CHAPTER 4

OPTIMIZATION PROBLEM

4.1 INTRODUCTION

The goal of any optimization problem can be finding the combination of parameters (independent variables) that optimize a given quantity, possibly subject to some restrictions on the allowed parameter ranges. The quantity to be optimized (maximized or minimized) is termed as the objective function; the parameters that may be changed in the quest for the optimum are called control or decision variables; the restrictions on allowed parameter values are known as constraints.

4.2 PROBLEM FORMULATION OF OPTIMIZATION

The problem formulation of any optimization task can be thought of as a sequence of steps and they are:

- Choosing design variables (control and state variables)
- Formulating constraints
- Formulating objective functions
- Setting up variable limits
- Choosing an algorithm to solve the problem
- Solving the problem to obtain the optimal solution
Decision (control) variables are parameters that are deemed to affect the output in a significant manner. Selecting the best set of decision variables can sometimes be a challenge because it is difficult to ascertain which variables affect each specific behavior in a simulation. Logic determining control flow can also be classified as a decision variable. The domain of potential values for decision variables is typically restricted by constraints set by the user. Optimization may involve only one objective (single objective optimization) or more than one objective (multi objective optimization) simultaneously.

4.3 MULTI OBJECTIVE OPTIMIZATION

The multi objective optimization problem (MOOP; also called the multi criteria optimization, multi performance, or vector optimization problem) can be defined (in words) as the problem of finding a vector of decision variables that satisfies constraints and optimizes a vector function whose elements represent the objective functions. These functions form a mathematical description of performance criteria that are usually in conflict with each other. Hence, the term optimizes means finding such a solution that would give the values of all the objective functions acceptable to the decision maker.

Multi objective optimization has created immense interest in the engineering field in the last two decades. Optimization methods are of great importance in practice, particularly in engineering design, scientific experiments, and business decision making. Most of the real-world problems involve more than one objective, making multiple conflicting objectives interesting to solve as multi objective optimization problems.
4.4 OPTIMIZATION TECHNIQUES

There are many optimization algorithms or techniques available for engineers to solve their engineering design. Many optimization methods are problem specific i.e. appropriate only for certain type of problems. Thus, it is important to be able to recognize the characteristics of a problem in order to identify an appropriate solution technique.

Table 4.1 Classification of the objective functions

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Property</th>
<th>Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of control variables</td>
<td>One</td>
<td>Univariate</td>
</tr>
<tr>
<td></td>
<td>More than one</td>
<td>Multivariate</td>
</tr>
<tr>
<td>Type of control variables</td>
<td>Continuous real numbers</td>
<td>Continuous</td>
</tr>
<tr>
<td></td>
<td>Integers</td>
<td>Integer or discrete</td>
</tr>
<tr>
<td>Problem functions</td>
<td>Both continuous real numbers and integers</td>
<td>Mixed integer</td>
</tr>
<tr>
<td></td>
<td>Linear functions of the control variables</td>
<td>Linear</td>
</tr>
<tr>
<td></td>
<td>Quadratic functions of the control variables</td>
<td>Quadratic</td>
</tr>
<tr>
<td></td>
<td>Other nonlinear functions of the control variables</td>
<td>Nonlinear</td>
</tr>
<tr>
<td>Problem formulation</td>
<td>Subject to constraints</td>
<td>Constrained</td>
</tr>
<tr>
<td></td>
<td>Not subject to constraints</td>
<td>Unconstrained</td>
</tr>
</tbody>
</table>

Within each class of problems there are different minimization methods, varying in computational requirements, convergence properties, and so on. Optimization problems are classified according to the mathematical characteristics of the objective function, the constraints, and the control variables. Probably the most important characteristic is the nature of the objective function. These classifications are summarized in Table 4.1.
4.5 CLASSES OF OPTIMIZATION METHODS

There are two basic classes of optimization methods according to the type of solution. They are

- Optimality Criteria
- Search Methods

4.5.1 Optimality Criteria

**Analytical methods:** Once the conditions for an optimal solution are established, then either:

- A candidate solution is tested to see if it meets the conditions.
- The equations derived from the optimality criteria are solved analytically to determine the optimal solution.

4.5.2 Search Methods

**Numerical methods:** An initial trial solution is selected, either using common sense or at random, and the objective function is evaluated. A move is made to a new point (second trial solution) and the objective function is evaluated again. If it is smaller than the value for the first trial solution, it is retained and another move is made. The process is repeated until the minimum is found. Search methods are used when:

- The number of variables and constraints is large.
- The problem functions (objective and constraints) are highly nonlinear.
• The problem functions (objective and constraint) are implicit in terms of the decision/control variables making the evaluation of derivative information difficult.

Other suggestions for classification of optimization methods are:

• The first is based on classic methods such as the nonlinear programming technique, the weights method, and the c-constraints method.

• The second is based on the evolutionary techniques such as the Evolutionary Programming method (EP), Genetic Algorithm (GA), Differential Evolution (DE), Particle Swarm Optimization (PSO) and Big Bang-Big Crunch Algorithm (BB-BC).

