit commitment problem of electric power system. The systematic procedural steps to apply the Harmony search method are described in the algorithm below, to the optimization problem defined below:

The optimization problem is defined in terms of scalar objective function $f(X)$ as follows:

$$\text{Minimize } f(X) \tag{2.6a}$$

$$\text{Subject to } x_i^{\min} \le x_i \le x_i^{\max} \qquad (i = 1, 2, ..., N) \tag{2.6b}$$

where, $X = [x_1, x_2, ..., x_N]^T$ is the set of design variables and N is the number of design or decision variables.

Algorithm

The optimization procedure for the harmony search algorithm consists of the following nine steps:

Step-I: *Define algorithm parameters.*

Specify the input parameters required by HS algorithm to solve the optimization problem defined by eqn.(2.6) such as; number of solution vectors in harmony memory (HM) i.e. harmony memory size (H_{MS}), pitch adjustment rate (P_{AR}), harmony memory consideration rate (H_{MCR}), and termination criterion (maximum number of iterations, $iter_{\max}$). H_{MCR} and P_{AR} are parameters that are used to improve the solution vector.

Step-II: *Initialization of Harmony Memory*

The initial population HM (Fig.2.1) consisting of H_{MS} vectors is generated randomly. The HM matrix is filled with H_{MS} vectors as follows:

$$\mathrm{HM} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1N} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2N} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{H_{MS}1} & x_{H_{MS}2} & x_{H_{MS}3} & \cdots & x_{H_{MS}N} \end{bmatrix}_{H_{MS}\times N} \tag{2.7a}$$

The element of HM matrix is initialized as:

$$X_{ij} = X_i^{\min} + rand()[X_i^{\max} - X_i^{\min}] \qquad (j = 1, 2, ..., N; i = 1, 2, , H_{MS}) \tag{2.7b}$$

where, $rand()$ is uniform random number within [0,1].

Fig.2.1: Random Initialization of HM

N is the number of decision variables and H_{MS} is the Harmony memory size.

Step-III: *Memory Improvisation*

In this step, the new harmony vector X_{ij}^{New} is improvised based on three rules: (i) harmony memory consideration rule (ii) random consideration rule, and (iii) pitch adjustment. The feasibility of the new solution is considered during the improvisation process.

Step-IV: *Harmony memory considering rule and random initialization rule*

The H_{MCR} is the probability of choosing one value from the historic values stored in the Harmony memory, and (1-H_{MCR}) is the probability of randomly choosing one feasible value outside of those stored in the Harmony memory (Fig.2.2).

For this rule, a new random number r_1 is generated within the range [0,1].

If $r_1 < H_{MCR}$, then the first decision variable in the new vector X_{ij}^{New} is chosen randomly from the values in the current HM as follows:

$$X_{ij}^{New} = X_{ij}\,; \quad (j = 1,2,...,N; i = 1,2,...,H_{MS}) \tag{2.8}$$

where, H_{MCR} is the Harmony Memory Consideration Rate

If the condition $r_1 < H_{MCR}$ fails, the new first decision variable in the new vector X_{ij}^{New} is generated randomly as follows:

$$X_{ij}^{New} = X_j^{min} + (X_j^{max} - X_j^{min}) \times rand(0,1) \qquad (j = 1,2,..., N; i = 1,2,...H_{MS}) \tag{2.9}$$

where, $X_j^{\min}$ and $X_j^{\max}$ are lower and upper bounds for decision or design variables and rand(0,1) is random variable within the range [0,1].

Fig.2.2: Harmony Memory Consideration

An H_{MCR} value of 0.95 indicates that the HS algorithm will choose the fitness value from previously stored values in the Harmony Memory with a probability of 95% and from the entire feasible range with a probability of 5%. H_{MCR} value of 1.0 is not suggested because of the possibility that the solution may be improved by values not stored in the HM i.e. it may be outside the stored values.

The eqns.(2.8) and (2.9) more succinctly mathematically represented as:

$$X_{ij}^{New} = \begin{cases} X_{kj} \quad (k = \text{int}[1 + rand() \times (H_{MS} - 1)]; & r_1 \le H_{MCR} \\ X_{ij}^{\min} + rand() \times (X_j^{\max} - X_j^{\min}) & ; \quad r_1 > H_{MCR} \end{cases} \qquad (j = 1, 2, ..., N; i = 1, 2, ..., H_{MS}) \tag{2.10}$$

If the newly generated vector X_{ij}^{New} of eqn. (2.10) is out of upper and lower boundary limits, it can be modified using eqn.(2.11) as follows:

$$X_{ij}^{New} = \begin{cases} X_j^{\min}; & X_{ij}^{New} < X_j^{\min} \\ X_j^{\max}; & X_{ij}^{New} > X_j^{\max} \\ X_{ij}^{New}; & X_j^{\min} \le X_{ij}^{New} \le X_j^{\max} \end{cases} \qquad (j = 1, 2, ..., N; i = 1, 2, ..., NP) \tag{2.11}$$

Step-III: *Pitch Adjusting Rate*

The obtained decision variables from the harmony memory consideration rule are further examined to determine if these need pitch adjustment or not (Fig.2.3), which is mathematically calculated as:

$$X_{ij}^{New} = \begin{cases} X_{ij} \pm rand(0,1) \times bw; & rand(0,1) \le P_{AR} \; and \quad rand(0,1) \le H_{MCR} \\ X_{ij} & ; otherwise \end{cases}$$

$$(j = 1, 2, ..., N; i = 1, 2, ..., H_{MS}) \qquad (2.12)$$

where, P_{AR} is pitch adjustment rate, whose value is selected within the range [0,1] and *bw* is a bandwidth factor, which is used to control the local search around the selected decision variable in the new vector and rand(0,1) is random number generated within the range [0,1].

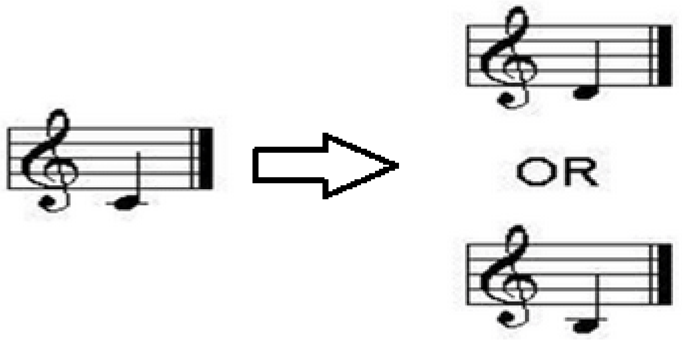


Fig.2.3: Pitch Adjustment Rate

Step-V: *Update the Harmony Memory*

After the harmony vector X_{ij}^{New} is generated, it will replace the worst harmony vector X_{ij}^{Worst} in the harmony memory if its objective function value is better than the objective function value of the worst harmony vector using eqn.(2.13). PSEUDO code for up gradation of worst harmony vector (X_{ij}^{Worst}) with new random harmony vector (X_{ij}^{New}) is mentioned below (Fig.2.4a) and its pictorial representation is described in Fig.2.4b:

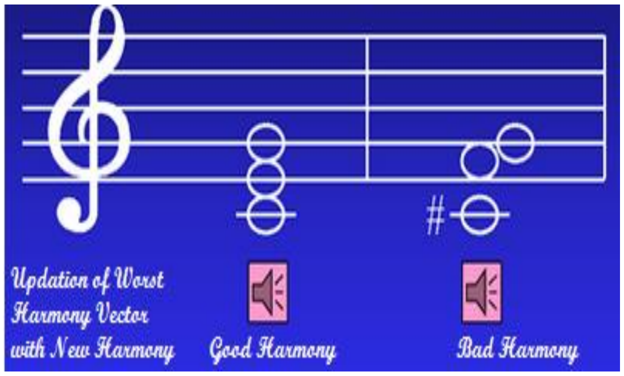
$$X_{ij}^{Worst} = \begin{cases} X_{ij}^{New}; & f(X_{ij}^{New} < f(X_{ij}^{Worst}) \\ X_{ij}^{Worst}; & otherwise \end{cases} \quad (j = 1, 2, ..., N; i = 1, 2, ... H_{MS}) \qquad (2.13)$$

The best fitness value f^{best} is evaluated from obtained solutions using eqn.(2.14) described below:

$$f^{best} = \min\{f_i \mid i \in 1, 2, ..., H_{MS}\} \qquad (2.14)$$

```
Do i=1 to $H_{MS}$
If $(f_i(X_{ij}^{New}) < f_i(X_{ij}^{Worst}))$ then
Update the HM using eqn.(2.13) and (2.14)
end if
end DO
```

(a) PSEUDO Code



(b) Pictorial Representation

Fig.2.4: Updation of Worst Harmony with Best Harmony

Step-VI: Verification of Stopping Criterion

Set the iteration number for *iter=iter+1*. Repeat the harmony memory improvisation process until the stopping criterion for maximum number of iteration, $iter_{max}$ is met.

