Chapter 2
Mathematical Optimization Techniques

Objectives The objectives of this chapter are:

- Explaining some optimization techniques.
- Explaining the minimum norm theorem and how it could be used as an optimization algorithm, where a set of equations can be obtained.
- Introducing the fuzzy system as an optimization technique.
- Introducing the simulated annealing algorithm (SAA) as an optimization technique.
- Introducing the tabu search algorithm (TSA) as an optimization technique.
- Introducing the genetic algorithm (GA) as an optimization technique.
- Introducing the particle swarm (PS) as an optimization technique.

2.1 Introduction

Growing interest in the application of Artificial Intelligence (AI) techniques to power system engineering has introduced the potential of using this state-of-the-art technology. AI techniques, unlike strict mathematical methods, have the apparent ability to adapt to nonlinearities and discontinuities commonly found in power systems. The best-known algorithms in this class include evolution programming, genetic algorithms, simulated annealing, tabu search, and neural networks.

In the last three decades many optimization techniques have been invented and successfully applied to the operation and control of electric power systems.

This chapter introduces the mathematical background behind the algorithms used in this book, without going deeply into the mathematical proofs of these algorithms, to help the reader understand the application of these algorithms. Different examples are offered, where they are needed, to help the reader understand a specific optimization algorithm.

S.A. Soliman and A.H. Mantawy, *Modern Optimization Techniques with Applications in Electric Power Systems*, Energy Systems,
DOI 10.1007/978-1-4614-1752-1_2,

2.2 Quadratic Forms [1]

An algebraic expression of the form

$$f_{(x,y)} = ax^2 + bxy + cy^2$$

is said to be a quadratic form. If we let

$$X = \begin{bmatrix} x \\ y \end{bmatrix}$$

Then we obtain

$$f_{(x,y)} = [x \quad y] \begin{bmatrix} a & \frac{b}{2} \\ \frac{b}{2} & c \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \text{or}$$

$$f_{(X)} = X^T AX$$

The above equation is in quadratic form. The matrix A in this form is a symmetrical matrix.

A more general form for the quadratic function can be written in matrix form as

$$F_{(X)} = X^T AX + B^T X + C$$

where X is an $n \times 1$ vector, A is an $n \times n$ matrix, B is an $n \times 1$ vector, and C is a 1×1 vector.

Example (2.1)

Given the function

$$f_{(x,y)} = 2x^2 + 4xy - y^2 = 0$$

it is necessary to write this function in a quadratic form.

Define the vector

$$X = \begin{bmatrix} x \\ y \end{bmatrix}$$

Then

$$f_{(x,y)} = [x \quad y] \begin{bmatrix} 2 & 2 \\ 2 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$F_{(x)} = X^T AX$$

where

$$A = \begin{bmatrix} 2 & 2 \\ 2 & -2 \end{bmatrix}$$

Example (2.2)

Obtain the quadratic form for the function

$$f_{(x_1,x_2)} = 3x_1^2 + 4x_1x_2 - 4x_2^2$$

Define the vector X as

$$X = [\, x_1 \quad x_2 \,]^T$$

Then

$$f_{(x_1,x_2)} = [\, x_1 \quad x_2 \,]^T \begin{bmatrix} 3 & 2 \\ 2 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Then

$$f_{(X)} = X^T AX$$

where

$$A = \begin{bmatrix} 3 & 2 \\ 2 & -4 \end{bmatrix}$$

Let A be an $n \times n$ matrix and let X be an $n \times 1$ vector. Then, irrespective of whether A is symmetric:

$$\begin{aligned} X^T AX &= \sum_{i=1}^{n} \sum_{j=1}^{n} x_i a_{ij} x_j \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} x_i a_{ji} x_j \end{aligned}$$

- An $n \times n$ matrix A is positive definite if and only if

$$X^T AX > 0 \quad (\forall X \in R^n, X \neq 0)$$

- and is positive semidefinite if and only if

$$X^T AX \geq 0 \quad (\forall X \in R^n)$$

- Similarly, A is negative definite if, and only if,

$$X^T AX < 0 \quad (\forall X \in R^n, X \neq 0)$$

- and A is negative semidefinite if and only if

$$X^T AX \leq 0 \quad (\forall X \in R^n)$$

2.3 Some Static Optimization Techniques [1–10]

In this section we discuss the general optimization problem without going into details of mathematical analysis. The first part of the section introduces unconstrained optimization, which has many applications throughout this book, and the second part introduces the constrained optimization problem. Generally speaking, the optimization problem has the form

$$\begin{aligned} &\text{Minimize} \\ &f(x_1, \ldots, x_n) \end{aligned} \tag{2.1}$$

$$\begin{aligned} &\text{Subject to} \\ &\phi_i(x_1, \ldots, x_n) = 0, (i = 1, \ldots, \ell) \end{aligned} \tag{2.2}$$

$$\Psi_j(x_1, \ldots, x_n) \leq 0, (j = 1, \ldots, m) \tag{2.3}$$

Equation 2.2 represents $\ell, \ell < n$, equality constraints, whereas Eq. 2.3 represents m inequality constraints. By using vector notation, we may express the general constrained optimization problem as follows.

$$\begin{aligned} &\text{Minimize} \\ &f(X) \end{aligned} \tag{2.4}$$

$$\begin{aligned} &\text{Subject to} \\ &\phi(X) = 0 \end{aligned} \tag{2.5}$$

$$\Psi(X) \leq 0, X \in R^n \tag{2.6}$$

The problem formulated in Eqs. 2.4, 2.5, and 2.6 is usually referred to as the general nonlinear programming problem. Any point X that satisfies these equations is called a feasible point.

2.3.1 Unconstrained Optimization

The calculus of variations is concerned with the determination of extrema (maxima and minima) or stationary values of functionals. A functional can be defined as a function of several other functions. The calculus of variations is a powerful method for the solution of problems in optimal economic operation of power systems. In this section we introduce the subject of variational calculus through a derivation of the Euler equations and associated transversality conditions.

In the unconstrained optimization problem, we need to find the value of the vector $X = [x_1, \ldots, x_n]^T$ that minimizes the function

$$f(x_1, \ldots, x_n) \tag{2.7}$$

provided that the function f is continuous and has a first-order derivative.

To obtain the minimum and/or maximum of the function f we set its first derivative with respect to x to zero

$$\frac{\partial f(x_1, \ldots, x_n)}{\partial x_1} = 0 \tag{2.8}$$

$$\frac{\partial f(x_1, \ldots, x_n)}{\partial x_2} = 0 \tag{2.9}$$

$$\vdots$$

$$\frac{\partial f(x_1, \ldots, x_n)}{\partial x_1} = 0 \tag{2.10}$$

Equations 2.8, 2.9, and 2.10 represent n equations in n unknowns. The solution of these equations produces candidate solution points. If the function f has second partial derivatives, then we calculate the Hessian matrix,

$$H = \frac{\partial^2 f(x_1, \ldots, x_n)}{\partial x_i^2}.$$

If the matrix H is positive definite, then the function f is a minimum at the candidate points, but if the matrix H is negative definite then f is a maximum at the candidate points. The following examples illustrate these steps.

Example (2.3)

Minimize

$$f(x_1, x_2) = x_1^2 + x_1 x_2 + x_2^2 \ \left(x \in R^2\right)$$

To obtain the candidate solution points, we have

$$\frac{\partial f(x_1, x_2)}{\partial x_1} = 2x_1 + x_2 = 0$$

and

$$\frac{\partial f(x_1, x_2)}{\partial x_2} = x_1 + 2x_2 = 0$$

Solving the above equations yields the candidate solution point as

$$\left[x_1^*, x_2^*\right]^T = [0, 0]^T$$

Next, we calculate the Hessian matrix using

$$H = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} \end{bmatrix}$$

to obtain

$$H = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

so

$$\begin{aligned} X^T H X &= [\, x_1 \quad x_2 \,] \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &= 2\left(x_1^2 + x_1 x_2 + x_2^2\right) \\ &= 2\left\{ \left(x_1 + \frac{1}{2} x_2\right)^2 + \frac{3}{4} x_2^2 \right\} \end{aligned}$$

Therefore

$$X^T H X > 0 \quad (\forall X \neq 0)$$

and so H is positive definite, and the function f is a minimum at the candidate point.

Note that the positive definiteness of H can also be verified just by calculating the values of the different determinants, produced from H as

$$\begin{aligned} \Delta_1(H) &= 2 = h_{11} \\ \Delta_2(H) &= (4 - 1) = 3 \end{aligned}$$

Because all Δs are positive, H is a positive definite matrix.

Example (2.4)

$$\text{Minimize}$$
$$f(x_1, x_2) = 34x_1^2 - 24x_1x_2 + 41x_2^2$$

Set the first derivatives to zero to obtain

$$\frac{\partial f(x_1, x_2)}{\partial x_1} = 68x_1 - 24x_2 = 0$$
$$\frac{\partial f(x_1, x_2)}{\partial x_2} = -24x_1 + 82x_2 = 0$$

The solution to the above equation gives

$$\begin{bmatrix} x_1^* \\ x_2^* \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Calculate the Hessian matrix as

$$H = \begin{bmatrix} 68 & -24 \\ -24 & 82 \end{bmatrix}$$

Check the definiteness for the Hessian matrix as

$$\Delta_1(H) = h_{11} = 68 \quad >0$$

$$\Delta_2(H) = 68 \times 82 - 24 \times 24 = 1328 \quad >0$$

Hence H is a positive definite matrix; or calculate the quadratic form:

$$\begin{aligned} X^T HX &= \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 68 & -24 \\ -24 & 82 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &= 68x_1^2 - 48x_1x_2 + 82x_2^2 \\ &= 2(4x_1 - 3x_2)^2 + 32x_1^2 + 64x_2^2 \end{aligned}$$

so

$$X^T HX > 0 \quad (\forall X \neq 0)$$

hence H is positive definite and f is a minimum at the feasible points.

Example (2.5)

$$\text{Minimize}$$
$$f(x_1, x_2) = x_1^3 - 2x_1^2 x_2 + x_2^2$$

We have

$$\frac{\partial f(x_1, x_2)}{\partial x_1} = 3x_1^2 - 4x_1 x_2 = 0$$
$$\frac{\partial f(x_1, x_2)}{\partial x_2} = -2x_1^2 + 2x_2 = 0$$

Solving the above two equations yields the critical points to be

$$x^* = \begin{bmatrix} x_1^* \\ x_2^* \end{bmatrix} = \begin{bmatrix} \frac{3}{4} \\ \frac{9}{16} \end{bmatrix}$$

The Hessian matrix is calculated as

$$H = \begin{bmatrix} (6x_1 - 4x_2) & -4x_1 \\ -4x_1 & 2 \end{bmatrix}$$

At the solution points, we calculate H as

$$H\left(x_1^*, x_2^*\right) = \begin{bmatrix} \frac{9}{4} & -3 \\ -3 & 2 \end{bmatrix}$$

$$\Delta_1(H) = \frac{9}{4} \quad > 0$$

$$\Delta_2(H) = \frac{18}{4} - 9 = -\frac{18}{4} \quad < 0$$

Hence $H\left(x_1^*, x_2^*\right)$ is positive semidefinite and so nothing can be concluded about the nature of the solution point x^*. The solution point in this case is called a saddle point.

2.3.2 Constrained Optimization

The problem examined in Sect. 2.3.1 excluded consideration of optimal control problems having constraint relationships between the scalar elements of the state trajectory, which occurs in many physical problems. The problem including such a

constraint can be formulated as follows. Find the function $x(t)$ that minimizes the following cost functional.

$$J = \int_{t_0}^{t_f} L[x(t), \dot{x}(t), t]dt \tag{2.11}$$

Subject to satisfying the following constraint:

$$\int_{t_0}^{t_f} g[x(t), \dot{x}(t), t]dt = 0 \tag{2.12}$$

An example of this problem that occurs in the power system area is the minimization of the fuel cost of a power plant subject to satisfying the active power balance equation for the system.

We can form an augmented cost functional by augmenting the cost functional of Eq. 2.11 by Eq. 2.12 via Lagrange's multiplier λ,

$$\tilde{J} = \int_{t_0}^{t_f} \tilde{L}[x(t)\dot{x}(t), t]dt \tag{2.13}$$

where

$$\tilde{L}(.) = L(.) + \lambda g(.) \tag{2.14}$$

As a result we obtain the following modified Euler equation.

$$\tilde{L}_{x(t)} - \frac{d}{dt}\left(\tilde{L}_{\dot{x}(t)}\right) = 0 \tag{2.15}$$

or

$$\left(L_{x(t)} + g_{x(t)}\right) - \frac{d}{dt}\left(L_{\dot{x}(t)} + \lambda g_{\dot{x}(t)}\right) = 0 \tag{2.16}$$

Example (2.6)

We wish to maximize

$$J = \int_{-1}^{1} x(t)dt$$

subject to satisfying

$$\int_{-1}^{1} \left[1 + \dot{x}^2(t)\right]^{1/2} dt = 1$$

The augmented cost functional is given by

$$\tilde{J} = \int_{-1}^{1} \left\{x(t) + \lambda \left[1 + \dot{x}^2(t)\right]^{1/2}\right\} dt$$

$$\tilde{L}(x, \dot{x}, t) = x(t) + \lambda \left[1 + \dot{x}^2(t)\right]^{1/2}$$

$$\tilde{L}_{x(t)} = \frac{\partial \tilde{L}(.)}{\partial x(t)} = 1$$

$$\tilde{L}_{\dot{x}(t)} = \frac{\partial \tilde{L}(.)}{\partial \dot{x}(t)} = \frac{\lambda \dot{x}(t)}{\left[1 + \dot{x}^2(t)\right]^{1/2}}$$

Substituting into Eq. 2.14, one obtains the following Euler equation,

$$1 - \lambda \frac{d}{dt} \frac{\dot{x}(t)}{\left[1 + \dot{x}^2(t)\right]^{1/2}} = 0$$

or

$$\frac{\dot{x}(t)}{[1 + \dot{x}(t)]^{1/2}} = \frac{1}{\lambda} t + C$$

The solution of the above equation is given by

$$(x - x_1)2 + (t - t_1)2 = r^2$$

where the parameters x_1, t_1, and r are chosen to satisfy the boundary conditions.

Another mathematical form of the constrained optimization problem has the form

Minimize

$$f(x_1, \ldots, x_n) \tag{2.17}$$

subject to satisfying

$$\phi_i(x_1, \ldots, x_n) = 0, (i = 1, \ldots, \ell) \tag{2.18}$$

and

$$\Psi_j(x_1, \ldots, x_n) \leq 0, (j = 1, \ldots, m) \tag{2.19}$$

Let us consider, for instance, the case when the objective function is subject only to equality constraints. We form the augmented objective function by adjoining the equality constraints to the function via Lagrange multipliers to obtain the alternative form.