4.6 CLASSIC METHODS (CONVENTIONAL TECHNIQUES)

The classic methods present some inconveniences due to the danger of convergence, the long execution time, algorithmic complexity, and the generation of a weak number of non-dominated solutions. Because of these inconveniences, evolutionary algorithms are more popular, thanks to their faculty to exploit vast amounts of research and the fact that they don’t require pre-recognition of the problem.

The two elements that most directly affect the success of an optimization technique are the quantity and domain of decision variables and the objective function. Identifying the decision variables and the objective function in an optimization problem often requires familiarity with the available optimization techniques and awareness of how these techniques interface with the system undergoing optimization.
The most appropriate method will depend on the type (classification) of problem to be solved. Some optimization techniques are computationally more expensive than others and thus the time required to complete an optimization is an important criterion. The setup time required for an optimization technique can vary from technique to technique and is dependent on the degree of knowledge required about the problem. All optimization techniques possess their own internal parameters that must be tuned to achieve good performances. The time required to tweak these parameters is part of the setup cost.

Conventional optimization techniques broadly consist of calculus-based, enumerated, and random techniques. These techniques are based on well established theories and work perfectly well for a case wherever applicable. But there are certain limitations to the above-mentioned methods. For example, the steepest descent method starts its search from a single point and finally ends up with an optimal solution. But this method does not ensure that this is the global optimum. Hence there is every possibility of these techniques getting trapped into local optima. Another great drawback of traditional methods is that they require complete information of the objective function, its dependence on each variable, and the nature of the function. They also make assumptions in realizing the function as a continuous one. All these characteristics of traditional methods make them inapplicable to many real-life problems where there is insufficient information on the mathematical model of the system, parameter dependence, and other such information. This calls for unconventional techniques to address many real-life problems.

The optimization methods that are incorporated in the optimal power flow tools can be classified based on optimization techniques such as:

- Linear Programming (LP) based methods
• Nonlinear Programming (NLP) based methods
• Integer Programming (IP) based methods
• Separable Programming (SP) methods

4.6.1 Linear Programming Based Methods

Notably, linear programming is recognized as a reliable and robust technique for solving a wide range of specialized optimization problems characterized by linear objectives and linear constraints. Many commercially available power system optimization packages contain powerful linear programming algorithms for solving power system problems for both planning and operator engineers.

Linear programming has extensions in the simple method, revised simplex method, and interior point techniques. Interior point techniques are based on the Karmarkar algorithm and encompass variants such as the projection scaling method, dual affine method, primal affine method, and barrier algorithm.

4.6.2 Nonlinear Programming Based Methods

In the case of the nonlinear programming optimization methods, the following techniques are introduced.

• Sequential Quadratic Programming (SQP) method
• Augmented Lagrangian (AL) method
• Generalized Reduced Gradient (GRG) method
• Projected Augmented Lagrangian (PAL) method
- Successive Linear Programming (SLP) method
- Interior Point (IP) method

4.6.2.1 Sequential quadratic programming method

Sequential quadratic programming is a technique for the solution of nonlinearly constrained problems. The main idea is to obtain a search direction by solving a quadratic program, that is, a problem with a quadratic objective function and linear constraints. This approach is a generalization of Newton’s method for unconstrained minimization. When solving optimization problems, SQP is not often used in its simple form. There are two major reasons for this: it is not guaranteed to converge to a local solution to the optimization problem, and it is expensive.

4.6.2.2 Augmented Lagrangian method

Augmented Lagrangian methods are a certain class of algorithms for solving constrained optimization problems. They have similarities to penalty methods in that they replace a constrained optimization problem by a series of unconstrained problems; the difference is that the augmented Lagrangian method adds an additional term to the unconstrained objective. This additional term is designed to mimic a Lagrange multiplier. The augmented Lagrangian is not the same as the method of Lagrange multipliers. Viewed differently, the unconstrained objective is the Lagrangian of the constrained problem, with an additional penalty term (the augmentation). Nonlinear optimization problems with multiple nonlinear constraints are often difficult to solve, because although the available mathematical theory provides the basic principles for solution, it does not guarantee convergence to the optimal point. The straightforward application of augmented Lagrangian techniques to such problems typically
results in slow (or lack of) convergence, and often in failure to achieve the optimal solution.

**4.6.2.3 Generalized reduced gradient method**

The basic concept of GRG method entails linearizing the Non-linear objective and constraint functions at a local solution with Taylor expansion equation. Then, the concept of reduced gradient method is employed which divides the variable set into two subsets of basic and non-basic variable and the concept of implicit variable elimination to express the basic variable by the non-basic variable. Finally, the constraints are eliminated and the variable space is deduced to only non-basic variables. The proven efficient method for non-constraints NLP problems is involved to solve the approximated problem and, then, the next optimal solution for the approximated problem should be found. The processes repeat again until it fulfills the optimal conditions. No less the generality, the following general form of NLP problem is used for explanation.