2.3.2 Differential Evolution Algorithm

Differential Evolution (DE) is a population based stochastic function minimizer (or maximizer) relating to evolutionary computation, whose simple yet powerful and straightforward features makes it very attractive for numerical optimization [340]. Differential evolution uses a rather greedy and less stochastic approach to problem solving than do evolutionary algorithms. Differential evolution combines simple arithmetic operators with the classical operators of recombination, mutation, and selection to evolve from a randomly generated starting population to a final solution. The differential evolution algorithm was first introduced by Stron and Price [115] and was successfully applied to nonlinear, non-differentiable, and non-convex optimization functions by Storn [139]. The differential evolution is a population based algorithm, which is similar genetic algorithm, as it uses the similar operators of crossover, mutation and selection. In differential evolution algorithm, the initial population is generated randomly and then evaluated. After mutation and crossover operations, the selection process takes place. DE generates a single offspring (instead of two like in the genetic algorithm) by adding the weighted difference vector between two parents to a third parent. If the resulting vector produces a lower objective function value than a predetermined population member, the newly generated vector replaces the vector to which it was compared. The steps of the differential evolution algorithm are described below (Fig.2.5):

```
Initialization
Evaluation
Repeat
    Apply Mutation
    Apply Crossover
    Evaluation of Fitness
    Apply Selection
Until (Termination criteria for maximum iterations are met)
```

Fig.2.5: Classical Differential Evolution algorithm

DE has three essential control parameters which are: the mutation factor (f_m), the crossover constant (C_R) and the population size (*NP*). The population size determines the number of individuals in the population and provides enough diversity to the algorithm to search the solution space. The termination criteria are the conditions under which the search process will stop. The algorithm can be terminated depending on predetermined maximum number of iterations $(iter_{\max})$ or whenever the global best solution does not improve over a predetermined number of generations. The procedural steps for applying Differential Evolution algorithm are given below:

Step-I: *Initialization*

Typically, each decision parameter in every vector of the initial population is assigned a randomly chosen value from within its corresponding feasible bounds [332]:

$$X_{ij}^{iter} = X_i^{\min} + rand(0,1)\left(X_i^{\max} - X_i^{\min}\right) \qquad (j = 1,2,...,N; \quad i = 1,2,...,NP) \qquad (2.15)$$

where, X_{ij}^{iter} represents the i^{th} decision parameter of j^{th} population and superscript *iter* represents initial generation; $rand(0,1)$ denotes a uniformly distributed random number within the range [0,1], generated for each value of i ; N is the number of decision variables. NP is the number of population, $X_i^{\min}$ and $X_i^{\max}$ is respectively the lower and upper bound on the i^{th} decision variable.

Step-II: *Mutation*

The mutation operator creates mutant vectors Z_{ij}^{iter} by perturbing a randomly selected vector $X_{R_1 j}$ with the difference of two other randomly selected vectors $X_{R_2 j}$ and $X_{R_3 j}$, according to the following equation [340]:

$$Z_{ij}^{iter} = X_{R_1 j}^{iter} + f_m \times (X_{R_2 j}^{iter} - X_{R_3 j}^{iter}) \qquad (j = 1, 2, ..., N; i = 1, 2, ..., NP) \qquad (2.16)$$

where, $X_{R_1 j}$, $X_{R_2 j}$ and $X_{R_3 j}$ are randomly chosen vectors among the NP population, and $R_\partial \neq R_2 \neq R_3$. $X_{R_2 j}$ and $X_{R_3 j}$ are selected for each parent vector. The mutation factor f_m is an algorithm control parameter used to adjust the perturbation size in the mutation operator and to improve algorithm convergence. The mutation factor is a value in the range [0, 2].

Step-III: *Crossover*

The crossover operation generates trial vectors U_{ij}^{iter} by mixing the parameters of the mutant vectors Z_{ij}^{iter} with its target or parent vectors X_{ij}^{iter} using following mathematical relation:

$$U_{ij}^{iter} = \begin{cases} X_{ij}^{iter}; & if \quad R_4(j) \leq C_R \quad or \quad (j = R_5(i)) \\ Z_{ij}^{iter}; & if \quad R_4(j) > C_R \quad or \quad (j \neq R_5(i)) \end{cases} \qquad (j = 1, 2, ..., N; i = 1, 2, ..., NP)$$

(2.17)

where, $R_4(j) \in \{1, 2, ..., N\}$; $R_4(j)$ is the jth evaluation of a uniform random number generation within [0,1]; C_R is the crossover or recombination rate in the range [0,1], which is differential evolution algorithm parameter that controls the diversity of the population and aids the algorithm to escape from local minima.

Step-IV: *Selection*

The selection operator forms the new population by choosing between the trial vectors and their predecessors (parent vectors) those individuals that present a better fitness or are more optimal according to following equation:

$$X_{ij}^{iter+1} = \begin{cases} U_{ij}^{iter+1} & (j = 1, 2, ..., N); \quad if \quad f(U_i^{iter+1}) \leq f(X_i^{iter}) \\ X_{ij}^{iter} & (j = 1, 2, ..., N); \quad otherwise \end{cases} \qquad (i = 1, 2, 3, ..., NP)$$

(2.18)

In this case, the cost of each trail vector U_{ij}^{iter+1} is compared with that of its parent target vector X_i^{iter}. If the cost f, of the target vector X_i^{iter}, is lower than that of the trial vector, the target is allowed to advance to the next generation. Otherwise, a trail vector replaces the target vector in the next generation as per eqn.(2.19).

$$f_i^{iter} = f(X_{ij}^{iter}) \qquad (i = 1, 2, ..., NP) \tag{2.19a}$$

$$f^{best} = \min\{ f_i^{iter} \mid i = 1, 2, ..., NP \} \tag{2.19b}$$

$$X_j^{best} = \begin{cases} X_{ij}^{iter+1}; & f_i < f^{best} \\ X_j^{best}; & otherwise \end{cases} \tag{2.19c}$$

This optimization process is repeated for several generations, allowing individuals to improve their fitness as they explore the solution space in search of optimal values.

Step-V: *Verification of the stopping criterion*

Set the iteration number for *iter=iter+1*. Repeat mutation, recombination and selection operation until stopping criterion of maximum iteration $iter_{\max}$ is met.

2.3.3 Random Search Algorithm

Random search is a direct search method, which does not require derivatives to search a continuous domain and belongs to the fields of stochastic and global optimization. The strategy of random search is to sample solutions across the entire search space using a uniform probability distribution, in which each future trial is independent of the samples that come before it [28]. Random search can return a reasonable approximation of the optimal solution within a reasonable time under low problem dimensionality, although the approach does not scale well with problem size (i.e. the number of dimensions). The results of a random search can be used to seed another search technique, like a local search technique that can be used to locate the best solution in the neighborhood of the 'good' candidate solution. This method generates trial solutions for the optimization model using random number generations for the decision variables. Random search algorithms are also frequently applied to stochastic optimization problems, where the objective function and constraints involve randomness [331]. The procedural steps of random search algorithm are described below:

Algorithm

Let $f(X)$ is the fitness function to be minimized and $X_{ij}^{iter} \in X_{ij}$ is a solution in the search-space (NP). The steps of random search algorithm is are described as:

Step-I*: Initialization*

Set the iteration counter, *iter*=0 and Initialize X_{ij}^{iter} with a random position in the search-space using eqn.(2.20) and evaluate $f_i = f(X_{ij}^{iter})$ for each iteration.

$$X_{ij}^{iter} = X_j^{\min} + rand() \times (X_j^{\max} - X_j^{\min}) \qquad (j = 1, 2, ..., N; i = 1, 2, ..., NP) \tag{2.20}$$

Step-II: *Updating variables*

Sample a new position Z_{ij}^{iter} by adding a normally distributed random vector, $N(0,1)$ to the current position X_{ij}^{iter} using eqn.(2.21) and evaluate $f_i = f(Z_{ij}^{iter})$ for each iteration.

$$Z_{ij}^{iter} = X_{ij}^{iter} + N(0,1) \tag{2.21}$$

Step-III: *Selection*

If $((f(Z_{ij}^{iter}) < f(X_{ij}^{iter}))$, then move to the new position by setting $X_{ij}^{iter} = Z_{ij}^{iter}$.

