Chapter 2
Random Multiple Objective Decision Making

Mathematical programming with multiple objective functions (also called vector programming) is a recent development in mathematical programming, and emerged from an attempt to tackle the problems raised by the present development of science, engineering, industry, economics, etc. Due to the complexity of these problems, several objectives had to be incorporated in the optimization process. Basically, the problem consists of optimizing several objectives functions (some of which must be maximized, some minimized) provided the variables satisfy the linear and nonlinear constraints.

Nowadays, mathematical programming problems with uncertain parameters are extremely important and many scholars have begun to pay attention to these. In this section, we mainly discuss a random multiple objective decision making model. Many mathematical programming problems are stochastic because some of the data involved are random variables. This may be due to

1. Errors and variations in the problem parameters, which are often associated with probabilities.
2. Risk and uncertainty, which may sometimes allow a significant numerical representation of the utility function of the decision-maker (e.g. the von Neumann axiomatic system).
3. The need for optimum decision rules connected with some statistical decision functions, etc.

Hence, one has to study vector programming under the assumption that some of the problem parameters are random variables.

2.1 Distribution Center Location Problem with Random Phenomena

With the appearance of the importance of logistics in global economics. The importance of distribution location problems has emerged as being one of the most critical issues for logistics managers reducing cost and improving service. Manufactorers,

J. Xu and L. Yao, *Random-Like Multiple Objective Decision Making*, Lecture Notes in Economics and Mathematical Systems 647, DOI 10.1007/978-3-642-18000-2_2,

customers and suppliers are important members of the supply chain. To some extent, the success of an enterprise depends on its ability to link these members seamlessly. Distribution centers (abbr. DC) have the competency to connect manufactorers with their customers. The number, size and locations of distribution centers along with the decision on which customers to serve from each center (i.e. the allocation of customers to distribution centers) significantly affects the cost of the distribution system and the level of customer service provided by the system. In real logistics systems, the sources (factories, vendors, etc.) supply the distribution centers, which in turn supply the demand locations or consumers. Thus, it is necessary for manufactorers to evaluate and select suitable DC locations from a potential set.

In recent years, increasing production economies of scale and reducing transport costs have focused attention on distribution centers and many methods for location selection have been developed [17, 132, 164, 308, 320, 362]. An integer programming model for the plant location was presented by Barahona and Jensen [17]. They considered not only the fixed costs and transport costs, but also inventory costs, which had been solved by the Dantzig–Wolfe (D-W) decomposition method. Tragantalerngsak et al. [320] considered a two-echelon facility location problem in which the facilities in the first echelon are uncapacitated and the facilities in the second echelon are capacitated. The goal is to determine the number and locations of facilities in both echelons in order to satisfy customer demand for the product. Zhou et al. [362] investigated the balanced allocation of customers to multiple distribution centers with a genetic algorithm approach. Syam [308] proposed a model and methodologies for a location problem with logistical components. Harkness [132] investigated a new type of facility location model: one in which production costs per unit increase once a certain scale of output is reached. Klose and Drexl [164] reviewed some of the contributions to the current state of facility location models for distribution systems.

However, although the facility location problem has been studied widely, the majority of these papers assume that the design parameters of the DC location problem are deterministic. For real decision making, the situation is often not deterministic, and some factors such as demand, and costs of transport are usually changing, hence we must consider the DC location problem under an uncertain environment. In practice, many parameters for location models are subject to uncertainty. For example, queuing location models are discussed by Berman and Larson [22]. They give certain distribution functions for the customer arrival process, waiting, and approximate service times. Waiting times are a function of demand allocation and, hence, of facility location. Further stochastic location models are proposed by Laporte et al. [186] and Listes and Dekker [202]. Laporte et al. [186] considers customer demand as a stochastic variable and proposes the branch-and-cut algorithm to solve location problems with stochastic parameters. Listes and Dekker [202] use stochastic models with recourse for the purposes of locating facilities in product recovery networks. More recently, since Zadeh's pioneering work [356], a lot of research on fuzzy parameters have been done [35, 197, 225, 238, 340, 345]. Many successful applications of fuzzy optimization theory in the area of facility the location can be found in literature. Chen [48] proposes a fuzzy multiple

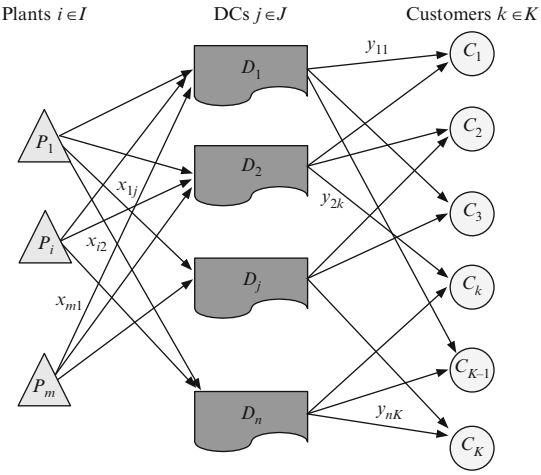

Fig. 2.1 Network model of logistics DCs location problem

criteria group decision-making method to solve distribution center location selection problems. Yang [351] investigates the logistics distribution centers location problem under a fuzzy environment and fuzzy chance-constrained programming is constructed as a decision model for the problem.

This chapter mainly concentrates on discussing a class of distribution center location problems with random parameters, i.e., customer demand, transportation costs and so on are considered as random variables. In fact, in real-life world, this case usually occurs. Let's recall a simple example proposed by Mirchandani et al. [220] (Fig. 2.1). They introduces a stochastic variant of the p-median problem. In particular, they consideres the demand and arc weights to be random variables. Under certain assumptions a finite number of states $i \in I$ of the graph with known probabilities can be enumerated. The model provided by them is as follows:

$$\begin{cases} \min \sum\limits_{i\in I}\sum\limits_{k\in K}\sum\limits_{j\in J} \pi_i c_{ikj} z_{ikj} \\ \text{s.t.} \begin{cases} \sum\limits_{j\in J} z_{ikj} = 1 \ \ \forall i \in I, j \in J \\ z_{ikj} - y_j \le 0 \ \ \forall i \in I, k \in K, j \in J \\ \sum\limits_{j\in J} y_j = p \\ z_{ikj}, y_j \in \mathbf{B} \ \ \forall i \in I, k \in K, j \in J \end{cases} \end{cases}$$

The symbol c_{ikj} denotes the demand weighted distance between nodes $k \in K$ and $j \in J$ in state $i \in I$. z_{ikj} are decision variables denoting demand allocation. y_j denotes location decisions. In this chapter, we will extend this model to be closer to the realistic problems and use the expected value operator and chance operator to resolve them. Readers can find these in the last section of this chapter.

2.2 Two Kinds of Random Variables

In the first chapter, we have introduced some basic properties about random variables. Here, we just research two special types of random variables which frequently appear in some realistic problems. One is the discrete, and the other is the continuous. Interested readers can refer to those related books such as [11, 76, 151, 152, 184, 281].

2.2.1 Discrete Random Variable

In this part, we consider certain types of discrete random variables.

Definition 2.1. Let ξ be a random variable on the probability space $(\Omega, \mathscr{A}, Pr)$. If $\Omega = \{\omega_1, \omega_2, \ldots\}$ is a set combined with finite or infinite discrete elements, where $Pr\{\omega = \omega_i\} = p_i$ and $\sum_{i=1}^{\infty} p_i = 1$, then ξ is called the discrete random variable.

From the above definition, we know that a discrete random variable ξ is a mapping from the discrete probability space Ω to the real space $\mathbf{R}$.

Example 2.1. Let $\Omega = \{1, 2, 3, 4\}$ be the probability space and $Pr\{\omega_i = i\} = 0.25$, $i = 1, \ldots, 4$. If $\xi(\omega_i) = 1/\omega_i$, then ξ is a discrete random variable.

Intuitively, we want to know what is the distribution of a discrete random variable. The following equation is usually used to describe the distribution,

$$\begin{pmatrix} \xi(\omega_1) & \xi(\omega_2) & \cdots & \xi(\omega_n) \\ p_1 & p_2 & \cdots & p_n \end{pmatrix}$$

In the following part, three special discrete random variables will be introduced.

As we know, in the trial of tossing a coin, the probabilities of the front and the back are all 0.5. Then we denote that $\omega_1 =$ 'Front' and $\omega_2 =$ 'Back' and let ξ be a mapping from $\{\omega_1, \omega_2\}$ to $\{0, 1\}$ satisfying $\xi(\omega_1) = 1$ and $\xi(\omega_2) = 0$.

Definition 2.2. (Binomial random variable) Suppose that n independent trials, each of which results in a "success" with probability p, are to be performed. If ξ represents the number of successes that occur in the n trials, then ξ is said to be a binomial random variable with parameters (n, p). Its probability mass function is given by

$$Pr\{\xi = i\} = \binom{n}{i} p^i (1-p)^{n-i}, \ i = 0, 1, \ldots, n \tag{2.1}$$

where

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}$$

is the binomial coefficient, equal to the number of different subsets of i elements that can be chosen from a set of n elements. Obviously, in this example, Ω has n elements combined with the natural number $i = 1, 2, \ldots, n$.

Since we assume that all trials are independent with each other, then the probability of any particular sequence of outcomes results in i successes and $n - i$ failures. Furthermore, it can be seen that (2.1) is valid since there are $\binom{n}{i}$ different sequences of the n outcomes that result in i successes and $n - i$ failures, which can be seen by noting that there are $\binom{n}{i}$ different choices of the i trials that result in successes.

Definition 2.3. A binomial random variable $(1, p)$ is called a Bernoulli random variable.

Since a binomial (n, p) random variable ξ represents the number of successes in n independent trials, each of which results in a success with probability p, we can represent it as follows:

$$\xi = \sum_{i=1}^{n} \xi_i \tag{2.2}$$

where

$$\xi_i = \begin{cases} 1, \text{ if the } i\text{th trial is a success} \\ 0, \text{ otherwise} \end{cases}$$

The following recursive formula expressing p_{i+1} in terms of p_i is useful when computing the binomial probabilities:

$$\begin{aligned} p_{i+1} &= \frac{n!}{(n-i-1)!(i+1)!} p^{i+1}(1-p)^{n-i-1} \\ &= \frac{n!(n-i)}{(n-i)!i!(i+1)} p^i (1-p)^{n-i} \frac{p}{1-p} \\ &= \frac{n-i}{i+1} \frac{p}{1-p} p_i . \end{aligned}$$

Definition 2.4. (Poisson random variable) A random variable ξ that takes on one of the values $0, 1, 2, \ldots$ is said to be a Poisson random variable with parameter λ, $\lambda > 0$, if its probability mass function is given by

$$p_i = Pr\{\xi = i\} = e^{-\lambda} \frac{\lambda^i}{i!}, \; i = 1, 2, \ldots \tag{2.3}$$

The symbol e, defined by $e = \lim_{n\to\infty}(1+1/n)^n$, is a famous constant in mathematics that is roughly equal to 2.7183.

Poisson random variables have a wide range of applications. One reason for this is that such random variables may be used to approximate the distribution of the number of successes in a large number of trials (which are either independent or at most "weakly dependent") when each trial has a small probability of being a success. To see why this is so, suppose that ξ is a binomial random variable with parameters (n, p), and so represents the number of successes in n independent trials when each trial is a success with probability p, and let $\lambda = np$. Then

$$\begin{aligned} Pr\{\xi = i\} &= \frac{n!}{(n-i)!\cdot i!}p^i(1-p)^{n-i} \\ &= \frac{n!}{(n-i)!\cdot i!}\left(\frac{\lambda}{n}\right)^i\left(1-\frac{\lambda}{n}\right)^{n-i} \\ &= \frac{n(n-1)\cdots(n-i+1)}{n^i}\cdot\frac{\lambda^i}{i!}\cdot\frac{(1-\lambda/n)^n}{(1-\lambda/n)^i} \end{aligned}$$

Now for n large and p small,

$$\begin{aligned} &\lim_{n\to\infty}\left(1-\frac{\lambda}{n}\right)^n \to e^{-\lambda}, \\ &\lim_{n\to\infty}\frac{n(n-1)\cdots(n-i+1)}{n^i} \to 1, \\ &\lim_{n\to\infty}\left(1-\frac{\lambda}{n}\right)^i \to 1. \end{aligned}$$

Hence, for n large and p small,

$$Pr\{\xi = i\} \approx e^{-\lambda}\frac{\lambda^i}{i!}.$$

To compute the Poisson probabilities we make use of the following recursive formula:

$$\frac{p_{i+1}}{p_i} = \frac{\frac{e^{-\lambda}\lambda^{(i+1)}}{(i+1)!}}{\frac{e^{-\lambda}\lambda^i}{i!}} = \frac{\lambda}{i+1}$$

or equivalently,

$$p_{i+1} = \frac{\lambda}{i+1}p_i, \quad i \geq 0.$$

Consider independent trials, each of which is a success with probability p. If ξ represents the number of the first trial that is a success, then

$$Pr\{\xi = n\} = p(1-p)^{n-1} \tag{2.4}$$

which is easily obtained by noting that in order for the first success to occur on the nth trial, the first $n-1$ must all be failures and the nth success. Equation (2.4) is said to be a geometric random variable with parameter p.

If we let ξ denote the number of trials needed to amass a total of r successes when each trial is independently a success with probability p, then ξ is said to be a negative binomial, sometimes called a Pascal random variable with parameters p and r. The probability mass function of such a random variable is given as follows,

$$Pr\{\xi = n\} = \binom{n-1}{r-1} p^r (1-p)^{n-r}, \quad n \geq r \tag{2.5}$$

To see why (2.5) is valid note that in order for it to take exactly n trials to amass r successes, the first $n-1$ trials must result in exactly $r-1$ successes, and the probability of this is $\binom{n-1}{r-1} p^r (1-p)^{n-r}$, and then the nth trial must be a success, and the probability of this is p.

Consider an urn containing $N+M$ balls, of which N are light colored and M are dark colored. If a sample of size n is randomly chosen (in the sense that each of the $\binom{N+M}{n}$ subsets of size n is equally likely to be chosen) then ξ, the number of light colored balls selected, has probability mass function,

$$Pr\{\xi = i\} = \frac{\binom{N}{i}\binom{M}{n-i}}{\binom{N+M}{n}}.$$

A random variable ξ whose probability mass function is given by the preceding equation is called a hypergeometric random variable.

2.2.2 *Continuous Random Variable*

In this part, we consider certain types of continuous random variables.

Definition 2.5. (Uniform random variable) A random variable ξ is said to be uniformly distributed over the interval (a, b), $a < b$, if its probability density function is given by

$$f(x) = \begin{cases} \dfrac{1}{b-a}, & \text{if } a < x < b, \\ 0, & \text{otherwise.} \end{cases}$$

In other words, ξ is uniformly distributed over (a, b) if it puts all its mass on that interval and it is equally likely to be "near" any point on that interval.

The distribution function of ξ is given, for $a < x < b$, by

$$F(x) = Pr\{\xi \leq x\} = \int_a^x (b-a)^{-1} dx = \frac{x-a}{b-a}.$$

Definition 2.6. (Normal random variable) A random variable ξ is said to be normally distributed with mean μ and variance σ^2 if its probability density function is given by

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2}, \quad -\infty < x < \infty.$$

The normal density is a bell-shaped curve that is symmetric about μ.

An important fact about normal random variables is that if ξ is normal with mean μ and variance σ^2, then for any constants a and b, $a\xi + b$ is normally distributed with mean $a\mu + b$ and variance $a^2\sigma^2$. It follows from this that if ξ is normal with mean μ and variance σ^2, then

$$\zeta = \frac{\xi - \mu}{\sigma}$$

is normal with mean 0 and variance 1. Such a random variable ζ is said to have a standard (or unit) normal distribution. Let Φ denote the distribution function of a standard normal random variable, that is

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^x e^{-x^2/2} dx, \quad -\infty < x < \infty.$$

The result that $\zeta = (\xi - \mu)/\sigma$ has a standard normal distribution when ξ is normal with mean μ and variance σ^2 is quite useful because it allows us to evaluate all probabilities concerning ξ in terms of Φ. For example, the distribution function of ξ can be expressed as

$$\begin{aligned} F(x) &= Pr\{\xi \leq x\} \\ &= Pr\left\{\frac{\xi - \mu}{\sigma} \leq \frac{x-\mu}{\sigma}\right\} \\ &= Pr\left\{\zeta \leq \frac{x-\mu}{\sigma}\right\} \\ &= \Phi\left(\frac{x-\mu}{\sigma}\right) \end{aligned}$$

The value of $\Phi(x)$ can be determined either by looking it up in a table or by writing a computer program to approximate it. For a in the interval $(0, 1)$, let ζ_a be such that

$$Pr\{\zeta > \zeta_a\} = 1 - \Phi(\zeta_a) = a.$$

That is, a standard normal will exceed ζ_a with probability a. The value of ζ_a can be obtained from a table of the values of Φ. For example, since

$$\Phi(1.64) = 0.95, \ \ \Phi(1.96) = 0.975, \ \ \Phi(2.33) = 0.99,$$

we see that

$$\zeta_{0.05} = 1.64, \ \ \zeta_{0.025} = 1.96, \ \ \zeta_{0.01} = 2.33.$$

The wide applicability of normal random variables results from one of the most important theorems of probability theory – the central limit theorem, which asserts that the sum of a large number of independent random variables has approximately a normal distribution.

Definition 2.7. (Exponential random variable) A continuous random variable having probability density function,

$$f(x) = \begin{cases} \lambda e^{-\lambda x}, & 0 \le x < \infty, \\ 0, & \text{otherwise.} \end{cases}$$

for some $\lambda > 0$ is said to be an exponential random variable with parameter λ.

Its cumulative distribution is given by

$$F(x) = \int_0^x \lambda e^{-\lambda x} dx = 1 - e^{-\lambda x}, \ 0 < x < \infty.$$

The key property of exponential random variables is that they possess the "memoryless property", where we say that the nonnegative random variable ξ is memoryless if

$$Pr\{\xi > s + t | \xi > s\} = Pr\{\xi > t\}, \text{ for all } s, t \ge 0. \tag{2.6}$$

To understand why the above is called the memoryless property, imagine that ξ represents the lifetime of some unit, and consider the probability that a unit of age s will survive an additional time t. Since this will occur if the lifetime of the unit exceeds $t + s$ given that it is still alive at time s, we see that

$$Pr\{\text{additional life of an item of age } s \text{ exceeds } t\} = Pr\{\xi > s + t | \xi > s\}.$$

Thus, (2.6) is a statement of fact that the distribution of the remaining life of an item of age s does not depend on s. That is, it is not necessary to remember the age of the unit to know its distribution of remaining life. Equation (2.6) is equivalent to

$$Pr\{\xi > s + t\} = Pr\{\xi > s\}Pr\{\xi > t\}.$$

As the above equation is satisfied whenever ξ is an exponential random variable – since, in this case, $Pr\{\xi > x\} = e^{-\lambda x}$ – we see that exponential random variables are memoryless (and indeed it is not difficult to show that they are the only memoryless random variables).

Another useful property of exponential random variables is that they remain exponential when multiplied by a positive constant. To see this suppose that ξ is exponential with parameter λ, and let c be a positive number. Then

$$Pr\{c\xi \leq x\} = Pr\left\{\xi \leq \frac{x}{c}\right\} = 1 - e^{-\lambda x/c},$$

which shows that $c\xi$ is exponential with parameter λ/c.

Let $\xi_1, \xi_2, \ldots, \xi_n$ be independent exponential random variables with respective rates $\lambda_1, \lambda_2, \ldots, \lambda_n$. A useful result is that $\min\{\xi_1, \xi_2, \ldots, \xi_n\}$ is exponential with rate $\sum_i \lambda_i$ and is independent of which one of the ξ_i is the smallest. To verify this, let $M = \min\{\xi_1, \xi_2, \ldots, \xi_n\}$. Then

$$\begin{aligned} Pr\left\{\xi_j = \min_i \xi_i | M > t\right\} &= Pr\{\xi_j - t = \min_i (\xi_i - t) | M > t\} \\ &= Pr\{\xi_j - t = \min_i (\xi_i - t) | \xi_i > t, i = 1, 2, \ldots, n\} \\ &= Pr\{\xi_j = \min_i \xi_i\}. \end{aligned}$$

The final equality follows because, by the lack of memory property of exponential random variables, given that ξ_i exceeds t, the amount by which it exceeds it is exponential with rate λ_i. Consequently, the conditional distribution of $\xi_1 - t, \ldots, \xi_n - t$ given that all the ξ_i exceed t is the same as the unconditional distribution of $\xi_1, \ldots, \xi_n$. Thus, M is independent of which of the ξ_i is the smallest.