Generalized reduced Gradient-based search methods are a category of optimization techniques that use the gradient of the objective function to find an optimal solution. Each iteration of the optimization algorithm adjusts the values of the decision variables so that the simulation behavior produces a lower objective function value. Each decision variable is changed by an amount proportionate to the reduction in objective function value. Gradient-based searches are prone to converging on local minima because they rely solely on the local values of the objective function in their search. They are best used on well-behaved systems where there is one clear optimum. Gradient-based methods will work well in high-dimensional spaces provided these spaces don’t have local minima. Frequently, additional dimensions make it harder to guarantee that there are not local minima that could trap the search routine. As a result, as the dimensions (parameters) of the search space
increase, the complexity of the optimization technique increases. The benefits of traditional use of gradient-based search techniques are that computation and setup time are relatively low. However, the drawback is that global minima are likely to remain undiscovered.

4.6.2.4 Projected augmented Lagrangian method

The algorithm is of the projected Lagrangian type, involving a sequence of sparse, linearly constrained sub problems whose objective functions include a modified Lagrangian term and a modified quadratic penalty function. It is used for solving large scale nonlinear programs whose objective and constraint functions are smooth and differentiable.

4.6.2.5 Successive linear programming method

Traditionally, different solution approaches have been developed to solve the different classes of the power system problem. These methods are nonlinear programming techniques with very high accuracy, but their execution time is very long and they cannot be applied to real-time power system operations. Since the introduction of sequential or successive programming techniques, it has become widely accepted that successive linear programming algorithms can be used effectively to solve the optimization problem. In SLP, the original problem is solved by successively approximating the original problem using Taylor series expansion at the current operating point and then moving in an optimal direction until the solution converges.

4.6.2.6 Interior point method

Interior point methods (also referred to as barrier methods) are a certain class of algorithms to solve linear and nonlinear convex
optimization problems. The interior point method was invented by John von Neumann. The method consists of a self-concordant barrier function used to encode the convex set. Contrary to the simplex method, it reaches an optimal solution by traversing the interior of the feasible region.

Any convex optimization problem can be transformed into minimizing (or maximizing) a linear function over a convex set. The idea of encoding the feasible set using a barrier and designing barrier methods was studied in the early 1960s. These ideas were mainly developed for general nonlinear programming, but they were later abandoned due to the presence of more competitive methods for this class of problems (e.g. sequential quadratic programming).

4.6.2.7  Newton method

Newton’s method requires the computation of the second-order partial derivatives of the power flow equations and other constraints (the Hessian) and is therefore called a second-order method. The necessary conditions of optimality commonly are the Kuhn–Tucker conditions. Newton’s method is favoured for its quadratic convergence properties. This method is an iterative approach for engineering optimization.

4.7  INTEGER PROGRAMMING BASED METHODS

An optimization problem is called a Linear Programming problem if the objective function and the constraint functions are linear functions of the design variables. The standard form of an integer linear program in canonical form is expressed as:

maximize $C^T X$

subject to $AX \leq b$, 
where, the entries of $C$, $b$ and $A$ are integer. Note that similar to linear programs, ILP is not in standard form can be converted to standard form by eliminating inequalities by introducing slack variables and replacing variables that are not sign-constrained with the difference of two sign-constrained variables. Some optimization problems, having nonlinear objective functions and linear constraints can be transformed in ILP optimization problems by simple approximation of the corresponding nonlinear functions by piecewise linear functions. Solving integer programming optimization problems, that is, finding an optimal solution to such kind of problems, can be a difficult task.

The linear integer programming problems are easier solvable than the convex nonlinear integer programming problems. An instance of problem can be transformed in polynomial time to an instance of a 0-1 linear integer programming problem. But the 0-1 linear integer programming problem can be solved by a brute-force enumerative algorithm. It should be noted, that there are many special cases that belong to the class of problems, solvable in polynomial time, i.e., there exist algorithms with polynomial time computational complexity, which can solve them.

The difficulty to solve (linear and/or nonlinear) integer programming problems arises from the fact that unlike linear programming, for example, whose feasible region is a convex set, in integer programming
problems, one must search for a lattice of feasible integer points to find an optimal solution.

Unlike LP where, due to the convexity of the problem, we can exploit the fact that any local solution is a global optimum, the integer programming problems have many local optima and finding a global optimum to the problem requires one to prove that a particular solution dominates all the feasible points by arguments other than the calculus based derivative approaches of convex programming with continuous variables. For this reason, the approximate algorithms solving ILP optimization problems are widely spread.

4.8 SEPARABLE PROGRAMMING METHODS

Separable programming is important because it allows a convex nonlinear program to be approximated with arbitrary accuracy with a linear programming model. The idea is to replace each nonlinear function with a piecewise linear approximation. Global solutions can then be obtained with any number efficient LP codes. For non-convex problems the approach is still valid but more work needs to be done. Either a mixed-integer linear programming problem must be solved, or a modified version of the simplex algorithm with a limited basis entry rule can be applied to the model directly. The candidates for the entering variable must be restricted to maintain the validity of the LP approximation. In this case a local optimum is obtained but it is possible to find the global optimum with the help of branch and bound. The technique separable programming basically replaces all separable functions, in objectives and constraints, by piecewise linear functions.
4.9 EVOLUTIONARY TECHNIQUES

Recently the advances in computer engineering and the increased complexity of the power system optimization problem have led to a greater need for and application of specialized programming techniques for large-scale problems. These include dynamic programming, Lagrange multiplier methods, heuristic techniques, and evolutionary techniques such as genetic algorithms. These techniques are often hybridized with many other intelligent system techniques, including Artificial Neural Networks (ANN), Expert Systems (ES), Tabu Search algorithms (TS), and Fuzzy Logic (FL).