Step-IV: *Find best Solution*

Replace X_{ij}^{iter} with the best-found position using eqn.(2.22).

$$X_j^{best} = \begin{cases} X_{ij}^{iter}; & f(X_{ij}^{iter}) < f^{best} \\ X_j^{best}; & otherwise \end{cases} \quad (j = 1, 2, ..., N) \text{ `} \tag{2.22}$$

Step-V: *Stopping criterion*

Repeat the Step-II to IV, until the termination criterion for maximum number of iteration *iter*$_{max}$ is satisfied.

The algorithm corresponds to a (*iter*+1) evolution strategy with constant step-size.

2.3.4 Integrated Differential Evolution-Harmony Search Algorithm

Harmony search (HS) is a recently proposed meta-heuristics algorithm by imitating music improvisation process, which has drawn much attention in the past few years. However, researches have revealed that the performance and the convergence rate of the method are suffered when dealing with high-dimensional or/and multimodal problems. To get a better control over exploitation and exploration, a hybrid version of Harmony

Search algorithm is proposed [405], which is controlled in two ways. Firstly, the memory consideration rule is modified by introducing the crossover and mutation operators of the Differential Evolution (DE) algorithm. Second, two control parameters, namely P_{AR} and *bw*, are either dynamically adjusted or are made self-learning along with the evolutionary process to fine-tune the solutions. To improve the performance of harmony search algorithm, a hybrid DE-HS algorithm is presented in [404].

Differential Evolution is a population-based stochastic search evolutionary algorithm developed by Storn [115, 139], applicable for non-linear, non-differentiable and non-convex optimization problem, whose simple yet powerful and straightforward features make it very attractive for numerical optimization. Differential evolution uses a rather greedy and less stochastic approach to problem solving than other evolutionary algorithms. DE combines simple arithmetic operators with the classical operators of recombination, mutation and selection to evolve from a randomly generated starting population to a final solution. DE algorithm is fast and simple with regard to application and requires few control parameters, having parallel processing nature and faster convergence, capable of providing multiple solutions in a single run, effective for integer, discrete and mixed parameter optimization, has the ability to find the optimal solution for a non-linear constrained optimization problem with penalty functions and has the ability to find the true global minimum regardless of the initial parameter values. In differential evolution algorithm, mutation operation is able to enhance the convergence rate and search speed, crossover operation is able to increase diversity of population and prevent trapping of algorithm into local optimum. It is intended to improve performance of HS algorithm and its ability for global search. In proposed integrated DE-HS algorithm, two global search algorithms i.e. Differential evolution and harmony search are integrated together to enhance the performance by acquiring their good qualities as described in flow chart of Fig.2.6.

2.3.5 Hybrid Harmony and Random Search algorithm

Harmony Search (HS) is a population based meta-heuristics search algorithm inspired from the musical process of searching for a perfect state of harmony. The pitch of each musical instrument determines the aesthetic quality, just as the fitness function value determines the quality of decision variables. In the musical improvisation process, all players sound pitches within possible range together to make one harmony [185, 404, 405, 412]. If all the pitches make a good harmony, each player stores in his memory that experience and the possibility of making a good harmony is increased next time. Even though, HS has the ability to escape from local minima, does not require differential gradients and initial value setting for the variables and free from divergence and has strong ability to explore the regions of solution space in a reasonable time, but it has lower exploitation ability in later period and it easily gets trapped into local optima and converges very slowly. To improve the exploitation ability of HS algorithm in later stage and provide global optimal solution, a novel and hybrid version of Harmony search combined with Random search algorithm i.e. Hybrid HS-random search algorithm is implemented in the subsequent section to solve single and multi-area unit commitment problems under single and multi-objective framework.

In the proposed Hybrid HS-RS algorithm, Random search algorithm is hybridized with harmony search algorithm for improvement of local search capability of Harmony search algorithm. In order to obtain the hybrid harmony search-random search (HS-RS) algorithm, the general operators of harmony search algorithm and random search algorithm are integrated recursively.

In proposed hybrid HS-random search algorithm, firstly the harmony memory is initialized with random vectors and local fitness is evaluated within local search space using random search algorithm. After evaluating local fitness value, harmony memory is updated using the Harmony search algorithm to determine global fitness. The flowchart of proposed hybrid harmony search-random search algorithm is shown in Fig.2.7.

2.3.6 Hybrid Differential Evolution and Random Search algorithm

Differential Evolution (DE) is a population-based stochastic search algorithm, whose simple yet powerful and straightforward features make it very attractive for numerical optimization. Differential evolution uses a rather greedy and less stochastic approach to problem solving than other evolutionary algorithms. DE combines simple arithmetic operators with the classical operators of recombination, mutation and selection to evolve from a randomly generated starting population to a final solution.