Minimize

$$\tilde{f}(x_1, \ldots, x_n, \lambda_i) = f(x_1, \ldots .x_n) + \sum_{i=1}^{l} \lambda_i \phi_i(x_1, \ldots, x_n) \tag{2.20}$$

or, in vector form,

$$\tilde{f}(X, \lambda) = f(X) + \lambda^T \phi(x) \tag{2.21}$$

Putting the first derivative to zero, we obtain

$$\frac{\partial \tilde{f}(X, \lambda)}{\partial x_i} = \frac{\partial f(x)}{\partial x_i} + \sum_{j=1}^{\ell} \lambda j \frac{\partial \phi_j}{\partial x_i} = 0 \tag{2.22}$$

Equation 2.22 is a set of n equations in $(n + \ell)$ unknowns $(x_i; \quad i = 1, \ldots, n : \lambda_j; j = 1, \ldots, \ell)$. To obtain the solution, the equality constraints must be satisfied; that is,

$$\phi_i(x_1, \ldots, x_n) = 0 \quad i = 1, \ldots, \ell \tag{2.23}$$

Solving Eqs. 2.22 and 2.23, we obtain x_i^* and λ_j^*. This is illustrated in the following examples.

Example (2.7)

Minimize

$f(x_1, x_2) = x_1^2 + x_2^2$

Subject to

$\phi[x_1, x_2] = x_1 + 2x_2 + 1 = 0$

For this problem $n = 2$, $\ell = 1$, $(n + \ell = 3)$. The augmented cost function is given by

$$\tilde{f}(x_1, x_2, \lambda) = x_1^2 + x_2^2 + \lambda(x_1 + 2x_2 + 1)$$

Putting the first derivatives to zero gives

$$\frac{\partial \tilde{f}}{\partial x_1} = 0 = 2x_1^* + \lambda$$
$$\frac{\partial \tilde{f}}{\partial x_2} = 0 = 2x_2^* + 2\lambda$$

and

$$\frac{\partial \tilde{f}}{\partial \lambda} = 0 = x_1^* + 2x_2^* + 1 \quad \text{(equalityconstraint)}$$

Solving the above three equations gives

$$x_1^* = -\frac{1}{5}, x_2^* = -\frac{2}{5}, \lambda = \frac{2}{5}$$

Example (2.8)

Minimize
$\tilde{f}(x_1, x_2, \lambda) = \left(10 + 5x_1 + 0.2x_1^2\right) + \left(20 + 3x_2 + 0.1x_2^2\right)$
Subject to
$x_1 + x_2 = 10$

The augmented cost function is

$$\tilde{f}(x_1, x_2, \lambda) = \left(30 + 5x_1 + 0.2x_1^2 + 3x_2 + 0.1x_2^2\right) + \lambda(10 - x_1 - x_2)$$

Putting the first derivatives to zero we obtain

$$\frac{\partial \tilde{f}}{\partial x_1} = 0 = 5 + 0.4x_1^* - \lambda$$
$$\frac{\partial \tilde{f}}{\partial x_2} = 0 = 3 + 0.2x_2^* - \lambda$$
$$\frac{\partial \tilde{f}}{\partial \lambda} = 0 = 10 - x_1^* - x_2^*$$

Solving the above three equations gives

$$x_1^* = 0, x_2^* = 10 \text{ and } \lambda = 5$$

and the minimum of the function is

$$f(0, 10) = 30 + 30 + 10 = 70$$

If there are inequality constraints, then the augmented function is obtained by adjoining these inequality constraints via Kuhn–Tucker multipliers, to obtain

$$\tilde{f}(X, \lambda, \mu) = f(X) + \lambda^T \phi(X) + \mu^T \Psi(X) \tag{2.24}$$

Putting the first derivative to zero, we obtain

$$\frac{\partial \tilde{f}}{\partial X} = 0 = \frac{\partial f(X^*)}{\partial X} + \lambda^T \frac{\partial \phi(X^*)}{\partial X} + \mu^T \frac{\partial \Psi(X^*)}{\partial X} \tag{2.25}$$

and

$$\frac{\partial \tilde{f}}{\partial \lambda} = 0 = \phi(X^*) \tag{2.26}$$

with

$$\mu^T \Psi(X^*) = 0 \tag{2.27}$$

If $\Psi(X^*)>0$, then $\mu = 0$.

Solving the above equations gives the candidate solution (X^*, λ, μ).

Example (2.9)

Recall the previous example; we have

Minimize

$f(x_1, x_2) = 0.1x_2^2 + 0.2x_1^2 + 3x_2 + 5x_1 + 30$

Subject to the following constraints

$x_1 + x_2 = 10$

with

$$x_1 \geq 0$$
$$0 \leq x_2 \leq 15$$

We form the augmented function as

$$\tilde{f}(x_1, x_2, \lambda, \mu_1, \mu_2, \mu_3) = f(x_1, x_2) + \lambda(10 - x_1 - x_2) + \mu_1 x_1 + \mu_2 x_2 + \mu_3(15 - x_2)$$

Putting the first derivatives to zero leads to

$$\frac{\partial \tilde{f}}{\partial x_1} = 0 = \frac{\partial f}{\partial x_1} - \lambda + \mu_1 + \mu_2$$
$$\frac{\partial \tilde{f}}{\partial x_2} = 0 = \frac{\partial f}{\partial x_2} - \lambda + \mu_2 - \mu_3$$
$$\frac{\partial \tilde{f}}{\partial \lambda} = 0 = 10 - x_1 - x_2$$

with

$$\mu_1 x_1 = 0;$$
$$\mu_2 x_2 = 0;$$
$$\mu_3(15 - x_2) = 0$$

Now we have six equations for six unknowns, however, solving these equations is very difficult. We assume that none of the variables violates its limits; thus we obtain

$$\mu_1 = 0$$
$$\mu_2 = 0$$
$$\mu_3 = 0$$

and we must check the solution obtained for these conditions. The solution in this case is:

$$x_1^* = 0, \quad x_2^* = 10, \quad \lambda = 5$$

Indeed, as we see, the variables do not violate their limits, and the optimal solution in this case is

$$x_1^* = 0, \quad x_2^* = 10, \quad \lambda^* = 5$$

$$\mu_1^* = 0, \quad \mu_2^* = 0, \quad \mu_3^* = 0$$

However, if we change the second inequality constraint to be

$$0 \leq x_2 \leq 8$$

then we can see that for the solution above, $x_2^* = 10$ violates the upper limit. In this case we put

$$x_2^* = 8; \qquad \text{with } \mu_3\left(8 - x_2^*\right) = 0$$

and recalculate x_1^* as

$$x_1^* = 10 - x_2^* = 10 - 8$$
$$x_1^* = 2, \quad \left(x_1^* > 0\right)$$

Under this solution $\mu_3 \neq 0$, but $\mu_1^* = 0$ and $\mu_2^* = 0$. To calculate λ^* and μ_3^* we use the first two equations as

$$0 = 0.4x_1^* + 5 - \lambda^*$$

or

$$\begin{aligned}\lambda^* &= 0.4(2) + 5\\ &= 5.8\end{aligned}$$

and

$$0 = 0.2x_2^* + 3 - \lambda^* - \mu_3^*$$

or

$$\begin{aligned}\mu_3^* &= 1.6 + 3 - 5.8\\ &= -1.2\end{aligned}$$

2.4 Pontryagin's Maximum Principle [11–14]

Let $u(t)$ be an admissible control and $x(t)$ be the corresponding trajectory of the system described by

$$\dot{\mathrm{x}}(\mathrm{t}) = (\mathrm{x}(\mathrm{t}),\mathrm{u}(\mathrm{t}),\mathrm{t}) \tag{2.28}$$

Let $x(t_0)$, $t_{0,}$ and t_f be specified and $x(t_f)$ be free. The necessary conditions for $u(t)$ to be an optimal control, that is, to be the control that takes $x(t)$ from $x(0)$ to some state $x(tf)$ while minimizing the functional J

$$\mathrm{J} = \mathrm{G}\left[\mathrm{x}(\mathrm{t_f},\mathrm{t_f}) + \int_{\mathrm{t_0}}^{\mathrm{t_f}} \mathrm{L}[\mathrm{x}(\mathrm{t}),\mathrm{u}(\mathrm{t}),\mathrm{t}]\mathrm{dt}\right] \tag{2.29}$$

are as follows.

1. There exists a function or vector $\lambda(t)$ such that $x(t)$ and $\lambda(t)$ are the solutions of the following equations.

$$\dot{\mathrm{x}}(\mathrm{t}) = \frac{\partial \mathrm{H}}{\partial\, \lambda(\mathrm{t})} \tag{2.30}$$

$$\dot{\lambda}(\mathrm{t}) = -\frac{\partial \mathrm{H}}{\partial \mathrm{x}(\mathrm{t})} \tag{2.31}$$

subject to the boundary conditions given by

$$\mathrm{x}(\mathrm{t_0}) = \mathrm{x}(0) \tag{2.32}$$

$$\lambda(t_f) = \left.\frac{\partial G(.)}{\partial x(t)}\right|_{t=t_f} \quad \text{at} \quad x(t) = x(t_f) \tag{2.33}$$

where the function H is a scalar function, which is called the Hamiltonian and is given by

$$H[x(t), u(t), \lambda(t), t] = L[x(t), u(t), t] + \lambda^T(t) f[x(t), u(t), t] \tag{2.34}$$

2. The functional $H[x(t), u(t), \lambda(t), t]$ has a local minimum at

$$\frac{\partial H}{\partial u(t)} = 0 \tag{2.35}$$

In many practical problems, there are inequality constraints on the admissible control and states and these constraints must be taken into account. As a result Eq. is no longer applicable. The basic contribution of the maximum principle addresses this difficulty. In place of Eq. the necessary condition is that the Hamiltonian function H $[x(t), u(t), \lambda(t), t]$ attains an absolute minimum as a function of $u(t)$ over the admissible region Ω for all t in the interval (t_0, t_f). In other words, the optimal $u\varepsilon$ satisfies

$$H[x(t), u_\varepsilon(t), \lambda(t), t] \leq H[x(t), u(t), \lambda(t), t] \quad u_\varepsilon \in \Omega \tag{2.36}$$

Example (2.10)

A linear differential system is described by

$$\dot{x} = Ax + Bu$$

where

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad x^T = [x_1, x_1], \quad u^T = [u_1, u_2]$$

Find $u(t)$ such that

$$J = \tfrac{1}{2} \int_0^2 \|u\|^2 dt$$

is minimum, given $xT(0) = [1, 1]$ and $x1(2) = 0$.

Define the Hamiltonian H as

$$H = \tfrac{1}{2} u^T u + \lambda^T (Ax + Bu)$$

Equation 2.24 leads to the following costate equations.

$$\dot{\lambda}(\mathrm{t}) = -\frac{\partial \mathrm{H}}{\partial \mathrm{x}}$$

$$\dot{\lambda}(\mathrm{t}) = \mathrm{A}^{\mathrm{T}}\ \lambda$$

or

$$\dot{\lambda}_1(\mathrm{t}) = 0 \tag{a}$$

$$\dot{\lambda}(\mathrm{t}) = -\ \lambda_1(\mathrm{t}) \tag{b}$$

with

$$\lambda_1(2) = \ \lambda_2(2) = 0 \quad (\text{because G} = 0) \tag{c}$$

Integration of Eqs. (a) and (b) with the boundary Eq. (c) gives

$$\lambda_1(t) = \mathrm{C}_1 \tag{d}$$

$$\lambda_2(t) = \mathrm{C}_1(2 - t) \tag{e}$$

H has a local minimum at

$$\frac{\partial \mathrm{H}}{\partial \mathrm{u}(\mathrm{t})} = 0 \tag{f}$$

or

$$u(t) = -BT\ \lambda(t) \tag{g}$$

We have

$$\dot{\mathrm{x}} = \mathrm{Ax} + \mathrm{Bu}$$

Substituting for $u(t)$ from Eq. (g) in the above equation, we obtain

$$\dot{\mathrm{x}} = \mathrm{Ax} - \mathrm{BB}^{\mathrm{T}}\ \lambda \tag{h}$$

The solution of the above equation with Eqs. (d) and (e) gives

$$\mathrm{x}_{\mathrm{t}}(\mathrm{t}) = -\mathrm{C}_1\mathrm{t}^2 + \tfrac{1}{6}\mathrm{C}_1\mathrm{t}^3 + \mathrm{t} - \mathrm{C}_1\mathrm{t} + \mathrm{C}_2$$

$$\mathrm{x}_2(\mathrm{t}) = -2\mathrm{C}_1\mathrm{t} + \tfrac{1}{2}\mathrm{C}_1\mathrm{t}^2 + \mathrm{C}_3$$

Using the boundary conditions at $t = 0$, $x_T(0) = [1, 1]$ gives

$$1 = C_2$$

$$1 = C_3$$

The state $x_1(t)$ now is given by

$$x_1(t) = C_1 t^2 + \tfrac{1}{6} C_1 t^3 + t - C_1 t + 1$$

By using the boundary condition at $t = 2$, $x1(2) = 0$,

$$0 = -4C_1 + \tfrac{8}{6} C_1 + 2 - 2C_1 + 1$$

$$C_1 = \tfrac{9}{14}$$

We are now in a position to write the system equations that satisfied the boundary conditions as

$$\lambda_1(t) = \tfrac{9}{14}$$

$$\lambda_2(t) = \frac{9}{14}(2 - t)$$

$$x_1(t) = -\tfrac{9}{14} t^2 + \tfrac{3}{28} t^2 + \tfrac{5}{14} t + 1$$

$$x_2(t) = -\tfrac{9}{7} t + \tfrac{9}{28} t^2 + 1$$

$$u_1(t) = -\tfrac{9}{14}$$

$$u_2(t) = -\frac{9}{14}(2 - t)$$

Example (2.11)

For the fixed plant dynamics given by

$$\dot{x}(t) = u(t), x(0) = x_0,$$

determine the optimal closed-loop control that minimizes for fixed *tf*,

$$J = \tfrac{1}{2} S x^2(t_f) + \tfrac{1}{2} \int_0^{t_f} u(t)^2 dt$$

where S is an arbitrary constant. Do this by first determining the optimum open-loop control and trajectory and then let $u(t) = k(t)\, x(t)$.