Many researchers agree that first, having a population of initial solutions increases the possibility of converging to an optimum solution, and second, updating the current information of the search strategy from the previous history is a natural tendency. Accordingly, attempts have been made by researchers to restructure these standard optimization techniques in order to achieve the two goals mentioned. To achieve these two goals, researchers have made concerted efforts in the last decade to invent novel optimization techniques for solving real-world problems, which have the attributes of memory update and population-based search solutions.

4.9.1 Heuristic Search

Several heuristic tools have evolved in the last decades that facilitate solving optimization problems that were previously difficult or impossible to solve. These tools include evolutionary computation, simulated annealing, tabu search, particle swarm, and so on. Reports of applications of each of these tools have been widely published. Recently, these new heuristic tools have been combined among themselves and with knowledge elements, as well as with more traditional approaches such as statistical analysis, to solve extremely challenging problems.
Developing solutions with these tools offers two major advantages:

- Development time is much shorter than when using more traditional approaches.
- The systems are very robust, being relatively insensitive to noisy and/or missing data.

Heuristic-based methods strike a balance between exploration and exploitation. This balance permits the identification of local minima, but encourages the discovery of a globally optimal solution. However, it is extremely difficult to fine-tune these methods to gain vital performance improvements. Because of their exploitation capabilities, heuristic methods may also be able to obtain the best value for decision variables. Heuristic techniques are good candidate solutions when the search space is large and nonlinear because of their exploration capabilities.

### 4.9.2 Evolutionary Computation

The mainstream algorithms for evolutionary computation are Genetic Algorithms (GAs), Evolutionary Programming (EP), evolution strategies, and Genetic Programming (GP). These algorithms have been recognized as important mathematical tools in solving continuous optimization problems. Evolutionary algorithms possess the following salient characteristics.

- Genetic variation is largely a chance phenomenon and stochastic processes play a significant role in evolution.
- A population of agents with nondeterministic recruitment is used.
• Inherent parallel search mechanisms are used during evolution.

• Evolution is a change in adaptation and diversity, not merely a change in gene frequencies.

• Evolutionary algorithms operate with a mechanism of competition–cooperation.

These algorithms simulate the principle of evolution (a two-step process of variation and selection), and maintain a population of potential solutions (individuals) through repeated application of some evolutionary operators such as mutation and crossover. They yield individuals with successively improved fitness, and converge; it is hoped, to the fittest individuals representing the optimum solutions. The evolutionary algorithms can avoid premature entrapment in local optima because of the stochastic search mechanism. Genetic algorithms and evolutionary programming are among the two most widely used evolutionary computation algorithms.

Evolutionary algorithms are robust and powerful global optimization techniques for solving large-scale problems that have many local optima. However, they require high computation times, and they are very poor in terms of convergence performance. On the other hand, local search algorithms can converge in a few iterations but lack a global perspective. The combination of global and local search procedures should offer the advantages of both optimization methods while offsetting their disadvantages.

Evolutionary algorithms seem particularly suitable to solve multi-objective optimization problems because they deal simultaneously with a set of possible solutions (the so-called population). This allows us to find several members of the Pareto optimal set in a single run of the algorithm, instead of having to perform a series of separate runs as in the case of traditional
mathematical programming techniques. In addition, evolutionary algorithms are less susceptible to the shape or continuity of the Pareto front (e.g., they can easily deal with discontinuous or concave Pareto fronts), whereas these two issues are a real concern with mathematical programming techniques.

4.9.3 Genetic Algorithm

The genetic algorithm is a search algorithm based on the conjunction of natural selection and genetics. The features of the genetic algorithm are different from other search techniques in several aspects. The algorithm is multipath, searching many peaks in parallel, and hence reducing the possibility of local minimum trapping. In addition, GA works with a coding of parameters instead of the parameters themselves. The parameter coding will help the genetic operator to evolve the current state into the next state with minimum computations.

The genetic algorithm evaluates the fitness of each string to guide its search instead of the optimization function. The GA only needs to evaluate the objective function (fitness) to guide its search. There is no requirement for derivatives or other auxiliary knowledge. Hence, there is no need for computation of derivatives or other auxiliary functions. Finally, GA explores the search space where the probability of finding improved performance is high.

At the start of a genetic algorithm optimization, a set of decision variable solutions is encoded as members of a population. There are multiple ways to encode elements of solutions including binary, value, and tree encodings. Crossover and mutation operators based on reproduction are used to create the next generation of the population. Crossover combines elements of solutions in the current generation to create a member of the next
generation. Mutation systematically changes elements of a solution from the current generation in order to create a member of the next generation.

Crossover and mutation accomplish exploration of the search space by creating diversity in the members of the next generation. Traditional uses of GAs leverage the fact that these algorithms explore multiple areas of the search space to find a global minimum. Through the use of the crossover operator, these algorithms are particularly strong at combining the best features from different solutions to find one global solution. Genetic algorithms are also well suited for searching complex, highly nonlinear spaces because they avoid becoming trapped in a local minimum. Genetic algorithms explore multiple solutions simultaneously. These sets of solutions make it possible for a user to gain, from only one iteration, multiple types of insight of the algorithm.

