CHAPTER 4

OPTIMIZATION TECHNIQUES IN PERSPECTIVE

Optimization means maximization or minimization of one or more functions with any possible constraints. In this chapter different types of optimization techniques are described briefly with emphasis on those that are used in the present dissertation.

4.1. Introduction

The origin of optimization methods can be traced from 300 BC when Euclid identified the minimal distance between two points to be length of straight line joining the two. He also proved that a square has the greatest area among the rectangles with given total length of edges. Heron proved in 100 BC that light travels between two points through the path with shortest length when reflecting from a mirror. Before the invention of calculus of variations, the optimization problems like, determining optimal dimensions of wine barrel in 1615 by J. Kepler, a proof that light travels between two points in minimal time in 1657 by P. De Fermat were solved. I. Newton (1660s) and G.W. von Leibniz (1670s) created mathematical analysis that forms the basis of calculus of variation. L. Euler's publication in 1740 began the research on general theory of calculus of variations.

The method of optimization for constrained problems, which involve the addition of unknown multipliers, became known by the name of its inventor, J. L. Lagrange. Cauchy made the first application of the gradient method to solve unconstrained optimization problems in 1847. G. Dantzig presented Simplex method in 1947. N. Karmarkar's polynomial time algorithm in 1984 begins a boom of interior point optimization methods. The advancement in solution techniques resulted several well defined new areas in optimization methods.

The linear and non-linear constraints arising in optimization problem can be easily handled by penalty method. In this method few or more expressions are added to make objective function less optimal as the solution approaches a constraint.

4.2. An Optimization Problem

The main components of an optimization problem are:

Objective Function

An objective function expresses one or more quantities which are to be minimized or maximized. The optimization problems may have a single objective function or more objective functions. Usually the different objectives are not compatible. The variables that optimize one objective may be far from optimal for the others. The problem with multi-objectives can be reformulated as single objective problems by either forming a weighted combination of the different objectives or by treating some of the objectives as constraints.

Variables

A set of unknowns, which are essential are called variables. The variables are used to define the objective function and constraints. One can not chose design variable arbitrarily, they have to satisfy certain specified functional and other requirements. The design variables can be continuous, discrete or Boolean.

Constraints

A set of constraints are those which allow the unknowns to take on certain values but exclude others. They are conditions that must be satisfied to render the design to be feasible.

Once the design variables, constraints, objectives and the relationship between them have been chosen, the optimization problem can be defined.

Statement of an optimization problem

An optimization problem can be stated as follows: To find $x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, which minimizes or maximizes$f(x)$;

Subject to the constraints

$$g_i(x) \leq 0; \;\; i = 1,2,3 \ldots m$$

$$h_j(x) = 0; \;\; j = 1,2,3 \ldots p$$

Where x is an n-dimensional vector called design variable, $f(x)$ is called the objective function, and $g_i(x)$ and $h_j(x)$ are known as inequality and equality constraints respectively. This type of problem is called constrained optimization problem. This problem can be represented in the following way,

Given: a function $f: A \to R$ (from some set A to set R)

Sought: an elements x_0 in A such that $f(x_0) \leq f(x)$for all x in A ("minimization") or such that $f(x_0) \geq f(x)$for all x in A ("maximization").

Typically, A is some subset of Euclidean space R^n, often specified by a set of constraints, equalities or inequalities that the members of A have to satisfy. The elements of A are called candidate solutions or feasible solutions. The function f is called an objective function, or cost function. A feasible solution that minimizes (or maximizes) the objective function is called an optimal solution. The domain A of f is called the search space. When the feasible region or the objective function of the problem does not present convexity, there may be several local minima and maxima, where a local minimum x* is defined as a point for which there exist some $\delta > 0$ so that for all x such that $||x - x^*|| \leq \delta; \quad and \quad f(x^*) \leq f(x)$

i.e., on some region around x^* all the function values are greater than or equal to the value at that point. In a similar manner local maxima can be defined.

A large number of algorithms for solving non-convex problem are not capable of making a distinction between local optimal solutions and global optimal solutions.