Define the Hamiltonian as

$$\mathrm{H} = \tfrac{1}{2}\mathrm{u}^2(\mathrm{t}) + \lambda(\mathrm{t})\mathrm{u}(\mathrm{t})$$

We have

$$\lambda(\mathrm{t}) = -\frac{\partial \mathrm{H}}{\partial \mathrm{x}} = 0$$

Integrating directly we obtain

$$\lambda(\mathrm{t}) = \mathrm{C}_1 \tag{a}$$

From the boundary term $\mathrm{G} = \frac{1}{2}\mathrm{Sx}^2(\mathrm{t_f})$, we have

$$\begin{aligned} \lambda(\mathrm{t_f}) &= \left.\frac{\partial \mathrm{G}}{\partial \mathrm{x}(\mathrm{t})}\right|_{\mathrm{t}=\mathrm{t_f}} \\ \lambda(\mathrm{t_f}) &= \mathrm{Sx}(\mathrm{t_f}) \end{aligned} \tag{b}$$

Substituting from Eq. b into Eq. 2.31a at $t = tf$, $\lambda(tf) = Sx(tf)$,

$$\begin{aligned} \mathrm{C}_1 &= \mathrm{Sx}(\mathrm{t_f}) \\ \lambda(\mathrm{t}) &= \mathrm{Sx}(\mathrm{t_f}) \end{aligned} \tag{c}$$

H has a local minimum at

$$\frac{\partial \mathrm{H}}{\partial \mathrm{u}(\mathrm{t})} = 0$$

or

$$0 = u(t) + \lambda(t)$$

Hence, the open-loop control is given by

$$u(t) = -Sx(t_f) \tag{d}$$

Also, we have

$$\dot{\mathrm{x}}(\mathrm{t}) = \mathrm{u}(\mathrm{t})$$

or

$$\dot{\mathrm{x}}(\mathrm{t}) = -\mathrm{Sx}(\mathrm{t_f})$$

Integrating the above equation directly and by using at $t = 0$, $x(t) = x(0)$, we obtain

$$x(t) = -Sx(t_f)t + x(0) \tag{e}$$

Now, the closed-loop control is given by

$$\mathrm{u(t) = k(t)[-Sx(t_f)t + x(0)]}$$

or

$$\mathrm{u(t) = -Sk(t)\,x(t_f)t + k(t)\,x(0)}$$

2.5 Functional Analytic Optimization Technique [6]

The aim of this section is to discuss the application of one important minimum norm result as an optimization technique that has been used as a powerful tool in the solution of problems treated in this book [6, 15]. Before we do this, a brief discussion of relevant concepts from functional analysis is given.

2.5.1 Norms

A norm, commonly denoted by $\|\cdot\|$, is a real-valued and positive definite scalar. The norm satisfies the following axioms.

(1) $\|x\| \geq 0$ for all $x \in X$, $\|x\| = 0 \leftrightarrow x = 0$

(2) $\|x + y\| \leq \|x\| + \|y\|$ for each $x, y \in X$

(3) $\|\alpha x\| = |\alpha| \cdot \|x\|$ for all scalars α and each $\mathrm{x} \in \mathrm{X}$

A normed linear (vector) space $\boldsymbol{X}$ is a linear space in which every vector $\boldsymbol{x}$ has a norm (length). The norm functional is used to define a distance and a convergence measure

$$\mathrm{d(x,y) = \|x - y\|}$$

For example, let [0, T] be a closed bounded interval. The space of continuous functions $x(t)$ on [0, T] can have one of the following norms.

$$\mathrm{\|x\|_1 = \int_0^T |x(t)|dt}$$

$$\|x\|_2 = \left[\int_0^T |x(t)|^2 dt\right]^{1/2}$$

The normed linear space becomes a metric space under the (usual) metric [5].

2.5.2 Inner Product (Dot Product)

Consider the real three-dimensional Euclidean space E_3. A vector or element in E_3 is an ordered real triple $x = (x_1, x_2, x_3)$ in which the norm is defined by

$$\|x\| = \left(|x_1|^2 + |x_2|^2 + |x_3|^2\right)^{1/2}$$

From linear algebra in a Euclidean space E, if x is multiplied with another vector $y = (y_1, y_2, y_3)$, the result is (dot product)

$$(x, y) = x_1y_1 + x_2y_2 + x_3y_3$$

or

$$<x, y> = \sum_{i=1}^{3} x_i y_i$$

In the space E_3 the angle θ between x and y is also defined and is related to the norm by the equation

$$\langle x,y\rangle = \|x\| \cdot \|y\| \cos\theta$$

If the two vectors are orthogonal, $\theta = 90°$, then their inner product is zero

$$\langle x,y\rangle = 0$$

and they are collinear, $\theta = 0$, if their inner product is equal to

$$\langle x,y\rangle = \pm\|x\| \cdot \|y\|$$

In the complex space E_3 the inner product between any two vectors $x = (x_1, x_2, x_3)$ and $y = (y_1, y_2, y_3)$ can be defined by

$$\langle x,y\rangle = x_1y_1^* + x_2y_2^* + x_3y_3^*$$

Using these ideas as a background, let us now formulate the basic definition of an inner product in an abstract linear space.

Let X be a linear space. A rule that assigns a scalar $\langle x,y\rangle$ to every pair of elements $x, y \in X$ is called an inner product function if the following conditions are satisfied.

1. $\langle x,y\rangle = \langle y,x\rangle$
2. $\langle \alpha x + \beta y, z\rangle = \alpha\langle x,z\rangle + \beta\langle y,z\rangle$
3. $\langle\ \lambda x,y\rangle = \ \lambda\langle x,y\rangle$
4. $\langle x, y\rangle \geq 0, \quad \langle x, x\rangle = 0 \leftrightarrow x = 0$
5. $\langle x, x\rangle = \|x\|^2$

A linear space X is called a Hilbert space if X is an inner product space that is complete with respect to the norm induced by the inner product.

Equivalently, a Hilbert space is a Banach space whose norm is induced by an inner product. Let us now consider some specific examples of Hilbert spaces. The space *En* is a Hilbert space with inner product as defined by

$$\langle x, y\rangle = x^T y$$

or

$$\langle x, y\rangle = \sum_{i=1}^{n} x_i y_i$$

The space L_2 [0, T] is a Hilbert space with inner product

$$\langle x, y\rangle = \int_0^T x(t)y(t)dt$$

We have in this book a very useful Hilbert space. The elements of the space are vectors whose components are functions of time such as active power generation by the system units over the interval $[0, T_f]$. Given a positive definite matrix $B(t)$ whose elements are functions of time as well, we can define the Hilbert space $L_{2B}^{n}(0,T_f)$. The inner product in this case is given by

$$\langle V(t), (t)\rangle = \int_0^{T_f} V^T(t)B(t)U(t)dt$$

for every $V(t)$ and $U(t)$ in the space.

2.5.3 Transformations

Let X and Y be linear vector spaces and let D be a subset of X. A rule that associates with every element $x \in D$ and element $y \in Y$ is said to be a transformation from X to Y with domain D. If y corresponds to x under T, we write $y = T(x)$ and y is referred to as the image of x under T. Alternatively, a transformation is referred to as an operator.

The transformation $T: X \rightarrow Y$ is said to be linear if

$$\mathrm{T}(\alpha_1 \mathrm{x}_1 + \alpha_2 \mathrm{x}_2) = \alpha_1 \mathrm{T}(\mathrm{x}_1) + \alpha_2 \mathrm{T}(\mathrm{x}_2)$$

for all $\alpha 1, \alpha 2 \in \mathrm{R}$ (the real line) and for every $x1, x2 \in X$.

The linear operator T from a normed space X to a normed space Y is said to be bounded if there is a constant M such that $\|\mathrm{Tx}\| \leq \mathrm{M}\|\mathrm{x}\|$ for all $x \in X$. The normed space of all bounded linear operators from the normed space X into the normed space Y is denoted by $B\ (X, Y)$. Examples of bounded linear operators include one transformation useful for our purposes. This is T: $\mathrm{L}_{1\mathrm{B}}^{\mathrm{n}}(0, \mathrm{T_f}) \rightarrow \mathrm{R}^{\mathrm{m}}$ defined by $b = T\ [U\ (t)]$

$$\mathrm{b} = \int_0^{\mathrm{T_f}} \mathrm{M}^{\mathrm{T}} \mathrm{I}(\mathrm{t}) \mathrm{dt}$$

In practical applications there are many transformations; among these transformations we have [6] the following.

1. If two power plants supply a demand $PD(t)$ such that the active power balance equation is satisfied

$$P_D(t) = P_1(t) + P_2(t)$$

2. If time is not included as a parameter the above equation can be written as

$$P_D(t) = M^T P(t)$$

where

$$M = \mathrm{col}(1, 1)$$

$$P(t) = \mathrm{col}[P_1(t), P_2(t)]$$

This defines a transformation T: $\mathrm{L}_2^2(0, \mathrm{T_f}) \rightarrow \mathrm{L}_2(0, \mathrm{T_f})$ sending functions $[P_1(t), P_2(t)]$ into their image $PD(t)$. Observe that $T = MT$.

A functional is a transformation from a linear space into the space of real or complex scalars. A typical functional is the objective functional of optimal economy operation of m thermal plants given by

$$\mathrm{J}(\mathrm{P_s}) = \int_0^{\mathrm{T_f}} \sum_{\mathrm{i=1}}^{\mathrm{m}} \left[\alpha_1 + \beta_1 \mathrm{P_{si}(t)} + \gamma_1 \mathrm{P_{si}^2(t)}\right] \mathrm{dt}$$

In the above equation the space $\mathrm{L_2^m}(0, \mathrm{T_f})$ of thermal power generation vector functions is mapped into the real scalar space. In a linear space X, a functional f is linear if for any two vectors $x_1, x_2 \in X$, and any two scalars $\alpha 1$ and $\alpha 2$ we have

$$\mathrm{f}(\alpha_1 \mathrm{x}_1 + \alpha_1 \mathrm{x}_2) = \alpha_1 \mathrm{f}(\mathrm{x}_1) + \alpha_2 \mathrm{f}(\mathrm{x}_2)$$

On a normed space a linear functional f is bounded if there is a constant M such that $|f(x) \leq \mathrm{M}\|\mathrm{x}\|\|$ for every $x \in X$. The space of these linear functionals is a normed linear space X^* . X^* is the normed dual space of X and is a Banach space. If X is a Hilbert space then $X = X^*$. Thus Hilbert spaces are self-dual. For the normed linear space to be a reflexive, $X = X^{**}$. Any Hilbert space is reflexive.

Let X and Y be normed spaces and let $T \in B\ (X, Y)$. The adjoint (conjugate) operator T^*: $Y^* \rightarrow X^*$ is defined by

$$\langle \mathrm{x}, \mathrm{T}^* \mathrm{y} \rangle = \langle \mathrm{Tx}, \mathrm{y}^* \rangle$$

An important special case is that of a linear operator T: $H \rightarrow G$ where H and G are Hilbert spaces. If G and H are real, then they are their own duals and the operator T^* can be regarded as mapping G into H. In this case the adjoint relation becomes

$$\langle \mathrm{Tx}, \mathrm{y} \rangle = \langle \mathrm{x}, \mathrm{T}^* \mathrm{y} \rangle$$

Note that the left-hand side inner product is taken in G whereas the right-hand side inner product is taken in H.

Composite transformations can be formed as follows. Suppose T and G are transformations $T: X \rightarrow Y$ and $G: Y \rightarrow Z$. We define the transformation $GT: X \rightarrow Z$ by

$$(GT)(x) = G(T(x))$$

We then say that GT is a composite of G and T, respectively.

2.5.4 The Minimum Norm Theorem

With the above-outlined definitions in mind, we can now introduce one powerful result in optimization theory. The theorem described here is only one of a wealth of results that utilize functional analytic concepts to solve optimization problems effectively.

Theorem. *Let B and D be Banach spaces. Let T be a bounded linear transformation defined on B with values in D. Let* $\hat{u}$ *be a given vector in B. For each* ξ *in the range of T, there exists a unique element* $u\xi \in B$ *that satisfies*

$$\xi = Tu$$

while minimizing the objective functional

$$J(u) = \|u - \hat{u}\|$$

The unique optimal $u\xi \in B$ is given by

$$u_{\xi} = T^{+}[\xi - T\hat{u}] + \hat{u}$$

where the pseudoinverse operator $T+$ in the case of Hilbert spaces is given by

$$T+\ \xi = T^{*}[TT^{*}] - 1\ \ \xi$$

provided that the inverse of $TT*$ exists.

The theorem as stated is an extension of the fundamental minimum norm problem where the objective functional is

$$J(u) = \|u\|$$

The optimal solution for this case is

$$u\ \xi = T+\ \xi$$

with $T+$ being the pseudoinverse associated with T.

The above equations for the optimal control vector u can be obtained by using the Lagrange multiplier argument. The augmented cost functional can be obtained by adjoining to the cost functional the equality constraint via the Lagrange multiplier as follows.

$$\tilde{J}(u) = \|u - \hat{u}\|^2 + \langle\ \lambda, (\xi - Tu)\rangle$$

where λ is a multiplier (in fact $\lambda \in D$) to be determined so that the constraint $\xi = Tu$ is satisfied. By utilizing properties of the inner product we can write

$$\tilde{J}(u) = \|u - \hat{u} - T^{*}(\ \lambda/2)\|^2 - \|T^{*}(\ \lambda/2)\|^2 + \langle\ \lambda, \xi\rangle$$

Only the first norm of the above equation depends explicitly on the control vector u. To minimize $\tilde{J}$ we consider only

$$\tilde{J}(u) = \|u - \hat{u} - T^{*}(\ \lambda/2)\|$$

The minimum of the above equation is clearly achieved when

$$\mathrm{u}_{\xi} = \hat{\mathrm{u}} + \mathrm{T}^{*}(\ \lambda/2)$$

To find the value of (λ/2) we use the equality constraint

$$\xi = Tu\ \xi$$

which gives

$$(\ \lambda/2) = [\mathrm{TT}^{*}]^{-1}[\xi - \mathrm{T}\hat{\mathrm{u}}]$$

It is therefore clear that with an invertible TT^{*} we write

$$\mathrm{u}_{\xi} = \mathrm{T}^{*}[\mathrm{TT}^{*}]^{-1}[\xi - \mathrm{T}\hat{\mathrm{u}}]$$

which is the required result. In the above equation if $\hat{\mathrm{u}} = 0$ we obtain

$$u\ \xi = T^{*}[TT^{*}] - 1\ \ \xi$$

which is the same result obtained for the fundamental minimum norm problem.

In applying this result to our physical problem we need to recall two important concepts from ordinary constrained optimization. These are the Lagrange multiplier rule and the Kuhn–Tucker multipliers. An augmented objective functional is obtained by adjoining to the original cost functional terms corresponding to the constraints using the necessary multipliers. The object in this case is to ensure that the augmented functional can indeed be cast as a norm in the chosen space. A set of linear constraints on the control vector is singled out; this set, under appropriate conditions, defines the bounded linear transformation T.

2.6 Simulated Annealing Algorithm (SAA) [16–26]

Annealing is the physical process of heating up a solid until it melts, followed by cooling it down until it crystallizes into a state with a perfect lattice. During this process, the free energy of the solid is minimized. Practice shows that the cooling must be done carefully in order not to get trapped in a locally optimal lattice structure with crystal imperfections. In combinatorial optimization, we can define a similar process. This process can be formulated as the problem of finding (among a potentially very large number of solutions) a solution with minimal cost. Now, by establishing a correspondence between the cost function and the free energy, and between the solutions and physical states, we can introduce a solution method in the field of combinatorial optimization based on a simulation of the physical annealing

process. The resulting method is called simulated annealing (SA). The salient features of the SA method may be summarized as follows.