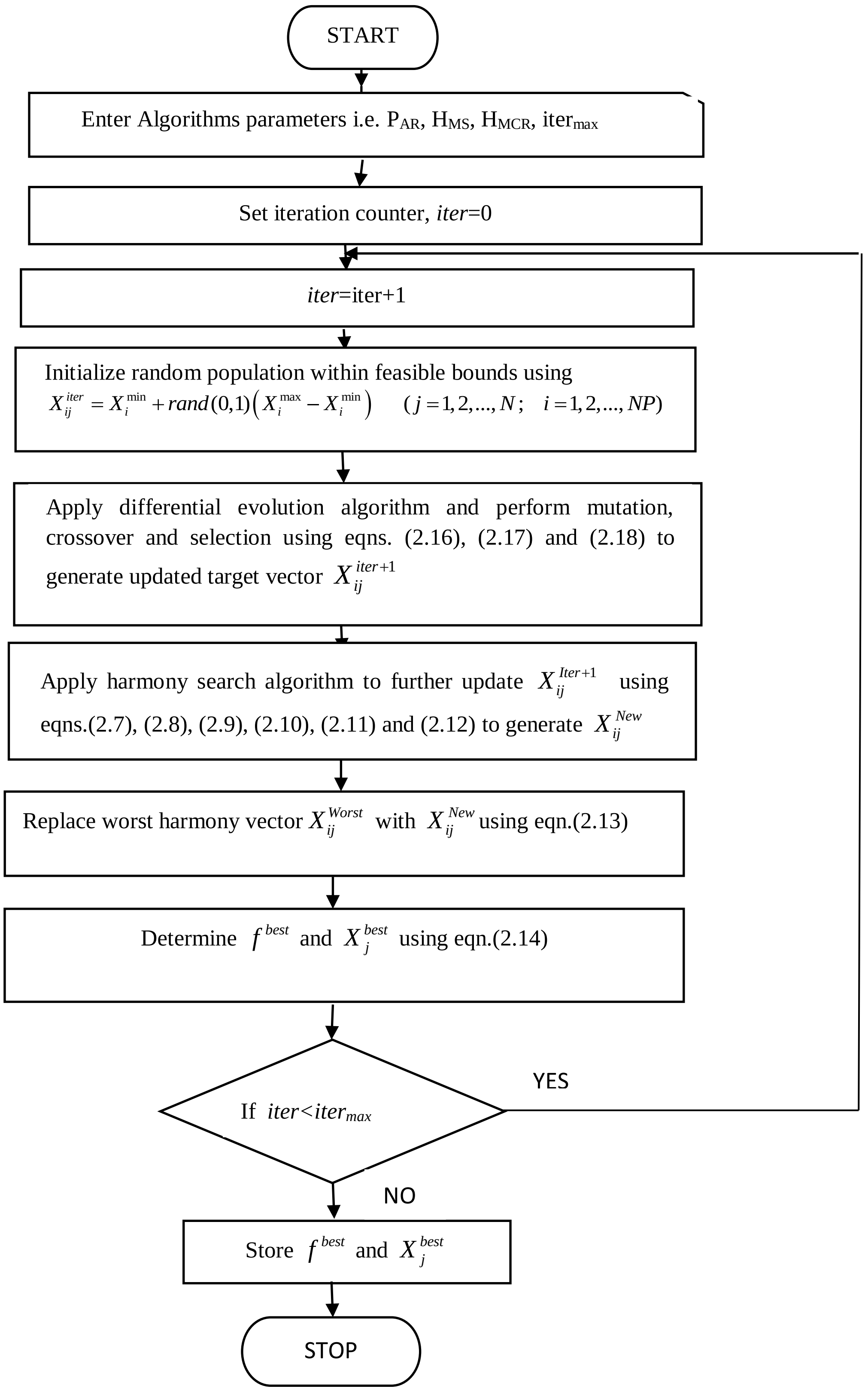

START
Enter Algorithms parameters i.e. P_{AR}, H_{MS}, H_{MCR}, $iter_{max}$
Set iteration counter, $iter=0$
$iter=iter+1$
Initialize random population within feasible bounds using $X_{ij}^{iter} = X_i^{\min} + rand(0,1)\left(X_i^{\max} - X_i^{\min}\right) \quad (j=1,2,...,N; \quad i=1,2,...,NP)$
Apply differential evolution algorithm and perform mutation, crossover and selection using eqns. (2.16), (2.17) and (2.18) to generate updated target vector X_{ij}^{iter+1}
Apply harmony search algorithm to further update X_{ij}^{Iter+1} using eqns.(2.7), (2.8), (2.9), (2.10), (2.11) and (2.12) to generate X_{ij}^{New}
Replace worst harmony vector X_{ij}^{Worst} with X_{ij}^{New} using eqn.(2.13)
Determine f^{best} and X_j^{best} using eqn.(2.14)
If $iter<iter_{max}$
YES
NO
Store f^{best} and X_j^{best}
STOP

Fig.2.6: Flow Chart of proposed integrated DE-HS algorithm

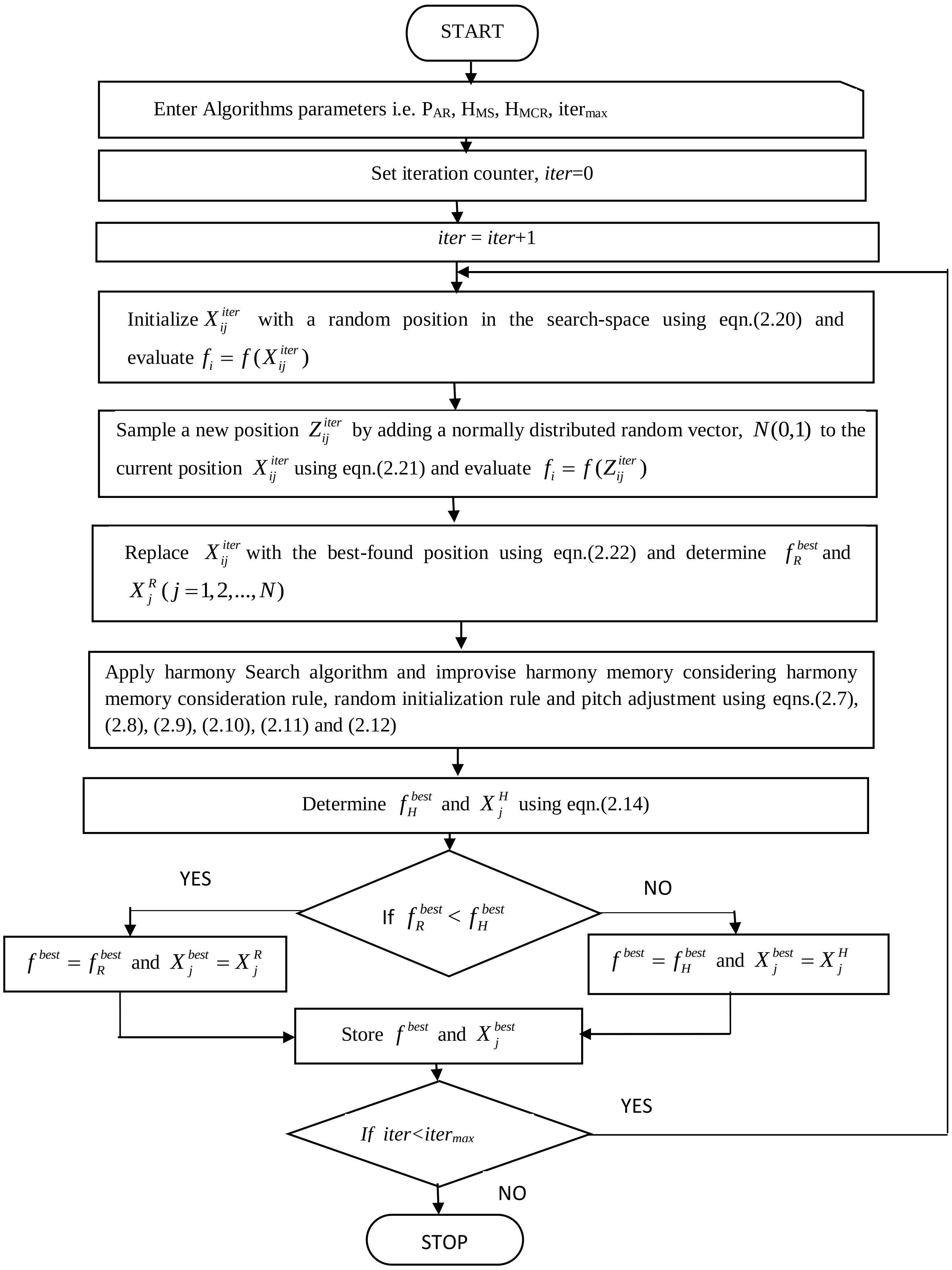

START
Enter Algorithms parameters i.e. P_{AR}, H_{MS}, H_{MCR}, $iter_{max}$
Set iteration counter, $iter=0$
$iter = iter+1$
Initialize X_{ij}^{iter} with a random position in the search-space using eqn.(2.20) and evaluate $f_i = f(X_{ij}^{iter})$
Sample a new position Z_{ij}^{iter} by adding a normally distributed random vector, $N(0,1)$ to the current position X_{ij}^{iter} using eqn.(2.21) and evaluate $f_i = f(Z_{ij}^{iter})$
Replace X_{ij}^{iter} with the best-found position using eqn.(2.22) and determine f_R^{best} and $X_j^R\ (j=1,2,...,N)$
Apply harmony Search algorithm and improvise harmony memory considering harmony memory consideration rule, random initialization rule and pitch adjustment using eqns.(2.7), (2.8), (2.9), (2.10), (2.11) and (2.12)
Determine f_H^{best} and X_j^H using eqn.(2.14)
If $f_R^{best} < f_H^{best}$
YES
NO
$f^{best} = f_R^{best}$ and $X_j^{best} = X_j^R$
$f^{best} = f_H^{best}$ and $X_j^{best} = X_j^H$
Store f^{best} and X_j^{best}
If $iter<iter_{max}$
YES
NO
STOP

Fig.2.7: Flowchart for hybrid HS-RS algorithm

Although, global exploration ability of differential evolution algorithm is adequate, but its local exploitation ability is feeble and convergence velocity is too low and it suffers from the problem of untimely convergence for multimodal objective function, in which search process may be trapped in local optima and it loses its diversity. Also, it suffers from the stagnation problem, where the search process may infrequently stop proceeding toward the global optimum even though the population has not converged to a local optimum or any other point.