The result that the distribution of M is exponential with rate $\sum_i \lambda_i$ follows from

$$Pr\{M > t\} = Pr\{\xi_i > t, i = 1, 2, \ldots, n\} = \prod_{i=1}^{n} Pr\{\xi_i > t\} = e^{-\sum_{i=1}^{n} \lambda_i t}.$$

The probability that ξ_j is the smallest is obtained from

$$\begin{aligned} Pr\{\xi_j = M\} &= \int Pr\{\xi_j = M | \xi_j = t\} \lambda_j e^{-\lambda_j t} dt \\ &= \int Pr\{\xi_j > t, i \neq j | \xi_j = t\} \lambda_j e^{-\lambda_j t} dt \\ &= \int Pr\{\xi_j > t, i \neq j\} \lambda_j e^{-\lambda_j t} dt \end{aligned}$$

$$= \int \left(\prod_{i \neq j} e^{-\lambda_i t} \right) e^{-\lambda_j t} dt$$
$$= \lambda_j \int e^{-\sum_i \lambda_i t} dt$$
$$= \frac{\lambda_j}{\sum_i \lambda_i}$$

There are also other special random variables following other distribution, readers can refer to the related literatures and we don't introduce here.

2.3 Random EVM

The *expectation* of a random variable is a central concept in the study of probability. It is the average of all possible values of a random variable, where a value is weighted according to the probability that it will appear. The expectation is sometimes also called *average*. It is also called the *expected value* or the *mean* of the random variable. These terms are all synonymous. The so-called *expected value model* (EVM) means to optimize some expected objective functions subject to some expected constraints, for example, minimizing expected cost, maximizing expected profit, and so on.

2.3.1 General Model for Random EVM

Now let us recall the well-known newsboy problem [206] in which a boy operating a news stall has to determine the number x of newspapers to order in advance from the publisher at a cost of c per one newspaper every day. It is known that the selling price is a per one newspaper. However, if the newspapers are not sold at the end of the day, then the newspapers have a small value of b per one newspaper at the recycling center. Assume that the demand for newspapers is denoted by ξ in a day, then the number of newspapers at the end of the day is clearly $x - \xi$ if $x > \xi$ or 0 if $x < \xi$. Thus the profit of the newsboy should be

$$f(x, \xi) = \begin{cases} (a-c)x, & \text{if } x \leq \xi \\ (b-c)x + (a-b)\xi, & \text{if } x > \xi \end{cases} \tag{2.7}$$

In practice, the demand ξ for newspapers is usually a stochastic variable, so is the profit function $f(x, \xi)$. Since we cannot predict how profitable the decision of ordering x newspapers will actually be, a natural idea is to employ the expected

profit, shown as follows,

$$E[f(x,\xi)] = \int_0^x [(b-c)x + (a-b)r]dF(r) + \int_x^{+\infty} (a-c)xdF(r) \quad (2.8)$$

where E denotes the expected value operator and $F(\cdot)$ is the distribution function of demand ξ. The newsboy problem is related to determining the optimal integer number x of newspapers such that the expected profit $E[f(x,\xi)]$ achieves the maximal value, i.e.,

$$\begin{cases} \max E[f(x,\xi)] \\ \text{s.t.} x \geq 0, \text{ integers} \end{cases} \quad (2.9)$$

This is a typical example of an expected value model.

Then we firstly should give the basic definition of the expected value. For the discrete random variable, we can define its expected value as follows.

Definition 2.8. Let ξ be a discrete random variable on the probability $(\Omega, \mathscr{A}, Pr)$ as follow,

$$\xi(\omega) = \begin{cases} x_1 \text{ if } \omega = \omega_1 \\ x_2 \text{ if } \omega = \omega_2 \\ \cdots \quad \cdots \end{cases} \quad (2.10)$$

where the probability of $\omega = \omega_i (i = 1, 2, \ldots)$ is p_i. If the series $\sum_{\omega\in\Omega} \xi(\omega_i) Pr\{\omega = \omega_i\}$ is absolutely convergent, then we call it the expected value of ξ, denoted by $E[\xi]$.

For the continuous random variable, its expected value can be defined as follows.

Definition 2.9. (Durrett [91]) Let ξ be a random variable on the probability space $(\Omega, \mathscr{A}, Pr)$. Then the expected value of ξ is defined by

$$E[\xi] = \int_0^{+\infty} Pr\{\xi \geq r\}dr - \int_{-\infty}^0 Pr\{\xi \leq r\}dr \quad (2.11)$$

There is another equivalent definition by the density function.

Definition 2.10. (Durrett [91]) The expected value of a random variable ξ with probability density function $f(x)$ is

$$E[\xi] = \int_{-\infty}^{+\infty} xf(x)dx \quad (2.12)$$

Expected value, average and mean are the same thing, but median is entirely different. The median is defined below, but only to make the distinction clear. After this, we won't make further use of the median.

Definition 2.11. (Durrett [91]) The median of a random variable ξ is the unique value r in the range of ξ such that $Pr\{\xi < r\} \leq 1/2$ and $Pr\{\xi > r\} < 1/2$.

For example, with an ordinary die, the median thrown value is 4, which not the same as the mean 3.5. The median and the mean can be very far apart. For example, consider a $2n$-side die, with n 0s and 100s. The mean is 50, and the median is 100.

To deeply understand random variables, the variance of a random variable is given as follows.

Definition 2.12. (Durrett [91]) The variance of a random variable ξ is defined by

$$V[\xi] = E[(\xi - E[\xi])^2].$$

The following properties about the expected value and variance of a random variable are very useful to the decision making problems with random parameters [91].

Lemma 2.1. *Let ξ and η be random variables with finite expected values. Then for any numbers a and b, we have*

$$E[a\xi + b\eta] = aE[\xi] + bE[\eta] \tag{2.13}$$

Lemma 2.2. *For two independent random variables ξ and η, we have*

$$E[\xi\eta] = E[\xi]E[\eta] \tag{2.14}$$

Lemma 2.3. *For the random variable ξ, we have*

$$V[\xi] = E[\xi^2] - (E[\xi])^2 \tag{2.15}$$

and for $a, b \in \mathbf{R}$, we have

$$V[a\xi + b] = a^2 V[\xi] \tag{2.16}$$

Let's consider the typical single objective with random parameters,

$$\begin{cases} \max f(\boldsymbol{x}, \boldsymbol{\xi}) \\ \text{s.t.} \begin{cases} g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, \ j = 1, 2, \ldots, p \\ \boldsymbol{x} \in X \end{cases} \end{cases} \tag{2.17}$$

where $f(\boldsymbol{x}, \boldsymbol{\xi})$ and $g_j(\boldsymbol{x}, \boldsymbol{\xi}), j = 1, 2 \ldots, p$ are continuous functions in X and $\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_n)$ is a random vector on the probability space $(\Omega, \mathscr{A}, Pr)$. Then it follows from the expected operator that,

$$\begin{cases} \max E[f(\boldsymbol{x}, \boldsymbol{\xi})] \\ \text{s.t.} \begin{cases} E[g_j(\boldsymbol{x}, \boldsymbol{\xi})] \leq 0, \ j = 1, 2, \ldots, p \\ \boldsymbol{x} \in X \end{cases} \end{cases} \tag{2.18}$$

After being dealt with by expected value operator, the problem (2.17) has been converted into a certain programming and then decision makers can easily obtain the optimal solution. However, whether the problem (2.18) has optimal solutions is a spot which decision makers pay more attention to, then its convexity is the focus we will discuss in the following part.

Definition 2.13. $\boldsymbol{x}$ is said to be a *feasible solution* of problem (2.18) if and only if $E[g_j(\boldsymbol{x}, \boldsymbol{\xi})] \leq 0\,(j = 1, 2, \ldots, p)$. For any feasible solution $\boldsymbol{x}$, if $E[f(\boldsymbol{x}^*, \boldsymbol{\xi})] \geq E[f(\boldsymbol{x}, \boldsymbol{\xi})]$, then $\boldsymbol{x}^*$ is an optimal solution of problem (2.18).

A mathematical programming model is called convex if both the objective function and the feasible set are convex. For the expected value model (2.18), we have the following result on convexity.

Theorem 2.1. *Assume that, for each $\boldsymbol{\xi}$, the functions $f(\boldsymbol{x}, \boldsymbol{\xi})$ and $g_j(\boldsymbol{x}, \boldsymbol{\xi})$, $j = 1, 2, \ldots, p$ are convex in X. Then the expected value model (2.18) is a convex programming.*

Proof. For each $\boldsymbol{\xi}$, since the function $f(\boldsymbol{x}, \boldsymbol{\xi})$ is convex in X, we have

$$f(\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2, \boldsymbol{\xi}) \leq \lambda f(\boldsymbol{x}_1, \boldsymbol{\xi}) + (1-\lambda) f(\boldsymbol{x}_2, \boldsymbol{\xi}),$$

for any given solutions $\boldsymbol{x}_1, \boldsymbol{x}_2$ and any scalar $\lambda \in [0, 1]$. It follows from the expected value operator that

$$E[f(\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2, \boldsymbol{\xi})] \leq \lambda E[f(\boldsymbol{x}_1, \boldsymbol{\xi})] + (1-\lambda) E[f(\boldsymbol{x}_2, \boldsymbol{\xi})],$$

which proves the convexity of the objective function $E[f(\boldsymbol{x}, \boldsymbol{\xi})]$ in X.

Let us prove the convexity of the feasible set by verifying that $\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2$ is feasible for any feasible solutions $\boldsymbol{x}_1$, and $\boldsymbol{x}_2$ constrained by $E[g_j(\boldsymbol{x}, \boldsymbol{\xi})] \leq 0$, $j = 1, 2, \ldots, p$ and any scalar $\lambda \in [0, 1]$. By the convexity of the functions $g_j(\boldsymbol{x}, \boldsymbol{\xi})$, $j = 1, 2, \ldots, p$, we know that

$$g_j(\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2, \boldsymbol{\xi}) \leq \lambda g_j(\boldsymbol{x}_1, \boldsymbol{\xi}) + (1-\lambda) g_j(\boldsymbol{x}_2, \boldsymbol{\xi}),$$

which yields that

$$E[g_j(\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2, \boldsymbol{\xi})] \leq \lambda E[g_j(\boldsymbol{x}_1, \boldsymbol{\xi})] + (1-\lambda) E[g_j(\boldsymbol{x}_2, \boldsymbol{\xi})],$$

for $j = 1, 2, \ldots, p$. It follows that $\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2$ is a feasible solution. Hence the feasible set is convex. This completes the proof. □

In many cases, there are usually multiple objectives which decision makers must consider. Thus we have to employ the following expected value model (EVM),

$$\begin{cases} \max[E[f_1(\boldsymbol{x}, \boldsymbol{\xi})], E[f_2(\boldsymbol{x}, \boldsymbol{\xi})], \ldots, E[f_m(\boldsymbol{x}, \boldsymbol{\xi})]] \\ \text{s.t.} \begin{cases} E[g_j(\boldsymbol{x}, \boldsymbol{\xi})] \leq 0, \ j = 1, 2, \ldots, p \\ \boldsymbol{x} \in X \end{cases} \end{cases} \tag{2.19}$$

where $f_i(\boldsymbol{x}, \boldsymbol{\xi})$ are return functions for $i = 1, 2, \ldots, m$. $\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_n)$ is a random vector on probability space $(\Omega, \mathscr{A}, Pr)$.

Definition 2.14. $\boldsymbol{x}^*$ is the Pareto solution of problem (2.19), if there doesn't exist feasible solutions $\boldsymbol{x}$ such that

$$E[f_i(\boldsymbol{x}, \boldsymbol{\xi})] \geq E[f_i(\boldsymbol{x}^*, \boldsymbol{\xi})], i = 1, 2, \ldots, m$$

and there exists at least one $j (j = 1, 2, \ldots, m)$ such that $E[f_j(\boldsymbol{x}, \boldsymbol{\xi})] > E[f_j(\boldsymbol{x}^*, \boldsymbol{\xi})]$.

We can also formulate a stochastic decision system as an expected value goal model (EVGM) according to the priority structure and target levels set by the decision maker:

$$\begin{cases} \min \sum_{j=1}^{l} P_j \sum_{i=1}^{m} (u_{ij} d_i^+ + v_{ij} d_i^-) \\ \text{s.t.} \begin{cases} E[f_i(\boldsymbol{x}, \boldsymbol{\xi})] + d_i^- - d_i^+ = b_i, & i = 1, 2, \ldots, m \\ E[g_j(\boldsymbol{x}, \boldsymbol{\xi})] \leq 0, & j = 1, 2, \ldots, p \\ d_i^-, d_i^+ \geq 0, & i = 1, 2, \ldots, m \\ \boldsymbol{x} \in X \end{cases} \end{cases} \tag{2.20}$$

where P_j is the preemptive priority factor which expresses the relative importance of various goals, $P_j >> P_{j+1}$, for all j, u_{ij} is the weighting factor corresponding to positive deviation for goal i with priority j assigned, v_{ij} is the weighting factor corresponding to negative deviation for goal i with priority j assigned, d_i^+ is the positive deviation from the target of goal i, defined as

$$d_i^+ = [E[f_i(\boldsymbol{x}, \boldsymbol{\xi})] - b_i] \vee 0,$$

d_i^- is the negative deviation from the target of goal i, defined as

$$d_i^- = [b_i - E[f_i(\boldsymbol{x}, \boldsymbol{\xi})]] \vee 0,$$

f_i is a function in goal constraints, g_j is a function in real constraints, b_i is the target value according to goal i, l is the number of priorities, m is the number of goal constraints, and p is the number of real constraints.

2.3.2 Linear Random EVM and the Weight Sum Method

Generally, many uncertain problems cannot be directly converted into crisp ones unless they have favorable properties and their random parameters have crisp distribution. For those which cannot be directly transformed, Monte Carlo simulation

is a useful tool to deal with them. Next, we will exhibit some examples which can be converted into crisp models. Let's consider the following linear random multi-objective programming (hereby, the linear relation is the relation among the coefficients $\bar{c}_i$ or $\bar{e}_r$, and the linearity and non-linearity in the following parts aims at this relationship),

$$\begin{cases} \max\left[\bar{c}_1^T\boldsymbol{x}, \bar{c}_2^T\boldsymbol{x}, \ldots, \bar{c}_m^T\boldsymbol{x}\right] \\ \text{s.t.} \begin{cases} \bar{e}_r^T\boldsymbol{x} \leq \bar{b}_r,\ r = 1, 2, \ldots, p \\ \boldsymbol{x} \geq 0 \end{cases} \end{cases} \tag{2.21}$$

where $\boldsymbol{x} \in X \subset \mathbf{R}^n$, $\bar{c}_i = (\bar{c}_{i1}, \bar{c}_{i2}, \ldots, \bar{c}_{in})^T$, $\bar{e}_r = (\bar{e}_{r1}, \bar{e}_{r2}, \ldots, \bar{e}_{rn})^T$ are random vectors, and $\bar{b}_r$ are random variables, $i = 1, 2, \ldots, m$, $r = 1, 2, \ldots, p$.

Because of the existence of some random parameters $\bar{c}_i$, $\bar{e}_r$ and $\bar{b}_r$, we cannot easily obtain its optimal solutions. Then we can obtain the following expected value model of problem (2.21),

$$\begin{cases} \max\left[E[\bar{c}_1^T\boldsymbol{x}], E[\bar{c}_2^T\boldsymbol{x}], \ldots, E[\bar{c}_m^T\boldsymbol{x}]\right] \\ \text{s.t.} \begin{cases} E[\bar{e}_r^T\boldsymbol{x}] \leq E[\bar{b}_r],\ r = 1, 2, \ldots, p \\ \boldsymbol{x} \geq 0 \end{cases} \end{cases} \tag{2.22}$$

2.3.2.1 Crisp Equivalent Model

One way to solve the expected value model is to convert it into crisp equivalent model, and then deal with it by the multi-objective programming technique.

Theorem 2.2. *Assume that random vectors $\bar{c}_i = (\bar{c}_{i1}, \bar{c}_{i2}, \ldots, \bar{c}_{in})^T$ is normally distributed with mean vectors $\mu_i^c = (\mu_{i1}^c, \mu_{i2}^c, \ldots, \mu_{in}^c)^T$ and positive definite covariance matrix V_i^c, written as $\bar{c}_i \sim \mathcal{N}(\mu_i^c, V_i^c)(i = 1, 2, \ldots, m)$ and random vectors $\bar{e}_r \sim \mathcal{N}(\mu_r^e, V_i^e)$, $\bar{b}_r \sim \mathcal{N}(\mu_r^b, (\sigma_r^b)^2)$ $(r = 1, 2, \ldots, p)$. Assume that for any $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$ and $r = 1, 2, \ldots, p$, $\bar{c}_{ij}$, $\bar{e}_{ij}$ and $\bar{b}_{ij}$ are independently random variables. Then problem (2.22) is equivalent to*

$$\begin{cases} \max\left[\mu_1^{cT}\boldsymbol{x}, \mu_2^{cT}\boldsymbol{x}, \ldots, \mu_m^{cT}\boldsymbol{x}\right] \\ s.t. \begin{cases} \mu_r^{eT}\boldsymbol{x} \leq \mu_r^b,\ r = 1, 2, \ldots, p \\ \boldsymbol{x} \geq 0 \end{cases} \end{cases} \tag{2.23}$$

Proof. Since random vector $\bar{c}_i$ is normally distributed on $(\Omega, \mathscr{A}, Pr)$ and $\bar{c}_i \sim \mathcal{N}(\mu_i^c, V_i^c)(i = 1, 2, \ldots, m)$, then

$$\bar{c}_i^T\boldsymbol{x} = \sum_{j=1}^{n} x_j\bar{c}_{ij} \sim \mathcal{N}\left(\sum_{j=1}^{n} x_j\mu_{ij}^c, \boldsymbol{x}^T V_i^c\boldsymbol{x}\right) = \mathcal{N}(\mu_i^{cT}\boldsymbol{x}, \boldsymbol{x}^T V_i^c\boldsymbol{x}),$$

so $E[\bar{c}_i^T \boldsymbol{x}] = \mu_i^{cT} \boldsymbol{x}$. Similarly, we obtain

$$E[\bar{e}_r^T \boldsymbol{x}] = \mu_r^{eT} \boldsymbol{x}, E[\bar{b}_r] = \mu_r^b, r = 1, 2, \ldots, p,$$

then it follows that problem (2.22) is equivalent to

$$\begin{cases} \max\left[\mu_1^{cT} \boldsymbol{x}, \mu_2^{cT} \boldsymbol{x}, \ldots, \mu_m^{cT} \boldsymbol{x}\right] \\ \text{s.t.} \begin{cases} \mu_r^{eT} \boldsymbol{x} \leq \mu_r^b, r = 1, 2, \ldots, p \\ \boldsymbol{x} \geq 0 \end{cases} \end{cases}$$

This completes the proof. □

Theorem 2.3. *Assume that random vector* $\bar{c}_i = (\bar{c}_{i1}, \bar{c}_{i2}, \ldots, \bar{c}_{in})^T$ *is exponentially distributed, written as* $\bar{c}_{ij} \sim exp(\lambda_{ij}^c)$ $(i = 1, 2, \ldots, m, j = 1, 2, \ldots, n)$ *and random variable* $\bar{e}_{rj} \sim exp(\lambda_{rj}^e)$, $\bar{b}_r \sim exp(\lambda_r^b)$ $(r = 1, 2, \ldots, p, j = 1, 2, \ldots, n)$. *Then problem (2.22) is equivalent to*

$$\begin{cases} \max\left[\lambda_1^{cT} \boldsymbol{x}, \lambda_2^{cT} \boldsymbol{x}, \ldots, \lambda_m^{cT} \boldsymbol{x}\right] \\ s.t. \begin{cases} \lambda_r^{eT} \boldsymbol{x} \leq \dfrac{1}{\lambda_r^b}, r = 1, 2, \ldots, p \\ \boldsymbol{x} \geq 0 \end{cases} \end{cases} \tag{2.24}$$

where $\lambda_i^c = \left(\frac{1}{\lambda_{i1}^c}, \frac{1}{\lambda_{i2}^c}, \ldots, \frac{1}{\lambda_{in}^c}\right)^T$ *and* $\lambda_i^e = \left(\frac{1}{\lambda_{r1}^e}, \frac{1}{\lambda_{r2}^e}, \ldots, \frac{1}{\lambda_{rn}^e}\right)^T$.