- It could find a high-quality solution that does not strongly depend on the choice of the initial solution.
- It does not need a complicated mathematical model of the problem under study.
- It can start with any given solution and try to improve it. This feature could be utilized to improve a solution obtained from other suboptimal or heuristic methods.
- It has been theoretically proved to converge to the optimum solution [16].
- It does not need large computer memory.

In this chapter we propose an implementation of a Simulated Annealing Algorithm (SAA) to solve the Unit Commitment Problem (UCP). The combinatorial optimization subproblem of the UCP is solved using the proposed SAA and the Economic Dispatch Problem (EDP) is solved via a quadratic programming routine. Two different cooling schedules are implemented and compared. Three examples are solved to test the developed computer model.

2.6.1 Physical Concepts of Simulated Annealing [79]

Simulated annealing was independently introduced by Kirkpatrick, Gela, and Vecchi in 1982 and 1983 [18] and Cerny in 1985 [19]. The slow cooling in annealing is achieved by decreasing the temperature of the environment in steps. At each step the temperature is maintained constant for a period of time sufficient for the solid to reach thermal equilibrium. At equilibrium, the solid could have many configurations, each corresponding to different spins of the electrons and to a specific energy level.

At equilibrium the probability of a given configuration, P_{confg}, is given by the Boltzman distribution,

$$P_{confg} = K.e^{\left(-\frac{E_{confg}}{Cp}\right)},$$

where E_{confg} is the energy of the given configuration and K is a constant [20].

Reference [21] proposed a Monte Carlo method to simulate the process of reaching thermal equilibrium at a fixed temperature Cp. In this method, a randomly generated perturbation of the current configuration of the solid is applied so that a trial configuration is obtained. Let E_c and E_t denote the energy level of the current and trial configurations, respectively. If $E_c > E_t$, then a lower energy level has been reached, and the trial configuration is accepted and becomes the current configuration. On the other hand, if $E_c \leq E_t$ then the trial configuration is accepted as the current configuration with probability $e^{[(E_c - E_t)/Cp]}$. The process continues where a transition to a higher energy level configuration is not necessarily rejected.

Eventually thermal equilibrium is achieved after a large number of perturbations, where the probability of a configuration approaches a Boltzman distribution. By gradually decreasing Cp and repeating Metropolis simulation, new lower energy levels become achievable. As Cp approaches zero, the least energy configurations will have a positive probability of occurring.

2.6.2 Combinatorial Optimization Problems

By making an analogy between the annealing process and the optimization problem, a large class of combinatorial optimization problems can be solved following the same procedure of transition from an equilibrium state to another, reaching the minimum energy level of the system. This analogy can be set as follows [20].

- Solutions in the combinatorial optimization problem are equivalent to states (configurations) of the physical system.
- The cost of a solution is equivalent to the energy of a state.
- A control parameter Cp is introduced to play the role of the temperature in the annealing process.

In applying the SAA to solve the combinatorial optimization problem, the basic idea is to choose a feasible solution at random and then get a neighbor to this solution. A move to this neighbor is performed if either it has a lower objective function value or, in the case of a higher objective function value, if $\exp(-\Delta E/Cp) \geq U(0,1)$, where ΔE is the increase in the objective function value if we moved to the neighbor. The effect of decreasing Cp is that the probability of accepting an increase in the objective function value is decreased during the search.

The most important part in using the SAA is to have good rules for finding a diversified and intensified neighborhood so that a large amount of the solution space is explored. Another important issue is how to select the initial value of Cp and how it should be decreased during the search.

2.6.3 A General Simulated Annealing Algorithm [16–26]

A general SAA can be described as follows.

Step (0): Initialize the iteration count $k = 0$ and select the temperature $Cp = Cp_o$ to be sufficiently high such that the probability of accepting any solution is close to 1.

Step (1): Set an initial feasible solution = current solution X_i with corresponding objective function value E_i.

Step (2): If the equilibrium condition is satisfied, go to Step (5); else execute Steps (3) and (4).

Step (3): Generate a trial solutionX_j, as a neighbor toX_i. Let E_j be the corresponding objective function value.

Step (4): Acceptance test: If $E_j \leq E_i$: accept the trial solution, set $X_i = X_j$, and go to Step (2). Otherwise: if $\exp[(E_i - E_j)/Cp] \geq U(0,1)$set $X_i = X_j$ and go to Step (2); else go to Step (2).

Step (5): If the stopping criterion is satisfied then stop; else decrease the temperature $Cp^k$and go to Step (2).

2.6.4 Cooling Schedules

A finite-time implementation of the SAA can be realized by generating homogeneous Markov chains of a finite length for a finite sequence of descending values of the control parameter Cp. To achieve this, one must specify a set of parameters that governs the convergence of the algorithm. These parameters form a cooling schedule. The parameters of the cooling schedules are as follows.

- An initial value of the control parameter.
- A decrement function for decreasing the control parameter.
- A final value of the control parameter specified by the stopping criterion.
- A finite length of each homogeneous Markov chain.

The search for adequate cooling schedules has been the subject of study in many papers [16, 23–26].

In this chapter, two cooling schedules are implemented, namely the polynomial-time and Kirk's cooling schedules. The description of these cooling schedules is presented in the following sections.

2.6.5 Polynomial-Time Cooling Schedule

This cooling schedule leads to a polynomial-time execution of the SAA, but it does not guarantee the convergence of the final cost, as obtained by the algorithm, to the optimal value. The different parameters of the cooling schedule are determined based on the statistics calculated during the search. In the following we describe these parameters [16, 23, 24].

2.6.5.1 Initial Value of the Control Parameter

The initial value of Cp, is obtained from the requirement that virtually all proposed trial solutions should be accepted. Assume that a sequence of m trials is generated at a certain value of Cp. Let m_1 denote the number of trials for which the objective function value does not exceed the respective current solution. Thus, $m_2 = m - m_1$ is the number of trials that result in an increasing cost.

It can be shown that the acceptance ratio X can be approximated by [16]:

$$X \approx (m_1 + m_2.\exp(-\overset{(+)}{\Delta f}/Cp))/(m_1 + m_2) \tag{2.37}$$

where, $\overset{(+)}{\Delta f}$ is the average difference in cost over the m_2 cost-increasing trials. From which the new temperature Cp is

$$\mathrm{Cp} = \overset{(+)}{\Delta \mathrm{f}}/\ln(\mathrm{m_2}/(\mathrm{m_2}.\mathrm{X} - \mathrm{m_1}(1 - \mathrm{X}) \tag{2.38}$$

2.6.5.2 Decrement of the Control Parameter

The next value of the control parameter,$\mathrm{Cp}^{\mathrm{k}+1}$, is related to the current value Cp^{k} by the function [16]:

$$\mathrm{Cp}^{\mathrm{k}+1} = \mathrm{Cp}^{\mathrm{k}}/(1 + (\mathrm{Cp}^{\mathrm{k}}.\ln(1 + \delta)/3\sigma\mathrm{Cp}^{\mathrm{k}} \tag{2.39}$$

where σ is calculated during the search. Small values of δ lead to small decrements in Cp. Typical values of δ are between 0.1 and 0.5.

2.6.5.3 Final Value of the Control Parameter

Termination in the polynomial-time cooling schedule is based on an extrapolation of the expected average cost at the final value of the control parameter. Hence, the algorithm is terminated if for some value of k we have [16, 23, 24]

$$\frac{Cp^k}{\langle f\rangle_\infty}.\frac{\partial\langle f\rangle_{Cp}}{\partial Cp}\bigg|_{Cp=Cp_k} < \varepsilon \tag{2.40}$$

where

$\langle f\rangle_\infty \approx \langle f\rangle_{Cpo}$ is the average cost at initial value of control parameter $\mathrm{Cp_o}$.

$\langle f\rangle_{Cp}$ is the average cost at the k<u>th</u> Markov chain.

$\frac{\partial\langle f\rangle_{Cp}}{\partial Cp}\Big|_{Cp=Cp_k}$ is the rate of change in the average cost at Cp^k.

ε is some small positive number. In our implementation $\varepsilon = 0.00001$.

2.6.5.4 The Length of Markov Chains

In [16] it is concluded that the decrement function of the control parameter, as given in Eq. 2.39, requires only a "small" number of trial solutions to rapidly approach the stationary distribution for a given next value of the control parameter. The word "small" can be specified as the number of transitions for which the algorithm has a sufficiently large probability of visiting at least a major part of the neighborhood of a given solution.

In general, a chain length of more than 100 transitions is reasonable [16]. In our implementation good results have been reached at a chain length of 150.

2.6.6 Kirk's Cooling Schedule

This cooling schedule was originally proposed by Kirkpatrick, Gelatt, and Vecchi [17]. It has been used in many applications of the SAA and is based on a number of conceptually simple empirical rules. The parameters of this cooling schedule are described in the following subsections [16, 18].

2.6.6.1 Initial Value of the Control Parameter

It is recommended to start with an arbitrary control parameter Cp [16]. If the percentage of the accepted trials solutions is close to 1, then this temperature is a satisfactory starting Cp. On the other hand, if this acceptance ratio is not close to 1, then Cp has to be increased iteratively until the required acceptance ratio is reached.

This can be achieved by starting off at a small positive value of Cp and multiplying it with a constant factor, larger than 1, until the corresponding value of the acceptance ratio, calculated from the generated transitions, is close to 1. In the physical system analogy, this corresponds to heating up the solid until all particles are randomly arranged in the liquid phase.

In our implementation, this procedure is accelerated by multiplying Cp by the reciprocal of the acceptance ratio.

2.6.6.2 Decrement of the Control Parameter

It is important to make "small" decrements in the values of the control parameter, to allow for a very slow cooling and consequently reaching an equilibrium at each value of the control parameter Cp. A frequently used decrement function is given by

$$\mathrm{Cp}^{k+1} = \alpha.\mathrm{Cp}^{k}, \quad k = 1, 2, \tag{2.41}$$

where α is a constant smaller than but close to 1. Typical values lie among 0.8 and 0.99.

2.6.6.3 Final Value of the Control Parameter

Execution of the algorithm is terminated if the value of the cost function of the solution obtained in the last trial of the Markov chain remains unchanged for a number of consecutive chains (*Lm*). In our implementation, *Lm* is taken as 500 chains.

2.6.6.4 Length of the Markov Chain

The length of Markov chains L^k is based on the requirement that equilibrium is to be restored at each value of Cp. This is achieved after the acceptance of at least some fixed number of transitions. However, because the transitions are accepted with decreasing probability, one would obtain $L^k \rightarrow \infty$ as $Cp^k \rightarrow 0$. Consequently, L^k is bounded by some constant L_{max} to avoid extremely long Markov chains for small values of Cp^k. In this work, the chain length is guided by the changes of the best solution that has been obtained thus far. The chain length is assumed equal to 150 unless the best solution changes. If so, the chain length is extended by another 150 iterations. In the next section, two algorithms based on tabu search methods are described.

2.7 Tabu Search Algorithm

Tabu search is a powerful optimization procedure that has been successfully applied to a number of combinatorial optimization problems [22–46]. It has the ability to avoid entrapment in local minima. TS employs a flexible memory system (in contrast to "memoryless" systems, such as SA and GAs, and rigid memory systems as in branch-and-bound). Specific attention is given to the short-term memory (STM) component of TS, which has provided solutions superior to the best obtained by other methods for a variety of problems [30]. Advanced TS procedures are also used for sophisticated problems. These procedures include, in addition to the STM, intermediate-term memory (ITM), long-term memory (LTM), and strategic oscillations (SO). In this section, two algorithms based on the TS method are discussed. The first algorithm uses the STM procedure, and the second algorithm is based on advanced TS procedures.

In general terms, TS is an iterative improvement procedure that starts from some initial feasible solution and attempts to determine a better solution in the manner of a greatest descent algorithm. However, TS is characterized by an ability to escape local optima (which usually cause simple descent algorithms to terminate) by using a short-term memory of recent solutions. Moreover, TS permits backtracking to previous solutions, which may ultimately lead, via a different direction, to better solutions [31].

The main two components of a TSA are the TL restrictions and the aspiration level (AV) of the solution associated with these restrictions. Discussion of these terms is presented in the following sections.

2.7.1 Tabu List Restrictions

TS may be viewed as a "metaheuristic" superimposed on another heuristic. The approach undertakes to surpass local optimality by a strategy of forbidding (or, more broadly, penalizing) certain moves. The purpose of classifying certain moves as forbidden – that is, "tabu" – is basically to prevent cycling. Moves that

hold tabu status are generally a small fraction of those available, and a move loses its tabu status to become once again accessible after a relatively short time.

The choice of appropriate types of the tabu restrictions "list" depends on the problem under study. The elements of the TL are determined by a function that utilizes historical information from the search process, extending up to Z iterations in the past, where Z (TL size) can be fixed or variable depending on the application or the stage of the search.

The TL restrictions could be stated directly as a given change of variables (moves) or indirectly as a set of logical relationships or linear inequalities. Usage of these two approaches depends on the size of the TL for the problem under study.

A TL is managed by recording moves in the order in which they are made. Each time a new element is added to the "bottom" of a list, the oldest element on the list is dropped from the "top." The TL is designed to ensure the elimination of cycles of length equal to the TL size. Empirically [30], TL sizes that provide good results often grow with the size of the problem and stronger restrictions are generally coupled with smaller lists.

The way to identify a good TL size for a given problem class and choice of tabu restrictions is simply to watch for the occurrence of cycling when the size is too small and the deterioration in solution quality when the size is too large (caused by forbidding too many moves). The best sizes lie in an intermediate range between these extremes. In some applications a simple choice of Z in a range centered around 7 seems to be quite effective [28].

2.7.2 Aspiration Criteria

Another key issue of TS arises when the move under consideration has been found to be tabu. Associated with each entry in the TL there is a certain value for the evaluation function called the aspiration level. If the appropriate aspiration criteria are satisfied, the move will still be considered admissible in spite of the tabu classification. Roughly speaking, AV criteria are designed to override tabu status if a move is "good enough" with the compatibility of the goal of preventing the solution process from cycling [28]. Different forms of aspiration criteria are available. The one we use in this study is to override the tabu status if the tabu moves yield a solution that has a better evaluation function than the one obtained earlier for the same move.

2.7.3 Stopping Criteria

There may be several possible stopping conditions for the search. In our implementation we stop the search if either of the following two conditions is satisfied.

- The number of iterations performed since the best solution last changed is greater than a prespecified maximum number of iterations.
- The maximum allowable number of iterations is reached.

2.7.4 General Tabu Search Algorithm

In applying the TSA to solve a combinatorial optimization problem, the basic idea is to choose a feasible solution at random and then get a neighbor to this solution. A move to this neighbor is performed if either it does not belong to the TL or, in the case of being in the TL, it passes the AV test. During these search procedures the best solution is always updated and stored aside until the stopping criteria are satisfied.