4.2. Classification of Optimization Problems

Optimization problems can be classified based on the type of constraints, nature of design variables, nature of the equations involved and type & number of objective functions. These classifications are briefly discussed below.

- ***Based on existence of constraints***

A problem is called constrained optimization problem if it is subject to one or more constraints otherwise it is called unconstrained.

- ***Based on the nature of the equations involved***

Based on the nature of equations for the objective function and the constraints, optimization problems can be classified as linear and nonlinear programming problems. The classification is very useful from a computational point of view since many predefined special methods are available for effective solution of a particular type of problem.

(i) Linear Programming problem

If the objective function and all the constraints are 'linear' functions of the design variables, the optimization problem is called a *linear programming problem* (LPP). In such case objective function $f(x)$, inequality constraints $g_i(x)$ and equality constraints $h_j(x)$ are linear.

(ii) Quadratic programming problem

If the objective function is a quadratic function and all constraint functions are linear functions of optimization variables, the problem is called a quadratic programming problem. It is possible to solve Quadratic Programming problems using extensions of the methods for LPP.

(iii) Nonlinear programming problem

If any of the functions among the objectives and constraint functions is nonlinear, the problem is called a *nonlinear programming* (NLP) *problem*. This is the most general form of a programming problem and all other problems can be considered as special cases of the NLP problem.

- ***Based on the permissible values of the decision variables***

(i) Integer programming problem

If some or all of the design variables of an optimization problem are restricted to take only integer (or discrete) values, the problem is called an *integer programming problem.* For example, the optimization is to find number of articles needed for an operation with least effort. Thus, minimization of the effort required for the operation being the objective, the decision variables, i.e. the number of articles used can take only integer values. Other restrictions on minimum and maximum number of usable resources may be imposed.

(ii) ***Real-valued programming problem***

A real-valued problem is that in which it is sought to minimize or maximize a real function by systematically choosing the values of real variables from within an allowed set. When the allowed set contains only real values, it is called a real-valued programming problem.

- ***Based on the number of objective functions***

Under this classification, objective functions can be classified as single-objective and multi-objective programming problems.

(i) ***Single-objective problem:*** Problem in which there is only a single objective function.

(ii) ***Multi-objective problem:*** A multi-objective programming problem can be stated as follows:

Find x which minimizes $f_1(x), f_2(x), \ldots f_k(x)$; Subject to $g_j(x) \leq 0, j = 1, 2, \ldots, m$

where $f_1, f_2, \ldots f_k$ denote the objective functions to be minimized simultaneously. There are m number of constraints also. For multi-objective optimization problems one tries to find good trade-offs rather than a single solution as in single objective problems. The most commonly used notion of the 'optimum' proposed by Pareto is depicted as follows. A vector of the decision variable x is called Pareto Optimal (efficient solution) if there does not exist another y such that $f_i(y) \leq f_i(x)$ for $i=1,2,3 \ldots k$ with $f_j(y) < f_i(x)$ for at least one j. In other words a solution vector x is called optimal if there is no other vector y that reduces some objective functions without causing simultaneous increase in at least one other objective function.

4.3. Solution of Optimization Problems

The choice of suitable optimization method depends on the type of optimization problem. Various classical methods were there to solve such problems. The major advances in optimization occurred only after the development of fast digital computers. Now days various advanced optimization techniques are used to solve the design and operation related nuclear reactor problems.

4.3.1. Classical optimization techniques

The classical optimization techniques are useful for single as well as multi dimensional optimization problems. Few popular classical optimization techniques are described below.

4.3.1.1. Direct methods

Direct methods are simple brute force approaches to exploit the nature of the function. These methods do not require evolution of derivatives at any points. For one-dimensional problem golden-section search or quadratic interpolation method can be used whereas, for multi-dimensional problem random search or univariate search method can be used.

The golden-section search [11] is a simple, general-purpose, single-variable search technique. The method starts with two initial guess (i.e. lower and upper bound points). The interior point is chosen according to the golden ratio ($\frac{\sqrt{5}-1}{2}$). The function is evaluated at new points and accordingly lower or upper bound point is changed.

Quadratic interpolation method [11] is based on the fact that there will be only one quadratic connecting three points and a quadratic polynomial often provides a good approximation to the objective function shape near an optimum.