Differential evolution algorithm, proposed by Storn and Price [115, 139] has been proved to be more robust, faster in convergence and more efficient than conventional evolutionary algorithms (EA) [139,115]. DE differs from other evolutionary algorithms in the mutation and recombination phases, as it uses weighted differences between solution vectors to change the population. However, in other stochastic techniques perturbation occurs in accordance with a random quantity. DE employs a greedy selection process with intrinsic discriminatory features [372, 373]. In order to attempt multimodal problems with a high number of global and/or local optima, different variants of DE algorithms are used. Thomsen [220] have combined classical deterministic crowding and fitness sharing technique with standard differential algorithm and developed two well-known algorithms known as Crowding DE (CDE) and Sharing DE. Li [912] has applied the concept of speciation to DE's structure, resulting in the widely used Species-based DE (SDE) [235, 347]. Moreover, DE with local selection designs a mutation strategy with two main components: a local and a global mutation rule. Throughout the evolutionary process, the two rules are probabilistically selected by a fixed and pre-specified probability. As expected, the global mutation rule is responsible for investigating unexplored regions of the search space, whilst the local mutation rule contributes towards its exploitative ability. The algorithm has been further hybridized with several mechanisms such as the crowding technique and a specialized multi-start local search procedure [346]. A parallel implementation of DE has been proposed by Zaharie to address multimodal problems [221].

Standard variant of DE algorithm was modified to solve combinatorial optimization problems, which is known as binary differential evolution (BDE) [338], in which sigmoid function is used to map the continuous space to binary search space.
A binary version of DE algorithm was also proposed to solve 0-1 Knapsack problem [295]. The use of ideas from Quantum Computing has been proposed to make the algorithms dramatically faster and extremely parallel. These have been shown to produce superior performance compared to their classical counterparts [128, 130].

Also, an adaptive quantum inspired differential evolution (AQDE) algorithm was proposed by Hota and Pat [374] for solving the 0-1 knapsack problem. An elitist quantum inspired differential evolution (QDE) algorithm is proposed to obtain an optimal subset of features for classification tasks using logistic regression for fitness function evaluation [421]. An effective hybrid differential evolution (HDE) was proposed for no-wait flow shop scheduling problem (FSSP) with the makespan criterion [302]. A hybrid algorithm based on differential evolution (HDE) was proposed to solve multi-objective permutation flow shop scheduling problem (MPFSSP) with limited buffers between consecutive machines [302]. An efficient local search, which was designed based on the landscape of MPFSSP with limited buffers combined with DE was applied to emphasize exploitation. The concept of Pareto dominance is used to handle the updating of solutions in sense of multi-objective optimization. The convergence property of HDE is analyzed by using the theory of finite Markov chain. The proposed HDE demonstrate the effectiveness and efficiency.

A self-adaptive mechanism was proposed to improve the performance of NSDE, SaDE and SaNSDE [294]. The algorithm SaNSDE showed the superiority over SaDE and NSDE. A differential evolution based on a variable neighborhood search algorithm (DE-VNS) is proposed in order to solve the constrained real-parameter optimization problems. The computational results show that the simple DE-VS algorithm was very competitive to some of the best performing algorithms from the literature [419]. A differential evolution algorithm with a variable neighborhood search was also proposed to solve the multidimensional knapsack problem. Computational results show its efficient in solving benchmark instances and its superiority to the best performing algorithms from the literature [420]. The above literature reveals that, differential evolution algorithm has the ability to find the true global minimum regardless of the initial parameters values and requires few control parameters. It has parallel processing nature and fast convergence as compared to conventional optimization algorithm. It always does not give an exact global optimum solution due to premature convergence and requires tremendously high computation time because of a large number of fitness evaluations. Random search algorithm is derivative free method for continuous domain, which is based on direct search and most suitable for stochastic and global optimization problem. To overcome the limitation of conventional DE, a hybrid combination of differential evolution (global search algorithm) and random search algorithm (local search algorithm) is presented in the proposed research and named as hybrid differential evolution-random search (DE-RS) algorithm.

In proposed hybrid DE-random search algorithm, firstly the trial vector is initialized randomly and local fitness is evaluated within local search space using random search algorithm. After evaluating local fitness value, the process of mutation, crossover and selection is applied to update the trial vector to determine global fitness. The flow chart for proposed hybrid differential evolution-random search algorithm is shown in Fig.2.8.

2.4 MULTI-OBJECTIVE OPTIMIZATION

Optimization problems are widely found in many domains of scientific research and engineering applications. According to the number of objectives to be optimized, they can be classified into two main categories, i.e., single-objective optimization problems (SOPs) and multi-objective optimization problems (MOPs). Compared to SOPs that aim to find a global optimal value, MOPs bring more challenges as they try to optimize several (often conflicting) objectives simultaneously. Due to the complex landscape in decision and objective space of MOPs, traditional deterministic approaches are not suitable for tackling MOPs as they can’t find a satisfactory result within a limited time. Therefore, evolutionary algorithms (EAs) are proposed to solve MOPs and they are experimentally validated to have the excellent global search capability when locating the optimal solution set.

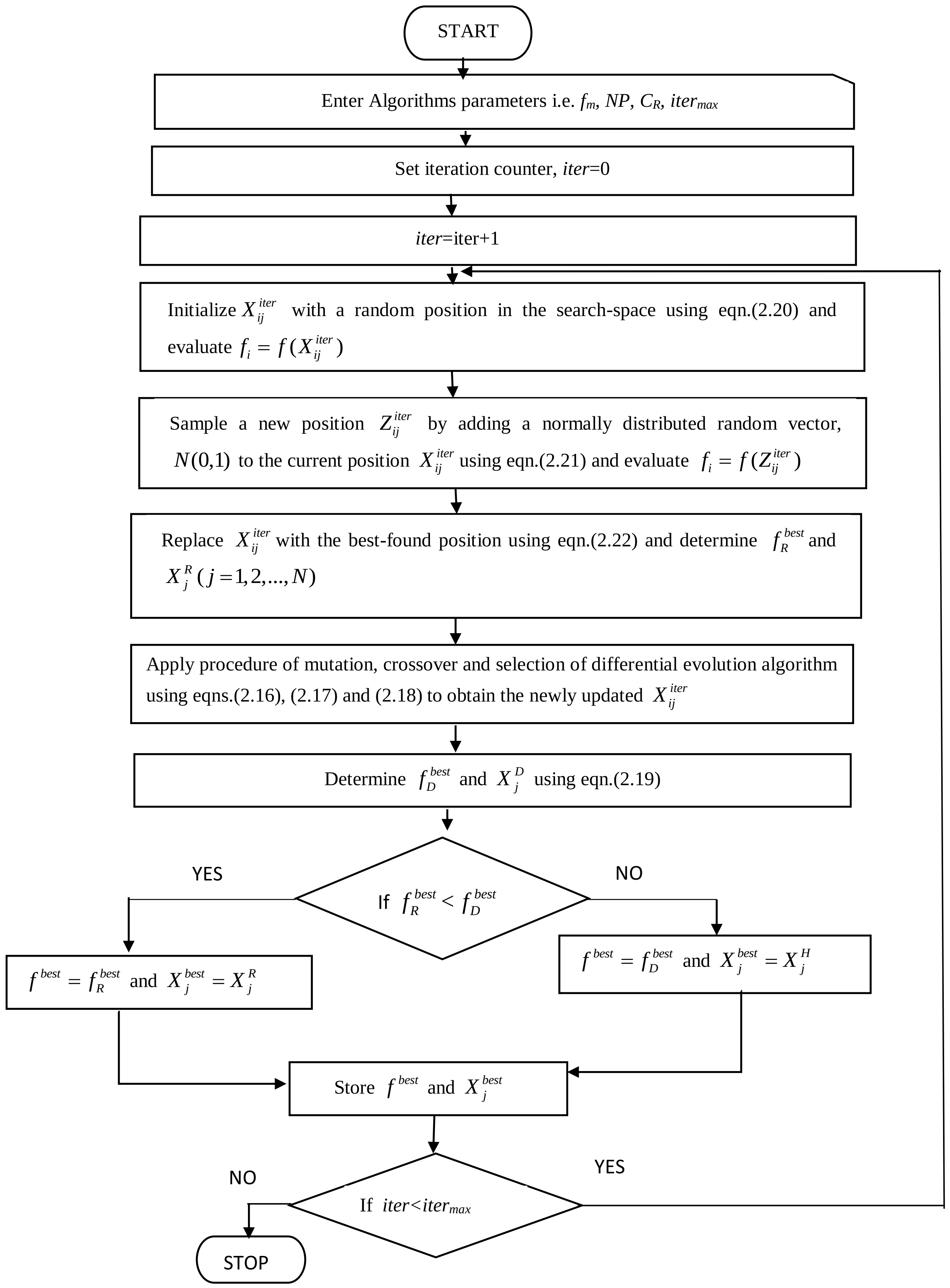

START
Enter Algorithms parameters i.e. f_m, NP, C_R, $iter_{max}$
Set iteration counter, $iter$=0
$iter$=iter+1
Initialize X_{ij}^{iter} with a random position in the search-space using eqn.(2.20) and evaluate $f_i = f(X_{ij}^{iter})$
Sample a new position Z_{ij}^{iter} by adding a normally distributed random vector, $N(0,1)$ to the current position X_{ij}^{iter} using eqn.(2.21) and evaluate $f_i = f(Z_{ij}^{iter})$
Replace X_{ij}^{iter} with the best-found position using eqn.(2.22) and determine f_R^{best} and $X_j^R\ (j=1,2,...,N)$
Apply procedure of mutation, crossover and selection of differential evolution algorithm using eqns.(2.16), (2.17) and (2.18) to obtain the newly updated X_{ij}^{iter}
Determine f_D^{best} and X_j^D using eqn.(2.19)
YES
If $f_R^{best} < f_D^{best}$
NO
$f^{best} = f_R^{best}$ and $X_j^{best} = X_j^R$
$f^{best} = f_D^{best}$ and $X_j^{best} = X_j^H$
Store f^{best} and X_j^{best}
NO
If $iter$<$iter_{max}$
YES
STOP