4.9.4 Evolutionary Programming

Evolution strategies employ real-coded variables and, in their original form, relied on mutation as the search operator, and a population size of one. Since then they have evolved to share many features with GAs. The major similarity between these two types of algorithms is that they both maintain populations of potential solutions and use a selection mechanism for choosing the best individuals from the population.

Evolutionary programming is a stochastic optimization strategy similar to GA, which places emphasis on the behavioral linkage between parents and their offspring, rather than seeking to emulate specific genetic operators as observed in nature. EP is similar to Evolutionary Strategies (ES), although the two approaches developed independently. Like both ES and GAs, EP is a useful method of optimization when other techniques such as gradient descent or direct analytical discovery are not possible. Combinatorial
and real-valued function optimizations, in which the optimization surface or fitness landscape is “rugged,” possessing many locally optimal solutions, are well suited for evolutionary programming.

As a powerful and general global optimization tool, EP seeks the optimal solution by evolving a population of candidate solutions over a number of generations or iterations. A new population is generated from an existing population through the use of a mutation operator. A mutation operator with high efficiency should fully reflect the principle of organic evolution in nature, that is, the lower the fitness score is, the higher the mutation possibility is, and vice versa.

Through the use of a competition scheme, the individuals in each population compete with each other. The winning individuals will form a resultant population that is regarded as the next generation. For optimization to occur, the competition scheme must ensure that the more optimal solutions have a greater chance of survival than the poorer solutions. Through this process, the population is expected to evolve toward the global optimum. It is known that there is more research needed in the mathematical foundation for the EP or its variants with regard to experimental and empirical research. The state-of-the-art of EP mainly focuses on the application of solving optimization problems, especially for the application to real-valued function optimization.

### 4.9.5 Tabu Search

Tabu search is basically a gradient descent search with memory. The memory preserves a number of previously visited states along with a number of states that might be considered unwanted. This information is stored in a tabu list. The definition of a state, the area around it and the length of the tabu list are critical design parameters. In addition to these tabu
parameters, two extra parameters are often used: aspiration and diversification. Aspiration is used when all the neighboring states of the current state are also included in the tabu list. In that case, the tabu obstacle is overridden by selecting a new state. Diversification adds randomness to this otherwise deterministic search. If the tabu search does not converge, the search is reset randomly.

Tabu search has the advantage of not using hill-climbing strategies. Its performance can also be enhanced by branch-and-bound techniques. However, the mathematics behind this technique is not as strong as those behind neural networks or simulated annealing. Furthermore, a solution space must be generated. Hence, tabu search requires knowledge of the entire operation at a more detailed level.

4.9.6 Simulated Annealing

In statistical mechanics, a physical process called annealing is often performed in order to relax the system to a state with minimum free energy. In the annealing process, a solid in a heat bath is heated up by increasing the temperature of the bath until the solid is melted into liquid, then the temperature is lowered slowly. In the liquid phase all particles of the solid arrange themselves randomly. In the ground state the particles are arranged in a highly structured lattice and the energy of the system is a minimum. The ground state of the solid is obtained only if the maximum temperature is sufficiently high and the cooling is done sufficiently slowly. Based on the annealing process in statistical mechanics, simulated annealing was introduced for solving complicated combinatorial optimization. The name “simulated annealing” originates from the analogy with the physical process of solids, and the analogy between the physical system and simulated annealing is that the cost function and the solution (configuration) in the
optimization process correspond to the energy function and the state of statistical physics, respectively.

In a large combinatorial optimization problem, an appropriate perturbation mechanism, cost function, solution space, and cooling schedule are required in order to find an optimal solution with simulated annealing. The process is effective in network reconfiguration problems for large-scale distribution systems, and its search capability becomes more significant as the system size increases. Moreover, the cost function with a smoothing strategy enables simulated annealing to escape more easily from local minima and to reach the vicinity of an optimal solution rapidly.

The major strengths of simulated annealing are that it can optimize functions with arbitrary degrees on nonlinearity, stochasticity, boundary conditions, and constraints. It is also statistically guaranteed to find an optimal solution. However, it has its disadvantages too. Like GAs it is very slow; its efficiency is dependent on the nature of the surface it is trying to optimize and it must be adapted to specific problems. The availability of supercomputing resources, however, mitigates these drawbacks and makes simulated annealing a good candidate.

4.9.7 Stochastic Approximation

Some non classical optimization techniques are able to optimize on discontinuous objective functions, however, they are unable to do so when the complexity of the data becomes very large. In this case the complexity of the system requires that the objective function be estimated. Furthermore, the models that are used to estimate the objective function may be stochastic due to the dynamic and random nature of the system and processes.
The basic idea behind the stochastic approximation method is the gradient descent method. Here the variable that the objective function is to be optimized upon is varied in small increments and the impact of this variation (measured by the gradient) is used to determine the direction of the next step. The magnitude of the step is controlled to have larger steps when the perturbations in the system are small and vice versa. Stochastic approximation algorithms based on various techniques have been developed recently. They have been applied to both continuous and discrete objective functions. Recently, their convergence has been proved for the degenerate case as well.