Proof. Since random variables $\bar{c}_{ij}$ are exponentially distributed on the probability space $(\Omega, \mathscr{A}, Pr)$ and $\bar{c}_{ij} \sim exp(\lambda_{ij}^c)$ $(i = 1, 2, \ldots, m, j = 1, 2, \ldots, n)$, then it follows from Theorem 2.1 that

$$E[\bar{c}_i^T \boldsymbol{x}] = E\left[\sum_{j=1}^n x_j \bar{c}_{ij}\right] = \sum_{j=1}^n x_j E[\bar{c}_{ij}] = \sum_{j=1}^n x_j / \lambda_{ij}^c = \lambda_i^{cT} \boldsymbol{x},$$

where $\lambda_i^c = (\frac{1}{\lambda_{i1}^c}, \frac{1}{\lambda_{i2}^c}, \ldots, \frac{1}{\lambda_{in}^c})^T, i = 1, 2, \ldots, m$. Similarly, we have

$$E[\bar{e}_r^T \boldsymbol{x}] = \lambda_r^{eT} \boldsymbol{x}, \quad E[\bar{b}_r] = \frac{1}{\lambda_r^b}$$

where $\lambda_r^e = (\frac{1}{\lambda_{r1}^e}, \frac{1}{\lambda_{r2}^e}, \ldots, \frac{1}{\lambda_{rn}^e})^T$, $r = 1, 2, \ldots, p$. Then problem (2.22) is equivalent to

$$\begin{cases} \max\left[\lambda_1^{cT} \boldsymbol{x}, \lambda_2^{cT} \boldsymbol{x}, \ldots, \lambda_m^{cT} \boldsymbol{x}\right] \\ \text{s.t.} \begin{cases} \lambda_r^{eT} \boldsymbol{x} \leq \dfrac{1}{\lambda_r^b}, r = 1, 2, \ldots, p \\ \boldsymbol{x} \geq 0 \end{cases} \end{cases}$$

where $\lambda_i^c = \left(\frac{1}{\lambda_{i1}^c}, \frac{1}{\lambda_{i2}^c}, \ldots, \frac{1}{\lambda_{in}^c}\right)^T$ and $\lambda_r^e = \left(\frac{1}{\lambda_{r1}^e}, \frac{1}{\lambda_{r2}^e}, \ldots, \frac{1}{\lambda_{rn}^e}\right)^T$. This completes the proof. □

We just take the normal distribution and the exponential distribution as examples, and readers can get the similar results when random parameters are subject to other distributions. If there are more than two different distributions in the same problem, readers can also deal with it by the expected value operator and convert it into the crisp one.

2.3.2.2 The Weight Sum Method

In this section, we will introduce the weight sum method to solve the multi-objective problem (2.23) and (2.24). Take the problem (2.23) as an example. Let $H_i(\boldsymbol{x}) = \mu_1^{cT}\boldsymbol{x}, i = 1, 2, \ldots, m$ and $X = \{\boldsymbol{x} | \mu_r^{eT}\boldsymbol{x} \leq \mu_r^b, \boldsymbol{x} \geq 0, r = 1, 2, \ldots, p\}$, then the problem (2.77) can be rewritten as,

$$\begin{cases} \max\ [H_1(\boldsymbol{x}), H_2(\boldsymbol{x}), \ldots, H_m(\boldsymbol{x})] \\ \text{s.t. } \boldsymbol{x} \in X \end{cases} \tag{2.25}$$

The weight sum method is one of the techniques which are broadly applied to solve the multi-objective programming problem. Assume that the related weight of the objective function $H_i(\boldsymbol{x})$ is w_i such that $\sum_{i=1}^m w_i = 1$ and $w_i \geq 0$. Construct the evaluation function as follows,

$$u(\boldsymbol{H}(\boldsymbol{x})) = \sum_{i=1}^{m} w_i H_i(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{H}(\boldsymbol{x}),$$

where w_i expresses the importance of the object $H_i(\boldsymbol{x})$ for DM. Then we get the following weight problem,

$$\max_{x \in X} u(\boldsymbol{H}(\boldsymbol{x})) = \max_{x \in X} \sum_{i=1}^{m} w_i H_i(\boldsymbol{x}) = \max_{x \in X} \boldsymbol{w}^T \boldsymbol{H}(\boldsymbol{x}) \tag{2.26}$$

Let $\bar{\boldsymbol{x}}$ be an optimal solution of the problem (2.26), we can easily deduce that if $\boldsymbol{w} > 0$, $\bar{\boldsymbol{x}}$ is an efficient solution of the problem (2.25). By changing $\boldsymbol{w}$, we can obtain a set composed of the efficient solutions of the problem (2.25) by solving the problem (2.26).

2.3.3 Nonlinear Random EVM and Random Simulation-Based SA

For some complex problems, it is usually difficult to convert them into crisp ones and obtain their expected values. For example, let's consider the problem:

$\max_{x \in X} E[\sqrt{(x_1 - \xi_1)^2 + (x_2 - \xi_2)^2}]$, where ξ_1 is a uniformly distributed random variable and ξ_2 is a normally distributed random variable. As we know, it is almost impossible to convert it into a crisp one. Thus, an intelligent algorithm should be provided to solve it. The technique of stochastic simulation-based SA is a useful and efficient tool when dealing with them. Let's consider the following random multi-objective problem,

$$\begin{cases} \max[E[f_1(\boldsymbol{x}, \boldsymbol{\xi})], E[f_2(\boldsymbol{x}, \boldsymbol{\xi})], \ldots, E[f_m(\boldsymbol{x}, \boldsymbol{\xi})]] \\ \text{s.t.} \begin{cases} E[g_j(\boldsymbol{x}, \boldsymbol{\xi})] \leq 0, j = 1, 2, \ldots, p \\ \boldsymbol{x} \in X \end{cases} \end{cases}$$

where $f_i(\boldsymbol{x}, \boldsymbol{\xi})$ or $g_j(\boldsymbol{x}, \boldsymbol{\xi})$ or both of them are nonlinear with respect to $\boldsymbol{\xi}$, $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, p$. $\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_n)$ is a random vector on probability space $(\Omega, \mathscr{A}, Pr)$. Because of the existence of the nonlinear functions, we cannot usually convert it into the crisp one. Then we have to apply the technique of the random simulation (Monte Carlo Simulation) to compute its expected value.

2.3.3.1 Random Simulation for EVM

Let $\boldsymbol{\xi}$ be an n-dimensional random vector defined on the probability space $(\Omega, \mathscr{A}, Pr)$ (equivalently, it is characterized by a probability distribution $F(\cdot)$), and $f : \mathbf{R}^n \to \mathbf{R}$ a measurable function. Then $f(\boldsymbol{x}, \boldsymbol{\xi})$ is also a random variable. In order to calculate the expected value $E[f(\boldsymbol{x}, \boldsymbol{\xi})]$ for given $\{\boldsymbol{x}\}$, we generate ω_k from Ω according to the probability measure Pr, and write $\boldsymbol{\xi}_k = \boldsymbol{\xi}(\omega_k)$ for $k = 1, 2, \ldots, N$, where $\omega_k = (\omega_k^1, \ldots, \omega_k^n)$ is an n-dimensional vector and ω_k^j is generated according to the random variable ξ_k. Equivalently, we generate random vectors $\boldsymbol{\xi}_k$, $k = 1, 2, \ldots, N$ according to the probability distribution $F(\cdot)$. It follows from the strong law of large numbers that

$$\frac{\sum_{k=1}^{N} f(\boldsymbol{x}, \boldsymbol{\xi}_k)}{N} \to E[f(\boldsymbol{x}, \boldsymbol{\xi})],$$

as $N \to \infty$. Therefore, the value $E[f(\boldsymbol{x}, \boldsymbol{\xi})]$ can be estimated by $\frac{1}{N} \sum_{k=1}^{N} f(\boldsymbol{x}, \boldsymbol{\xi}_k)$ provided that N is sufficiently large. Then the procedure simulating the expected value of the function $f(\boldsymbol{x}, \boldsymbol{\xi})$ can be summarized as follows:

Procedure `Random simulation for EVM`

Input: The decision vector $\boldsymbol{x}$
Output: The expected value $E[f(\boldsymbol{x}, \boldsymbol{\xi})]$
Step 1. Set $L = 0$;
Step 2. Generate ω_k from Ω according to the probability measure Pr, $k = 1, 2, \ldots, N$;
Step 3. $L \leftarrow L + f(\boldsymbol{x}, \boldsymbol{\xi}_k)$;
Step 4. Repeat the second and third steps N times;
Step 5. Return $E[f(\boldsymbol{x}, \boldsymbol{\xi})] = L/N$.

Example 2.2. Let ξ_1 be an exponentially distributed variable $exp(1)$, ξ_2 a normally distributed variable $\mathcal{N}(3, 1)$, and ξ_3 a uniformly distributed variable $\mathcal{U}(0, 1)$. A run of stochastic simulation with 10000 cycles shows that $E[\sqrt{\xi_1^2 + \xi_2^2 + \xi_3^2}] = 3.3566$.

2.3.3.2 Simulated Annealing Algorithm

Simulated annealing algorithm (abbr. SA) are proposed by Kirkpatrick et al. [160, 161] for the problem of finding, numerically, a point of the global minimum of a function defined on a subset of a n-dimensional Euclidean space. The name of the algorithm derived from an analogy between the simulation of the annealing of solid first proposed by Metropolis et al. [218] and the strategy of solving combinatorial optimization problems. The motivation of the methods lies in the physical process of annealing, in which a solid is heated to a liquid state and, when cooled sufficiently slowly, takes up the configuration with minimal inner energy. Metropolis et al. [218] described this process mathematically. Simulating annealing uses this mathematical description for the minimization of other functions than the energy. The first results have been published by Černý [39], Kirkpatrick et al. [160,161] and Geman and Geman [107]. For a related earlier result, see Hasminskij [232]. Most of the early considerations concern minimization of functions defined on a finite set. Kushner [179] and Gelfand and Mitter [106] obtained results for functions with infinite domains. Laarhoven and Aarts [183], and Laarhoven [182] are monographs on simulated annealing. Steel [302], in a review of [182], calls simulated annealing the most exciting algorithmic development of the decade.

Annealing, physically, refers to the process of heating up a solid to a high temperature followed by slow cooling 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 specific energy level. Simulated annealing is a computational stochastic technique for obtaining near global optimum solutions to combinatorial and function optimization problems. The method is inspired from the thermodynamic process of cooling (annealing) of molten metals to attain the lowest free energy state. When molten metal is cooled slowly enough it tends to solidify in a structure of minimum energy. This annealing process is mimicked by a search strategy. The key principle of the method is to allow occasional worsening moves so that these can eventually help locate the neighborhood to the true (global) minimum. The associated mechanism is given by the Boltzmann probability, namely,

$$\text{probability } (p) = \exp\left(\frac{-\triangle E}{K_B T}\right), \tag{2.27}$$

where $\triangle E$ is the change in the energy value from one point to the next, K_B is the Boltzmanns constant and T is the temperature (control parameter). For the purpose

of optimization the energy term, $\triangle E$ refers to the value of the objective function and the temperature, T, is a control parameter that regulates the process of annealing. The consideration of such a probability distribution leads to the generation of a Markov chain of points in the problem domain. The acceptance criterion given by (2.27) is popularly referred to as the Metropolis criterion [218]. Another variant of this acceptance criterion (for both improving and deteriorating moves) has been proposed by Galuber [113] and can be written as

$$\text{probability } (p) = \frac{\exp(-\triangle E/T)}{1 + \exp(-\triangle E/T)} \tag{2.28}$$

In simulated annealing search strategy: at the start any move is accepted. This allows us to explore solution space. Then, gradually the temperature is reduced which means that one becomes more and more selective in accepting new solution. By the end, only the improving moves are accepted in practice. The temperature is systematically lowered using a problem-dependent schedule characterized by a set of decreasing temperatures.

Next, we introduce the general framework for the simulated annealing algorithm. The standard SA technique makes the analogy between the state of each molecule that determines the energy function and the value of each parameter that affects the objective functions. It then uses the statistical mechanics principle for energy minimization to minimize the objective function and optimize the parameter estimates. Starting with a high temperature, it randomly perturbs the parameter values and calculates the resulting objective function. The new state of objective function after perturbation is then accepted by a probability determined by the Metropolis criterion. The system temperature is then gradually reduced as the random perturbation proceeds, until the objective function reaches its global or nearly global minimum. A typical SA algorithm is described as follows (Fig. 2.2):

Step 1. Specify initial temperature $T_k = T_0$ for $k = 0$; randomly initialize the parameter set estimate $\theta^* = \theta_0$.

Step 2. Under kth temperature, if the inner loop break condition is met, go to step 3; otherwise, for $(j + 1)$th perturbation, randomly produce a new parameter set θ_{j+1}, compute the change in objective function $\triangle f = f(\theta^*) - f(\theta_{j+1})$. If $\triangle f \leq 0$, accept $\theta_{j+1}(\theta^* = \theta_j)$; if not, follow the Metropolis criterion to accept θ_{j+1} with a probability of $\min(1, e^{-\triangle f/T_k})$ and step 2 continues.

Step 3. Reduce T_k to T_{k+1} following a specified cooling schedule. If outer loop break condition is met, computation stops and optimal parameter set is reached; if not, return back to step 2.

The steps outlined above consist of one inner loop (step 2) and one outer loop (step 3). The proceeding of SA is mainly controlled by (1) the choice of T_0; (2) the way a new perturbation is generated; (3) the inner loop break conditions; (4) the choice of cooling schedule; and (5) the outer loop break conditions. The pseudocode can be seen in Table 2.1.

For the multiobjective optimization problem, many researchers have proposed many kinds of SA algorithms to obtain the Pareto-optimal solutions.

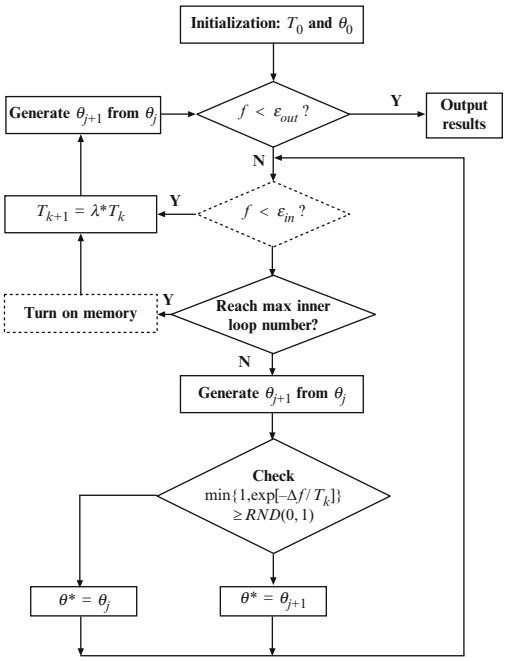

Fig. 2.2 The flow chart of modified SA algorithm

Table 2.1 Simulated annealing algorithm in pseudo-code

Select an initial temperature $T_0 > 0$;
Select an initial solution, S_0, and make it the current solution, S, and the current best solution, S^*;
repeat
  set repetition counter $n = 1$;
    repeat
      generates solution S_n in the neighborhood of S;
      calculate $\Delta = f(S_n) - f(S)$
      if $(\Delta \leq 0)$, then $S = S_n$
        else $S = S_n$ with probability of $p = e^{-\Delta/T}$
      if $(f(S_n) \leq f(S^*))$, then $S^* = S_n$
      n = n + 1;
    until $n >$ number of repetitions allowed at each temperature level (L);
  reduce the temperature T;
until stop criterion is true.

Suppapitnarm et al. [307] has used the concept of archiving the Pareto-optimal solutions coupled with return to base strategy to solve multiobjective problems with simulated annealing. They use the objective function values but not the weight vector as an acceptance criterion after penalizing them and annealing temperature and consider multiple annealing temperatures (usually one per objective). Therefore, the key probability step can be given as:

$$\text{probability } (p) = \min\left(1, \prod_{i=1}^{m} \exp\left\{\frac{-\triangle s_i}{T_i}\right\}\right), \tag{2.29}$$

where $\triangle s_i = f_i(\boldsymbol{y}) - f_i(\boldsymbol{x})$, $\boldsymbol{x}$ is the current solution, $\boldsymbol{y}$ is the generated solution, f_i is the ith objective function, T_i is the ith annealing temperature and m is the number of objective functions. On the other hand, the penalty function approach can help us to convert the constrained problem to an unconstrained one.

Above all, the SA algorithm which is proposed to solve the multiobjective programming problem(m objective functions and n decision variables) by Suppapitnarm et al. [307] (abbr. SMOSA) can be summarized as follows:

Procedure `The SMOSA algorithm`

Input: The initial temperature T_0

Output: The Pareto-solution $\boldsymbol{x}^*$

Step 1. Randomly generate a feasible $\boldsymbol{x}$ by random simulation and put $\boldsymbol{x}$ into a Pareto set of solutions. Compute all objective values;

Step 2. Generate a new solution $\boldsymbol{y}$ in the neighborhood of $\boldsymbol{x}$ by the random perturbation. Compute the objective values and apply the penalty function approach to the corresponding objective functions, if necessary;

Step 3. Compare the generated solution with all solutions in the Pareto set and update the Pareto set if necessary;

Step 4. Replace the current solution $\boldsymbol{x}$ with the generated solution $\boldsymbol{y}$ if $\boldsymbol{y}$ is archived and go to Step 7;

Step 5. Accept the generated solution $\boldsymbol{y}$ as the current solution if it is not archived with the probability:

$$\text{probability } (p) = \min\left(1, \prod_{i=1}^{m} \exp\left\{\frac{-\triangle s_i}{T_i}\right\}\right),$$

where $\triangle s_i = f_i(\boldsymbol{y}) - f_i(\boldsymbol{x})$. If the generated solution is accepted, replace $\boldsymbol{x}$ with $\boldsymbol{y}$ and go to Step 7;

Step 6. If the generated solution as current solution vector by $\boldsymbol{x} = \boldsymbol{x}$ and go to Step 7;

Step 7. Periodically, restart with a randomly selected solution from the Pareto set. While periodically restarting with the archived solutions, Suppapitnarm et al. [307] have recommended biasing towards the extreme ends of the trade-off surface;

Step 8. Periodically reduce the temperature using a problem-dependent annealing schedule;

Step 9. Repeat steps 2–8, until a predefined number of iterations is carried out.

There are also many other SA algorithms designed to solve multiobjective programming problems by many scholars. For example, Ulungu et al. [323, 324]

proposed UMOSA to project the multidimensional objective space into a mono dimensional space using weighed sum-scalarizing technique. Czyżak et al. [70] and Czyżak and Jaszkiewicz [71] proposed the PSA algorithm which modified the procedure Ulungu et al. [323] introduced by using the concept of interacting efficient solutions which are obtained by combining unicriterion simulated annealing and genetic algorithm. Suman [304, 305] proposed the WMOSA algorithm to handle constraints with its main algorithm by using weight vector in the acceptance criterion. Suman [306] also proposed the PDMOSA algorithm which uses the fitness value in the acceptance criteria to handle the multiobjective optimization problems. We consider only the random simulation-based SMOSA in this book and readers can find more detail about the multiobjective optimization problem solved by SA in [306].

2.3.4 Numerical Examples

Example 2.3. Let's consider the following programming problem,

$$\begin{cases} \max\ f_1(\boldsymbol{x}, \boldsymbol{\xi}) = 3\xi_1 x_1 - 2\xi_2 x_2 + 1.3\xi_3 x_3 \\ \max\ f_2(\boldsymbol{x}, \boldsymbol{\xi}) = -2.5\xi_1 x_1 + 3\xi_2 x_2 + 5\xi_3 x_3 \\ \text{s.t.} \begin{cases} x_1 + x_2 + x_3 \le 10 \\ 3x_1 + 5x_2 + 3x_3 \ge 4 \\ x_1, x_2, x_3 \ge 0 \end{cases} \end{cases} \tag{2.30}$$

where ξ_1 is a random variable with 0–1 distribution, $Pr\{\xi_1 = 1\} = 0.8$, $\xi_2 \sim B(20, 0.6)$ is a binomially distributed random variable and $\xi_3 \sim P(5)$ is a poisson distributed random variable. Since there are uncertain variables in objective functions, we usually aims at the expected values of objective functions, then problem (2.30) can be rewritten as

$$\begin{cases} \max\ E[f_1(\boldsymbol{x}, \boldsymbol{\xi})] = E[3\xi_1 x_1 - 2\xi_2 x_2 + 1.3\xi_3 x_3] \\ \max\ E[f_2(\boldsymbol{x}, \boldsymbol{\xi})] = E[-2.5\xi_1 x_1 + 3\xi_2 x_2 + 5\xi_3 x_3] \\ \text{s.t.} \begin{cases} x_1 + x_2 + x_3 \le 10 \\ 3x_1 + 5x_2 + 3x_3 \ge 4 \\ x_1, x_2, x_3 \ge 0 \end{cases} \end{cases} \tag{2.31}$$

Then we have

$$E[\xi_1] = 0.8,\ E[\xi_2] = 12,\ E[\xi_3] = 5$$

It follows from the linearity of expected value operator that, (2.31) can be rewritten as

Table 2.2 The optimal solution of expected value model

w_1	w_2	x_1	x_2	x_3	$H_1(\boldsymbol{x}^*)$	$H_2(\boldsymbol{x}^*)$
0.1	0.9	0	10	0	−240	360
0.2	0.8	0	10	0	−240	360
0.3	0.7	0	0	10	65	250
0.4	0.6	0	0	10	65	250
0.5	0.6	0	0	10	65	250

$$\begin{cases} \max H_1(\boldsymbol{x}) = 2.4x_1 - 24x_2 + 6.5x_3 \\ \max H_2(\boldsymbol{x}) = -2x_1 + 36x_2 + 25x_3 \\ \text{s.t.} \begin{cases} x_1 + x_2 + x_3 \leq 10 \\ 3x_1 + 5x_2 + 3x_3 \geq 4 \\ x_1, x_2, x_3 \geq 0 \end{cases} \end{cases} \tag{2.32}$$

Let's firstly obtain the weight by solving $H_i^0 = \max_{\boldsymbol{x} \in X} H_i(\boldsymbol{x})$ and $w_i = H_i^0 / (H_1^0 + H_2^0)$. Then we get

$$\begin{aligned} &H_1^0 = 65.00,\ H_2^0 = 360.00 \\ &w_1 = 0.153,\ \ w_2 = 0.847 \end{aligned}$$

Then we obtain the optimal solution $\boldsymbol{x}^* = (0, 10.00, 0)^T$ and $H_1(\boldsymbol{x}^*) = -240$ and $H_2(\boldsymbol{x}^*) = 360$.