A general TSA, based on the STM, for combinatorial optimization problems is described below, with this notation used in the algorithm:

X: The set of feasible solutions for a given problem
x: Current solution, $x \in X$
x'': Best solution reached
x': Best solution among a sample of trial solutions
$E(x)$: Evaluation function of solution x
$N(x)$: Set of neighborhood of $x \in X$ (trial solutions)
$S(x)$: Sample of neighborhood, of x, $S(x) \in N(x)$
$SS(x)$: Sorted sample in ascending order according to their evaluation functions, E (x)

Step (0): Set the TL as empty and the AV to be zero.
Step (1): Set iteration counter $K = 0$. Select an initial solution $x \in X$, and set $x'' = x$.
Step (2): Generate randomly a set of trial solutions $S(x) \in N(x)$ (neighbor to the current solution x) and sort them in an ascending order, to obtain $SS(x)$. Let x' be the best trial solution in the sorted set $SS(x)$ (the first in the sorted set).
Step (3): If $E(x') > E(x'')$, go to Step (4); else set the best solution $x'' = x'$ and go to Step (4).
Step (4): Perform the tabu test. If x' is not in the TL, then accept it as a current solution, set $x = x'$, update the TL and AV, and go to Step (6); else go to Step (5).
Step (5): Perform the AV test. If satisfied, then override the tabu state, set $x = x'$, update the AV, and go to Step (7); else go to Step (6).
Step (6): If the end of the $SS(x)$ is reached, go to Step (7); otherwise, let x' be the next solution in the SS(x) and go to Step (3).
Step (7): Perform the termination test. If the stopping criterion is satisfied then stop; else set $K = K + 1$ and go to Step (2).

The main steps of the TSA are also shown in the flowchart of Fig. 2.1.
In the following section we describe the details of the general TSA.

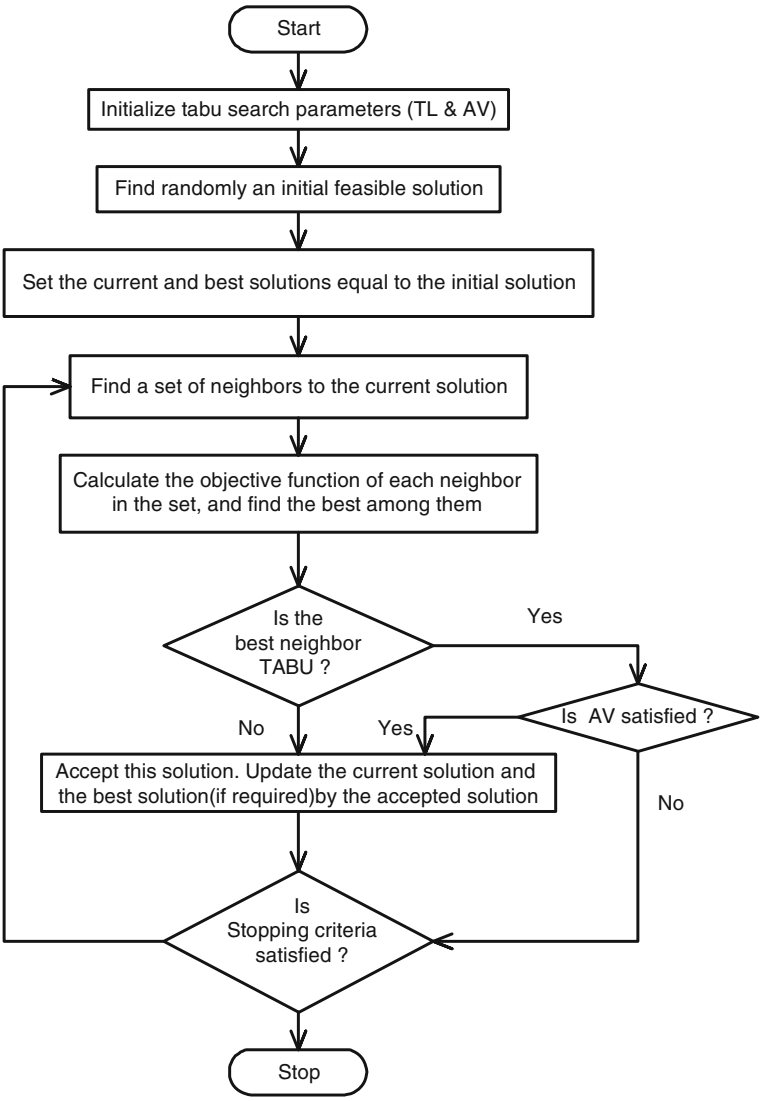

Fig. 2.1 Flowchart of a general Tabu search algorithm

2.8 The Genetic Algorithm (GA)

GAs are general-purpose search techniques based on principles inspired by the genetic and evolution mechanisms observed in natural systems and populations of living beings. Their basic principle is the maintenance of a population of solutions to a problem (genotypes) in the form of encoded individual information

that evolves in time. A GA for a particular problem must have the following five components [39, 40, 47].

- A genetic representation for a potential solution to the problem.
- A way to create an initial population of potential solutions.
- An evaluation function that plays the role of the environment, rating solutions in terms of their "fitness."
- Genetic operators that alter the composition of children.
- Values for various parameters that the GA uses (population size, probabilities of applying genetic operators, etc.)

A genetic search starts with a randomly generated initial population within which each individual is evaluated by means of a fitness function. Individuals in this and subsequent generations are duplicated or eliminated according to their fitness values. Further generations are created by applying GA operators. This eventually leads to a generation of high-performing individuals [44].

2.8.1 Solution Coding

GAs require the natural parameter set of the optimization problem to be coded as a finite-length string over some finite alphabet. Coding is the most important point in applying the GA to solve any optimization problem. Coding could be in a real or binary form. Coded strings of solutions are called "chromosomes." A group of these solutions (chromosomes) is called a population.

2.8.2 Fitness Function

The fitness function is the second important issue in solving optimization problems using GAs. It is often necessary to map the underlying natural objective function to a fitness function through one or more mappings. The first mapping is done to transform the objective function into a maximization problem rather than minimization to suit the GA concepts of selecting the fittest chromosome that has the highest objective function.

A second important mapping is the scaling of the fitness function values. Scaling is an important step during the search procedures of the GA. This is done to keep appropriate levels of competition throughout a simulation. Without scaling, there is a tendency early on for a few superindividuals to dominate the selection process. Later, when the population has largely converged, competition among population members is weaker and simulation tends to wander. Thus, scaling is a useful process to prevent both the premature convergence of the algorithm and the random improvement that may occur in the late iterations of the algorithm. There are many methods for scaling such as linear, sigma truncation, and power law scaling [42]. Linear scaling is the most commonly used. In the sigma truncation method,

population variance information to preprocess raw fitness values prior to scaling is used. It is called sigma (σ) truncation because of the use of population standard deviation information; a constant is subtracted from raw fitness values as follows:

$$\mathbf{f}^{'} = \mathbf{f} - (\mathbf{f}^{'} - \mathbf{c}.\sigma) \tag{2.41a}$$

In Eq. 2.41a the constant c is chosen as a reasonable multiple of the population standard deviation and negative results ($\mathbf{f}^{'} < 0$) are arbitrarily set to 0. Following sigma truncation, fitness scaling can proceed as described without the danger of negative results.

2.8.3 Genetic Algorithms Operators

There are usually three operators in a typical GA [44]. The first is the production operator which makes one or more copies of any individual that possesses a high fitness value; otherwise, the individual is eliminated from the solution pool.

The second operator is the recombination (also known as the "crossover") operator. This operator selects two individuals within the generation and a crossover site and performs a swapping operation of the string bits to the right-hand side of the crossover site of both individuals. The crossover operator serves two complementary search functions. First, it provides new points for further testing within the hyperplanes already represented in the population. Second, crossover introduces representatives of new hyperplanes into the population, which are not represented by either parent structure. Thus, the probability of a better performing offspring is greatly enhanced.

The third operator is the "mutation" operator. This operator acts as a background operator and is used to explore some of the unvisited points in the search space by randomly flipping a "bit" in a population of strings. Frequent application of this operator would lead to a completely random search, therefore a very low probability is usually assigned to its activation.

2.8.4 Constraint Handling (Repair Mechanism)

Constraint-handling techniques for the GAs can be grouped into a few categories [40]. One way is to generate a solution without considering the constraints but to include them with penalty factors in the fitness function. This method has been used previously [48–52].

Another category is based on the application of a special repair algorithm to correct any infeasible solution so generated.

The third approach concentrates on the use of special representation mappings (decoders) that guarantee (or at least increase the probability of) the generation of a feasible solution or the use of problem-specific operators that preserve feasibility of the solutions.

In our implementation, we are always generating solutions that satisfy the minimum up/down constraints. However, due to applying the crossover and mutation operations the load demand and/or the reserve constraints might be violated. A mechanism to restore feasibility is applied by randomly committing more units at the violated time periods and keeping the feasibility of the minimum up/down time constraints.

2.8.5 A General Genetic Algorithm

In applying the GAs to optimization problems, certain steps for simulating evolution must be performed. These are described as follows [39]:

Step (1): Initialize a population of chromosomes.
Step (2): Evaluate each chromosome in the population.
Step (3): Create new chromosomes by mating current chromosomes; apply mutation and recombination as the parent chromosomes mate.
Step (4): Delete members of the population to make room for the new chromosomes.
Step (5): Evaluate the new chromosomes and insert them into the population.
Step (6): If the termination criterion is satisfied, stop and return the best chromosomes; otherwise, go to Step (3).

2.9 Fuzzy Systems [78]

Human beings make tools for their use and also think to control the tools as they desire [53–60]. A feedback concept is very important to achieve control of the tools. As modern plants with many inputs and outputs become more and more complex, the description of a modern control system requires a large number of equations. Since about 1960 modern control theory has been developed to cope with the increased complexity of modern plants. The most recent developments may be said to be in the direction of optimal control of both deterministic and stochastic systems as well as the adaptive and learning control of time-variant complex systems. These developments have been accelerated by the digital computer.

Modern plants are designed for efficient analysis and production by human beings. We are now confronted by control of living cells, which are nonlinear, complex, time-variant, and "mysterious." They cannot easily be mastered by classical or control theory and even modern artificial intelligence (AI) employing

a powerful digital computer. Thus our problems are seen in terms of decision, management, and predictions. Solutions are seen in terms of faster access to more information and of increased aid in analyzing, understanding, and utilizing that information to discern its usefulness. These two elements, a large amount of information coupled with a large amount of uncertainty, taken together constitute the ground of many of our problems today: complexity. How do we manage to cope with complexity as well as we do and how could we manage to cope better? These are the reasons for introducing fuzzy notations, because the fuzzy sets method is very useful for handling uncertainties and essential for a human expert's knowledge acquisitions. First we have to know the meaning of fuzzy, which is vague or imprecise information.

Everyday language is one example of the ways in which vagueness is used and propagated such as driving a car or describing the weather and classifying a person's age, and so on. Therefore fuzzy is one method engineers use to describe the operation of a system by means of fuzzy variables and terms. To solve any control problem you might have a variable; this variable is a crisp set in the conventional control method (i.e., it has a definite value and a certain boundary). We define two groups as follows.

1. *Members:* Those that certainly belong in the set inside the boundary.
2. *Nonmembers:* Those that certainly don't.

But sometimes we have collections and categories with boundaries that seem vague and the transition from member to nonmember appears gradual rather than abrupt. These are what we call fuzzy sets. Thus fuzzy sets are a generalization of conventional set theory. Every fuzzy set can be represented by a membership function, and there is no unique membership. A membership function for any fuzzy set exhibits a continuous curve changing from 0 to 1 or vice versa, and this transition region represents a fuzzy boundary of the term.

In computer language we define fuzzy logic as a method of easily representing analog processes with continuous phenomena that are not easily broken down into discrete segments; the concepts involved are often difficult to model. In conclusion, we can use fuzzy when:

1. One or more of the control variables is continuous.
2. When a mathematical model of the process does not exist, or exists but is too difficult to encode.
3. When a mathematical model is too complex to be evaluated fast enough for real-time operation.
4. When a mathematical model involves too much memory on the designated chip architecture.
5. When an expert is available who can specify the rules underlying the system behavior and the fuzzy sets that represent the characteristics of each variable.
6. When a system has uncertainties in either the input or definition.

On the other hand, for systems where conventional control equations and methods are already optimal or entirely equal we should avoid using fuzzy logic.

One of the advantages of fuzzy logic is that we can implement systems too complex, too nonlinear, or with too much uncertainty to implement using traditional techniques. We can also implement and modify systems more quickly and squeeze additional capability from existing designs. Finally it is simple to describe and verify. Before we introduce fuzzy models we need some definitions.

- *Singletons*:
  A deterministic term or value, for example: male and female, dead and alive, 80°C, 30 Kg. These deterministic words and numerical values have neither flexibility nor intervals. So a numerical value to be substituted into a mathematical equation representing a scientific law is a singleton.
- *Fuzzy number*:
  A fuzzy linguistic term that includes an imprecise numerical value, for example, around 80°C, bigger than 25.
- *Fuzzy set*:
  A fuzzy linguistic term that can be regarded as a set of singletons where the grades are not only [1] but also range from [0 to 1]. It can also be a set that allows partial membership states. Ordinary or crisp sets have only two membership states: inclusion and exclusion (member or nonmember). Fuzzy sets allow degrees of membership as well. Fuzzy sets are defined by labels and membership functions, and every fuzzy set has an infinite number of membership functions (μFs) that may represent it.
- *Fuzzy linguistic terms*:
  Elements of which are ordered, are fuzzy intervals, and the membership function is a bandwidth of this fuzzy linguistic term. Elements of fuzzy linguistic terms such as "robust gentleman" or "beautiful lady" are discrete and also disordered. This type of term cannot be defined by a continuous membership function, but defined by vectors.
- *A characteristic function of*:
  Singletons, an interval and a fuzzy linguistic term are given by:

  (a) Singleton.
  (b) An interval.
  (c) Fuzzy linguistic term.

- *Control variable*:
  A variable that appears in the premise of a rule and controls the state of the solution variables.
- *Defuzzification*:
  The process of converting an output fuzzy set for a solution variable into a single value that can be used as an output.
- *Overlap*:
  The degree to which the domain of one fuzzy set overlaps that of another.
- *Solution fuzzy set*:
  A temporary fuzzy set created by the fuzzy model to resolve the value of a corresponding solution variable. When all the rules have been fired the solution fuzzy set is defuzzified into the actual solution variable.

- *Solution variable*:
  The variable whose value the fuzzy logic system is meant to find.
- *Fuzzy model*:
  The components of conventional and fuzzy systems are quite alike, differing mainly in that fuzzy systems contain "fuzzifers" which convert inputs into their fuzzy representations and "defuzzifiers" which convert the output of the fuzzy process logic into "crisp" (numerically precise) solution variables.

In a fuzzy system, the values of a fuzzified input execute all the values in the knowledge repository that have the fuzzified input as part of their premise. This process generates a new fuzzy set representing each output or solution variable. Defuzzification creates a value for the output variable from that new fuzzy set. For physical systems, the output value is often used to adjust the setting of an actuator that in turn adjusts the states of the physical systems. The change is picked up by the sensors, and the entire process starts again. Finally we can say that there are four steps to follow to design a fuzzy model.