Stochastic approximation did not have as many applications reported as the other techniques. This could have been because of various factors such as the lack of a metaphorical concept to facilitate understanding and proofs that are complex. It has recently shown great promise, however, especially in optimizing non discrete problems.

4.9.8 Differential Evolution

Differential Evolution (DE) is an improved version of the genetic algorithm for faster optimization. Unlike a simple GA that uses binary coding for representing problem parameters, differential evolution uses real coding of floating point numbers. Among the DE’s advantages are its simple structure, ease of use, speed, and robustness.

Differential strategies can be adopted in a DE algorithm depending upon the type of problem to which DE is applied. The strategies can vary based on the vector to be perturbed, number of difference vectors considered for perturbation, and finally the type of crossover used. The general convention used in these strategies is DE/x/y/z. DE stands for differential evolution, x represents a string denoting the vector to be perturbed, y is the
number of difference vectors considered for the perturbation of x, and z stands for the type of crossover being used.

Differential evolution has been successfully applied in various fields. Some of the successful applications include digital filter design, batch fermentation process, estimation of heat transfer parameters in a trickle bed reactor, optimal design of heat exchangers, synthesis and optimization of a heat-integrated distillation system, scenario-integrated optimization of dynamic systems, optimization of nonlinear functions, optimization of thermal cracker operation, optimization of nonlinear chemical processes, global optimization of nonlinear chemical engineering processes, and optimization of water pumping systems, among others.

4.9.9  Bacteria Foraging Algorithm

Bacterial Foraging Optimization Algorithm (BFOA) has been widely accepted as a global optimization algorithm of current interest for distributed optimization and control. BFOA is inspired by the social foraging behavior of Escherichia coli. BFOA has already drawn the attention of researchers because of its efficiency in solving real-world optimization problems arising in several application domains. The underlying biology behind the foraging strategy of E.coli is emulated in an extraordinary manner and used as a simple optimization algorithm.

During foraging of the real bacteria, locomotion is achieved by a set of tensile flagella. Flagella help an E.coli bacterium to tumble or swim, which are two basic operations performed by a bacterium at the time of foraging. When they rotate the flagella in the clockwise direction, each flagellum pulls on the cell. That results in the moving of flagella independently and finally the bacterium tumbles with lesser number of
tumbling whereas in a harmful place it tumbles frequently to find a nutrient gradient. Moving the flagella in the counterclockwise direction helps the bacterium to swim at a very fast rate. In the above-mentioned algorithm the bacteria undergoes chemotaxis, where they like to move towards a nutrient gradient and avoid noxious environment.

Generally the bacteria move for a longer distance in a friendly environment. When they get food in sufficient, they are increased in length and in presence of suitable temperature they break in the middle to from an exact replica of itself. This phenomenon inspired to introduce an event of reproduction in BFOA. Due to the occurrence of sudden environmental changes or attack, the chemotactic progress may be destroyed and a group of bacteria may move to some other places or some other may be introduced in the swarm of concern. This constitutes the event of elimination-dispersal in the real bacterial population, where all the bacteria in a region are killed or a group is dispersed into a new part of the environment.

4.9.10 Shuffled Frog Leaping Algorithm

Shuffled Frog Leaping Algorithm (SFLA) is a heuristic search algorithm presented for the first time by Eusuff and Lansey in 2003. The main purpose of this algorithm was achieving a method to solve complicated optimization problems without any use of traditional mathematical optimization tools. In fact, the SFL algorithm is combination of “meme-based genetic algorithm or Memetic Algorithm” and “Particle Swarm Optimization (PSO)”. This algorithm has been inspired from memetic evolution of a group of frogs when seeking for food. In this method, a solution to a given problem is presented in the form of a string, called “frog” which has been considered as a control vector.
The initial population of frogs is partitioned into groups or subsets called “memeplexes” and the number of frogs in each subset is equal. The SFL algorithm is based on two search techniques: local search and global information exchange techniques. Based on local search, the frogs in each subset improve their positions to have more foods (to reach the best solution). In second technique, obtained information between subsets is compared to each other (after each local search in subsets).

4.10 PROPOSED APPROACHES

Optimization algorithms have constituted some of the most significant subjects in mathematics and engineering optimization. For all the traditional algorithms available to seek best solutions to a given function, however, optimization continues to pose a challenge in most real world cases because of the huge and complex solution space. Indeed, there are still large-scale optimization problems that necessitate speedy resolution in a time span between ten milliseconds and a few minutes, resulting in the exchange of optimality with speed gains.

In the main, speed and precision are goals considered to be at variance, at least with respect to probabilistic algorithms: optimization accuracy can be enhanced only if there is more time available. These obstacles, which are correlated with the utilization of mathematical optimization in large-scale problems, have gradually paved the way for the development of alternative solutions. One of these alternative solutions is to implement the Evolutionary Algorithm. The Evolutionary Algorithm was developed by mimicking or simulating processes found in nature and mainly includes Genetic Algorithms (GA), Differential Evolution (DE), Particle Swarm Optimization (PSO), and Big Bang Big Crunch (BB-BC) algorithms.
The PSO is very simple, easy to implement and has balanced local and global searching mechanism. It has good convergence characteristics and exploited for many engineering optimizations. The BB-BC algorithm is a recently developed one. It is based on the big bang theory about evolution of the universe. It is with very less number of operators and parameters. Its convergence behavior is highly commendable. These two algorithms have almost similar searching quality and with less number of parameters. This is the reason why preference is given for PSO and BB-BC among other evolutionary techniques.