As the name implies, the random search method repeatedly evaluates the function at randomly selected values of independent variables. If a sufficient number of samples are used, the optimum will eventually be located. This method works well even for the discontinuous and non-differentiable functions and it always finds global rather than local optimum point. The main drawback of this method is that it does not account the behavior of function at already evaluated points. Hence if the number of independent variables grows, the required amount of effort increases enormously. Hence it is not an efficient method. In univariate search method, change is made in one variable at a time to improve the approximation while the other variables are held constant. The univariate search method is more efficient than random search method. The advanced optimization methods provide more sophisticated search because they utilize the information gathered at previously solved points.

4.3.1.2. Gradient methods

The optimization method that uses knowledge of derivative information to locate optimum point is called gradient method. The first derivatives provide slope of the function being differentiated and at optima it become zero. The slope or gradient of the function tells what direction to move locally. For one-dimensional problem Newton's method use the following technique to find optimum of *f(x)*

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

The gradient of n-dimensional function $f(x_1, x_2, x_3, \dots x_n)$ can be represented as $\nabla f^T = [\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial x_3} \dots .]$. The iterative scheme followed in gradient method is $x_{k+1} = x_k + \propto_k . D_k$, where $\propto_k$ and D_k are step size and direction vector respectively. In the method of steepest descent ∇f is chosen as direction vector whereas, in conjugate gradient method the direction vector is chosen such that two successive direction vectors are conjugate to each other (they can never be in same direction). The appropriate step size is determined accordingly [11].

4.3.1.3. Linear programming methods

In Linear Programming the term linear connotes that the mathematical functions representing both the objective and constraints are linear. The term Programming connotes ‘scheduling’ or ‘setting an agenda’. For two or three dimensional linear programming problem graphical solution method can be used. The feasible solution region is determined by plotting equality and inequality constraints. The objective function can be plotted by another line for a particular point lying in feasible region. The point in feasible region for which objective function is optimum can be determined in this graphical representation method. This method has very limited practical utility. Simplex method [11] provides optima in an extremely efficient manner for linear programming problems. In the simplex method an external (slack) variable is added to convert the inequality constraints into equality constraints. The basic solution for m linear equations with n unknowns is developed by setting n-m variables to zero, and solving the m equations for m remaining unknowns. The zero variables are formally referred to as nonbasic variables, whereas the remaining m variables are called basic variables. If all the basic variables are non-negative, the result is called basic feasible solution. The optimum will be one of them.

4.3.1.4. Interior point methods

The interior point methods were popular during 1960s for solving nonlinearly constrained optimization problems because of the total dominance of simplex method for linear programming problems. After the Karmarkar’s fast interior method for linear programming, interior methods are playing a growing role in the study of all kind of optimization problems. Khatchian’s ellipsoid method and Karmarkar’s projective scaling method [12] are the interior point method which produce solution to a linear programming problem by moving through the interior of the feasible region. The time required to solve an linear programming problem of size *n* by the simplex method is of the order 2^n (known as exponential time algorithm) where, the time consumed by interior point method is of the order of n^i (i=2 or 3, known as polynomial time algorithm). The polynomial time algorithms are computationally superior to exponential algorithms for large linear programming problems.

4.3.2. Advanced optimization techniques

Most of the real world optimization problems involve complexities like discrete, continuous or mixed variables, multiple conflicting objectives, non-linearity, discontinuity etc. The search space may be so large that the global optimum can not be found in reasonable time. The classical methods may not be efficient to solve such problems. Various stochastic methods like simulated annealing or evolutionary optimization algorithms can be used in such situations.

4.3.2.1. Simulated annealing

The name and inspiration came from annealing process in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. The heat causes the atoms to become unstuck from their initial positions (a local minimum of the internal energy) and wander randomly through states of higher energy; the slow cooling gives them more chances of finding configurations with lower internal energy than the initial one. In the simulated annealing method [13, 14], each point of the search space is compared to a state of some physical system, and the function to be minimized is interpreted as the internal energy of the system in that state. Therefore the goal is to bring the system, from an arbitrary initial state, to a state with the minimum possible energy.