By changing the weight, we can get different solutions under different weights as shown in Table 2.2.

Example 2.4. Consider the following programming problem,

$$\begin{cases} \max f_1(\boldsymbol{x}, \boldsymbol{\xi}) = \sqrt{(\xi_1 - x_1)^2 + (\xi_2 - x_2)^2} \\ \max f_2(\boldsymbol{x}, \boldsymbol{\xi}) = \sqrt{(\xi_1 + x_1)^2 + (\xi_2 + x_2)^2} \\ \text{s.t.} \begin{cases} x_1 + x_2 \leq 5 \\ x_1 \geq 0, x_2 \geq 0 \end{cases} \end{cases} \tag{2.33}$$

where $\xi_1 \sim \mathscr{N}(2, 0.5)$ is a normally distributed random variable and $\xi_2 \sim \mathscr{U}(1, 2)$ is also a normally distributed random variable. Then by the expected value operator, we have

$$\begin{cases} \max H_1(\boldsymbol{x}, \boldsymbol{\xi}) = E\left[\sqrt{(\xi_1 - x_1)^2 + (\xi_2 - x_2)^2}\right] \\ \max H_2(\boldsymbol{x}, \boldsymbol{\xi}) = E\left[\sqrt{(\xi_1 + x_1)^2 + (\xi_2 + x_2)^2}\right] \\ \text{s.t.} \begin{cases} x_1 + x_2 \leq 5 \\ x_1 \geq 0, x_2 \geq 0 \end{cases} \end{cases} \tag{2.34}$$

Next, we will use the random simulation-based SA to solve the above problem. Set the initial temperature $T_0 = 500$, the last temperature be 0 and the cooling method be 1 decrement once. The neighborhood can be constructed as follows,

$$x_1^1 = x_1^0 + rh, \quad x_2^1 = x_2^0 + rh,$$

where r is a random number in (0,1) and h is the step length (here $h = 2.0$). After the simulation with many cycles, we get the optimal solution under different weights as shown in Table 2.3. Figure 2.3 shows the cooling process when the weight is 0.5. The real line expresses the weight sum of two objective functions, and it shows that it gradually converges from T = 360. Figure 2.4 shows the changes of two objective values when the temperature decreases.

Table 2.3 The optimal solution by random simulation-based SA

w_1	w_2	x_1	x_2	$H_1(\boldsymbol{x})$	$H_2(\boldsymbol{x})$	$H(\boldsymbol{x})$	T_0
0.1	0.9	2.2448	2.7160	1.4333	6.1385	5.6680	500
0.2	0.8	2.2823	0.0925	1.7342	6.4436	5.5017	500
0.3	0.7	0.1658	2.8159	2.5864	6.3471	5.2189	500
0.4	0.6	1.0663	1.1634	1.3053	6.1609	4.2187	500
0.5	0.5	0.0307	0.7439	1.3862	5.9708	3.6785	500

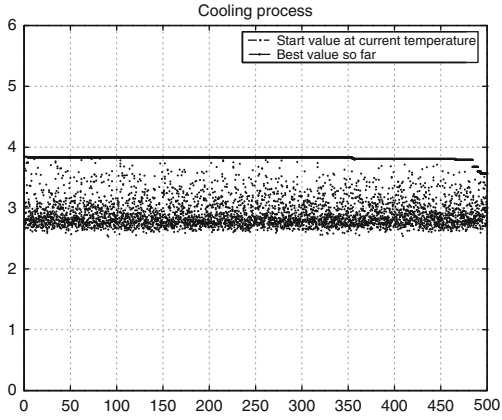

Fig. 2.3 The simulation process of random simulation-based SA

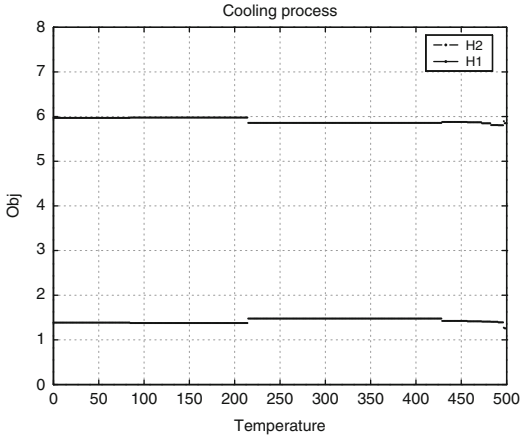

Fig. 2.4 Two objective values by random simulation-based SA

2.4 Random CCM

In 1959, Charnes and Cooper [45] developed another technique to deal with random programming problems, that is chance-constrained model (CCM). It is a powerful tool to help decision makers to make the decision in stochastic decision systems with assumption that the random constrains will hold at least α level value, where α is referred to as the confidence level provided as an appropriate safety margin by the decision maker.

This section will introduce some basic theories of chance-constrained model including chance-constrained operator, basic model, chance-constrained multi-objective model, and crisp equivalent model. Finally, the random simulation for CCM and some numerical examples will be provided.

2.4.1 General Model for Random CCM

In practice, the goal of decision makers is to maximize the objective value on the condition of probability α, where α is predetermined confidence level. Next, we will introduce the concept of the chance measure of random variables. The chance measure of a random event is considered as the probability of the event $f(\boldsymbol{x}, \boldsymbol{\xi}) \geq \bar{f}$. Then the chance constraint is considered as $Pr\{f(\boldsymbol{x}, \boldsymbol{\xi}) \geq \bar{f}\} \geq \alpha$, where α is the predetermined confidence level, and $\bar{f}$ is called the *critical value*. A natural idea is to provide a confidence level α at which it is desired that random constrains hold. Let's still consider the following model,

$$\begin{cases} \max f(\boldsymbol{x}, \boldsymbol{\xi}) \\ \text{s.t.} \begin{cases} g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, \ j = 1, 2, \ldots, p \\ \boldsymbol{x} \in X \end{cases} \end{cases}$$

where $f(\boldsymbol{x}, \boldsymbol{\xi})$ and $g_j(\boldsymbol{x}, \boldsymbol{\xi})$, $j = 1, 2 \ldots, p$ are continuous functions in X and $\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_n)$ is a random vector on probability space $(\Omega, \mathscr{A}, Pr)$. Based on the chance-constraint operator, the random chance-constrained model (CCM):

$$\begin{cases} \max \bar{f} \\ \text{s.t.} \begin{cases} Pr\{f(\boldsymbol{x}, \boldsymbol{\xi}) \geq \bar{f}\} \geq \beta \\ Pr\{g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0\} \geq \alpha_j, \ j = 1, 2, \ldots, p \\ \boldsymbol{x} \in X \end{cases} \end{cases} \tag{2.35}$$

where β and α_j are the predetermined confidence levels, $\bar{f}$ is the critical value which needs to determine.

Definition 2.15. A solution $\boldsymbol{x} \in X$ is said to be the feasible solution of problem (2.35) if and only if $Pr\{f(\boldsymbol{x}, \boldsymbol{\xi}) \geq \bar{f}\} \geq \beta$ and $Pr\{g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0\} \geq \alpha_j$ hold for

all j. For any feasible $\boldsymbol{x}$, if there is a solution $\boldsymbol{x}^*$ such $\bar{f}^* > \bar{f}$, then $\boldsymbol{x}^*$ is called the optimal solution.

If the objective is to be minimized (for example, the objective is a cost function), the CCM should be as follows,

$$\begin{cases} \min \bar{f} \\ \text{s.t.} \begin{cases} Pr\{f(\boldsymbol{x}, \boldsymbol{\xi}) \leq \bar{f}\} \geq \beta \\ Pr\{g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0\} \geq \alpha_j, \ j = 1, 2, \ldots, p \\ \boldsymbol{x} \in X \end{cases} \end{cases} \tag{2.36}$$

where β and α_j are the predetermined confidence levels. Similarly, we have the following definition.

Definition 2.16. A solution $\boldsymbol{x} \in X$ is called the feasible solution of problem (2.36) if and only if $Pr\{f(\boldsymbol{x}, \boldsymbol{\xi}) \leq \bar{f}\} \geq \beta$ and $Pr\{g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0\} \geq \alpha_j$ holds for all j. For any feasible $\boldsymbol{x}$, if there is a solution $\boldsymbol{x}^*$ such $\bar{f}^* < \bar{f}$, then $\boldsymbol{x}^*$ is called the optimal solution.

In practice, the real-life problems are more complex, and there usually exist multiple objectives which need decision makers to decide. Thus we have to employ the following multi-objective CCM:

$$\begin{cases} \max[\bar{f_1}, \bar{f_2}, \ldots, \bar{f_m}] \\ \text{s.t.} \begin{cases} Pr\{f_i(\boldsymbol{x}, \boldsymbol{\xi}) \geq \bar{f_i}\} \geq \beta_i, \ i = 1, 2, \ldots, m \\ Pr\{g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0\} \geq \alpha_j, \ j = 1, 2, \ldots, p \\ \boldsymbol{x} \in X \end{cases} \end{cases} \tag{2.37}$$

where β_i and α_j are the predetermined confidence levels, $\bar{f_i}$ are critical values which need to be determined.

Definition 2.17. $\boldsymbol{x} \in X$ is called the Pareto solution of problem (2.37) if there doesn't exist a feasible $\boldsymbol{x}$ such that

$$\bar{f_i} \geq \bar{f_i}^*, \ i = 1, 2, \ldots, m \tag{2.38}$$

and there at least exists one $j (j = 1, 2, \ldots, m)$ such that $\bar{f_j} > \bar{f_j}^*$.

Sometimes, we may formulate a stochastic decision system as a chance-constrained goal model (CCGM) according to the priority structure and target levels set by the decision-maker:

$$\begin{cases} \min \sum_{j=1}^{l} P_j \sum_{i=1}^{m} (u_{ij} d_i^+ + v_{ij} d_i^-) \\ \text{s.t.} \begin{cases} Pr\{f_i(\boldsymbol{x}, \boldsymbol{\xi}) - b_i \leq d_i^+\} \geq \beta_i^+, \ i = 1, 2, \ldots, m \\ Pr\{b_i - f_i(\boldsymbol{x}, \boldsymbol{\xi}) \leq d_i^-\} \geq \beta_i^-, \ i = 1, 2, \ldots, m \\ Pr\{g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0\} \geq \alpha_j, \qquad j = 1, 2, \ldots, p \\ d_i^-, d_i^- \geq 0, \qquad\qquad\qquad i = 1, 2, \ldots, m \end{cases} \end{cases} \tag{2.39}$$

where P_j is the preemptive priority factor which express the relative importance of various goals, $P_j >> P_{j+1}$, for all j, u_{ij} is the weighting factor corresponding to positive deviation for goal i with priority j assigned, v_{ij} is the weighting factor corresponding to negative deviation for goal i with priority j assigned, d_i^+ is the β_i^+-optimistic positive deviation from the target of goal i, defined as

$$\min\{d \vee 0 | Pr\{f_i(\boldsymbol{x}, \boldsymbol{\xi}) - b_i \leq d_i^+\} \geq \beta_i^+\} \tag{2.40}$$

d_i^- is the β_i^--optimistic positive deviation from the target of goal i, defined as

$$\min\{d \vee 0 | Pr\{b_i - f_i(\boldsymbol{x}, \boldsymbol{\xi}) \leq d_i^-\} \geq \beta_i^-\} \tag{2.41}$$

f_i is a function in goal constraints, g_j is a function in system constraints, b_i is the target value according to goal i, l is the number of priorities, m is the number of goal constraints,and p is the number of system constraints.

Remark 2.1. If the random vector $\boldsymbol{\xi}$ degenerates to the deterministic case, then the two probabilities $Pr\{f_i(\boldsymbol{x}, \boldsymbol{\xi}) - b_i \leq d_i^+\}$ and $Pr\{b_i - f_i(\boldsymbol{x}, \boldsymbol{\xi}) \leq d_i^-\}$ should be always 1 provided that $\beta_i^+, \beta_i^- > 0$, and

$$Pr\{f_i(\boldsymbol{x}, \boldsymbol{\xi}) - b_i \leq d_i^+\} \geq \beta_i^+, d_i^+ \geq 0,$$
$$Pr\{b_i - f_i(\boldsymbol{x}, \boldsymbol{\xi}) \leq d_i^-\} \geq \beta_i^-, d_i^- \geq 0$$

imply that

$$d_i^+ = [f_i(\boldsymbol{x}, \boldsymbol{\xi}) - b_i] \vee 0, d_i^- = [b_i - f_i(\boldsymbol{x}, \boldsymbol{\xi})] \vee 0.$$

This coincides with the deterministic goal programming.

2.4.2 Linear Random CCM and the Lexicographic Method

As we know, traditional solution methods need the conversion of the chance constraints to their respective deterministic equivalents. However, this process is usually hard to perform and only successful for some special cases. Many scholars has developed it a lot and made great achievements.

Theorem 2.4. *Assume that the random vector $\boldsymbol{\xi}$ degenerates to a random variable ξ with distribution function Φ, and the function $g(\boldsymbol{x}, \xi)$ has the form $g(\boldsymbol{x}, \xi) = h(\boldsymbol{x}) - \xi$. Then $Pr\{g(\boldsymbol{x}, \xi) \leq 0\} \geq \alpha$ if and only if $h(\boldsymbol{x}) \leq K_\alpha$, where $K_\alpha = sup\{K | K = \Phi^{-1}(1-\alpha)\}$.*

Proof. The assumption implies that $Pr\{g(\boldsymbol{x}, \xi) \leq 0\} \geq \alpha$ can be written in the following form,

$$Pr\{h(\boldsymbol{x}) \leq \xi\} \geq \alpha \tag{2.42}$$

It is clear that, for each given confidence level $\alpha(0 < \alpha < 1)$, there exists a number K_α (may be multiple or ∞) such that

$$Pr\{K_\alpha \leq \xi\} = \alpha \tag{2.43}$$

and the probability $Pr\{\alpha \leq \xi\}$ will increase if K_α is replaced with a smaller number. Hence $Pr\{h(\boldsymbol{x}) \leq \xi\} \geq \alpha$ if and only if $h(\boldsymbol{x}) \leq K_\alpha$.

Notice that the equation $Pr\{\alpha \leq \xi\} = 1 - \Phi(K_\alpha)$ always holds, and we have, by (2.43),

$$K_\alpha = \Phi^{-1}(1-\alpha),$$

where Φ^{-1} is the inverse function of Φ. Sometimes, the solution of (2.43) is not unique. Equivalently, the function Φ^{-1} is multi-valued. For this case, we should choose it as the largest one, i.e.,

$$K_\alpha = sup\{K | K = \Phi^{-1}(1-\alpha)\}.$$

Thus the deterministic equivalent is $h(\boldsymbol{x}) \leq K_\alpha$. The theorem is proved. □

Theorem 2.5. *Assume that the random vector* $\boldsymbol{\xi} = (a_1, a_2, \ldots, a_n, b)$ *and the function* $g(\boldsymbol{x}, \boldsymbol{\xi})$ *has the form* $g(\boldsymbol{x}, \boldsymbol{\xi}) = a_1x_1 + a_2x_2 + \cdots + a_nx_n - b$. *If* a_i *and* b *are assumed to be independently normally distributed variables, then* $Pr\{g(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0\} \geq \alpha$ *if and only if*

$$\sum_{i=1}^{n} E[a_i]x_i + \Phi^{-1}(\alpha)\sqrt{\sum_{i=1}^{n} V[a_i]x_i^2 + V[b]} \leq E[b] \tag{2.44}$$

where Φ *is the standardized normal distribution.*

Proof. The chance constraint $Pr\{g(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0\} \geq \alpha$ can be written in the following form,

$$Pr\left\{\sum_{i=1}^{n} a_i x_i \leq b\right\} \geq \alpha \tag{2.45}$$

Since a_i and b are assumed to be independently normally distributed variables, the function

$$y(\boldsymbol{x}) = \sum_{i=1}^{n} a_i x_i - b$$

is also normally distributed with the following expected value and variance,

$$E[y(\boldsymbol{x})] = \sum_{i=1}^{n} E[a_i]x_i - E[b],$$
$$V[y(\boldsymbol{x})] = \sum_{i=1}^{n} V[a_i]x_i^2 + V[b].$$

We note that

$$\frac{\sum_{i=1}^{n} a_i x_i - b - (\sum_{i=1}^{n} E[a_i]x_i - E[b])}{\sqrt{\sum_{i=1}^{n}\sum_{i=1}^{n} V[a_i]x_i^2 + V[b]}}$$

must be standardized normally distributed. Since the inequality $\sum_{i=1}^{n} a_i x_i \leq b$ is equivalent to

$$\frac{\sum_{i=1}^{n} a_i x_i - b - \left(\sum_{i=1}^{n} E[a_i]x_i - E[b]\right)}{\sqrt{\sum_{i=1}^{n}\sum_{i=1}^{n} V[a_i]x_i^2 + V[b]}} \leq -\frac{\sum_{i=1}^{n} E[a_i]x_i - E[b]}{\sqrt{\sum_{i=1}^{n}\sum_{i=1}^{n} V[a_i]x_i^2 + V[b]}}.$$

The chance constraint (2.45) is equivalent to

$$Pr\left\{\eta \leq -\frac{\sum_{i=1}^{n} E[a_i]x_i - E[b]}{\sqrt{\sum_{i=1}^{n}\sum_{i=1}^{n} V[a_i]x_i^2 + V[b]}}\right\} \geq \alpha \tag{2.46}$$

where η is the standardized normally distributed variable. Then the chance constraint (2.46) holds if and only if

$$\Phi^{-1}(\alpha) \leq -\frac{\sum_{i=1}^{n} E[a_i]x_i - E[b]}{\sqrt{\sum_{i=1}^{n}\sum_{i=1}^{n} V[a_i]x_i^2 + V[b]}}\alpha$$

That is, the deterministic equivalent of chance constraint is (2.44). The theorem is proved. □

2.4.2.1 Crisp Equivalent Model

Then let's still consider a class of multi-objective linear models as follows,

$$\begin{cases} \max\left[\bar{c}_1^T\boldsymbol{x}, \bar{c}_2^T\boldsymbol{x}, \ldots, \bar{c}_m^T\boldsymbol{x}\right] \\ \text{s.t.} \begin{cases} \bar{e}_r^T\boldsymbol{x} \le \bar{b}_r, \ r = 1, 2, \ldots, p \\ \boldsymbol{x} \ge 0 \end{cases} \end{cases} \tag{2.47}$$

where $\boldsymbol{x} \in X$, $\bar{c}_i = (\bar{c}_{i1}, \bar{c}_{i2}, \ldots, \bar{c}_{in})^T$, $\bar{e}_r = (\bar{e}_{r1}, \bar{e}_{r2}, \ldots, \bar{e}_{rn})^T$ are random vectors, and $\bar{b}_r$ are random variables, $i = 1, 2, \ldots, m$, $r = 1, 2, \ldots, p$.

Based on the chance-constrained operator, the CCM of problem (2.47) can be defined as

$$\begin{cases} \max[\bar{f}_1, \bar{f}_2, \ldots, \bar{f}_m] \\ \text{s.t.} \begin{cases} Pr\{\bar{c}_i^T\boldsymbol{x} \ge \bar{f}_i\} \ge \beta_i, & i = 1, 2, \ldots, m \\ Pr\{\bar{e}_r^T\boldsymbol{x} \le \bar{b}_r\} \ge \alpha_i, & r = 1, 2, \ldots, p \\ \boldsymbol{x} \ge 0, 0 \le \alpha_r, \beta_i \le 1 \end{cases} \end{cases} \tag{2.48}$$

where α_i and β_i are predetermined confidence levels decision makers gave.