In this research work, PSO and BB-BC are proposed to optimize the problem of real power loss minimization and Voltage stability maximization of the system by inserting single and multi type FACS devices. Implementation of PSO and BB-BC for voltage stability limit improvement is given in chapter 5. Results are obtained from these algorithms and compared when single type and multi type FACTS controllers are used. These results are discussed in detail in chapter 6.

4.11 NEWTON METHOD OF OPTIMIZATION

Newton’s method is an optimization technique applied to find the minimum or the maximum of a function $f(x)$. It is very similar to the Newton-Raphson method in finding an optimal solution of a function such that $f(x)=0$. Since the derivative of the function $f(x)$, $f'(x)$ at the functions maximum and minimum, the minima and the maxima can be found by applying the Newton-Raphson method to the derivative, essentially obtaining,

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$  \hspace{1cm} (4.1)
We caution that before using Newton’s method to determine the minimum or the maximum of a function, one should have a reasonably good estimate of the solution to ensure convergence, and that the function should be easily twice differentiable.

![Newton's method diagram](image)

**Figure 4.1 Newton’s method**

Slope at point

\[
C \approx \frac{f(x_i) - f(x_{i+1})}{x_i - x_{i+1}} \tag{4.2}
\]

We “wish” that in the next iteration \( X_{i+1} \) will be the root, or \( f(x_{i+1}) = 0 \). Thus:

Slope at point

\[
C \approx \frac{f(x_i) - 0}{x_i - x_{i+1}} \tag{4.3}
\]

Or
The value of first derivative can be calculated from the slope of the objective function. Here, $x_i$ and $x_{i+1}$ are the value of $x$ at $i$th and $i+1$th iterations respectively.

$$f'(x) = \frac{f(x_i)}{x_i - x_{i+1}}$$

Hence:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

The algorithm is repeated until the difference between the values of $x$ in consecutive iterations is acceptably low (tolerance).
PARTICLE SWARM OPTIMIZATION ALGORITHM

4.12.1 Overview

The concept of PSO was first suggested by Kennedy and Eberhart in 1995. Since its development, PSO has become one of the most promising optimizing techniques for solving global optimization problems. Its mechanism is inspired by the social and cooperative behavior displayed by
various species like birds, fish, termites, ants and even human beings. The PSO system consists of a population (swarm) of potential solutions called particles. These particles move through the search domain with a specified velocity in search of optimal solution. Each particle maintains a memory which helps it in keeping the track of its previous best position.

4.12.2 Searching mechanism of PSO

The positions of the particles are distinguished as personal best and global best. In the past several years, PSO has been successfully applied in many research and application areas. It has been demonstrated that PSO gets better results in a faster and cheaper way in comparison to other methods like GA, simulated annealing (SA) etc. The particles or members of the swarm fly through a multidimensional search space looking for a potential solution. Each particle adjusts its position in the search space from time to time according to the flying experience of its own and of its neighbors (or colleagues). The flow chart of the PSO algorithm is given in Figure 4.1.

For a $D$-dimensional search space, the position of the $i^{th}$ particle is represented as:

$$X_i = (x_{i1}, x_{i2}, \ldots x_{id}, \ldots x_{iD})$$  \hspace{1cm} (4.6)

Each particle maintains a memory of its previous best position which is represented as:

$$P_i = (p_{i1}, p_{i2}, \ldots p_{id}, \ldots p_{iD})$$  \hspace{1cm} (4.7)

The best one among all the particles in the population is represented as:

$$P_g = (p_{g1}, p_{g2}, \ldots p_{gd}, \ldots p_{gD})$$  \hspace{1cm} (4.8)
The velocity of each particle is represented as:

\[ V_i = (v_{i1}, v_{i2}, \ldots, v_{id}, \ldots, v_{id_s}) \]  \hspace{1cm} (4.9)

The maximum velocity is represented as:

\[ V_{max} = (v_{max1}, v_{max2}, \ldots, v_{maxd}, \ldots, v_{maxd_s}) \]  \hspace{1cm} (4.10)

The velocity \( V_i \) of each particle is clamped to a maximum velocity \( V_{max} \) which is specified by the user. \( V_{max} \) determines the resolution with which regions between the present position and the target position are searched. Large values of \( V_{max} \) facilitate global exploration, while smaller values encourage local exploitation. If \( V_{max} \) is too small, the swarm may not explore sufficiently beyond locally good regions. On the other hand, too large values of \( V_{max} \) risk the possibility of missing a good region. At each iteration a new velocity value for each particle is evaluated according to its current velocity, the distance from the global best position. The new velocity value is then used to calculate the next position of the particle in the search space. This process is then iterated a number of times or until a minimum error is achieved.
Figure 4.3 Flow chart for PSO algorithm
The two basic equations which govern the working of PSO are that of velocity vector and position vector given by:

\[ V_{id} = w v_{id} + c_1 r a n d_1 (p_{id} - x_{id}) + c_2 r a n d_2 (p_{ga} - x_{ga}) \]  