Fig.2.8: Flowchart for proposed hybrid DE-RS algorithm

Moreover, the ability to handle complex MOPs that are characterized with discontinuities, multimodality, disjoint feasible space and noisy function evaluations, reinforces the potential effectiveness of multi-objective EAs (MOEAs). One of the inherent characteristics associated with real world decision-making problems is their inescapably manifold nature. One of the multifarious features of such problems is multiple objectives that are usually non-commensurable and are often in conflict. Real-world decision making problems, thus, often lead to a multi-objective optimization problem formulation. The ultimate goal in such problems is to seek a most preferred solution among the set of non-inferior solutions. Interactive methods are desirable in certain decision situations where little a priori knowledge and experience are known about a decision problem in hand. The interactive techniques allow the solution to progress towards a preferred solution through an adaptive process during which the decision maker's preferences are elicited progressively. Among the very different forms of local preference information, explicit trade-offs between different objectives functions are widely used in different approaches. Geoffrion's method [14] is a pioneer approach in explicit tradeoff analysis, where marginal rates of substitution are used to facilitate interactive trade-off analysis. Trade-off step size is chosen using a trade-off table constructed inside the feasible solution space. Geoffrion's method is limited to convex problems and also little guidance is given to support the elicitation of the local preferences and no direct relationship between the elicited preference and the termination criteria is provided in interactive trade-off process. Consequently, the interactive process may be unexpectedly terminated when the decision maker is not aware that why this happened and what this meant. Yang and Li [199] have presented a new explicit interactive trade-off analysis method based on the identification of normal vector on a non-inferior frontier.

In recent years there has been an increase in research on multi-objective optimization methods, also called multi-performance, multi-criterion or vector optimization. The situation is formulated as a multi-objective optimization problem in which the engineer's goal is to maximize or minimize not a single objective function but several objective functions simultaneously. In various practical applications, many MOPs are encountered that need to handle constraints and optimize multiple (often conflicting) objectives simultaneously. Without loss of generality, the mathematical description of MOPs for minimization can be expressed as follows:

Minimize $F(X) = [f_1(X), f_2(X), ..., f_m(X)]^T$ (2.23)

Such that: $g_i(X) \leq 0 \quad (i = 1, 2, ..., q)$ (2.24)

$h_j(X) = 0 \quad (j = 1, 2, 3, ..., p)$ (2.25)

where, $X = [x_1, x_2, ..., x_n]^T \in \Omega$ is a decision vector with *n* dimensions; Ω is the decision space, *m* is the number of objectives; $g_i(x)$ (*i* =1,2,...,*q*) are *q* inequality constraints and $h_j(X)$ (*j*=1,2,..., *p*) are *p* equality constraints.

The goal of MOPs is to minimize all the objective functions in eqn. (2.23). The concepts of Pareto optimum theory are important for MOPs to find out the trade-off solutions, as described below.

Definition 1. (Pareto-dominance): Suppose that there are two vectors such as: $\vec{x} = (x_1, x_2, x_3, ..., x_k)$ and $\vec{y} = (y_1, y_2, y_3, ..., y_k)$. Vector x dominates vector y (denoted as $x \succ y$) iff $\forall i \in \{1, 2, 3, ..., k\}, [f(y_i) \geq f(x_i)] \wedge [\exists i \in 1, 2, 3, ..., k : f(x_i)]$ (2.26)

Definition 2 (Pareto-optimal): A solution $\vec{x} \in X$ is said to be Pareto-optimal if and only if there does not exist any vector $\vec{y}$ belong to X such that $F(\vec{y})$ dominates $F(\vec{x})$ and mathematically may be represented as:

$\nexists \ \vec{y} \in X \mid f(\vec{y}) \succ f(\vec{x})$ (2.27)

Definition 3 (Pareto-optimal set): The set of all Pareto-optimal solutions is called Pareto-optimal set.

This set (P$_s$) includes all the Pareto-optimal solutions and mathematically defined as:

$P_s := \{x, y \in X \mid \exists F(y) \succ F(x)\}$ (2.28)

Definition 4 (Pareto-optimal front): A set containing the corresponding objective values of Pareto optimal solutions in Pareto optimal set is called Pareto optimal front. The set P_f includes the value of all the objective functions corresponding to the Pareto-optimal solutions in P_S and mathematically described as:

$P_f := \{F(x) \mid x \in P_s\}$ (2.29)

It is usually not possible to simultaneously maximize/minimize all the objectives, because generally the objectives are in conflict with each other and are incommensurable. Instead of a single optimum solution, there are several solutions, called efficient solutions that have the property, that no improvement in any objective is possible without sacrificing one or more of the other objectives. The set of all efficient solutions in the continuous case is known as the efficient frontier. An efficient solution is also called a non-dominated solution, non-inferior solution or pareto-optimal solution. The final solution out of these non-dominated solutions is selected according to DM’s criteria. It is called the compromised solution and is defined as an efficient solution that maximizes the DM’s preference function.

2.4.1 Generation of Non-inferior Solutions

Methodologies used for solving multi-objective problems, principally differ in two ways, the procedure used to generate non-inferior solutions and the ways and means used to interact with the DM and the type of information made available to the DM such as trade-offs. The weighting and constraint methods identify the non-inferior set, within which the best compromised solution lies. In almost all decision-making problems, there are several criteria for judging the possible alternatives. The main concern of decision maker is to fulfill the conflicting goals while satisfying the constraints of the system. There are two different approaches to solve such problems. One approach assumes that there exists a utility function for the particular problem. The other approach makes no assumption regarding the existence of any utility function, but provides the DM with a set of simple but effective tools to obtain the best alternative. A few methods for generating efficient points or non-inferior solutions are reviewed in the following sub sections:

2.4.1.1 Weighting method

The weighting method, also known as the parametric approach, is the most common method used for solving multi-objective problems until recently. Multi-objective problem is converted into scalar optimization problem as given below:

$$\text{Minimize } \sum_{i=1}^{N_{ob}} w_i f_i(x) \tag{2.30}$$

Subject to $x \in X$

$$\sum_{i=1}^{N_{ob}} w_i = 1 \quad ; w_i \geq 0 \qquad (i = 1, 2, ..., N_{ob}) \tag{2.31}$$

Where, w_i is the ith weighting coefficient. The approach yields meaningful results to the decision maker only when solved many times for different values of w_i $(i = 1, 2, ..., N_{ob})$, which are chosen on the basis of DMs intuition. N_{ob} is the number of objectives. By varying these weighting coefficients, different points are located in the non-inferior set. Till now very little is usually known about how to choose weighting coefficients. Still, the engineers choose them, presumably on the basis of their intuition.

2.4.1.2 Weighted min-max method

In this method the following minimization problem is solved:

$$\min_{x}[\max_{1 \leq i \leq N_{ob}} w_i f_i(x)] \tag{2.32}$$

Subject to $x \in X$

$$\sum_{i=1}^{N_{ob}} w_i = 1 \quad ; w_i \geq 0 \qquad (i = 1, 2, ..., N_{ob}) \tag{2.33}$$

A systematic variation of w_i's will generate the non-inferior solutions. The solution obtained in this way is a local efficient point.