1. First step: "Define the model function and operational characteristics"
   The goal is to establish the architectural characteristics of the system and also to define the specific operating properties of the proposed fuzzy system. The fuzzy system designer's task lies in defining what information (datapoint) flows into the system, what basic information is performed on the data, and what data elements are output from the system. Even if the designer lacks a mathematical model of the system process, it is essential that she have a deep understanding of these three phenomena. This step is also the time to define exactly where the fuzzy subsystem fits into the total system architecture, which provides a clear picture of how inputs and outputs flow to and from the subsystem. The designer can then estimate the number and ranges of input and output that will be required. It also reinforces the input-process-output design step.
2. The second step: "Define the control surfaces"
   Each control and solution variable in the fuzzy model is decomposed into a set of a fuzzy regions. These regions are given a unique name, called labels, within the domain of the variable. Finally a fuzzy set that semantically represents the concept associated with the label is created. Some rules of thumb help in defining fuzzy sets.

   (a) First, the number of labels associated with a variable should generally be an odd number from [56–58].
   (b) Second, each label should overlap somewhat with its neighbors. To get a smooth stable surface fuzzy controller, the overlap should be between 10% and 50% of the neighboring space. And the sum of vertical points of the overlap should always be less than one.
   (c) Third, the density of the fuzzy sets should be the highest around the optimal control point of the system, and this should be out as the distance from that point increases.

3. The third step: "Define the behavior of the control surfaces"
   This step involves writing the rules that tie the input values to the output model properties. These rules are expressed in natural language with syntax such as

   IF<fuzzy proposition>, then<fuzzy proposition>

   that is, IF, THEN rule, where fuzzy proposition are "x is y" or "x is not y"
   x is a scalar variable, and y is a fuzzy set associated with that variable. Generally the number of rules a system requires is simply related to the number of control variables.
4. The fourth step: "Select a method of defuzzification"
   It is a way to convert an output fuzzy set into a crisp solution variable. The two most common ways are:

   - The composite maximum.
   - Calculation of the concentration.

   Once the fuzzy model has been constructed, the process of solution and protocycling begins. The model is compared against known test cases to validate the results. When the results are not as desired, changes are made either to the fuzzy set descriptions or to the mappings encoded in the rules.

5. Fuzzy Sets and Membership
   Fuzzy set theory was developed to improve the oversimplified model, thereby developing a more robust and flexible model in order solve real-world complex systems involving human aspects. Furthermore, it helps the decision maker not only to consider the existing alternatives under given constraints (optimize a given system), but also to develop new alternatives (design a system). Fuzzy set theory has been applied in many fields, such as operations research, management science, control theory, artificial intelligence/expert systems, and human behavior, among others.

6. Membership Functions
   A classical (crisp or hard) set is a collection of distinct objects, defined in such a manner as to separate the elements of a given universe of discourse into two groups: those that belong (members), and those that do not belong (nonmembers). The transition of an element between membership and nonmembership in a given set in the universe is abrupt and well defined. The crisp set can be defined by the so-called characteristic function, for example, let U be a universe of discourse, the characteristic function of a crisp set.

2.9.1 Basic Terminology and Definition

Let X be a classical set of objects, called the universe, whose generic elements are denoted by x. The membership in a crisp subset of X is often viewed as the characteristic function μ_A from X to $\{0, 1\}$ such that:

$$\begin{aligned} \mu_A(x) &= 1 \text{ if and only if } x \in A \\ &= 0 \text{ otherwise} \end{aligned} \tag{2.42}$$

where $\{0, 1\}$ is called a valuation set.

If the valuation set is allowed to be the real interval [0, 1], $\tilde{A}$ is called a fuzzy set as proposed by Zadeh. $\mu_A(x)$ is the degree of membership of x in A. The closer the value of $\mu_A(x)$ is to 1, the more x belongs to A. Therefore, $\tilde{A}$ is completely characterized by the set of ordered pairs:

$$\tilde{A} = \{(x, \mu_A(x))|x \in X\} \tag{2.43}$$

It is worth noting that the characteristic function can be either a membership function or a possibility distribution. In this study, if the membership function is preferred, then the characteristic function is denoted as$\mu_A(x)$. On the other hand, if the possibility distribution is preferred, the characteristic function is specified as $\pi(x)$. When X is a finite set$\{x_1, x_2, ..., x_n\}$, a fuzzy set A is then expressed as

$$\tilde{A} = \mu_A(x_1)/x_1 + ... + \mu_A(x_n)/x_n = \sum_i \mu_A(x_i)/x_i \tag{2.44}$$

When X is not a finite set, A then can be written as

$$A = \int_X \mu_A(x)/x \tag{2.45}$$

Sometimes, we might only need objects of a fuzzy set (but not its characteristic function), in order to transfer a fuzzy set. To do so, we need two concepts: support and α-level cut.

2.9.2 *Support of Fuzzy Set*

The support of a fuzzy set A is the crisp set of all $x \in U$such that $(x) > 0$. That is,

$$\text{supp}(A) = \{x \in U|\mu_A{>}0\} \tag{2.46}$$

- **α-Level Set (α-Cut)**

  The α-level set (α-cut) of a fuzzy set A is a crisp subset of X and is shown in Fig. 2.2. An α-cut of a fuzzy set $\tilde{A}$ is a crisp set A which contains all the elements of the universe $\cup$ that have a membership grade in $\tilde{A}$ greater than or equal to α. That is,

$$A_\alpha = \{x|\mu_A(x) \geq \alpha \; and \; \text{x} \in \text{X}\} \tag{2.47}$$

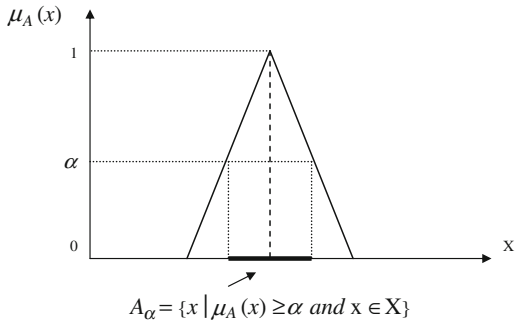

Fig. 2.2 Membership and α-cut

If $A_\alpha = \{x|\mu_A(x)>\alpha\}$, then A_α is called a strong α-cut of a given fuzzy set A is called a level set of A. That is,

$$\prod\nolimits_A = \{\alpha|\mu_A(x) = \alpha, \text{ for some } x \in \cup\} \tag{2.48}$$

2.9.3 Normality

A fuzzy set A is normal if and only if $\text{Sup}_x\ \mu_A(x) = 1$; that is, the supreme of $\mu_A(x)$over X is unity. A fuzzy set is subnormal if it is not normal. A nonempty subnormal fuzzy set can be normalized by dividing each $\mu_A(x)$ by the factor $\text{Sup}_x\mu_A(x)$. (A fuzzy set is empty if and only if $\mu_A(x) = 0$ for$\forall x \in X)\forall x \in X$.

2.9.4 Convexity and Concavity

A fuzzy set A in X is convex if and only if for every pair of point x^1 and x^2 in X, the membership function of A satisfies the inequality:

$$\mu_A(\partial x^1 + (1 - \partial)x^2) \geq \min(\mu_A(x^1), \mu_A(x^2)) \tag{2.49}$$

where $\partial \in [0, 1]$(see Fig. 2.3). Alternatively, a fuzzy set is convex if all α-level sets are convex.

Dually, A is concave if its complement A^c is convex. It is easy to show that if A and B are convex, so is $A \cap B$. Dually, if A and B are concave, so is $A \cup B$.

2.9.5 Basic Operations [53]

This section is a summary of some basic set-theoretic operations useful in fuzzy mathematical programming and fuzzy multiple objective decision making. These operations are based on the definitions from Bellman and Zadeh.

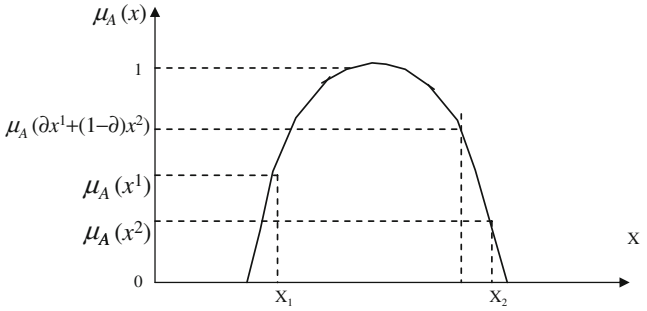

Fig. 2.3 A convex fuzzy set

2.9.5.1 Inclusion

Let A and B be two fuzzy subsets of X; then A is included in B if and only if:

$$\mu_A(x) \leq \mu_B(x) \text{ for } \forall x \in X \tag{2.50}$$

2.9.5.2 Equality

A and B are called equal if and only if:

$$\mu_A(x) = \mu_B(x) \text{ for } \forall x \in X \tag{2.51}$$

2.9.5.3 Complementation

A and B are complementary if and only if:

$$\mu_A(x) = 1 - \mu_B(x) \text{ for } \forall x \in X \tag{2.52}$$

2.9.5.4 Intersection

The intersection of A and B may be denoted by $A \cap B$ which is the largest fuzzy subset contained in both fuzzy subsets A and B. When the min operator is used to express the logic "and," its corresponding membership is then characterized by

$$\begin{aligned} \mu_{A\cap B}(x) &= \min(\mu_A(x), \mu_B(x)) \text{ for } \forall x \in X \\ &= \mu_A(x) \wedge \mu_B(x) \end{aligned} \tag{2.53}$$

where $\wedge$ is a conjunction.

2.9.5.5 Union

The union ($A \cup B$) of A and B is dual to the notion of intersection. Thus, the union of A and B is defined as the smallest fuzzy set containing both A and B.

The membership function of $A \cup B$ is given by

$$\begin{aligned}\mu_{A\cup B}(x) &= \max(\mu_A(x), \mu_B(x)) \text{ for } \forall x \in X \\ &= \mu_A(x) \vee \mu_B(x)\end{aligned} \tag{2.54}$$

2.9.5.6 Algebraic Product

The algebraic product AB of A and B is characterized by the following membership function,

$$\mu_{AB}(x) = \mu_A(x)\mu_B(x) \text{ for } \forall x \in X \tag{2.55}$$

2.9.5.7 Algebraic Sum

The algebraic sum $A \oplus B$ of A and B is characterized by the following membership function,

$$\mu_{A\oplus B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x) \tag{2.56}$$

2.9.5.8 Difference

The difference $A - B$ of A and B is characterized by

$$\mu_{A\cap B^c}(x) = \min(\mu_A(x), \mu_{B^c}(x)) \tag{2.57}$$

2.9.5.9 Fuzzy Arithmetic

(a) **Addition of Fuzzy Number**
The addition of X and Y can be calculated by using the α-level cut and max–min convolution.

- **α-level cut**. Using the concept of confidence intervals, the α-level sets of X and Y are $X_\alpha = [{X_\alpha}^L, {X_\alpha}^U]$ and $Y_\alpha = [{Y_\alpha}^L, {Y_\alpha}^U]$ where the result Z of the addition is:

$$Z_\alpha = X_\alpha(+)Y_\alpha = [{X_\alpha}^L + {Y_\alpha}^L, {X_\alpha}^U + {Y_\alpha}^U] \tag{2.58}$$

for every $\alpha \in [0, 1]$.

- **Max–Min Convolution.** The addition of the fuzzy number X and Y is represented as

$$Z(z) = \max_{z=x+y} [\min[\mu_X(x), \mu_Y(y)]] \tag{2.59}$$

(b) **Subtraction of Fuzzy Numbers**

- **α-level cut**. The subtraction of the fuzzy numbers X and Y in the α-level cut representation is:

$$Z_\alpha = X_\alpha(-)Y_\alpha = \left[{X_\alpha}^L - {Y_\alpha}^U, {X_\alpha}^U - {Y_\alpha}^L\right] \text{ for every } \alpha \in [0, 1]. \tag{2.60}$$

- **Max–Min Convolution.** The subtraction of the fuzzy number X and Y is represented as

$$\begin{aligned} \mu_Z(Z) = &\max_{z=x-y} \{[\mu_x(x), \mu_Y(y)]\} \\ &\max_{z=x+y} \{[\mu_x(x), \mu_Y(-y)]\} \\ &\max_{z=x+y} \{[\mu_x(x), \mu_{-Y}(y)]\} \end{aligned} \tag{2.61}$$

(c) **Multiplication of Fuzzy Numbers**

- **α-level cut**. The multiplication of the fuzzy numbers X and Y in the α-level cut representation is

$$Z_\alpha = X_\alpha(.)Y_\alpha = [[{X_\alpha}^L {y_\alpha}^L {X_\alpha}^U {Y_\alpha}^U]] \tag{2.62}$$

- for every $\alpha \in [0, 1]$.
- **Max–Min Convolution**. The multiplication of the fuzzy number X and Y is represented by Kaufmann and Gupta in the following procedure as

1. First, find $Z1$ (the peak of the fuzzy number Z) such that $\mu_Z(z^1) = 1$; then we calculate the left and right legs.
2. The left leg of $\mu_Z(z)$ is defined as

$$\mu_z(z) = \max_{xy \le z} \{\min[\mu_x(x), \mu_Y(y)]\} \tag{2.63}$$

3. The right leg of $\mu_Z(z)$ is defined as

$$\mu_z(z) = \max_{xy \ge z} \{\min[\mu_x(x), \mu_Y(y)]\} \tag{2.64}$$

(d) **Division of Fuzzy Numbers**

$$\boldsymbol{\alpha} - \textbf{level cut}.\ Z_\alpha = X_\alpha(:)Y_\alpha = [[x_{\alpha/}{}^{L} y_\alpha{}^{U}, x_{\alpha/}{}^{U} y_\alpha{}^{L}]] \tag{2.65}$$

- **Max–Min Convolution.** As defined earlier we must find the peak, then the left and right legs.