(4.6)

\[ x_{id} = x_{id} + v_{id} \]  

(4.7)

Here \( w \) is the inertia constant, \( c_1 \) and \( c_2 \) are acceleration constants. They represent the weighting of the stochastic acceleration terms that pull each particle towards personal best and global best positions. Therefore, adjustment of these constants changes the amount of tension in the system. Small values of these constants allow particles to roam far from the target regions before tugged back, while high values result in abrupt movement toward, or past, target regions. The constants \( r a n d_1, r a n d_2 \) are the uniformly generated random numbers in the range of \((0, 1)\).

The first part of Equation (4.6), \( w v_{id} \), represents particle’s previous velocity, which serves as a memory of the previous flight direction. This memory term can be visualized as a momentum, which prevents the particle from drastically changing its direction and biases it towards the current direction. The second part, \( c_1 r a n d_1 (p_{id} - x_{id}) \), is called the cognition part and it indicates the personal experience of the particle. We can say that, this cognition part resembles individual memory of the position that was best for the particle. The effect of this term is that particles are drawn back to their own best positions, resembling the tendency of individuals to return to situations or places that were most satisfying in the past. The third part, \( c_2 r a n d_2 (p_{ga} - x_{id}) \), represents the cooperation among particles and is therefore named as the social component. This term resembles a group norm or standard which individuals seek to attain. The effect of this term is that each particle is also drawn towards the best position found by its neighbor.
4.13 BIG BANG-BIG CRUNCH ALGORITHM

This algorithm involves two phases in its searching mechanism, viz big bang phase and big crunch phase as explained below.

4.13.1 Big Bang Phase

The BB–BC is a meta heuristic global optimization method and is developed by Erol and Eksin. It involves two phases: The Big Bang phase and the Big Crunch phase. In the Big Bang phase, candidate solutions are randomly distributed over the search space. Randomness can be seen as equivalent to the energy dissipation in nature while convergence to a local or global optimum point can be viewed as gravitational attraction. Since energy dissipation creates disorder from ordered particles, we will use randomness as a transformation from a converged solution to the birth of totally new solution candidates. The creation of the initial population randomly is called the Big Bang phase. In this phase, the candidate solutions are spread all over the search space in a uniform manner.

4.13.2 Big Crunch Phase

The Big Bang phase is followed by the Big Crunch phase. The Big Crunch is a convergence operator that has many inputs but only one output, which is named as the “centre of mass”, since the only output has been derived by calculating the centre of mass. The point representing the centre of mass that is denoted by $X_c$ is calculated according to the following equation.

$$X_c = \frac{\sum_{i=1}^{NP} \frac{1}{f(X_i)} X_i}{\sum_{i=1}^{NP} \frac{1}{f(X_i)}}$$  \hspace{1cm} (4.8)
where $X_i$ is a point within an D-dimensional search space generated, $f(X_i)$ is a fitness function value of this point, $NP$ is the population size in Big Bang phase. The convergence operator in the Big Crunch phase is different from ‘exaggerated’ selection since the output term may contain additional information (new candidate or member having different parameters than others) than the participating ones, hence differing from the population members. This one step convergence is superior compared to selecting two members and finding their centre of gravity. This method takes the population members as a whole in the Big Crunch phase that acts as a squeezing or contraction operator; and it, therefore, eliminates the necessity for two-by-two combination calculations.

![Flow chart for BB-BC algorithm](image)

**Figure 4.4 Flow chart for BB-BC algorithm**
After the Big Crunch phase, the algorithm must create new members to be used as the Big Bang of the next iteration step. This can be done in various ways, the simplest one being jumping to the first step and creating an initial population. The algorithm will have no difference than random search method by so doing since latter iterations will not use the knowledge gained from the previous ones; hence, the convergence of such an algorithm will most probably be very low. In this work, the new candidates are generated around the centre of mass and knowledge of centre of mass of previous iteration is used for better convergence. The parameters to be supplied to normal random point generator are the centre of mass of the previous step and the standard deviation. The deviation term can be fixed, but decreasing its value along with the elapsed iterations produces better results.

\[ X^{new} = X_c + \frac{r\alpha(X^{max} - X^{min})}{t} \]  

(4.9)

where, \( r \) is a normal random number, \( \alpha \) is a parameter limiting the size of the search space, \( X^{max} \) and \( X^{min} \) are the upper and lower limits, and \( t \) is the iteration step. Since normally distributed numbers can be exceeding \( \pm1 \), it is necessary to limit the population to the prescribed search space boundaries. This narrowing down restricts the candidate solutions into the search space boundaries. This BB-BC algorithm is similar to PSO in searching behaviour.

4.14 CONCLUSION

This chapter exposes the basics and classification of optimization. The different conventional and evolutionary techniques are also discussed. The two nature inspired algorithms of PSO and BB-BC are explained in a detailed manner. These two algorithms are used in this research for optimizing the voltage stability limit improvement.