Theorem 2.6. *Assume that random vector $\bar{c}_i = (\bar{c}_{i1}, \bar{c}_{i2}, \ldots, \bar{c}_{in})^T$ is normally distributed with mean vector $\mu_i^c = (\mu_{i1}^c, \mu_{i2}^c, \ldots, \mu_{in}^c)^T$ and positive definite covariance matrix V_i^c, written as $\bar{c}_i \sim \mathcal{N}(\mu_i^c, V_i^c)(i=1, 2, \ldots, m)$ and random vector $\bar{e}_r \sim \mathcal{N}(\mu_r^e, V_r^e)$, $\bar{b}_r \sim \mathcal{N}(\mu_r^b, (\sigma_r^b)^2)$ $(r = 1, 2, \ldots, p)$. Assume that for any $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$ and $r = 1, 2, \ldots, p$, $\bar{c}_{ij}$, $\bar{e}_{ij}$ and $\bar{b}_{ij}$ are independently random variables. Then problem (2.48) is equivalent to*

$$\begin{cases} \max[H_1(\boldsymbol{x}), H_2(\boldsymbol{x}), \ldots, H_m(\boldsymbol{x})] \\ s.t. \begin{cases} g_r(\boldsymbol{x}) \le 0, r = 1, 2, \ldots, p \\ \boldsymbol{x} \ge 0, 0 \le \alpha_r, \beta_i \le 1 \end{cases} \end{cases}$$

where $H_i(\boldsymbol{x}) = \Phi^{-1}(1-\beta_i)\sqrt{\boldsymbol{x}^T V_i^c \boldsymbol{x}} + \mu_i^{cT}\boldsymbol{x}$, $g_r(\boldsymbol{x}) = \Phi^{-1}(\alpha_r)$ $\sqrt{\boldsymbol{x}^T V_r^e \boldsymbol{x} + (\sigma_r^b)^2} + \mu_r^{eT}\boldsymbol{x} - \mu_r^b$ and Φ is the standardized normal distribution.

Proof. Since random vector $\bar{c}_i$ is normally distributed with mean vector μ_i^c and positive definite covariance matrix V_i^c, then we have that

$$\bar{c}_i^T\boldsymbol{x} = \sum_{j=1}^{n} x_j\bar{c}_{ij} \sim \mathcal{N}(\sum_{j=1}^{n} x_j\mu_{ij}^c, \boldsymbol{x}^T V_i^c \boldsymbol{x}) = \mathcal{N}(\mu_i^{cT}\boldsymbol{x}, \boldsymbol{x}^T V_i^c \boldsymbol{x}).$$

It follows that

$$\frac{\bar{c}_i^T\boldsymbol{x} - \mu_i^{cT}\boldsymbol{x}}{\sqrt{\boldsymbol{x}^T V_i^c \boldsymbol{x}}}$$

must be standardized normally distributed. For any $i\,(i = 1, 2, \ldots, m)$, we have

$$
\begin{aligned}
Pr\{\bar{c}_i^T \boldsymbol{x} \geq \bar{f}_i\} \geq \beta_i \Leftrightarrow \beta_i &\leq Pr\left\{\frac{\bar{c}_i^T x - \mu_i^{cT} x}{\sqrt{x^T V_i^c x}} \geq \frac{\bar{f}_i - \mu_i^{cT} x}{\sqrt{x^T V_i^c x}}\right\} \\
&\Leftrightarrow \beta_i \leq 1 - Pr\left\{\frac{\bar{c}_i^T x - \mu_i^{cT} x}{\sqrt{x^T V_i^c x}} \leq \frac{\bar{f}_i - \mu_i^{cT} x}{\sqrt{x^T V_i^c x}}\right\} \\
&\Leftrightarrow \beta_i \leq 1 - \Phi\left(\frac{\bar{f}_i - \mu_i^{cT} x}{\sqrt{x^T V_i^c x}}\right) \\
&\Leftrightarrow \bar{f}_i \leq \Phi^{-1}(1 - \beta_i)\sqrt{x^T V_i^c x} + \mu_i^{cT} x
\end{aligned}
$$

where Φ is the standardized normal distribution. Similarly, we know that

$$\bar{e}_r^T \boldsymbol{x} \sim \mathcal{N}(\mu_r^{eT} \boldsymbol{x}, \boldsymbol{x}^T V_r^e \boldsymbol{x}).$$

Since $\bar{b}_r \sim \mathcal{N}(\mu_r^b, (\sigma_r^b)^2)$, it follows from, $\bar{e}_{ij}$ and $\bar{b}_{ij}$ are independently random variables for any r, j, that

$$
\begin{aligned}
E[\bar{e}_r^T \boldsymbol{x} - \bar{b}_r] &= E[\bar{e}_r^T \boldsymbol{x}] - E[\bar{b}_r] = \mu_r^{eT} \boldsymbol{x} - \mu_r^b, \\
V[\bar{e}_r^T \boldsymbol{x} - \bar{b}_r] &= V[\bar{e}_r^T \boldsymbol{x}] + V[\bar{b}_r] = \boldsymbol{x}^T V_r^e \boldsymbol{x} + (\sigma_r^b)^2,
\end{aligned}
$$

then $\bar{e}_r^T \boldsymbol{x} - \bar{b}_r$ is also normally distributed with

$$\bar{e}_r^T \boldsymbol{x} - \bar{b}_r \sim \mathcal{N}(\mu_r^{eT} \boldsymbol{x} - \mu_r^b, \boldsymbol{x}^T V_r^e \boldsymbol{x} + (\sigma_r^b)^2).$$

For any $r\,(r = 1, 2, \ldots, p)$, we have

$$
\begin{aligned}
Pr\{\bar{e}_r^T \boldsymbol{x} \geq \bar{b}_r\} \geq \alpha_r \Leftrightarrow \alpha_r &\leq Pr\left\{\frac{\bar{e}_r^T \boldsymbol{x} - \bar{b}_r - (\mu_r^{eT} \boldsymbol{x} - \mu_r^b)}{\sqrt{x^T V_r^e \boldsymbol{x} + (\sigma_r^b)^2}} \right. \\
&\qquad \left. \leq -\frac{\mu_r^{eT} \boldsymbol{x} - \mu_r^b}{\sqrt{\boldsymbol{x}^T V_r^e \boldsymbol{x} + (\sigma_r^b)^2}}\right\} \\
&\Leftrightarrow \alpha_r \leq \Phi\left(-\frac{\mu_r^{eT} \boldsymbol{x} - \mu_r^b}{\sqrt{\boldsymbol{x}^T V_r^e \boldsymbol{x} + (\sigma_r^b)^2}}\right) \\
&\Leftrightarrow \Phi^{-1}(\alpha_r)\sqrt{\boldsymbol{x}^T V_r^e \boldsymbol{x} + (\sigma_r^b)^2} + \mu_r^{eT} \boldsymbol{x} - \mu_r^b \leq 0
\end{aligned}
$$

where Φ is the standardized normal distribution. Then we have (2.48) is equivalent to

$$\begin{cases} \max[\bar{f}_1, \bar{f}_2, \ldots, \bar{f}_m] \\ \text{s.t.} \begin{cases} \bar{f}_i \le \Phi^{-1}(1-\beta_i)\sqrt{\boldsymbol{x}^T V_i^c \boldsymbol{x}} + \mu_i^{cT}\boldsymbol{x},\ i = 1, 2, \ldots, m \\ \Phi^{-1}(\alpha_r)\sqrt{\boldsymbol{x}^T V_r^e \boldsymbol{x} + (\sigma_r^b)^2} + \mu_r^{eT}\boldsymbol{x} - \mu_r^b \le 0,\ r = 1, 2, \ldots, p \\ \boldsymbol{x} \ge 0,\ 0 \le \alpha_r, \beta_i \le 1 \end{cases} \end{cases} \tag{2.49}$$

Let $H_i(\boldsymbol{x}) = \Phi^{-1}(1-\beta_i)\sqrt{\boldsymbol{x}^T V_i^c \boldsymbol{x}} + \mu_i^{cT}\boldsymbol{x}$ and $g_r(\boldsymbol{x}) = \Phi^{-1}(\alpha_r)\sqrt{\boldsymbol{x}^T V_r^e \boldsymbol{x} + (\sigma_r^b)^2} + \mu_r^{eT}\boldsymbol{x} - \mu_r^b$. In view of the purpose of maximizing $\bar{f}_i$ in problem (2.49), it also can be converted into

$$\begin{cases} \max[H_1(\boldsymbol{x}), H_2(\boldsymbol{x}), \ldots, H_m(\boldsymbol{x})] \\ \text{s.t.} \begin{cases} g_r(\boldsymbol{x}) \le 0, r = 1, 2, \ldots, p \\ \boldsymbol{x} \ge 0, 0 \le \alpha_r, \beta_i \le 1 \end{cases} \end{cases} \tag{2.50}$$

This completes the proof. □

For the crisp multi-objective programming problem (2.50), whether there are some optimal solutions is the spot we pay more attention to. Sometimes, the convexity of the multi-objective problem provides an useful rule researchers consult. Next, let's introduce the definition of the convexity of the multi-objective problem.

Definition 2.18. For the following multi-objective programming problem,

$$\begin{cases} \min f(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})) \\ s.t.\ g_r(\boldsymbol{x}) \ge 0, r = 1, 2, \ldots, p \end{cases} \tag{2.51}$$

If $f_i(\boldsymbol{x})$ are convex functions for all i, and $g_r(\boldsymbol{x})$ are concave functions for all r, then (2.51) is a convex programming.

Then let's discuss the convexity of the problem (2.50).

Theorem 2.7. *Let* $X = \{\boldsymbol{x} \in R^n | \Phi^{-1}(\alpha_r)\sqrt{\boldsymbol{x}^T V_r^e \boldsymbol{x} + (\sigma_r^b)^2} + \mu_r^{eT}\boldsymbol{x} - \mu_r^b \le 0, r = 1, 2, \ldots, p; \boldsymbol{x} \ge 0\}$. *If* α_r *and* β_i *are more than 0.5,* $\bar{c}_{ij}$, $\bar{e}_{rj}$ *and* $\bar{b}_r$ *are respectively independent,* $i = 1, 2, \ldots, m, j = 1, 2, \ldots, n, r = 1, 2, \ldots, p$, *then the problem (2.50) is convex.*

Proof. Since $\bar{c}_{ij}$ are independent random variable with each other, the covariance matrix has the following form,

$$V_i^c = \begin{pmatrix} (\sigma_{i1}^c)^2 & 0 & \cdots & 0 \\ \vdots & & \cdots & \vdots \\ 0 & 0 & \cdots & (\sigma_{in}^c)^2 \end{pmatrix}.$$

It's obvious that V_i^c is a positive definite matrix. Then according to [301], $\sqrt{\boldsymbol{x}^T V_i^c \boldsymbol{x}}$ is a convex function. In addition, since $\alpha_r \geq 0.5$ and $\beta_i \geq 0.5$, it follows that $\Phi^{-1}(\alpha_r) \geq 0$ and $\Phi^{-1}(1-\beta_i) \leq 0$, we know $H_i(\boldsymbol{x})$ are concave functions, $i = 1, 2, \ldots, m$. Really, let $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ be any two points in feasible set, and $\lambda \in [0, 1]$, we have that

$$H_i[\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2] \geq \lambda H_i(\boldsymbol{x}_1) + (1-\alpha) H_i(\boldsymbol{x}_2).$$

Next, we prove that X is convex. Since

$$X = \{\boldsymbol{x} \in R^n | \Phi^{-1}(\alpha_r)\sqrt{\boldsymbol{x}^T V_r^e \boldsymbol{x} + (\sigma_r^b)^2} + \mu_r^{eT}\boldsymbol{x} - \mu_r^b \leq 0, r = 1, 2, \ldots, p; \boldsymbol{x} \geq 0\},$$

and $\Phi^{-1}(\alpha_r) \geq 0$, it follows that $g_r(\boldsymbol{x}) = \Phi^{-1}(\alpha_r)\sqrt{\boldsymbol{x}^T V_r^e \boldsymbol{x} + (\sigma_r^b)^2} + \mu_r^{eT}\boldsymbol{x} - \mu_r^b \leq 0$ are convex functions, $r = 1, 2, \ldots, p$. Really, let $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ be two feasible solutions, then

$$g_r(\boldsymbol{x}_1) \leq 0, g_r(\boldsymbol{x}_2) \leq 0,$$

according to g_r's convexity, we have

$$g_r[\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2] \leq \lambda g_r(\boldsymbol{x}_1) + (1-\lambda) g_r(\boldsymbol{x}_2) \leq 0,$$

where $0 \leq \lambda \leq 1$, $r = 1, 2, \ldots, p$. This means that $\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2$ is also a feasible solution. So X is a convex set. Above all, we can conclude from Definition 2.18 that problem (2.50) is a convex programming and its global optimal solution can be obtained easily. □

2.4.2.2 Lexicographic Method

The basic idea of lexicographic method is to rank the objective function by its importance to decision makers and then resolve the next objective function after resolving the above one. We take the solution of the last programming problem as the final solution.

Consider the following multi-objective programming problem,

$$\begin{cases} \min[f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})] \\ \text{s.t. } \boldsymbol{x} \in X. \end{cases} \tag{2.52}$$

Without loss of generality, assume the rank as $f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})$ according to different importance. Solve the following single objective problem in turn,

$$\min_{x \in X} f_1(\boldsymbol{x}) \tag{2.53}$$

$$\begin{cases} \min f_i(\boldsymbol{x}) \\ \text{s.t.} \begin{cases} f_k(\boldsymbol{x}) = f_k(\boldsymbol{x}^k),\ k = 1, 2, \ldots, i-1 \\ \boldsymbol{x} \in X \end{cases} \end{cases} \tag{2.54}$$

where $i = 1, 2, \ldots, m$, X is the feasible area and denote the feasible area of problem (2.54) as X^i.

Theorem 2.8. *Let $X \subset \mathbf{R}^n$, $\boldsymbol{f} : X \to \mathbf{R}^m$. If $\boldsymbol{x}^m$ be the optimal solution by the lexicographic method, then $\boldsymbol{x}^m$ is an efficient solution of problem (2.52).*

Proof. If $\boldsymbol{x}^m$ is not an efficient solution of problem (2.52), there exists $\bar{\boldsymbol{x}} \in X$ such that $\boldsymbol{f}(\bar{\boldsymbol{x}}) \leq \boldsymbol{f}(\boldsymbol{x}^m)$. Since $f_1(\boldsymbol{x}^m) = f_1^* = f_1(\boldsymbol{x}^1)$, $f_1(\bar{\boldsymbol{x}}) < f_1(\boldsymbol{x}^m)$ cannot hold. It necessarily follows that $f_1(\bar{\boldsymbol{x}}) = f_1(\boldsymbol{x}^m)$.

If we have proved $f_k(\bar{\boldsymbol{x}}) = f_k(\boldsymbol{x}^m)$ $(k = 1, 2, \ldots, i-1)$, but $f_i(\bar{\boldsymbol{x}}) < f_i(\boldsymbol{x}^m)$. It follows that $\bar{\boldsymbol{x}}$ is a feasible solution of problem (2.54). Since $f_i(\bar{\boldsymbol{x}}) < f_i(\boldsymbol{x}^m) = f_i(\boldsymbol{x}^i)$, this results in the conflict with that $\boldsymbol{x}^i$ the optimal solution of problem (2.54). Thus, $f_k(\bar{\boldsymbol{x}}) = f_k(\boldsymbol{x}^m)(k = 1, 2, \ldots, i)$ necessarily holds. Then we can prove $f_k(\bar{\boldsymbol{x}}) = f_k(\boldsymbol{x}^m)(k = 1, 2, \ldots, m)$ by the mathematical induction. This conflicts with $\boldsymbol{f}(\bar{\boldsymbol{x}}) \leq \boldsymbol{f}(\boldsymbol{x}^m)$. This completes the proof. □

2.4.3 Nonlinear Random CCM and Random Simulation-Based ASA

Let's return to the nonlinear multi-objective programming problem,

$$\begin{cases} \max[\bar{f_1}, \bar{f_2}, \ldots, \bar{f_m}] \\ \text{s.t.} \begin{cases} Pr\{f_i(\boldsymbol{x}, \boldsymbol{\xi}) \geq \bar{f_i}\} \geq \beta_i,\ i = 1, 2, \ldots, m \\ Pr\{g_r(\boldsymbol{x}, \boldsymbol{\xi}) \leq b_r\} \geq \alpha_r,\ r = 1, 2, \ldots, p \\ \boldsymbol{x} \geq 0, 0 \leq \alpha_r, \beta_i \leq 1 \end{cases} \end{cases} \tag{2.55}$$

where $f_i(\boldsymbol{x}, \boldsymbol{\xi})$ or $g_r(\boldsymbol{x}, \boldsymbol{\xi})$ or both of them are nonlinear functions with respect $\boldsymbol{\xi}$ is a random vector, $i = 1, 2, \ldots, m$, $r = 1, 2, \ldots, p$. If their distributions are all crisp, we can convert it into a crisp programming problem. For the class of problems which cannot be converted into crisp ones, the random simulation is a useful tool to compute the probability of a random variable.

2.4.3.1 Random Simulation for CCM

Suppose that $\boldsymbol{\xi}$ is an n-dimensional random vector defined on the probability space $(\Omega, \mathscr{A}, Pr)$, and $f : \mathbf{R}^n \to \mathbf{R}$ is a measurable function. The problem is to determine the maximal value $\bar{f}$ for given $\boldsymbol{x}$ such that

$$Pr\{f(\boldsymbol{x}, \boldsymbol{\xi}) \geq \bar{f}\} \geq \alpha \tag{2.56}$$

where α is a predetermined confidence level with $0 < \alpha < 1$. We generate ω_k from Ω according to the probability measure Pr, and write $\boldsymbol{\xi}_k = \boldsymbol{\xi}(\omega_k)$ for $k = 1, 2, \ldots, N$, where $\boldsymbol{\omega_k} = (\omega_k^1, \ldots, \omega_k^n)$ is an n-dimensional vector and ω_k^j is generated according to the random variable ξ_k. Now we define

$$h(\boldsymbol{x}, \boldsymbol{\xi}_k) = \begin{cases} 1, \text{ if } f(\boldsymbol{x}, \boldsymbol{\xi}_k) \geq \bar{f} \\ 0, \text{ otherwise} \end{cases} \tag{2.57}$$

for $k = 1, 2, \ldots, N$, which are a sequence of random variables, and $E[h(\boldsymbol{x}, \boldsymbol{\xi}_k)] = \alpha$ for all k. By the strong law of large numbers, we obtain

$$\frac{\sum_{k=1}^{N} h(\boldsymbol{x}, \boldsymbol{\xi}_k)}{N} \to \alpha,$$

as $N \to \infty$. Note that the sum $\sum_{k=1}^{N} h(\boldsymbol{x}, \boldsymbol{\xi}_k)$ is just the number of $\boldsymbol{\xi}_k$ satisfying $f(\boldsymbol{x}, \boldsymbol{\xi}_k) \geq \bar{f}$ for $k = 1, 2, \ldots, N$. Thus the value $\bar{f}$ can be taken as the N'th largest element in the sequence $\{f(\boldsymbol{x}, \boldsymbol{\xi}_1), f(\boldsymbol{x}, \boldsymbol{\xi}_2), \ldots, f(\boldsymbol{x}, \boldsymbol{\xi}_N)\}$, where N' is the integer part of αN. Then the procedure simulating the critical value $\bar{f}$ of $Pr\{f(\boldsymbol{x}, \boldsymbol{\xi}) \geq \bar{f}_i\}$ can be summarized as follows:

Procedure `Random simulation for CCM`

Input: The decision vector $\boldsymbol{x}$
Output: The critical value $\bar{f}$
Step 1. Set $L = 0$;
Step 2. Generate $\omega_1, \omega_2, \ldots, \omega_N$ from Ω according to the probability measure Pr;
Step 3. Return the N'th largest element in $\{f(\boldsymbol{x}, \boldsymbol{\xi}_1), f(\boldsymbol{x}, \boldsymbol{\xi}_2), \ldots, f(\boldsymbol{x}, \boldsymbol{\xi}_N)\}$.

Example 2.5. Let us employ the random simulation to search for the maximal $\bar{f}$ such that

$$Pr\left\{\sqrt{\xi_1^2 + \xi_2^2 + \xi_3^2} \geq \bar{f}\right\} \geq 0.8,$$

where $\xi_1 \sim exp(1)$ is an exponentially distributed variable, $\xi_2 \sim \mathcal{N}(3, 1)$ is a normally distributed variable, and $\xi_3 \sim \mathcal{U}(0, 1)$ is a uniformly distributed variable. A run of stochastic simulation with 10000 cycles shows that $\bar{f} = 2.0910$.

2.4.3.2 Adaptive Simulated Algorithm

Too often the management of complex systems is ill-served by not utilizing the best tools available. For example, requirements set by decision-makers often are not formulated in the same language as constructs formulated by powerful mathematical

formalisms, and so the products of analyses are not properly or maximally utilized, even if and when they come close to faithfully representing the powerful intuitions they are supposed to model. In turn, even powerful mathematical constructs are ill-served, especially when dealing with approximations to satisfy constraints of numerical algorithms familiar to particular analysts, but which tend to destroy the power of the intuitive constructs developed by decision-makers. In order to deal with fitting parameters or exploring sensitivities of variables, as models of systems have become more sophisticated in describing complex behavior, it has become increasingly important to retain and respect the nonlinearities inherent in these models, as they are indeed present in the complex systems they model. The adaptive simulated algorithm (abbr. ASA) can help to handle these fits of nonlinear models of real-world data.