2.4.2 Decision Making

Decisions with multiple objectives are quite prevalent in government, military, industry and other organizations. Researchers from a wide variety of disciplines such as mathematics, management, science, economics, engineering and others have contributed to the solution methods for *multiple objective optimization problems* (MOOPs). Due to the conflicting nature of the objectives, an optimal solution that simultaneously maximizes all the criteria is usually not obtainable. Instead there are several solutions, called efficient solutions that have the property that no improvement in one objective is possible without sacrificing on one or more of the other objectives. The solution methods developed in MOOPs can be categorized by the basic assumptions made with respect to preference function: (i) when complete information of the preference function is available from the *decision maker* (DM) ; (ii) when no information is available; and (iii) when partial information is obtainable *progressively* from the DM. In the first approach, the DM's preference function is assessed or the DM's aspirations are disclosed, even before the MOOP is solved. In the second approach, attention is focused on the generation of all efficient solutions, and the set of efficient solutions is either partially or completely enumerated and presented to the DM. The third approach does not require a priori preference information; instead it elicits the DM's preference structure through man-machine interactions. In the interactive method, the DM provides preference information about the current solution either implicitly or explicitly. Since the DM is involved in the entire solution process, interactive methods are more accepted in practice. When interactive algorithms are applied to real world problems, the most critical factor is functional restrictions placed on the objective functions, constraints and the unknown preference function.

Multiple criteria decision making (MCDM) is an extremely important discipline that deals with decision-making problems with multiple objectives. The DM is the key player in the MCDM process. The DM is responsible for the selection of the best solution from among all the generated solutions. Several methods have been proposed in recent years to solve multi-objective optimization problems. These methods are broadly congregated under two major titles: *non-interactive* and *interactive*. In the non-interactive method, the global preference function of the objectives is identified and optimized with respect to the constraints. In the interactive method, the decision maker identifies a local preference function or trade-off among objectives and the solution process gradually proceeds towards the globally satisfactory solution.

Several techniques have been applied by different researchers for the solution of optimization problems. Modern optimization methods were pioneered by Caurant's paper on penalty functions [2], and Karush, Kuhn and Tucker [1,3] who derived the *Karush Kuhn Tucker (KKT)* optimality conditions for constrained problems. The use of non-linear optimization techniques in structural design has been pioneered by Schmit [5]. The books on engineering optimization [6, 7, 33, 91] made an impact in educating engineers to apply optimization techniques.

There are a number of excellent literature reviews on multi-objective problems. Cohon and Marks [18] classified the multi-objective approach into generating techniques, which rely on the prior articulation of preferences and techniques, which are faster interactive to the definitions of preferences. The methods in various classes are reviewed and evaluated in terms of the hypothesized criteria by them. In another survey paper, Brayton et. al. [26] has described multi-objective constrained optimization techniques which appropriated in the design of integrated circuits. Shin and Ravindran [80] conducted an excellent survey of the interactive methods developed for solving continuous multiple objective optimization problems and other applications. Most of the methods fall into the following categories: feasible region reduction method, feasible direction methods, criterion weight space methods, trade-off cutting plane methods, Lagrange multiplier methods, visual interactive methods, branch and bound methods, relaxation methods, sequential methods and function scalarizing methods. The methods of each category have been reviewed based on the nature of preference assessments, functional assumption and relationship between methods. Saber and Ravindran [96] outlined a literature of *Non-linear Goal Programming* (NLGP) methods and applications. The methodological articles have been categorized into four major approaches viz. (a) simplex method; (b) direct search; (c) gradient search and (d) interactive approaches; to help practitioners in selecting the proper NLGP method for use. In order to identify areas where NLGP methods can be used, the application articles have been categorized into nine major areas including engineering design. The number of publications in the field of MOOP is too many. So, it is difficult to discuss all of them here due to space constraint, however a few representative ones are being discussed here.

A computational procedure for calculating a set of the so-called non-inferior vectors to an unconstrained optimization problem has been described by Vemuri [15]. The Surrogate Worth trade-off method has been developed by Haimes and Hall [16] for solving non-commensurable multi-objective functions. The trade-off and surrogate worth functions have been constructed in the functional space, where an interaction with the DM has taken place. In another attempt Chankong et. al. [27] have integrated two existing methodologies into a single objective dynamic programming method for capacity expansion and surrogate worth trade-off method for optimizing multiple objectives into a unified scheme. Rarig and Haimes [32] have developed the risk/dispersion index method to find solution to the multi-objective problem that minimizes sensitivity and maximizes overall utility, using the information provided by dispersion index. The index has been interpreted as a first order approximation to the standard deviation in the optimal solution of the non-linear program. Kaunas and Haimes [38] have applied Risk/Dispersion method to solve the groundwater contamination problem.

A MOOP is generally solved for finding the set of all non-inferior solutions to the problem. A man machine interactive approach to solve multiple objective linear programming problems has been elaborated by Quaddus and Holzman [41]. An interactive paired comparison method has been developed by Malakooti [54] using the heuristic approach for finding local adjacent discrete points to approximate the gradient and identifying discrete points for the one-dimensional search. A new method for generating efficient points for the multi-criterion optimization by defining a shifted min-max function has been brought out by Charalambous [53] and has been applied to design 1-D filters.

Most interactive methods are based on the elicitation of the DM's local preferences. Among the very different forms of local preference information, explicit trade-offs between different objective functions are widely used in several approaches [14, 20, 21, 54, 81, 114, 140, 154, 164, 165, 166]. Trade-off rates developed in some typical generating methods have been connected with each other [280]. In a trade-off analysis, the DM could provide local preference information such as indifference trade-offs during the interactive solution process for generating a most preferred solution [16, 23, 59]. Wei et. al. [123] have applied interior point quadratic programming algorithm to solve power system optimization problems. The proposed algorithm can start from either a feasible (interior point) or an infeasible point (non-interior point) and has quadratic convergence. In most, situations, DM's "global" disutility function is not available. Interactive methods are desirable in certain decision situations where a little prior knowledge and experience are known about a decision problem at hand. Recent years have seen an increasing number of papers in literature, reporting applications of interactive methods to optimization problems in engineering design [124, 125, 126, 155, 167]. This growing interest results from the recognition that interactive techniques allow the solution to progress toward a preferred solution through an adaptive process during which the DM’s preferences have been obtained progressively. In this self-learning process, the DM is supported to investigate what is achievable and what should be done to arrive at a most preferred solution [34, 46, 69, 92, 97, 124]. This mirrors the common adaptive process in real-world decision making in engineering and management.

Yang [231] and Yang and Li [195] have implemented the interactive process using weighted mini-max formulation by regulating the relative weights of objectives in a systematic manner. It has been proved under a mild condition that a normal vector can be identified using the weights and Kuhn-Tucker multipliers in the min-max formulation. Utility gradient has been estimated using local preference information such as marginal rates of substitution for estimating trade-off directions and sizes. However, this estimation has been supported through the identification of normal vector on a non-inferior frontier [127]. Leung and Wang [210] have proposed a quality measure called U-measure to measure the uniformity of a given set of non-dominated solutions over the Pareto frontier. This frontier has a non-linear hyper-surface. They measured the uniformity over this hyper-surface in three main steps: (i) determined the domains of the Pareto frontier over which uniformity has been measured, (ii) determined the nearest neighbors of each solution in 'objective space, and (iii) computed the discrepancy among the distances between nearest neighbors.

In general, a large-scale system typified by an electric power system also possesses multiple objectives to be achieved viz. economic operation, reliability, security and environment. Nanda et. al. [56] has applied linear and non-linear goal programming algorithm to solve the economic-emission load dispatch problem. Wadhwa and Jain [70] have formulated the optimal load flow problem as multiple objective programming problem. They minimized both the cost of generation and transmission loss simultaneously through priority goal programming. Hang et. al. [98] developed a technique for order preference by similarity to ideal solution for multiple objective decision making in which the chosen solution should be as close to positive ideal solution as possible and as far away from negative ideal solution as possible. To resolve the conflict between two distance objective functions, membership functions and satisfactory/preference levels have been employed. Types of conflict among planning objectives have been illustrated and categorized by Crousillat et. al. [99]. Purpose of their paper was to document new discoveries and observations in dealing with conflicting objectives and risk. Risk has been defined as the hazard to which a utility has been exposed to because of uncertainty. Differentiating uncertainty and risk, they concluded that uncertainty cannot be eliminated but risk can certainly be managed.