4.3.2.2. Evolutionary optimization algorithms

Evolutionary algorithms (EAs) are developed to arrive at near-optimum solutions to a large scale optimization problem. The problem having very large number of decision variables and non-linear objective functions are often solved by EAs. EAs mimic the metaphor of natural biological evolution or social behavior like how ants find the shortest route to a source of food and how birds find their destination during migration. The behavior of such species is guided by learning and adaptation. A flow chart diagram is shown in Fig.4.1. The evolutionary algorithms are based on population based search procedures that incorporate random variation and selection. The first evolutionary-based optimization technique was the genetic algorithm (GA) [15]. GA was developed based on the Darwinian principle of the survival of the fittest and the natural process of evolution through reproduction. There are so many algorithms like Particle Swarm Optimization (PSO) [16], Ant Colony Optimization (ACO) [17] and Estimation of Distribution Algorithm (EDA) [18] etc. have been introduced during the past 10 years.

EAs start from a population of possible solutions (called individuals) and move towards the optimal by incorporating generation and selection. Objects forming possible solution sets to the original problem are called phenotype and the encoding (representation) of the individuals in the EAs are called genotype. The way by which mapping of phenotype to genotype is done and the EA's operators are applied to genotype affects the computational time. An individual consist a genotype and a fitness function. Fitness represents the quality of the solution and forms the basis for selecting the individuals. A flow chart indicating the steps of a simple evolutionary algorithm is shown below.

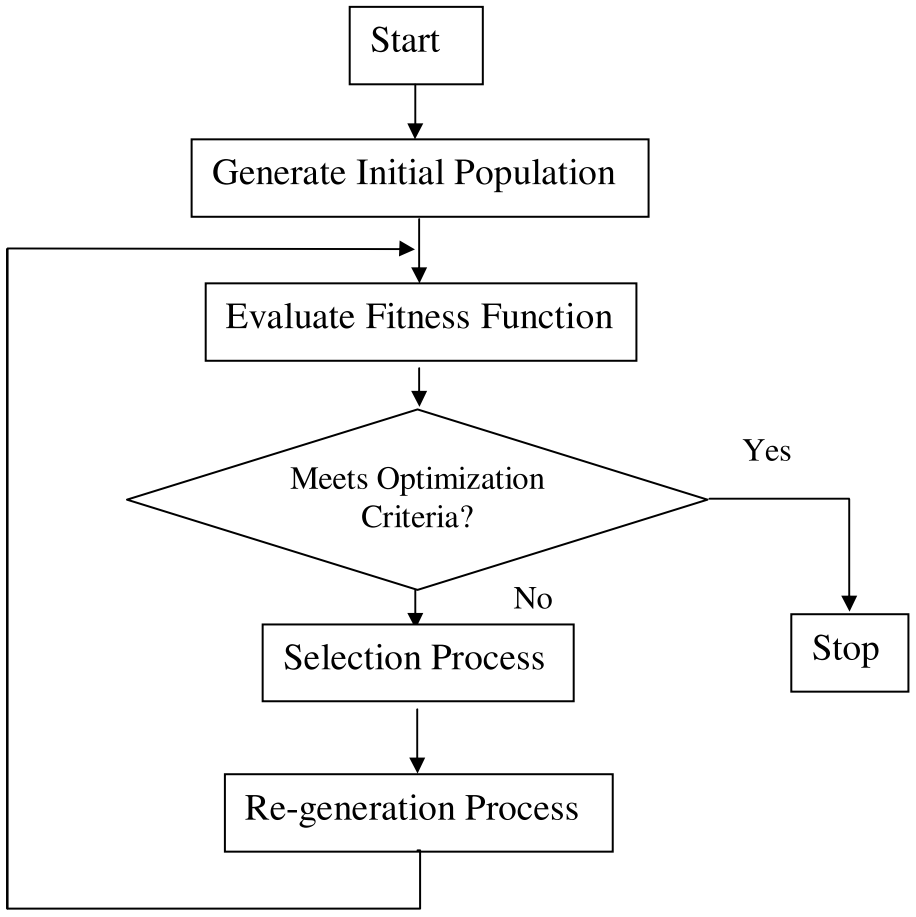
Start
Generate Initial Population
Evaluate Fitness Function
Meets Optimization Criteria?
Yes
No
Selection Process
Stop
Re-generation Process