ASA, also known as the very fast simulated reannealing, is a very efficient version of SA. ASA is a global optimization technique with certain advantages. It is versatile and needs very few parameters to tune. The distinct feature of this method is the temperature change mechanism, which is an important part of the transition probability equation. The conventional can allows a higher chance of transition to a worse solution when beginning with a high temperature. By doing so, the search can move out of the local optimization. However, as the search process develops, the continuously declining temperature will result in a reduced chance of uphill transition. Such an approach could be useful if the local optimization is near the start point, but may not lead to a near optimal solution if some local optimization is encountered at a relatively low temperature toward the end of the search. Therefore, a improved method should be considered to alleviated this difficulty. In conventional method, the cooling schedule is usually monotonically nonincreasing. In ASA, an adaptive cooling schedule based on the profile of the search path to dynamically adjust the temperature. Such adjustments could be in any direction including the possibility of reheating. Some scholars have proposed the idea of reversing the temperature before. Dowsland [83] considered two functions together to control temperature in the application to packing problem. The first function is a function that reduces the temperature, whereas the second function is used as a heating up function that gradually increases the temperature if needed. In ASA, a single function is proposed to maintain the temperature above a minimum level. If there is any upward move, the heating process gradually takes place, but the cooling process may suddenly take place by the first downhill move. Azizi [12] proposed the following temperature control function,

$$T_i = T_{\min} + \lambda ln(1 + r_i), \tag{2.58}$$

where $T_{\min}$ is the minimum value that the temperature can take, λ is a coefficient that controls the rate of temperature rise, and r_i is the number of consecutive upward moves at iteration i. The initial value of r_i is zero, thus the initial temperature $T_0 = T_{\min}$. The purpose of the minimum temperature, $T_{\min}$, is two-fold. Firstly, it prevents the probability function from becoming invalid when r_i is zero. Secondly, it determines the initial value of the temperature. $T_{\min}$ can take any value greater

than zero. The parameter λ controls the rate of temperature rise. The greater value of λ, the faster the temperature rises. The search spends less time looking for good solutions in its current neighborhood when a large value is assigned to λ. Similarly, by assigning a small value to the parameter λ, the search spends more time looking for better solutions in the neighborhood. Choosing a value for the parameter λ could be linked to computation time, which also depends on the size and complexity of the problem. We don't clarify other parts about ASA, readers can find more detailed analysis of the algorithm in [12, 52, 141–143].

Consider the following general optimization problem,

$$\min_{\boldsymbol{x}\in X} f(\boldsymbol{x}) \tag{2.59}$$

where $\boldsymbol{x} = [x_1, \ldots, x_n]^T$ is the n-dimensional decision vector to be optimized, X is the feasible set of $\boldsymbol{x}$. The cost function $f(\boldsymbol{x})$ can be multi-modal and non-smooth. The ASA is a global optimization scheme for solving this kind of constrained optimization problems. Next, let's introduce how to use ASA to solve the above problem.

Although there are many possible realizations of the ASA, an implementation is illustrated in Fig. 2.5, and this algorithm is detailed as follows:

Step 1. An initial $\boldsymbol{x} \in X$ is randomly generated, the initial temperature of the acceptance probability function, $T_{\text{accept}}(0)$, is set to $f(\boldsymbol{x})$, and the initial temperatures of the parameter generating probability functions, $T_{i,gen}(0)$, $1 \leq i \leq n$, are set

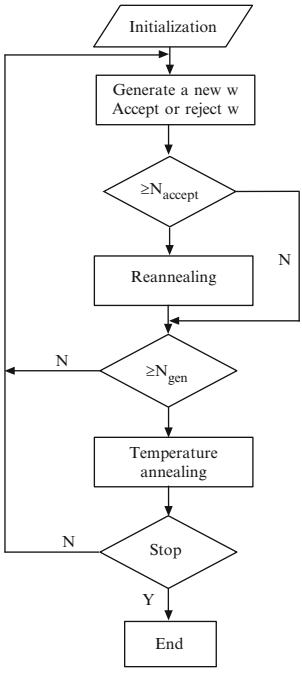

Fig. 2.5 Flow chart of ASA algorithm

to 1.0. A user-defined control parameter c in annealing is given, and the annealing times, k_i for $1 \leq i \leq n$ and k_a, are all set to 0.

Step 2. The algorithm generates a new point in the parameter space with

$$x_i^{\text{new}} = x_i^{\text{old}} + q_i M_i, \ 1 \leq i \leq n \tag{2.60}$$

where M_i is a positive number determined by the difference between the upper and lower boundary of x_i and

$$\boldsymbol{x}^{\text{new}} \in X \tag{2.61}$$

where q_i is calculated as

$$q_i = sgn(\upsilon_i - \frac{1}{2}) T_{i,gen}(k_i) \times \left(\left(1 + \frac{1}{T_{i,gen}(k_i)} \right)^{|2\upsilon_i - 1|} - 1 \right) \tag{2.62}$$

and υ_i a uniformly distributed random variable in [0,1]. Notice that if a generated $\boldsymbol{x}^{\text{new}}$ is not in X, it is simply discarded and a new point is tried again until $\boldsymbol{x}^{\text{new}} \in X$.

The value of the cost function $f(\boldsymbol{x}^{\text{new}})$ is then evaluated and the acceptance probability function of $\boldsymbol{x}^{\text{new}}$ is given by

$$P_{\text{accept}} = \frac{1}{1 + exp((f(\boldsymbol{x}^{\text{new}}) - f(\boldsymbol{x}^{\text{old}}))/T_{\text{accept}}(K_a))} \tag{2.63}$$

A uniform random variable P_{unif} is generated in [0,1]. If $P_{unif} \leq P_{\text{accept}}$, $\boldsymbol{x}^{\text{new}}$ is accepted; otherwise it is rejected.

Step 3. After every N_{accept} acceptance points, reannealing takes place by first calculating the sensitivies

$$s_i = \left| \frac{f(\boldsymbol{x}^{\text{best}} + e_i \delta) - f(\boldsymbol{x}^{\text{best}})}{\delta} \right|, \ 1 \leq i \leq n \tag{2.64}$$

where $\boldsymbol{x}^{\text{best}}$ is the best point found so far, δ is a small step size, the n-dimensional vector e_i has unit ith element and the rest of elements of e_i are all zeros. Let $s_{\max} = \max\{s_i, 1 \leq i \leq n\}$. Each parameter generating temperature $T_{i,gen}$ is scaled by a factor $s_{\max}/s_i$ and the annealing time k_i is reset,

$$T_{i,\,gen}(k_i) = \frac{s_{\max}}{s_i} T_{i,gen}(k_i), \ k_i = \left(-\frac{1}{c} log \left(\frac{T_{i,\,gen}(k_i)}{T_{i,\,gen}(0)} \right) \right)^n.$$

Similarly, $T_{\text{accept}}(0)$ is reset to the value of the last accepted cost function, $T_{\text{accept}}(k_a)$ is reset to $f(\boldsymbol{x}^{\text{best}})$ and the annealing time k_a is rescaled accordingly,

$$k_a = \left(-\frac{1}{c} log \left(\frac{T_{i,\,gen}(k_a)}{T_{i,\,gen}(0)} \right) \right)^n \tag{2.65}$$

Step 4. After every N_{gen} generated points, annealing takes place with

$$k_i = k_i + 1,\ T_{i,\ gen}(k_i) = T_{i,\ gen}(0)exp(-ck_i^{1/n});$$

and

$$k_a = k_a + 1,\ T_{\text{accept}}(k_a) = T_{\text{accept}}(0)exp(-ck_a^{1/n});$$

otherwise, go to step 2.

Step 5. The algorithm is terminated if the parameters have remained unchanged for a few successive reannealing or a preset maximum number of cost function evaluations has been reached; otherwise, go to step 2.

As in a standard SA algorithm, this ASA contains two loops. The inner loop ensures that the parameter space is searched sufficiently at a given temperature, which is necessary to guarantee that the algorithm finds a global optimum. The ASA also uses only the value of the cost function in the optimization process and is very simple to program.

Last, we discuss about the algorithm parameter tuning. For the above ASA algorithm, most of the algorithm parameters are automatically set and "tuneda", and the user only needs to assign a control parameter c and set two values N_{accept} and N_{genera}. Obviously, the optimal values of N_{accept} and N_{genera} are problem dependent, but our experience suggests that an adequate choice for N_{accept} is in the range of tens to hundreds and an appropriate value for N_{gen} is in the range of hundreds to thousands. The annealing rate control parameter c can be determined form the chosen initial temperature, final temperature and predetermined number of annealing steps. We have found out that a choice of c in the range 1.0–10.0 is often adequate.

It should be emphasized that, as the ASA has excellent self adaptation ability, the performance of the algorithm is not critically influenced by the specific chosen values of c, N_{accept} and N_{gen}.

2.4.4 Numerical Examples

Example 2.6. Let's still consider the following problem,

$$\begin{cases} \max f_1(\boldsymbol{x}, \boldsymbol{\xi}) = \xi_1 x_1 + \xi_2 x_2 + \xi_3 x_3 + \xi_4 x_4 + \xi_5 x_5 \\ \max f_2(\boldsymbol{x}, \boldsymbol{\xi}) = c_1\xi_6 x_1 + c_2\xi_7 x_2 + c_3\xi_8 x_3 + c_4\xi_9 x_4 + c_5\xi_{10} x_5 \\ \text{s.t.} \begin{cases} x_1 + x_2 + x_3 + x_4 + x_5 \le 350 \\ x_1 + x_2 + x_3 + x_4 + x_5 \ge 300 \\ 4x_1 + 2x_2 + 1.5x_3 + x_4 + 2x_5 \le 1085 \\ x_1 + 4x_2 + 2x_3 + 5x_4 + 3x_5 \le 660 \\ x_1 \ge 20, x_2 \ge 20, x_3 \ge 20, x_4 \ge 20, x_5 \ge 20 \end{cases} \end{cases} \tag{2.66}$$

where $c = (c_1, c_2, c_3, c_4, c_5) = (1.2, 0.5, 1.3, 0.8, 0.9)$,

$$\xi_1 \sim \mathcal{N}(113, 1),\ \xi_2 \sim \mathcal{N}(241, 4),\ \ \xi_3 \sim \mathcal{N}(87, 1),\ \ \xi_4 \sim \mathcal{N}(56, 2),$$
$$\xi_5 \sim \mathcal{N}(92, 1),\ \ \xi_6 \sim \mathcal{N}(628, 1),\ \ \xi_7 \sim \mathcal{N}(143, 2),\ \xi_8 \sim \mathcal{N}(476, 2),$$
$$\xi_9 \sim \mathcal{N}(324, 2),\ \xi_{10} \sim \mathcal{N}(539, 2).$$

and $\xi_i (i = 1, 2, \ldots, 9)$ are independently random variables. We set $\beta_1 = \beta_2 = 0.9$, from (2.48) we have the following chance-constrained model,

$$\begin{cases} \max[\bar{f_1}, \bar{f_2}] \\ \text{s.t.} \begin{cases} Pr\{\xi_1 x_1 + \xi_2 x_2 + \xi_3 x_3 + \xi_4 x_4 + \xi_5 x_5 \geq \bar{f_1}\} \geq \beta_1 \\ Pr\{c_1\xi_6 x_1 + c_2\xi_7 x_2 + c_3\xi_8 x_3 + c_4\xi_9 x_4 + c_5\xi_{10} x_5 \geq \bar{f_2}\} \geq \beta_2 \\ x_1 + x_2 + x_3 + x_4 + x_5 \leq 350 \\ x_1 + x_2 + x_3 + x_4 + x_5 \geq 300 \\ 4x_1 + 2x_2 + 1.5x_3 + x_4 + 2x_5 \leq 1085 \\ x_1 + 4x_2 + 2x_3 + 5x_4 + 3x_5 \leq 660 \\ x_1 \geq 20, x_2 \geq 20, x_3 \geq 20, x_4 \geq 20, x_5 \geq 20 \end{cases} \end{cases} \tag{2.67}$$

Since $\Phi^{-1}(1-\beta_i) = -1.28, i = 1, 2$, from Theorem 2.6, problem (2.67) is equivalent to

$$\begin{cases} \max H_1(\boldsymbol{x}) = 113x_1 + 241x_2 + 87x_3 + 56x_4 + 92x_5 \\ \qquad\qquad -1.28\sqrt{x_1^2 + 4x_2^2 + x_3^2 + 2x_4^2 + x_5^2} \\ \max H_2(\boldsymbol{x}) = 753.6x_1 + 71.5x_2 + 618.8x_3 + 259.2x_4 + 485.1x_5 \\ \qquad\qquad -1.28\sqrt{x_1^2 + 2x_2^2 + 2x_3^2 + 2x_4^2 + 2x_5^2} \\ \text{s.t.} \begin{cases} x_1 + x_2 + x_3 + x_4 + x_5 \leq 350 \\ x_1 + x_2 + x_3 + x_4 + x_5 \geq 300 \\ 4x_1 + 2x_2 + 1.5x_3 + x_4 + 2x_5 \leq 1085 \\ x_1 + 4x_2 + 2x_3 + 5x_4 + 3x_5 \leq 660 \\ x_1 \geq 20, x_2 \geq 20, x_3 \geq 20, x_4 \geq 20, x_5 \geq 20 \end{cases} \end{cases} \tag{2.68}$$

Then we use the lexicographic method to resolve the above programming problem. Assume that the objective function $H_1(\boldsymbol{x})$ is more important than $H_2(\boldsymbol{x})$ to DM. Let's firstly solve the following problem,

$$\begin{cases} \max H_1(\boldsymbol{x}) = 113x_1 + 241x_2 + 87x_3 + 56x_4 + 92x_5 \\ \qquad\qquad -1.28\sqrt{x_1^2 + 4x_2^2 + x_3^2 + 2x_4^2 + x_5^2} \\ \text{s.t.} \begin{cases} x_1 + x_2 + x_3 + x_4 + x_5 \leq 350 \\ x_1 + x_2 + x_3 + x_4 + x_5 \geq 300 \\ 4x_1 + 2x_2 + 1.5x_3 + x_4 + 2x_5 \leq 1085 \\ x_1 + 4x_2 + 2x_3 + 5x_4 + 3x_5 \leq 660 \\ x_1 \geq 20, x_2 \geq 20, x_3 \geq 20, x_4 \geq 20, x_5 \geq 20 \end{cases} \end{cases} \tag{2.69}$$

Then we get the optimal solution $\boldsymbol{x}^* = (218.18, 59.10, 22.73, 20.00, 20.00)$ and the objective value $H_2(\boldsymbol{x}^*) = 197290.30$.

Secondly, construct the following programming problem,

$$\begin{cases} \max H_2(\boldsymbol{x}) = 753.6x_1 + 71.5x_2 + 618.8x_3 + 259.2x_4 + 485.1x_5 \\ \qquad\qquad -1.28\sqrt{x_1^2 + 2x_2^2 + 2x_3^2 + 2x_4^2 + 2x_5^2} \\ \text{s.t.} \begin{cases} 113x_1 + 241x_2 + 87x_3 + 56x_4 + 92x_5 \\ \quad - 1.28\sqrt{x_1^2 + 4x_2^2 + x_3^2 + 2x_4^2 + x_5^2} = 43510.72 \\ x_1 + x_2 + x_3 + x_4 + x_5 \le 350 \\ x_1 + x_2 + x_3 + x_4 + x_5 \ge 300 \\ 4x_1 + 2x_2 + 1.5x_3 + x_4 + 2x_5 \le 1085 \\ x_1 + 4x_2 + 2x_3 + 5x_4 + 3x_5 \le 660 \\ x_1 \ge 20, x_2 \ge 20, x_3 \ge 20, x_4 \ge 20, x_5 \ge 20 \end{cases} \end{cases} \tag{2.70}$$

Then we get the final optimal solution $\boldsymbol{x}^* = (218.18, 59.10, 22.73, 20.00, 20.00)$ and the objective value $H_1(\boldsymbol{x}^*) = 43510.72$.

Example 2.7. Consider the following multi-objective programming problem with random parameters,

$$\begin{cases} \max[\bar{f_1}, \bar{f_2}] \\ \text{s.t.} \begin{cases} Pr\{\sqrt{(x_1 - \xi_1)^2 + (x_2 - \xi_2)^2} \ge \bar{f_1}\} \ge \beta_1 \\ Pr\{\sqrt{(x_1 + \xi_1)^2 + (x_2 + \xi_2)^2} \ge \bar{f_2}\} \ge \beta_2 \\ x_1 + x_2 \le 5 \\ x_1 \ge 0, x_2 \ge 0 \end{cases} \end{cases} \tag{2.71}$$

where $\xi_1 \sim \mathscr{N}(2, 0.5)$ ia normally distributed random variable and $\xi_2 \sim \mathscr{U}(1, 2)$ is also a normally distributed random variable.

Next we will use the random simulation-based ASA to solve the above problem. Set the initial temperature T_0, the last temperature be 1 and the cooling method be 0.05% decrement once. The neighborhood can be constructed as follows,

$$x_1^1 = x_1^0 + rh, \;\; x_2^1 = x_2^0 + rh,$$

where r is a random number in (0,1) and h is the step length (here $h = 2.0$). After the simulation with many cycles, we get the optimal solution under different weights as shown in Table 2.4. Figure 2.6 shows the cooling process when the weight is 0.5. The real line expresses the weight sum of two objective functions, and it shows that it gradually converges from T = 25. The second figure shows the changes of two objective values when the temperature decreases.

Table 2.4 The optimal solution by random simulation-based ASA

w_1	w_2	x_1	x_2	$\bar{f}_1$	$\bar{f}_2$	$\bar{f}$	T_0
0.1	0.9	2.2823	0.0925	1.1123	6.0069	5.5175	1000
0.2	0.8	1.1542	2.7342	1.4170	5.8217	4.9407	1000
0.3	0.7	0.2140	1.9297	1.5559	5.8935	4.5922	1000
0.4	0.6	1.5625	0.2912	0.9358	5.8060	3.8579	1000
0.5	0.5	2.6555	1.1940	0.8505	5.7968	3.3236	1000

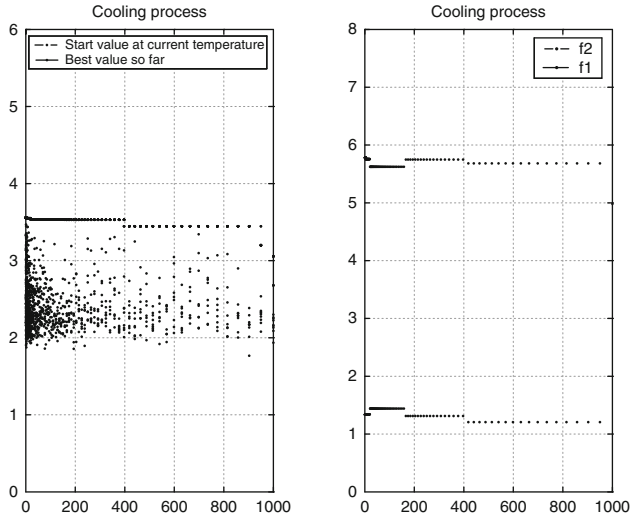

Fig. 2.6 The cooling process of random simulation-based ASA

2.5 Random DCM

In practice, there usually exist multiple events in a complex random decision system. Sometimes, the decision-maker wishes to maximize the chance functions of these events (i.e. the probabilities of satisfying the events). In order to model this type of random decision system, Schneider [280] developed another technique, called probability maximization model (also called dependent-chance model (abbr. DCM) by some scholars, we will use this name in this book), in which the underlying philosophy is based on selecting the decision with maximal chance to meet the event. Then [150] discussed some coverage probability maximization problems. From then on, it was widely used in many fields to solve some realistic problems [133, 146, 272].

DCM theory breaks the concept of feasible set and replaces it with uncertain environment. Roughly speaking, DCM involves maximizing chance functions of events in an uncertain environment. In deterministic model, expected value model (EVM), and chance-constrained programming (CCM), the feasible set is essentially assumed

to be deterministic after the real problem is modeled. That is, an optimal solution is given regardless of whether it can be performed in practice. However, the given solution may be impossible to perform if the realization of uncertain parameter is unfavorable. Thus DCM theory never assumes that the feasible set is deterministic. In fact, DCM is constructed in an uncertain environment. This special feature of DCM is very different from the other existing types of random programming.

In this section, we introduce the concept of chance function and discuss the DCM models. We also give some crisp equivalent models. Finally, some numerical examples are exhibited.

2.5.1 General Model for Random DCM

Let's consider the following typical DCM model,

$$\begin{cases} \max Pr\{f(\boldsymbol{x}, \boldsymbol{\xi}) \geq f\} \\ \text{s.t.} \begin{cases} g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, j = 1, 2, \ldots, p \\ \boldsymbol{x} \geq 0 \end{cases} \end{cases} \tag{2.72}$$

Since a complex decision system usually undertakes multiple events, there undoubtedly exist multiple potential objectives (some of them are chance functions) in a decision process. A typical formulation of dependent-chance multi-objective programming model (DCM) is represented as maximizing multiple chance functions subject to an uncertain environment,

$$\begin{cases} \max \begin{bmatrix} Pr\{h_1(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0\} \\ Pr\{h_2(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0\} \\ \cdots \\ Pr\{h_m(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0\} \end{bmatrix} \\ \text{s.t.} \quad g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, j = 1, 2, \ldots, p \end{cases} \tag{2.73}$$

where $h_i(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0$ are represent events ε_i for $i = 1, 2, \ldots, m$, respectively.