  1. The peak $Z = X\ (:)\ Y$ is used.
  2. The left leg is presented as

$$\begin{aligned} \mu_z(z) = & \max_{x/y \le z} \{\min[\mu_x(x), \mu_Y(y)]\} \\ & \max_{xy \le z} \{\min[\mu_x(x), \mu_Y(1/y)]\} \\ & \max_{xy \le z} \left\{\min[\mu_x(x), \mu_{1/Y}(y)]\right\} \end{aligned} \tag{2.66}$$

  3. The right leg is presented as

$$\begin{aligned} \mu_z(z) = & \max_{x/y \ge z} \{\min[\mu_x(x), \mu_Y(y)]\} \\ & \max_{xy \ge z} \{\min[\mu_x(x), \mu_Y(1/y)]\} \\ & \max_{xy \ge z} \left\{\min[\mu_x(x), \mu_{1/Y}(y)]\right\} \end{aligned} \tag{2.67}$$

2.9.5.10 *LR*-Type Fuzzy Number

A fuzzy number is defined to be of the *LR* type if there are reference functions L and R and positive scalars as shown in Fig. 2.4: α(left spread)**,** β(right spread), and m (mean) such that [53]:

$$\mu_M(x) = \left\{ \begin{array}{ll} L\left(\dfrac{m-x}{\alpha}\right) & for \quad \mathrm{x} \le \mathrm{m} \\ R\left(\dfrac{x-m}{\beta}\right) & for \quad \mathrm{x} \ge \mathrm{m} \end{array} \right\} \tag{2.68}$$

As the spread increases, M becomes fuzzier and fuzzier. Symbolically we write:

$$M = (m, \alpha, \beta)_{LR} \tag{2.69}$$

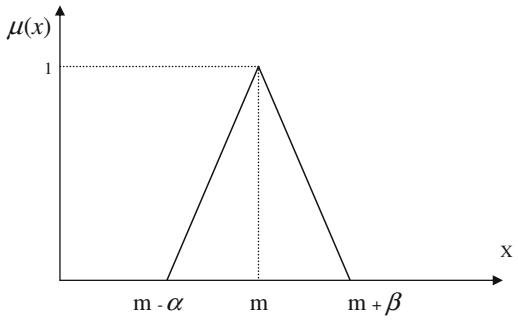

Fig. 2.4 Triangular membership

2.9.5.11 Interval Arithmetic

Interval arithmetic is normally used with uncertain data obtained from different instruments if we enclose those values obtained in a closed interval on the real line R; that is, this uncertain value is inside an interval of confidence R,$x \in [a_1, a_2]$, where $a_1 \leq a_2$.

2.9.5.12 Triangular and Trapezoidal Fuzzy Numbers

The triangular and trapezoidal fuzzy number is considered one of the most important and useful tools in solving possibility mathematical programming problems. Tables 2.1 and 2.2 show all the formulae used in the *L–R* representation of the fuzzy number and interval arithmetic methods.

2.10 Particle Swarm Optimization (PSO) Algorithm

One of the most difficult parts encountered in practical engineering design optimization is handling constraints [61–76]. Real-world limitations frequently introduce multiple nonlinear and nontrivial constraints in engineering design problems [77]. Constraints often limit the feasible solutions to a small subset of the design space. A general engineering optimization problem can be defined as

Minimize f (X), $X = \{x_1, x_2, \ldots, x_n\} \in R$
Subject to $g_i\,(X) \leq 0$, $i = 1, 2, \ldots, p$

$$h_i(X) = 0, \quad i = 1, 2, \ldots, m$$

Where

$$x_i^{(L)} \leq x_i \leq x_i^{(U)}, \; i = 1, 2, \ldots, n$$

Table 2.1 Fuzzy arithmetic on triangular *L–R* representation of fuzzy numbers $X = (x, \alpha, \beta)\&\ \ Y = (y, r, \delta)$

Image of Y: $\quad -Y = (-y, \delta, r) - Y = (-y, \delta, r)$
Inverse of Y: $\quad Y^{-1} = (y^{-1}, \delta y^{-2}, r y^{-2})$
Addition: $\quad X(+)Y = (x + y, \alpha + r, \beta + \delta)$
Subtraction: $\quad X(-)Y = X(+) - Y = (x - y, \alpha + \delta, \beta + r)$
Multiplication:
$X{>}0, Y{>}0 : X(\bullet)Y = (xy, xr + y\alpha, x\delta + y\beta)$
$X{<}0, Y{>}0 : X(\bullet)Y = (xy, y\alpha - x\delta, y\beta - xr)$
$X{<}0, Y{<}0 : X(\bullet)Y = (xy, -x\delta - y\beta, -xr - y\alpha)$
Scalar multiplication:
$a{>}0, a \in R : a(\bullet)X = (ax, a\alpha, a\beta)$
$a{<}0, a \in R : a(\bullet)X = (ax, -a\beta, -a\alpha)$
Division:
$X{>}0, Y{>}0 : X(:)Y = \left(x/y, (x\delta + y\alpha)/y^2, (xr + y\beta)/y^2\right)$
$X{<}0, Y{>}0 : X(:)Y = \left(x/y, (y\alpha - xr)/y^2, (y\beta - x\delta)/y^2\right)$
$X{<}0, Y{<}0 : X(:)Y = \left(x/y, (-xr - y\beta)/y^2, (-x\delta - y\alpha)/y^2\right)$

Table 2.2 Fuzzy interval arithmetic on triangular fuzzy numbers $X = (x^m, x^p, x^o)\ \&\ Y = (y^m, y^p, y^o)$

Image of Y: $\ -Y = (-y^m, -y^o, -y^p)$
Inverse of Y: $Y^{-1} = (1/y^m, 1/y^o, 1/y^p)$
Addition: $X(+)Y = (x^m + y^m, x^p + y^p, x^o + y^o)$
Subtraction: $X(-)Y = X(+) - Y = (x^m - y^m, x^p - y^o, x^o - y^p)$
Multiplication:
$X{>}0, Y{>}0 : X(\bullet)Y = (x^m y^m, x^p y^p, x^o y^o)$
$X{<}0, Y{>}0 : X(\bullet)Y = (x^m y^m, x^p y^o, x^o y^p)$
$X{<}0, Y{<}0 : X(\bullet)Y = (x^m y^m, x^o y^o, x^p y^p)$
Scalar multiplication:
$a{>}0, a \in R : a(\bullet)X = (ax^m, ax^p, ax^o)$
$a{<}0, a \in R : a(\bullet)X = (ax^m, ax^o, ax^p)$
Division:
$X{>}0, Y{>}0 : X(:)Y = (x^m/y^m, x^p/y^o, x^o/y^p)$
$X{<}0, Y{>}0 : X(:)Y = (x^m/y^m, x^o/y^o, x^p/y^p)$
$X{<}0, Y{<}0 : X(:)Y = (x^m/y^m, x^o/y^p, x^p/y^o)$

Due to the complexity and unpredictability of constraints, a general deterministic solution is difficult to find. In recent years, several evolutionary algorithms have been proposed for constrained engineering optimization problems and many methods have been proposed for handling constraints, the key point of the optimization process. Recently a new evolutionary computational technique, called particle swarm optimization (PSO) has been proposed and introduced [61–64].

Particle swarm optimization is a population-based stochastic optimization technique developed in [65], inspired by the social behavior of flocks of birds or schools of fish [65]. PSO shares many similarities with evolutionary computation techniques such as genetic algorithms. The system is initialized with a population of random feasible solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. The PSO algorithm has also been demonstrated to perform well on genetic algorithm test functions [66].

In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles [67]. The particles change their positions by flying around in a multidimensional search space until a relatively unchanged position has been encountered, or until computational limitations are exceeded [68]. In a social science context, a PSO system combines a social-only model and a cognition-only model. The social-only component suggests that individuals ignore their own experience and fine-tune their behavior according to the successful beliefs of the individual in the neighborhood. On the other hand, the cognition-only component treats individuals as isolated beings. A particle changes its position using these models.

Each particle keeps track of its co-ordinates in the problem space, which is associated with the best solution, fitness, it has achieved so far. The fitness value is also stored. This value is called *p*best. Another best value that is tracked by the particle swarm optimizer is the best value obtained thus far by any particle in the neighbors of the particle. This location is called *l*best. When a particle takes the whole population as its topological neighbors, the best value is a global best and is called *g*best.

The concept of the PSO consists of, at each timestep, changing the velocity of (accelerating) each particle toward its *p*best and *l*best locations (local version of PSO). Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward *p*best and *l*best locations. In the past few 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 compared with other methods. As shown in the literature, the PSO algorithm has been successfully applied to various problems [68–75].

Another reason that PSO is attractive is that there are few parameters to adjust. One version, with slight variations, works well in a wide variety of applications. Particle swarm optimization has been used for approaches that can be used across a wide range of applications, as well as for specific applications focused on a specific requirement. Many advantages of PSO over other traditional optimization techniques can be summarized as follows [68].

(a) PSO is a population-based search algorithm (i.e., PSO has implicit parallelism). This property ensures that PSO is less susceptible to being trapped on local minima.
(b) PSO uses payoff (performance index or objective function) information to guide the search in the problem space. Therefore, PSO can easily deal with

nondifferentiable objective functions. In addition, this property relieves PSO of assumptions and approximations, which are often required by traditional optimization models.

(c) PSO uses probabilistic transition rules and not deterministic rules. Hence, PSO is a kind of stochastic optimization algorithm that can search a complicated and uncertain area. This makes PSO more flexible and robust than conventional methods.

(d) Unlike the genetic and other heuristic algorithms, PSO has the flexibility to control the balance between global and local exploration of the search space. This unique feature of a PSO overcomes the premature convergence problem and enhances search capability.

(e) Unlike traditional methods, the solution quality of the proposed approach does not depend on the initial population. Starting anywhere in the search space, the algorithm ensures convergence to the optimal solution.

2.11 Basic Fundamentals of PSO Algorithm

The basic fundamentals of the PSO technique are stated and defined as follows [68].

1. **Particle *X*(*i*)**: A candidate solution represented by a k-dimensional real-valued vector, where k is the number of optimized parameters; at iteration i, the jth particle $X\ (i, j)$ can be described as

$$X_j(i) = [x_{j,1}(i); x_{j,2}(i); \ldots; x_{j,k}(i); \ \ldots \ \ldots .; x_{j,d}(i)] \tag{2.70}$$

   Where: xs are the optimized parameters
   $x_k\ (i,j)$ is the kth optimized parameter in the jth candidate solution
   d represents the number of control variables

2. **Population**: This is a set of n particles at iteration i.

$$\text{pop}\ (i) = [X_1(i), X_2(i), \ \ldots \ \ldots \ \ldots . X_n(i)]^T \tag{2.71}$$

   where n represents the number of candidate solutions.

3. **Swarm**: This is an apparently disorganized population of moving particles that tend to cluster together and each particle seems to be moving in a random direction.

4. **Particle velocity *V*(*i*)**: The velocity of the moving particles represented by a d-dimensional real-valued vector; at iteration i, the jth particle $V_j\ (i)$ can be described as

$$V_j(i) = [v_{j,1}(i); v_{j,2}(i); \ \ldots; v_{j,k}(i); \ \ldots \ \ldots .; v_{j,d}(i)] \tag{2.72}$$

where: $v_{j,k}(i)$ is the velocity component of the jth particle with respect to the kth dimension.

5. **Inertia weight $w(i)$**: This is a control parameter, used to control the impact of the previous velocity on the current velocity. Hence, it influences the tradeoff between the global and local exploration abilities of the particles. For initial stages of the search process, a large inertia weight to enhance global exploration is recommended whereas it should be reduced at the last stages for better local exploration. Therefore, the inertia factor decreases linearly from about 0.9 to 0.4 during a run. In general, this factor is set according to Eq. (2.73):

$$W = W_{\max} - \frac{(W_{\max} - W_{\min})}{iter_{\max}} * iter \tag{2.73}$$

where $iter_{\max}$ is the maximum number of iterations and $iter$ is the current number of iterations.

6. **Individual best $X^*(i)$**: During the movement of a particle through the search space, it compares its fitness value at the current position to the best fitness value it has ever reached at any iteration up to the current iteration. The best position that is associated with the best fitness encountered thus far is called the individual best $X^*(i)$. For each particle in the swarm, $X^*(i)$ can be determined and updated during the search. For the jth particle, individual best can be expressed as

$$X_j{}^*(i) = [x_{j1}{}^*(i), x_{j,2}{}^*(i), \ \ldots \ , \ x_{j,d}{}^*(i)]^T \tag{2.74}$$

In a minimization problem with only one objective function f, the individual best of the jth particle $X_j^*(i)$ is updated whenever $f(X_j^*(i)) < f(X_j^*(i-1))$. Otherwise, the individual best solution of the jth particle will be kept as in the previous iteration.

7. **Global best $X^{**}(t)$**: This is the best position among all of the individual best positions achieved thus far.
8. **Stopping criteria**: The search process will be terminated whenever one of the following criteria is satisfied.

   - The number of iterations since the last change of the best solution is greater than a prespecified number.
   - The number of iterations reaches the maximum allowable number.

The particle velocity in the kth dimension is limited by some maximum value, $v_k^{\max} v_k{}^{\max}$. This limit enhances local exploration of the problem space and it realistically simulates the incremental changes of human learning. The maximum velocity in the kth dimension is characterized by the range of the kth optimized parameter and given by

$$V_k{}^{\max} = \frac{(x_k{}^{\max} - x_k{}^{\max})}{N} \tag{2.75}$$

where N is a chosen number of intervals in the kth dimension.

2.11.1 General PSO Algorithm

In a PSO algorithm, the population has n particles that represent candidate solutions. Each particle is a k-dimensional real-valued vector, where k is the number of the optimized parameters [69]. Therefore, each optimized parameter represents a dimension of the problem space. The PSO technique steps can be described as follows.

Step 1: **Initialization**: Set $i = 0$ and generate random n particles $\{X_j\ (0), j = 1, 2, \ldots, n\}$. Each particle is considered to be a solution for the problem and it can be described as $X_j(0) = [x_{i,1}(0);\ x_{i,2}(0);\ \ldots,\ x_{i,k}(0)]$. Each control variable has a range $[x_{\min}, x_{\max}]$. Each particle in the initial population is evaluated using the objective function f. If the candidate solution is a feasible solution (i.e., all problem constraints have been met), then go to Step 2; else repeat this step.

Step 2: **Counter updating**: Update the counter $i = i + 1$.

Step 3: **Compute the objective function.**

Step 4: **Velocity updating**: Using the global best and individual best, the jth particle velocity in the kth dimension in this study (integer problem) is updated according to the following equation.

$$V(k,j,i+1) = w^*V(k,j,i) + C_1^* rand^*(pbestx(j,k) - x(k,j,i)) + C_2^* rand^*(gbestx(k) - x(k,j,i)) \tag{2.76}$$

Where i is the iteration number
j is the particle number
k is the kth control variable
w is the inertia weighting factor
c_1, c_2 are acceleration constants
rand () is a uniform random value in the range of [0,1]
$V(k,j,i)$ is the velocity of particle j at iteration i
$x(k,j,i)$ is the current position of particle j at iteration j

Then, check the velocity limits. If the velocity violated its limit, set it at its proper limit. The second term of the above equation represents the cognitive part of the PSO where the particle changes its velocity based on its own thinking and memory. The third term represents the social part of PSO where the particle changes its velocity based on the social–psychological adaptation of knowledge.