Nangia et. al. [156] has formulated a MOP. They have used weighting method to convert the problem into single objective problem. The optimal weight assigned to each objective has been derived on the basis of maximum range of each objective function. Hota et. al. [177] have described an interactive fuzzy satisfying method to formulate a MOP. Operating cost, NOx emission and transmission line security has been considered as competing objectives. Dhillon et. al. [184, 197] has applied the multi-objective problem formulation for the solution of stochastic economic-emission problem for short range as well as long range hydrothermal scheduling. Fuzzy set theory has been utilized for decision-making. Abido [211] has developed multi-objective evolutionary algorithm for environmental/economic power dispatch optimization problem. A fuzzy dynamic programming approach for multi-objective multi-stage decision making problems has been developed by Chen and Fu [246].

In recent years, the analysis of optimization problems having multiple and conflicting objectives has been a focal issue in various fields. Decision-making is a complex problem in such real world problems. A decomposition-coordination method called the envelope approach has been proposed by Li and Haimes [47]. It is used to generate the set of non-inferior solutions, to solve some classes of multi-objective optimization problems. A flexible method for nonlinear multi-criteria decision making problems was proposed by Narula and Weistroffer [65]. It is an interactive approach to formulate and solve multi-criteria decision-making problems. Decision maker is allowed to designate criterion functions as objective functions or as constraints or as something in between. The proposed algorithm guarantees all intermediate solutions to be efficient and the final solution to be a most preferred solution. An interactive projection method for multi-criteria optimization problems was explored by Ferreira and Geromel [71]. As an application a bi-objective optimization problem is solved with the relaxation-projection approach. A two-phase interactive solution method for multiple-objective programming problems was also discussed in a research paper by Buchanan [82].

In the present study weighting method has been used for generation of non-inferior solutions and interactive approach has been adopted for optimization process. The weighting method is employed to simulate the trade-off relation between the conflicting objectives in the non-inferior domain. In order to determine the Pareto Front from a large set of multi-objective points, two different efficient algorithms are implemented in proposed research. The first algorithm considers the logical relationship between dominated and non-dominated points to avoid duplication of comparisons as much as possible so that the overall operations reduced. The second algorithm takes the advantage of vectorization to split the given objective set into several smaller groups to be examined by the first algorithm. Then, the Pareto fronts of each group are combined as one set to be re-checked by the first algorithm again to determine the overall Pareto front.

2.5 RESULTS AND DISCUSSION

The proposed integrated DE-HS, hybrid HS-RS and hybrid DE-RS algorithms are tested for different unimodal, multimodal and discontinuous benchmark functions. Table-2.1 shows the test results for different benchmark functions using integrated DE-HS algorithm. Table-2.2 represents the test results for benchmark functions using hybrid HS-RS algorithm and Table-2.3 shows the test results hybrid DE-RS algorithm. Simulations results are recorded for the minimum $(f_{\min})$, average (f_{avg}) and maximum $(f_{\max})$ value of objective function along with standard deviation (*std*) and number of fitness evaluations (N_{FE}).

Table-2.1: Results of benchmark functions using integrated DE-HS algorithm

Function Type	**Dim**	**Range**	$f_{\min}$	f_{avg}	$f_{\max}$	***SD***	N_{FE}
$f_1(x)=\sum_{i=1}^{n} x_i^2$	30	[-100, 100]	1.60E-15	6.41E-13	5.44E-12	1.69E-12	303030
$f_2(x)=\sum_{i=1}^{n}\|x_i\|+\prod_{i=1}^{n}\|x_i\|$	30	[-10, 10]	2.39E-10	1	10	3.162278	303030
$f_3(x)=\max_i\{\|x_i\|,1\le i\le n\}$	30	[-100, 100]	0.000784	3.410478	20.4935	7.109703	303030
$f_4(x)=\sum_{i=1}^{n} ix_i^4+random[0,1)$	30	[-1.28, 1.28]	0.004634	0.012068	0.023054	0.007	303030
$f_5(x)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10]$	30	[-5.12, 5.12]	7.9597	16.41681	26.8638	5.121835	303030

Table-2.2: Results of benchmark functions using hybrid HS-RS algorithm

Function Type	Dim	Range	$f_{\min}$	f_{avg}	$f_{\max}$	SD	NFE
$f_1(x)=\sum_{i=1}^{n} x_i^2$	30	[-100, 100]	6.91E-29	1.15E-27	3.61E-27	1.18E-27	45030
$f_2(x)=\sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[-10, 10]	2.38E-17	6.42E-17	1.42E-16	3.95E-17	45030
$f_3(x)=\max_i\{\lvert x_i \rvert, 1 \le i \le n\}$	30	[-100, 100]	1.85E-07	5.42E-07	1.22E-06	3.54E-07	45030
$f_4(x)=\sum_{i=1}^{n} i x_i^4 + random[0,1)$	30	[-1.28, 1.28]	0.000453	0.002121	0.004184	0.001115	45030
$f_5(x)=\sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i)+10]$	30	[-5.12, 5.12]	5.68E-14	2.58303	7.7332	3.129082	45030

Table-2.3: Results of benchmark functions using hybrid DE-RS algorithm

Function Type	Dim	Range	$f_{\min}$	f_{avg}	$f_{\max}$	SD	NFE
$f_1(x)=\sum_{i=1}^{n} x_i^2$	30	[-100, 100]	1.94E-16	9.17E-13	6.41E-12	2.00E-12	45030
$f_2(x)=\sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[-10, 10]	1.75E-12	4.47E-10	2.24E-09	6.95E-10	45030
$f_3(x)=\max_i\{\lvert x_i \rvert, 1 \le i \le n\}$	30	[-100, 100]	1.87E-06	0.001042	0.00622	0.001916	45030
$f_4(x)=\sum_{i=1}^{n} i x_i^4 + random[0,1)$	30	[-1.28, 1.28]	0.00016	0.003848	0.01913	0.005594	45030
$f_5(x)=\sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i)+10]$	30	[-5.12, 5.12]	0	0.001856	0.012152	0.00414	45030

2.6 COMPARISON OF RESULTS

The proposed integrated DE-HS, hybrid HS-RS and hybrid DE-RS algorithms are successfully tested for unimodal, multimodal and discontinuous benchmark functions and results are recorded in Table-2.1, Table-2.2 and Table-2.3. The comparative analysis of proposed algorithms with respect to average value of objective function (*mean*) and standard deviation (*SD*) is shown in Table-2.4.

Table-2.4: Comparison of Results for different benchmark functions

Function Type	Integrated DE-HS		Hybrid HS-RS		Hybrid DE-RS	
	Mean	***SD***	***Mean***	***SD***	***Mean***	***SD***
$f_1(x)=\sum_{i=1}^{n} x_i^2$	6.41E-13	1.69E-12	1.15E-27	1.18E-27	9.17E-13	2.00E-12
$f_2(x)=\sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	1	3.162278	6.42E-17	3.95E-17	4.47E-10	6.95E-10
$f_3(x)=\max_i\{\lvert x_i \rvert, 1 \le i \le n\}$	3.410478	7.109703	5.42E-07	3.54E-07	0.001042	0.001916
$f_4(x)=\sum_{i=1}^{n} i x_i^4 + random[0,1)$	0.012068	0.007	0.002121	0.001115	0.003848	0.005594
$f_5(x)=\sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)+10]$	16.41681	5.121835	2.58303	3.129082	0.001856	0.00414

2.7 CONCLUSION

Conventional Harmony search algorithm, differential evolution algorithm and random search algorithm are described to explore the formulation of integrated DE-HS, hybrid HS-RS and hybrid DE-RS algorithm. The multi objective optimization problem is described to explore the concept of multi objective unit commitment problem. The proposed integrated DE-HS, hybrid HS-RS and hybrid DE-RS algorithm are tested for different unimodal, multimodal and discontinuous benchmark functions and their comparative analysis is represented to explore the effectiveness of proposed hybrid methodologies. In the succeeding chapters, proposed integrated DE-HS, hybrid HS-RS and hybrid DE-RS algorithms are explored for the solution of scalar and multi area unit commitment problem under single and multi-objective framework.