It follows from the principle of uncertainty that we can construct a relationship between decision vectors and chance function, thus calculating the chance functions by stochastic simulations or traditional methods. Then we can solve DCM by utility theory if complete information of the preference function is given by the decision-maker or search for all of the efficient solutions if no information is available. In practice, the decision maker can provide only partial information. In this case, we have to employ the interactive methods.

Sometimes, the objective function may minimize the deviations, positive, negative, or both, with a certain priority structure set by the decision maker. Then we may formulate the stochastic decision system as the following model,

$$\begin{cases} \min \sum_{j=1}^{l} P_j \sum_{i=1}^{m} (u_{ij} d_i^+ + v_{ij} d_i^-) \\ \text{s.t.} \begin{cases} Pr\{h_i(\boldsymbol{x}, \boldsymbol{\xi}) \le 0\} + d_i^- - d_i^+ = b_i, & i = 1, 2, \ldots, m \\ g_j(\boldsymbol{x}, \boldsymbol{\xi}) \le 0, & j = 1, 2, \ldots, p \\ d_i^-, d_i^+ \ge 0, & i = 1, 2, \ldots, m \end{cases} \end{cases} \tag{2.74}$$

where P_j is the preemptive priority factor which express the relative importance of various goals, $P_j >> P_{j+1}$, for all j, u_{ij} is the weighting factor corresponding to positive deviation for goal i with priority j assigned, v_{ij} is the weighting factor corresponding to negative deviation for goal i with priority j assigned, d_i^+ is the positive deviation from the target value according to goal i, d_i^- is the negative deviation from the target of goal i, b_i is the target value according to goal i, l is the number of priorities, and m is the number of goal constraints.

2.5.2 Linear Random DCM and the Fuzzy Programming Method

As we know, traditional solution methods need conversion of the chance constraints to their respective deterministic equivalents. However, this process is usually hard to perform and only successful for some special cases. Next, let's consider a special case in which the model is linear and parameters are subject to normal distribution (see the following linear model)

$$\begin{cases} \max \left[\bar{c}_1^T \boldsymbol{x}, \bar{c}_2^T \boldsymbol{x}, \ldots, \bar{c}_m^T \boldsymbol{x}\right] \\ \text{s.t.} \begin{cases} \bar{e}_r^T \boldsymbol{x} \le \bar{b}_r, r = 1, 2, \ldots, p \\ \boldsymbol{x} \ge 0 \end{cases} \end{cases} \tag{2.75}$$

where $\bar{c}_i = (c_{i1}(\omega), c_{i2}(\omega), \ldots, c_{in}(\omega))$, $\bar{e}_r = (e_{r1}(\omega), e_{r2}(\omega), \ldots, e_{rn}(\omega))$ and $\bar{b}_r$ are independently random vectors on the probability space $(\Omega, \mathscr{A}, Pr)$, f_i is the predetermined objective value. By the definition of chance function, we can get the following model,

$$\begin{cases} \max \left[Pr\{\bar{c}_1^T \boldsymbol{x} \ge f_1\}, Pr\{\bar{c}_2^T \boldsymbol{x} \ge f_2\}, \ldots, Pr\{\bar{c}_m^T \boldsymbol{x} \ge f_m\}\right] \\ \text{s.t.} \begin{cases} Pr\{\bar{e}_r^T \boldsymbol{x} \le \bar{b}_r\} \ge \beta_r, \ r = 1, 2, \ldots, p \\ \boldsymbol{x} \ge 0 \end{cases} \end{cases} \tag{2.76}$$

where f_i is predetermined objective value and β_r is the predetermined level value.

2.5.2.1 Crisp Equivalent Model

By the equivalent transformation, we get the following theorem.

Theorem 2.9. *Assume that random vector* $\bar{c}_i = (\bar{c}_{i1}, \bar{c}_{i2}, \ldots, \bar{c}_{in})^T$ *is normally distributed with mean vector* $\mu_i^c = (\mu_{i1}^c, \mu_{i2}^c, \ldots, \mu_{in}^c)^T$ *and positive definite*

covariance matrix V_i^c, written as $\bar{c}_i \sim \mathcal{N}(\mu_i^c, V_i^c)(i = 1, 2, \ldots, m)$ and random vector $\bar{e}_r \sim \mathcal{N}(\mu_r^e, V_i^e)$, $\bar{b}_r \sim \mathcal{N}(\mu_r^b, (\sigma_r^b)^2)$ $(r = 1, 2, \ldots, p)$. Assume that for any $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$ and $r = 1, 2, \ldots, p$, $\bar{c}_{ij}$, $\bar{e}_{rj}$ and $\bar{b}_r$ are independent with each other, respectively. Then problem (2.76) is equivalent to

$$\begin{cases} \max \left[1 - \Phi \left(\dfrac{f_i - \mu_i^{cT} \boldsymbol{x}}{\sqrt{\boldsymbol{x}^T V_i^c \boldsymbol{x}}} \right), i = 1, 2, \ldots, m \right] \\ s.t. \begin{cases} \Phi^{-1}(\beta_r) \sqrt{\boldsymbol{x}^T V_r^e \boldsymbol{x} + (\sigma_r^b)^2} + \mu_r^{eT} \boldsymbol{x} - \mu_r^b \leq 0 \\ \boldsymbol{x} \geq 0, r = 1, 2, \ldots, p \end{cases} \end{cases}$$

Proof. Since the random vector $\bar{c}_i$ is normally distributed with mean vector μ_i^c and positive definite covariance matrix V_i^c, then we have that

$$\bar{c}_i^T \boldsymbol{x} = \sum_{j=1}^{n} x_j \bar{c}_{ij}(\omega) \sim \mathcal{N} \left(\sum_{j=1}^{n} x_j \mu_{ij}^c, \boldsymbol{x}^T V_i^c \boldsymbol{x} \right) = \mathcal{N}(\mu_i^{cT} \boldsymbol{x}, \boldsymbol{x}^T V_i^c \boldsymbol{x}).$$

It follows that

$$\frac{\bar{c}_i^T \boldsymbol{x} - \mu_i^{cT} \boldsymbol{x}}{\sqrt{\boldsymbol{x}^T V_i^c \boldsymbol{x}}}$$

must be standardized normally distributed. Then

$$\begin{aligned} Pr\{c_i^T \boldsymbol{x} \geq f_i\} &= Pr \left\{ \frac{\bar{c}_i^T \boldsymbol{x} - \mu_i^{cT} \boldsymbol{x}}{\sqrt{\boldsymbol{x}^T V_i^c \boldsymbol{x}}} \geq \frac{f_i - \mu_i^{cT} \boldsymbol{x}}{\sqrt{\boldsymbol{x}^T V_i^c \boldsymbol{x}}} \right\} \\ &= 1 - Pr \left\{ \frac{\bar{c}_i^T \boldsymbol{x} - \mu_i^{cT} \boldsymbol{x}}{\sqrt{\boldsymbol{x}^T V_i^c \boldsymbol{x}}} \leq \frac{f_i - \mu_i^{cT} \boldsymbol{x}}{\sqrt{\boldsymbol{x}^T V_i^c \boldsymbol{x}}} \right\} \\ &= 1 - \Phi \left(\frac{f_i - \mu_i^{cT} \boldsymbol{x}}{\sqrt{\boldsymbol{x}^T V_i^c \boldsymbol{x}}} \right) \end{aligned}$$

where Φ is the standardized normally distributed function. Similarly, we know that

$$\bar{e}_r^T \boldsymbol{x} \sim \mathcal{N}(\mu_r^{eT} \boldsymbol{x}, \boldsymbol{x}^T V_r^e \boldsymbol{x}).$$

Since $\bar{b}_r \sim \mathcal{N}(\mu_r^b, (\sigma_r^b)^2)$, it follows from, $\bar{e}_{rj}(\omega)$ and $\bar{b}_{rj}(\omega)$ are independently random variables for any r, j, that

$$E[\bar{e}_r^T \boldsymbol{x} - \bar{b}_r] = E[\bar{e}_r^T \boldsymbol{x}] - E[\bar{b}_r] = \mu_r^{eT} \boldsymbol{x} - \mu_r^b,$$
$$V[\bar{e}_r^T \boldsymbol{x} - \bar{b}_r] = V[\bar{e}_r^T \boldsymbol{x}] + V[\bar{b}_r] = \boldsymbol{x}^T V_r^e \boldsymbol{x} + (\sigma_r^b)^2,$$

then $\bar{e}_r^T \boldsymbol{x} - \bar{b}_r$ is also normally distributed with

$$\bar{e}_r^T \boldsymbol{x} - \bar{b}_r \sim \mathcal{N}(\mu_r^{eT} \boldsymbol{x} - \mu_r^b, \boldsymbol{x}^T V_r^e \boldsymbol{x} + (\sigma_r^b)^2).$$

For any $r (r = 1, 2, \ldots, p)$, we have

$$\begin{aligned} Pr\{\bar{e}_r^T \boldsymbol{x} \geq \bar{b}_r\} \geq \beta_r \Leftrightarrow \beta_r \leq Pr \Bigg\{ & \frac{\bar{e}_r^T \boldsymbol{x} - \bar{b}_r - (\mu_r^{eT} \boldsymbol{x} - \mu_r^b)}{\sqrt{\boldsymbol{x}^T V_r^e \boldsymbol{x} + (\sigma_r^b)^2}} \\ & \leq -\frac{\mu_r^{eT} \boldsymbol{x} - \mu_r^b}{\sqrt{\boldsymbol{x}^T V_r^e \boldsymbol{x} + (\sigma_r^b)^2}} \Bigg\} \\ \Leftrightarrow \beta_r \leq \Phi & \left(-\frac{\mu_r^{eT} \boldsymbol{x} - \mu_r^b}{\sqrt{\boldsymbol{x}^T V_r^e \boldsymbol{x} + (\sigma_r^b)^2}} \right) \\ \Leftrightarrow \Phi^{-1}(\beta_r) & \sqrt{\boldsymbol{x}^T V_r^e \boldsymbol{x} + (\sigma_r^b)^2} + \mu_r^{eT} \boldsymbol{x} - \mu_r^b \leq 0 \end{aligned}$$

where Φ is the standardized normal distribution. Then we have that problem (2.76) is equivalent to

$$\begin{cases} \max \left[1 - \Phi \left(\dfrac{f_i - \mu_i^{cT} \boldsymbol{x}}{\sqrt{\boldsymbol{x}^T V_i^c \boldsymbol{x}}} \right), i = 1, 2, \ldots, m \right] \\ \text{s.t.} \begin{cases} \Phi^{-1}(\beta_r) \sqrt{\boldsymbol{x}^T V_r^e \boldsymbol{x} + (\sigma_r^b)^2} + \mu_r^{eT} \boldsymbol{x} - \mu_r^b \leq 0 \\ \boldsymbol{x} \geq 0, r = 1, 2, \ldots, p \end{cases} \end{cases} \tag{2.77}$$

This completes the proof. □

2.5.2.2 Fuzzy Programming Method

The fuzzy programming method for multi-objective programming problems was proposed by Zimmermann [364] and has been advanced by Sakawa and colleagues [271]. An interactive fuzzy satisficing method for multiobjective linear programming problems and the interactive fuzzy decision making for multiobjective nonlinear programming using augmented minimax problems have been also introduced [212, 270, 282]. The fuzzy programming method in which fuzziness in the decision making process is represented by using the fuzzy concept has also been studied extensively and many results have been published [265, 317]. This method can be applied to not only the linear multi-objective but also the nonlinear multi-objective programming.

Take the problem (2.77) as an example. Let $H_i(\boldsymbol{x}) = 1 - \Phi\left(\frac{f_i - \mu_i^{cT}\boldsymbol{x}}{\sqrt{\boldsymbol{x}^T V_i^c \boldsymbol{x}}}\right)$, and $X = \{\boldsymbol{x} | \Phi^{-1}(\beta_r)\sqrt{\boldsymbol{x}^T V_r^e \boldsymbol{x} + (\sigma_r^b)^2} + \mu_r^{eT}\boldsymbol{x} - \mu_r^b \leq 0, r = 1, 2, \ldots, p; \boldsymbol{x} \geq 0\}$, then problem (2.77) is equivalent to

$$\begin{cases} \max[H_1(\boldsymbol{x}), H_2(\boldsymbol{x}), \ldots, H_m(\boldsymbol{x})] \\ \text{s.t. } \boldsymbol{x} \in X \end{cases} \tag{2.78}$$

Considering the imprecise nature of the decision maker's judgements for each objective function of problem (2.78), a new fuzzy objective goal such as "make $H_i(\boldsymbol{x})$ approximately larger than a certain value" is introduced. Then problem (2.78) is converted into

$$\begin{cases} \max[\rho_1(H_1(\boldsymbol{x})), \rho_2(H_2(\boldsymbol{x})), \ldots, \rho_m(H_m(\boldsymbol{x}))] \\ \text{s.t. } \boldsymbol{x} \in X \end{cases} \tag{2.79}$$

where the random goal is characterized by the following linear membership function,

$$\rho_i(H_i(\boldsymbol{x})) = \begin{cases} 1, & H_i(\boldsymbol{x}) > H_i^1 \\ \dfrac{H_i(\boldsymbol{x}) - H_i^0}{H_i^1 - H_i^0}, & H_i^0 \leq H_i(\boldsymbol{x}) \leq H_i^1 \\ 0, & H_i(\boldsymbol{x}) < H_i^0 \end{cases}$$

where H_i^1 and H_i^0 respectively denote the maximal and minimal values of the objective functions $H_i(\boldsymbol{x})$ as follows,

$$H_i^0 = \min_{\boldsymbol{x} \in X} H_i(\boldsymbol{x}), \ H_i^1 = \max_{\boldsymbol{x} \in X} H_i(\boldsymbol{x}), \ i = 1, 2, \ldots, m.$$

For each objective function $\rho_i(H_i(\boldsymbol{x}))$, assume that the DM can specify the so-called reference probability function value $\bar{\rho_i}$ which reflects the probability function value of $\rho_i(H_i(\boldsymbol{x}))$. The corresponding Pareto optimal solution, which is nearest to the requirements in the minimax sense or better than that if the reference probability function value is attainable, is obtained by solving the following minimax problem,

$$\begin{cases} \min \max\limits_{i=1,2,\ldots,m} \{\bar{\rho_i} - \rho_i(H_i(\boldsymbol{x}))\} \\ \text{s.t. } \boldsymbol{x} \in X \end{cases} \tag{2.80}$$

By introducing auxiliary variable λ, problem (2.80) is equivalent to

$$\begin{cases} \min \lambda \\ \text{s.t.} \begin{cases} \bar{\rho_i} - \rho_i(H_i(\boldsymbol{x})) \leq \lambda, \ i = 1, 2, \ldots, m \\ 0 \leq \lambda \leq 1 \\ \boldsymbol{x} \in X \end{cases} \end{cases} \tag{2.81}$$

or equivalently

$$\begin{cases} \min \lambda \\ \text{s.t.} \begin{cases} \lambda_i^{cT} \boldsymbol{x} \geq H_i^0 + (\bar{\rho}_i - \lambda)(H_i^1 - H_i^0), i = 1, 2, \ldots, m \\ 0 \leq \lambda \leq 1 \\ \boldsymbol{x} \in X \end{cases} \end{cases} \tag{2.82}$$

Obviously, it follows that (2.82) is a convex programming problem of which the global optimal solution is easily obtained.

The relationship between the optimal solution of problem (2.81) and the Pareto optimal solution of problem (2.78) can be characterized by the following theorem.

Theorem 2.10. *1. If $\boldsymbol{x}^* \in X$ is a unique optimal solution to problem (2.81) for some $\bar{\rho}_i, i = 1, 2, \ldots, m$, then $\boldsymbol{x}^*$ is a Pareto optimal solution to problem (2.78).*

2. If $\boldsymbol{x}^*$ is a Pareto optimal solution to problem (2.78) with $0 < \rho_i(H_i(\boldsymbol{x}^*)) < 1$ holding for all i, then there exist ε_i such that $\boldsymbol{x}^*$ is an optimal solution to problem (2.81).

Proof. 1. For some $\bar{\rho_k}$, if a unique optimal solution $\boldsymbol{x}^*$ to problem (2.81) is not the Pareto optimal solution of problem (2.78), then there exists $\boldsymbol{x} \in X$ such that $H_k(\boldsymbol{x}) > H_k(\boldsymbol{x}^*)$, where $H_k(\boldsymbol{x}^*) = \max\{H_1(\boldsymbol{x}^*), H_2(\boldsymbol{x}^*), \ldots, H_m(\boldsymbol{x}^*)\}$. Then,

$$\begin{aligned} &\frac{H_k(x) - H_k^0}{H_k^1 - H_k^0} > \frac{H_k(\boldsymbol{x}^*) - H_k^0}{H_k^1 - H_k^0} \\ &\Leftrightarrow \rho(H_k(x)) > \rho(H_k(\boldsymbol{x}^*)) \\ &\Leftrightarrow \bar{\rho_k} - \rho(H_k(\boldsymbol{x})) < \bar{\rho_k} - \rho(H_k(\boldsymbol{x}^*)) \end{aligned}$$

This means that there exits a λ, satisfying $\lambda < \lambda^*$. It follows that $\boldsymbol{x}^*$ is not the optimal solution of problem (2.81), which contradicts the assumption that $\boldsymbol{x}^*$ is a unique optimal solution to problem (2.81).

2. If $\boldsymbol{x}^*$ is not an optimal solution to problem (2.81), then there exists $\boldsymbol{x}^{'} \in X$ such that $\bar{\rho}_i - \rho(H_i(\boldsymbol{x}^{'})) < \bar{\rho}_i - \rho(H_i(\boldsymbol{x}^*)), i = 1, 2, \ldots, m$, because of $0 < \rho_i(H_i(\boldsymbol{x}^*)) < 1$, then

$$\begin{aligned} &\rho(H_i(\boldsymbol{x}^{'})) > \rho(H_i(\boldsymbol{x}^*)) \\ &\quad \Leftrightarrow \frac{H_i(\boldsymbol{x}^{'}) - H_i^0}{H_i^1 - H_i^0} > \frac{H_i(\boldsymbol{x}^*) - H_i^0}{H_i^1 - H_i^0} \\ &\quad \Leftrightarrow H_i(\boldsymbol{x}^{'}) > H_i(\boldsymbol{x}^*) \end{aligned}$$

This means that there exists $\boldsymbol{x}^{'}$ such that $H_i(\boldsymbol{x}^{'}) > H_i(\boldsymbol{x}^*)$, then $\boldsymbol{x}^*$ is not the Pareto optimal solution to problem (2.78), which contradicts the assumption that $\boldsymbol{x}^*$ is a Pareto optimal solution to problem (2.78). This completes the proof. □

If $\boldsymbol{x}^*$, an optimal solution to problem (2.81), is not unique, then the Pareto optimality test for $\boldsymbol{x}^*$ can be performed by solving the following problem

$$\begin{cases} \max \sum_{i=1}^{m} \varepsilon_i \\ \text{s.t.} \begin{cases} \rho_i(H_i(\boldsymbol{x}^*)) + \varepsilon_i = \bar{\rho}_i, i = 1, 2, \ldots, m \\ \boldsymbol{x} \in X, \varepsilon_i \geq 0 \end{cases} \end{cases}$$

From Theorem 2.10, we know that the optimal solution $\boldsymbol{x}^*$ of problem (2.81) is a Pareto optimal solution to problem (2.78). Then the interactive random satisfying method, which is similar to the interactive fuzzy satisfying method, can be constructed to obtain a satisfactory solution of problem (2.78),

Step 1. The DM is required to present reference probability values $\bar{\rho}_i$, $i = 1, 2, \ldots, m$.

Step 2. The optimal solution of problem (2.80), which is also a Pareto optimal solution of problem (2.78), is considered as a satisfactory solution to problem (2.78).

Step 3. If the obtained $\rho_i(H_i(\boldsymbol{x}^*))$ are satisfying, the process stops and $\boldsymbol{x}^*$ is selected as satisfactory solution to problem (2.78); Or else, the DM should update his or her reference random probability values $\bar{\rho}_i$ and return to Step 2.

2.5.3 Nonlinear Random DCM and Random Simulation-Based PSA

Similarly to the EVM and CCM, for the problems which are easily converted into crisp ones, we deal with them by the chance measure, and the random simulation is used to deal with those which cannot be converted into crisp ones. Next, let's introduce the process of the random simulation dealing with the dependent chance models. Consider the nonlinear multi-objective programming problem as follows,

$$\begin{cases} \max[Pr\{f_1(\boldsymbol{x}, \boldsymbol{\xi}) \geq \bar{f_1}\}, \ldots, Pr\{f_m(\boldsymbol{x}, \boldsymbol{\xi}) \geq \bar{f_m}\}] \\ \text{s.t.} \begin{cases} Pr\{g_r(\boldsymbol{x}, \boldsymbol{\xi}) \leq b_r\} \geq \alpha_i, r = 1, 2, \ldots, p \\ \boldsymbol{x} \geq 0, 0 \leq \alpha_r \leq 1 \end{cases} \end{cases} \tag{2.83}$$

where $f_i(\boldsymbol{x}, \boldsymbol{\xi})$ and $g_r(\boldsymbol{x}, \boldsymbol{\xi})$ are nonlinear functions with respect to $\boldsymbol{\xi}$, $i = 1, 2, \ldots, m$, $r = 1, 2, \ldots, p$, $\bar{f_i}$ are predetermined confidence levels, and $\boldsymbol{\xi}$ is a random vector.