Step 5: **Position updating**: Based on the updated velocity, each particle changes its position according to Eq. 2.77 (Fig. 2.5).

$$x(k,j,i+1) = x(k,j-1,i) + v(k,j,i) \tag{2.77}$$

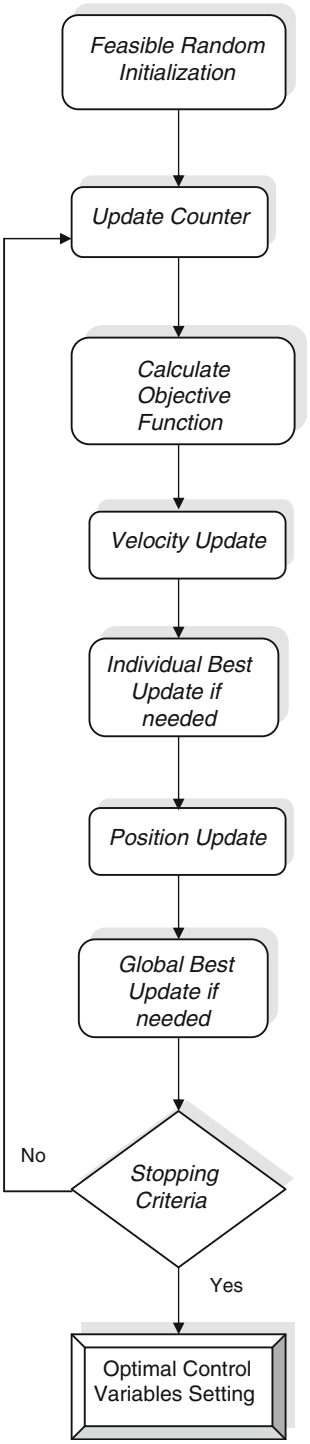

Fig. 2.5 Particle swarm optimization algorithm

Step 6: **Individual best updating**: Each particle is evaluated and updated according to the update position.

Step 7: Search for the minimum value in the individual best where its solution has ever been reached in every iteration and considered it as the minimum.

Step 8: **Stopping criteria**: If one of the stopping criteria is satisfied, then stop; otherwise go to Step 2.

References

1. El-Hawary, M.E., Christensen, G.S.: Optimal Economic Operation of Electric Power Systems. Academic, New York (1979)
2. Horst, R., Pardalos, P.M. (eds.): Handbook of Global Optimization. Kluwer, Netherlands (1995)
3. Kuo, B.C.: Automatic Control Systems, 4th edn. Prentice-Hall, Englewood Cliffs (1982)
4. Nemhauser, G.L., Rinnooy Kan, A.H.G., Todd, M.J. (eds.): Optimization. Elsevier Science, Netherlands (1989)
5. Wolfe, M.A.: Numerical Methods for Unconstrained Optimization: An Introduction. Van Nostrand Reinhold, New York (1978)
6. Zill, D.G., Cullen, M.R.: Advanced Engineering Mathematics. PWS, Boston (1992)
7. Porter, W.A.: Modern Foundations of Systems Engineering. Macmillan, New York (1966)
8. Luenberger, D.G.: Optimization by Vector Space Methods. Wiley, New York (1969)
9. Sage, A.: Optimum System Controls. Prentice-Hall, Englewood Cliffs (1968)
10. Sage, A.P., White, C.C.: Optimum Systems Control. Prentice-Hall, Englewood Cliffs (1977)
11. Bryson, A.E., Ho, Y.C.: Applied Optimal Control. Wiley, New York (1975)
12. Rao, S.S.: Optimization Theory and Applications. Wiley Eastern, New Delhi (1979)
13. Leitmann, G.: The Calculus of Variations and Optimal Control. Plenum Press, New York (1981)
14. Kirk, D.E.: Optimal Control Theory: An Introduction. Prentice-Hall, Englewood Cliffs (1970)
15. Narici, B.: Functional Analysis. Academic, New York (1966)
16. Aarts, E., Korst, J.: Simulated Annealing and Boltzman Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing. Wiley, New York (1989)
17. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**, 671–680 (1983)
18. Cerny, V.: Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. J. Optim. Theor. Appl. **45**(1), 41–51 (1985)
19. Selim, S.Z., Alsultan, K.: A simulated annealing algorithm for the clustering problem. Pattern Recogn. **24**(10), 1003–1008 (1991)
20. Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., Teller, E.: Equations of state calculations by fast computing machines. J. Chem. Phys. **21**, 1087–1092 (1953)
21. Tado, M., Kubo, R., Saito, N.: Statistical Physics. Springer, Berlin (1983)
22. Aarts, E.H.L., van Laarhoven, P.J.M.: Statistical cooling: a general approach to combinatorial optimization problems. Philips J. Res. **40**, 193–226 (1985)
23. Aarts, E.H.L., van Laarhoven, P.J.M.: A new polynomial time cooling schedule. In: Proceedings of the IEEE International Conference on Computer-Aided Design, pp. 206–208. Santa Clara (1985)
24. Aarts, E.H.L., van Laarhoven, P.J.M.: Simulated annealing: a pedestrian review of the theory and some applications. In: Devijver, P.A., Kittler, J. (eds.) Pattern Recognition Theory and Applications. NASI Series on Computer and Systems Sciences 30, pp. 179–192. Springer, Berlin (1987)

25. Glover, F., Greenberg, H.J.: New approach for heuristic search: a bilateral linkage with artificial intelligence. Eur. J. Oper. Res. **39**, 119–130 (1989)
26. Glover, F.: Future paths for integer programming and links to artificial intelligence. Comput. Oper. Res. **13**(5), 533–549 (1986)
27. Glover, F.: Tabu search-part I. Orsa J. Comput. **1**(3), 190–206 (1989). Summer
28. Glover, F.: Artificial intelligence, heuristic frameworks and tabu search. Manage. Decis. Econ. **11**, 365–375 (1990)
29. Glover, F.: Tabu search-part II. Orsa J. Comput. **2**(1), 4–32 (1990). Winter
30. Bland, J.A., Dawson, G.P.: Tabu search and design optimization. Comput. Aided Des. **23**(3), 195–201 (1991). April
31. Glover, F.: A user's guide to tabu search. Ann. Oper. Res. **41**, 3–28 (1993)
32. Laguna, M., Glover, F.: Integrating target analysis and tabu search for improved scheduling systems. Expert Syst. Appl. **6**, 287–297 (1993)
33. Kelly, J.P., Olden, B.L., Assad, A.A.: Large-scale controlled rounding using tabu search with strategic oscillation. Ann. Oper. Res. **41**, 69–84 (1993)
34. Barnes, J.W., Laguna, M.: A tabu search experience in production scheduling. Ann. Oper. Res. **41**, 141–156 (1993)
35. Charest, M., Ferland, J.A.: Preventive maintenance scheduling of power generating units. Ann. Oper. Res. **41**, 185–206 (1993)
36. Daniels, R.L., Mazzola, J.B.: A tabu search heuristic for the flexible-resource flow shop scheduling problem. Ann. Oper. Res. **41**, 207–230 (1993)
37. Amico, M.D., Trubian, M.: Applying tabu search to the jop-shop scheduling problem. Ann. Oper. Res. **41**, 231–252 (1993)
38. Mooney, E.L., Rardin, R.L.: Tabu search for a class of scheduling problems. Ann. Oper. Res. **41**, 253–278 (1993)
39. Davis, L. (ed.): Handbook of Genetic Algorithms. Van Nostrand, New York (1991)
40. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer, Berlin/Heidelberg/New York (1992)
41. Grefenstette, J.J.: Optimization of control parameters for genetic algorithms. IEEE Trans. Syst. Man Cybern. **16**(1), 122–128 (1986)
42. Grefenstette, J.J., Baker, J.E.: How genetic algorithm work: a critical look at implicit parallelism. In: The Proceedings of the Third International Conference on Genetic Algorithms. Morgan Kaufmann, San Mateo (1989)
43. Buckles, B.P., Petry, F.E., Kuester, R.L.: Schema survival rates and heuristic search in genetic algorithms. In: Proceedings of Tools for AI, pp. 322–327. Washington, DC (1990)
44. Awadh, B., Sepehri, N., Hawaleshka, O.: A computer-aided process planning model based on genetic algorithms. Comput. Oper. Res. **22**(8), 841–856 (1995)
45. Goldberg, D.E., Deb, K., Clark, J.H.: Genetic algorithms, noise, and the sizing of populations. Complex Syst. **6**, 333–362 (1992)
46. Homaifar, A., Guan, S., Liepins, G.E.: Schema analysis of the traveling salesman problem using genetic algorithms. Complex Syst. **6**, 533–552 (1992)
47. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesely, Reading Mass (1989)
48. Mantawy, H., Abdel-Magid, Y.L., Selim, S.Z.: A simulated annealing algorithm for unit commitment. IEEE Trans. Power Syst. **13**(1), 197–204 (1998)
49. Dasgupta, D., Mcgregor, D.R.: Thermal unit commitment using genetic algorithms. IEE Proc. Gener. Transm. Distrib. **141**(5), 459–465 (1994). September
50. Ma, X., El-Keib, A.A., Smith, R.E., Ma, H.: A genetic algorithm based approach to thermal unit commitment of electric power systems. Electr. Power Syst. Res. **34**, 29–36 (1995)
51. Kazarilis, S.A., Bakirtzis, A.G., Petridis, V.: A genetic algorithm solution to the unit commitment problem. IEEE Trans. Power Syst. **11**(1), 83–91 (1996). February

52. Yang, P.-C., Yang, H.-T., Huang, C.-L.: Solving the unit commitment problem with a genetic algorithm through a constraint satisfaction technique. Electr. Power Syst. Res. **37**, 55–65 (1996)
53. Ross, T.J.: Fuzzy Logic with Engineering Applications. McGraw-Hill, New York (1995)
54. Nazarka, J., Zalewski, W.: An application of the fuzzy regression analysis to the electrical load estimation. Electrotechnical Conference: MELECON'96, Bari, Italy, vol. 3, pp. 1563–1566. IEEE Catalog #96CH35884, 13–16 May 1996
55. Tanaka, H., Uejima, S., Asai, K.: Linear regression analysis with fuzzy model. IEEE Trans. Syst. Man Cybern. **12**(6), 903–907 (1983)
56. Chang, P.T., Lee, E.S.: Fuzzy least absolute deviations regression based on the ranking of fuzzy numbers. IEEE World Congress on Fuzzy Systems, Orlando, FL, USA, IEEE Proceeding, vol. 2, pp. 1365–1369 (1994)
57. Watada, J., Yabuchi, Y.: Fuzzy robust regression analysis. In: IEEE World Congress on Fuzzy Systems, Orlando, FL, USA, IEEE Proceeding, vol. 2, pp. 1370–1376 (1994)
58. Alex, R., Wang, P.Z.: A new resolution of fuzzy regression analysis. In: IEEE International Conference on Systems, Man, and Cybernetics, San Diego, California, USA, vol. 2, pp. 2019–2021. (1998)
59. Ishibuchi, H., Nii, M.: Fuzzy regression analysis by neural networks with non-symmetric fuzzy number weights. In: Proceedings of IEEE International Conference on Neural Networks, Washington, DC, USA, vol. 2, pp. 1191–1196 (1996)
60. Ghoshray, S.: Fuzzy linear regression analysis by symmetric triangular fuzzy number coefficients. In: Proceedings of IEEE International Conference on Intelligent Engineering Systems, Budapest, Hungary, pp. 307–313 (1997)
61. Hu, X., Eberhart, R.C., Shi, Y.: Engineering optimization with particle swarm. In: IEEE International Conference on Evolutionary Computation, pp. 53–57 (2003)
62. Kennedy, J.: The particle swarm: social adaptation of knowledge. In: Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97), pp. 303–8. Indianapolis (1997)
63. Angeline, P.: Evolutionary optimization versus particle swarm optimization: philosophy and performance differences. In: Proceedings of the 7th Annual Conference on Evolutionary Programming, San Diego, California, USA, pp. 601–10 (1998)
64. Shi, Y., Eberhart, R.: Parameter selection in particle swarm optimization. In: Proceedings of the 7th Annual Conference on Evolutionary Programming, San Diego, California, USA, pp. 591–600 (1998)
65. Stott, B., Hobson, E.: Power system security control calculation using linear programming. IEEE Transactions on Power Apparatus and Systems, vol. PAS-97, pp. 1713–1731 (1978)
66. Ozcan, E., Mohan, C.: Analysis of a simple particle swarm optimization system. Intell. Eng. Syst. Artif. Neural Networks **8**, 253–258 (1998)
67. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE International Conference on Evolutionary Computation, Perth, WA, Australia, pp. 1942–1948 (1995)
68. Eberhart, R.C., Shi, Y.: Comparing inertia weights and constriction factors in particle swarm optimization. In: IEEE International Conference on Evolutionary Computation, San Antonio, TX, USA, pp. 84–88 (2000)
69. Abido, M.A.: Optimal design of power-system stabilizers using particle swarm optimization. IEEE Trans. Energy Convers. **17**(3), 406–413 (2002). September
70. Gaing, Z.L.: Particle swarm optimization to solving the economic dispatch considering the generator constraints. IEEE Trans. Power Syst. **18**(3), 11871–195 (2003). August
71. Hirotaka, Y., Kawata, K., Fukuyama, Y.: A particle swarm optimization for reactive power and voltage control considering voltage security assessment. IEEE Trans. Power Syst. **15**(4), 1232–1239 (2000). November
72. Miranda, V., Fonseca, N.: EPSO-evolutionary particle swarm optimization, a new algorithm with applications in power systems. In: IEEE Trans. Power Syst. pp. 745–750 (2000)

73. Shi, Y., Eberhart, R.C.: A modified particle swarm optimizer. Proceedings of IEEE International Conference on Evolutionary Computation, pp. 69–73. Anchorage (1998)
74. Zhenya, H., et al.: Extracting rules from fuzzy neural network by particle swarm optimization. Proceedings of IEEE International Conference on Evolutionary Computation, pp. 74–77. Anchorage (1998)
75. Kennedy, J., Spears, W.: Matching algorithm to problems: an expermental test of the particle swarm optimization and some genetic algorithms on the multimodal problem generator. In: Proceedings of IEEE International Conference on Evolutionary Computation, pp. 78–83. Anchorage (1998)
76. Angeline, P.: Using selection to improve particle swarm optimization. In: Proceedings of IEEE International Conference on Evolutionary Computation, pp. 84–89. Anchorage (1998)
77. Talaq, J.H., El-Hawary, F., El-Hawary, M.E.: A summary of environmental/economic dispatch algorithms. IEEE Trans. Power Syst. **9**, 1508–1516 (1994). August
78. Soliman, S.A., Al-Kandari, M.A.: Electrical Load Forecasting; Modeling and Model Construction. Elsevier, New York (2010)
79. Mantawy, H., Abdel-Magid, Y.L., Selim, S.Z., Salah, M.A.: An improved simulated annealing algorithm for unit commitment-application to Sceco-East. In: 3rd International Conference on Intelligent Applications in Communications and Power Systems, IACPS'97, pp. 133–139. UAE (1997)

http://www.springer.com/978-1-4614-1751-4

Modern Optimization Techniques with Applications in Electric Power Systems

Soliman, S.A.-H.; Mantawy, A.-A.H.

2012, XVIII, 414 p., Hardcover
ISBN: 978-1-4614-1751-4