Figure 4.1: Evolutionary algorithm flow chart

• Genetic algorithms (GA)

Genetic algorithm (GA) [15] improves fitness through evolution. A solution to a given problem is represented in the form of a string, called ‘chromosome’, consisting of a set of elements, called ‘genes’ that holds a set of values for the optimization variables. GAs works with a random population of solutions. The fitness of each chromosome is determined by evaluating it against an objective function. To simulate the natural survival of the fittest process, best chromosomes exchange information (through crossover) to produce offspring chromosome. There are many methods to select the best chromosomes, for example roulette wheel selection, Boltzman selection, tournament selection, termination selection and others. The chromosome is then mutated by changing a few genes to create a new individual. The newly generated individuals are evaluated and used to evolve the population if they provide better solutions than weak population members. This process is continued for a large number of generations to obtain a near-optimum solution. Four main parameters affect the performance of GAs are population size, number of generations, crossover rate and mutation rate.

• Estimation of distribution algorithm (EDA)

EDAs [18] are motivated by the idea of discovering and exploiting the interaction between variables in the solution. In EDAs, the two GA operations of recombination and mutation are replaced by estimation of distribution and sampling. In each generation, N individuals are generated by sampling probability distribution. The fitness is evaluated for each of them and the best M individuals are selected to estimate probability distribution. The probability distribution function is updated slowly. The performance of an EDA highly depends on how well it estimates and samples the probability distribution.

• Some other algorithms

The Particle Swarm Optimization (PSO) [16] is inspired by the social behavior of a flock of migrating birds trying to reach an unknown destination. In PSO, each solution is a bird in the flock and is referred to as a particle. A particle is analogous to a chromosome in GAs. As opposed to GAs, the evolutionary process in the PSO does not create new birds from parent ones. Rather, the birds in the population only evolve their social behavior and accordingly their movement towards a destination. Each bird looks in a specific direction, and then when communicates together, they identify the bird that is in the best location. Accordingly, each bird speeds towards the best bird using a velocity that depends on its current position. Each bird, then investigates the search space from its new local position, and the process repeats until the flock reaches a desired destination.

Similar to PSO, ACO [17] population evolves not by their genetics but by their social behavior. In the real world, ants (initially) wander randomly, and upon finding food return to their colony while laying down pheromone trails. If other ants find such a path, they are likely not to keep traveling at random, but instead follow the trail laid by earlier ants, returning and reinforcing it, if they eventually find any food. Over time, however, the pheromone trail starts to evaporate, thus reducing its attractive strength. The more time it takes for an ant to travel down the path and back again, the more time the pheromones have to evaporate. A short path, by comparison, gets marched over faster, and thus the pheromone density remains high as it is laid on the path as fast as it can evaporate. Thus, when one ant finds a good (short) path from the colony to a food source, other ants are more likely to follow that path, and such positive feedback eventually leaves all the ants following a single path.

There are other evolutionary algorithms like shuffled frog leaping, cellular automata, virtual bee algorithms for solving optimization problems. In this thesis the two optimization algorithms i.e. genetic algorithm and estimation of distribution algorithm are used to solve the PHWR physics optimization problems.

• *Some interesting characteristics of evolutionary algorithms*

The evolutionary algorithms are based on selection of fitter individuals from the population and regeneration of new individuals guided by previous fitter individuals. In general if selection pressure is high (often called exploitation), the convergence of algorithm is fast. However, one may end up with an inferior result. On the other hand, if selection pressure is low (also called exploration), one is likely to get a superior result though convergence is poor. Here some kind of optimization is needed based on experience to decide selection pressure and diversity in population.

Another interesting property is the “no free lunch theorem” [15, 16, 17]. According to this, one algorithm can not be superior to other algorithms in all kinds of cases. Hence, for the class of problem being studied, one has to find out which algorithm is better. As will be seen later in present studies, we found that EDA is better than GA for problem at hand.