2.5.3.1 Random Simulation for DCM

Let $\boldsymbol{\xi}$ be an n-dimensional random vector defined on the probability space $(\Omega, \mathscr{A}, Pr)$, and $f : \mathbf{R}^n \to \mathbf{R}$ a measurable function. In order to obtain the probability for given $\boldsymbol{x}$,

$$L = Pr\{f(\boldsymbol{x}, \boldsymbol{\xi}) \geq \bar{f}\},$$

we generate ω_k from Ω according to the probability measure Pr, and write $\boldsymbol{\xi}_k = \boldsymbol{\xi}(\omega_k)$ for $k = 1, 2, \ldots, N$, where $\omega_k = (\omega_k^1, \ldots, \omega_k^n)$ is an n-dimensional vector

and ω_k^j is generated according to the random variable ξ_k. Let $N^{'}$ denote the number of occasions on which $f(\boldsymbol{x}, \boldsymbol{\xi}_k) \geq \bar{f}$ for $k = 1, 2, \ldots, N$ (i.e., the number of random vectors satisfying the system of inequalities). Let us define

$$h(\boldsymbol{x}, \boldsymbol{\xi}_k) = \begin{cases} 1, \text{ if } f(\boldsymbol{x}, \boldsymbol{\xi}_k) \geq \bar{f} \\ 0, \text{ otherwise} \end{cases}$$

Then we have $E[h(\boldsymbol{x}, \boldsymbol{\xi}_k)] = L$ for all k, and $N^{'} = \sum_{k=1}^{N} h(\boldsymbol{x}, \boldsymbol{\xi}_k)$. It follows from the strong law of large numbers that

$$\frac{N^{'}}{N} = \frac{\sum\limits_{k=1}^{N} h(\boldsymbol{x}, \boldsymbol{\xi}_k)}{N}$$

converges a.s. to L. Thus the probability L can be estimated by $N^{'}/N$ provided that N is sufficiently large. Then the procedure simulating the probability of the event $f(\boldsymbol{x}, \boldsymbol{\xi}) \geq \bar{f}$ can be summarized as follows:

Procedure `Random simulation for DCM`

Input: The decision vector $\boldsymbol{x}$
Output: The probability $Pr\{f(\boldsymbol{x}, \boldsymbol{\xi}) \geq \bar{f}\}$
Step 1. Set $N^{'} = 0$;
Step 2. Generate ω from Ω according to the probability measure Pr;
Step 3. If $f(\boldsymbol{x}, \boldsymbol{\xi}_k) \geq \bar{f}$, then $N^{'} + +$;
Step 4. Repeat the second and third steps N times;
Step 5. Return $L = N^{'}/N$.

Example 2.8. Let $\xi_1 \sim exp(2)$ be an exponentially distributed variable, $\xi_2 \sim \mathcal{N}(4, 1)$ a normally distributed variable, and $\xi_3 \sim \mathcal{U}(0, 2)$ a uniformly distributed variable. A run of random simulation with 10000 cycles shows that

$$Pr\left\{\sqrt{\xi_1^2 + \xi_2^2 + \xi_3^2} \leq 8\right\} = 0.9642.$$

2.5.3.2 The Parallel SA Algorithm

The parallel SA algorithm, which is usually shorten as PSA, is a more efficient tool to deal with complex problems. Parallel processing appears to be the only viable way to substantially speed up the method and thus expand its applicability. For fast tailored SAs, parallel implementations may also reduce the loss of robustness. In an effort to increase convergence speed, parallel SA optimization algorithms have been implemented with mixed results in various scientific fields [50]. Czech and Czarnas [69] presented a parallel simulated annealing for the vehicle routing problem. Many scholars [16, 16, 41, 69, 121, 235, 268, 336] introduced the PSA with the feature that multiple processing elements follow a single search path (Markov chain) (to be

referred to as SMC PSA), but Aarts and Korst [1] described the idea of the multiple Markov chains PSA (to be referred to as MMC PSA), namely the divison algorithm and Lee and Lee [190] proposed the MMC PSA and applied it to graph partition and Li, Cha and Lu [195] applied it for 3D engineering layout design.

Although optimization performance can be greatly improved through parallelization of SA [78, 163], there has few applications of this techniques in multiobjective programming problems. Since the "annealing community" has so far not achieved a common agreement with regards to a general approach for the serial SA, the best parallel scheme is still the object of current research. Bevilacqua [23] claimed that Many scholars has done many attempts of classifying methods: that is, if parallelism is used to generated a move or to run independent moves on different processors, if these methods depend on the problem or not, if they are synchronous or not, and others.

In any case, a key issue in developing a PSA is to ensure that it maintains the same convergence property as the sequential version, for which a formal proof to converge to a global minimum exists. For this purpose, it is easy to show that it is not necessary for a parallel implementation to generate exactly the same sequence of solutions as the sequential version. Frequently, it is enough to keep the same ratio between the number of accepted moves and the number of attempted moves (acceptance rate), averaged over a given temperature, as the correspondent sequential version.

Next, let's discuss the detailed algorithm about PSA. This section mainly refers to [23, 195, 217, 352], and readers can also consult the related literatures. Let's firstly introduce the parallel Markov chain in [217]. Let X be a finite set and $U : X \to R^+$ be a non-constant function to be minimized. We denote by $X_{\min}$ the set of global minima of U. Suppose that we have $p > 1$ processors. The optimization algorithm which is the center of our interest is described as follows. Choose any starting point $x_0 \in X$ and let each processor individually run a Metropolis Markov chain of fixed length $L > 1$ at inverse temperature $\beta(0)$ starting in x_0. After L transitions the simulation is stopped and only one state x_1 is selected from the p states according to a selection strategy. Again each processor individually simulates a Metropolis Markov chain of length L at an updated inverse temperature $\beta(1)$ starting in x_1. Again at the end of the simulation a state x2 is selected from the p states and the next simulation starts from x_2, etc. Fig. 2.7. This, algorithm is closely related to the so-called parallel chain algorithm. However, the main difference is that the number of parallel chains and the length of the Markov chains L is kept fixed in our model. In the parallel chain algorithm the length L is usually increased and the number of parallel Markov chains is decreased during the run of the algorithm so that the parallel chain

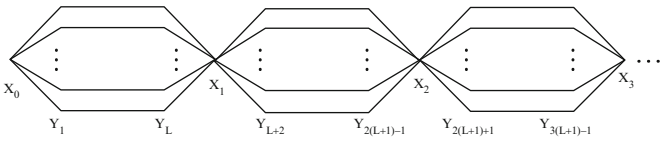

Fig. 2.7 Illustration of the algorithm's working method

algorithm asymptotically behaves like the so-called one-chain algorithm. Secondly, let's introduce the process of PSA. PSA starts with a high temperature. After generating an initial solution, it attempts to move randomly from the current solution to one of its neighborhood solutions. The changes in the objective function values Δf are computed(we usually consider a weighted sum of all objective functions in a multiobjective programming problem). If the new solution results in a better objective value, it is accepted. However, if the new solution yields a worse value, it can still be accepted according to the acceptance probability function, $P(T_j)$. The PSA algorithm repeats this Metropolis algorithm L times at each temperature to reach thermal equilibrium, where L is a control parameter, usually called the Markov Chain Length. After some temperature annealing, solutions migrate from the neighboring processors at each interval with migration probability, P_m. Parameter T_j is gradually decreased by a cooling function as parallel SA proceeds until the

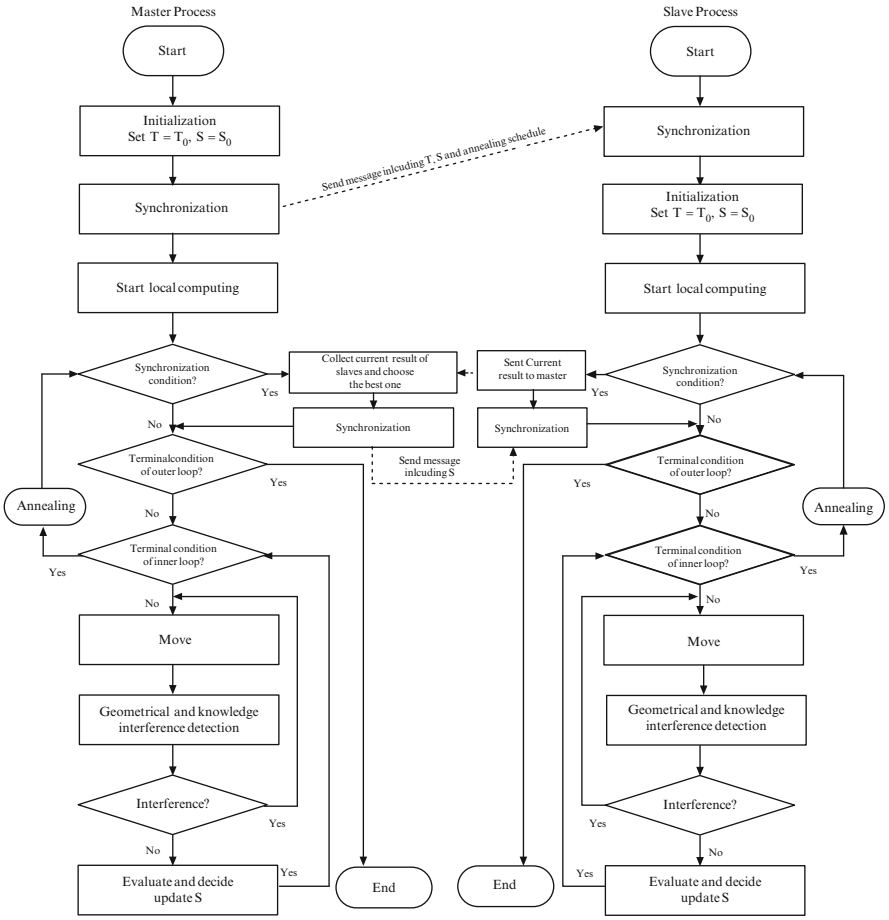

Fig. 2.8 Flow chart of the PSA

terminating condition is met. Readers can refer to Fig. 2.8 to know the work flow of PSA. The parallel SA algorithms proposed can be summarized as follows:

Procedure `The PSA algorithm`

Input: The initial parameters: T_0, T_f, P_m, M_i
Output: The Pareto-solution $\boldsymbol{x}^*$
Step 1. Initialize PSA parameters: starting temperature T_0, final temperature T_f, Markov chain length L, migration rate P_m, and migration interval M_i;
Step 2. Randomly generate an initial feasible $\boldsymbol{x}$ and assign to;
Step 3. Repeat the following L times: (a) Generate a neighborhood solution $\boldsymbol{x}_j^{new}$ through a random number generator; (b) Compute $\Delta f = f(\boldsymbol{x}_j^{new}) - f(\boldsymbol{x}_j^{old})$; (c) If $\Delta f \leq 0$(when we minimize the objective function), set $\boldsymbol{x}_j^{old} = \boldsymbol{x}_j^{new}$; (d) If $\Delta f > 0$, generate a random number X in (0,1), and compute $P(T_j)$. If $P(T_j) > X$, set $\boldsymbol{x}_j^{old} = \boldsymbol{x}_j^{new}$;
Step 4. the migration interval (M_i) is reached, then solutions migrate from the neighboring processors with migration rate P_m;
Step 5. If the terminating condition is met, stop; otherwise, let T_j decrease by the cooling schedule and go to Step 3.

2.5.4 Numerical Examples

Example 2.9. Let's consider the following problem,

$$\begin{cases} \max f_1(\boldsymbol{x}, \boldsymbol{\xi}) = \xi_1 x_1 + \xi_2 x_2 + \xi_3 x_3 + \xi_4 x_4 + \xi_5 x_5 \\ \max f_2(\boldsymbol{x}, \boldsymbol{\xi}) = c_1 \xi_6 x_1 + c_2 \xi_7 x_2 + c_3 \xi_8 x_3 + c_4 \xi_9 x_4 + c_5 \xi_{10} x_5 \\ \text{s.t.} \begin{cases} x_1 + x_2 + x_3 + x_4 + x_5 \leq 10 \\ x_1 + x_2 + x_3 - x_4 - x_5 \geq 0 \\ x_1 + 4x_2 + 2x_3 - x_4 + 5x_5 \leq 20 \\ x_i \geq 0, i = 1, 2, \ldots, 5 \end{cases} \end{cases} \tag{2.84}$$

where $c = (c_1, c_2, c_3, c_4, c_5) = (1.2, -0.5, 1.3, -0.1, 2)$, and $\xi_i (i = 1, 2, \ldots, 9)$ are independently random variables as follows,

$$\begin{aligned} &\xi_1 \sim \mathcal{N}(2, 1), \quad \xi_2 \sim \mathcal{N}(3, 0.5), \quad \xi_3 \sim \mathcal{N}(1, 0.2), \xi_4 \sim \mathcal{N}(5, 1), \\ &\xi_5 \sim \mathcal{N}(2, 0.1), \xi_6 \sim \mathcal{N}(7, 0.5), \quad \xi_7 \sim \mathcal{N}(3, 0.2), \xi_8 \sim \mathcal{N}(4, 1), \\ &\xi_9 \sim \mathcal{N}(5, 1), \quad \xi_{10} \sim \mathcal{N}(1, 0.1). \end{aligned}$$

If DM predetermines two level values $\bar{f_1}$ and $\bar{f_2}$, and aim at obtaining the maximum probability based on the predetermined level value $\bar{f_1}$ and $\bar{f_2}$. Set $\bar{f_1} = 14$ and $\bar{f_2} = 22$, then we get the following dependent-chance model,

$$\begin{cases} \max g_1(\boldsymbol{x}, \boldsymbol{\xi}) = Pr\{\xi_1 x_1 + \xi_2 x_2 + \xi_3 x_3 + \xi_4 x_4 + \xi_5 x_5 \geq 14\} \\ \max g_2(\boldsymbol{x}, \boldsymbol{\xi}) = Pr\{c_1\xi_6 x_1 + c_2\xi_7 x_2 + c_3\xi_8 x_3 + c_4\xi_9 x_4 + c_5\xi_{10} x_5 \geq 22\} \\ \text{s.t.} \begin{cases} x_1 + x_2 + x_3 + x_4 + x_5 \leq 10 \\ x_1 + x_2 + x_3 - x_4 - x_5 \geq 0 \\ x_1 + 4x_2 + 2x_3 - x_4 + 5x_5 \leq 20 \\ x_i \geq 0, i = 1, 2, \ldots, 5 \end{cases} \end{cases} \tag{2.85}$$

It follows from Theorem 2.9 that, (2.85) is equivalent to

$$\begin{cases} \max H_1(\boldsymbol{x}) = 1 - \Phi\left(\dfrac{14 - (2x_1 + 3x_2 + x_3 + 5x_4 + 2x_5)}{\sqrt{x_1^2 + 0.5x_2^2 + 0.2x_3^2 + x_4^2 + 0.1x_5^2}}\right) \\ \max H_2(\boldsymbol{x}) = 1 - \Phi\left(\dfrac{22 - (8.4x_1 - 1.5x_2 + 5.2x_3 - 0.5x_4 + 2x_5)}{\sqrt{0.5x_1^2 + 0.2x_2^2 + x_3^2 + x_4^2 + 0.1x_5^2}}\right) \\ \text{s.t.} \begin{cases} x_1 + x_2 + x_3 + x_4 + x_5 \leq 10 \\ x_1 + x_2 + x_3 - x_4 - x_5 \geq 0 \\ x_1 + 4x_2 + 2x_3 - x_4 + 5x_5 \leq 20 \\ x_i \geq 0, i = 1, 2, \ldots, 5 \end{cases} \end{cases} \tag{2.86}$$

Next, we will use the fuzzy programming method to solve the above problem. Then we have that

$$H_1^1 = 43944.64,\ H_1^0 = 30220.00,\ H_2^1 = 225444.00,\ H_2^0 = 158178.70$$

By the fuzzy programming method, we have the satisfactory solutions to problem (2.86) as shown in Table 2.5.

Example 2.10. Consider the following multi-objective programming problem with random parameters,

Table 2.5 Interactive process of expected value model

$\bar{\mu_1}$	$\bar{\mu_2}$	H_1	H_2	$\mu_1(H_1)$	$\mu_2(H_2)$	x_1	x_2	x_3	x_4	x_5	λ
1	1	41649.56	214197.2	0.8328	0.8328	216.08	39.62	54.30	20.00	20.00	0.1672
1	0.95	42033.36	212717.6	0.8607	0.8108	215.56	42.20	52.24	20.00	20.00	0.1392
0.95	1	41265.50	215675.4	0.8048	0.8548	216.59	37.04	56.37	20.00	20.00	0.1452
1	0.90	42418.70	211232.6	0.8888	0.7887	215.04	44.79	50.17	20.00	20.00	0.1112
1	1	41649.56	21419.72	0.8328	0.8328	216.08	39.62	54.30	20.00	20.00	0.1672
0.90	1	40881.7	217154.9	0.7768	0.8768	217.11	34.46	58.43	20.00	20.00	0.1232
1	0.85	42796.17	209722.0	0.9163	0.7663	214.57	47.36	48.00	20.00	20.00	0.0837
0.85	1	38756.36	206263.9	0.6220	0.7149	217.63	31.87	60.50	20.00	20.00	0.1012
1	0.8	42999.18	207354.5	0.9311	0.7311	215.28	49.66	43.04	20.00	20.00	0.0689
0.8	1	40112.30	220118.1	0.7208	0.9208	218.14	29.29	62.57	20.00	20.00	0.0792

$$\begin{cases} \max H_1(\boldsymbol{x}, \boldsymbol{\xi}) = Pr\{\sqrt{(x_1 - \xi_1)^2 + (x_2 - \xi_2)^2} \geq 1.75\} \\ \max H_2(\boldsymbol{x}, \boldsymbol{\xi}) = Pr\{\sqrt{(x_1 + \xi_1)^2 + (x_2 + \xi_2)^2} \geq 6.2\} \\ \text{s.t.} \begin{cases} x_1 + x_2 \leq 5 \\ x_1 \geq 0, x_2 \geq 0 \end{cases} \end{cases} \tag{2.87}$$

where $\xi_1 \sim \mathcal{N}(2, 0.5)$ ia normally distributed random variable and $\xi_2 \sim \mathscr{U}(1, 2)$ is also a normally distributed random variable.

Next we will use the random simulation-based PSA algorithm to solve the above problem. Set the initial temperature T_0, the last temperature be 1 and the cooling method be 1 decrement once. The neighborhood can be constructed as follows,

$$x_1^1 = x_1^0 + rh, \;\; x_2^1 = x_2^0 + rh,$$

where r is a random number in (0,1) and h is the step length (here $h = 2.0$). After the simulation with many cycles, we get the optimal solution under different weights as shown in Table 2.6. Figure 2.9 shows the cooling process when the weight is 0.5. The real line expresses the weight sum of two objective functions, and it shows that it gradually converges from T = 440. Figure 2.10 shows the changes of two objective values when the temperature decreases.

Table 2.6 The optimal solution by random simulation-based PSA

w_1	w_2	x_1	x_2	$\bar{f}_1$	$\bar{f}_2$	$\bar{f}$	T_0
0.1	0.9	0.7407	0.7124	0.0400	0.6500	0.5890	500
0.2	0.8	1.0209	1.8018	0.1800	0.5500	0.4760	500
0.3	0.7	0.5342	0.5561	0.0200	0.6000	0.4260	500
0.4	0.6	3.5738	1.4076	0.4400	0.8500	0.6860	500
0.5	0.5	0.2657	2.0287	0.7400	0.6000	0.6700	500

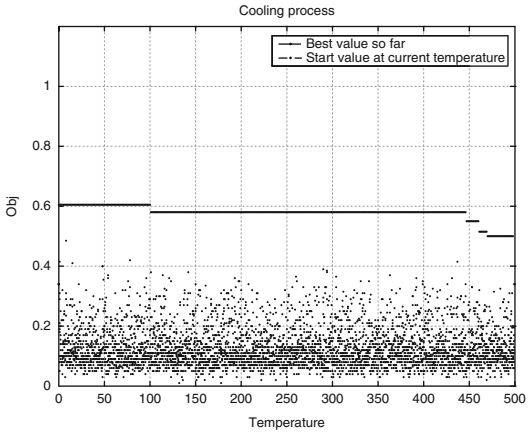

Fig. 2.9 The cooling process of random simulation-based PSA

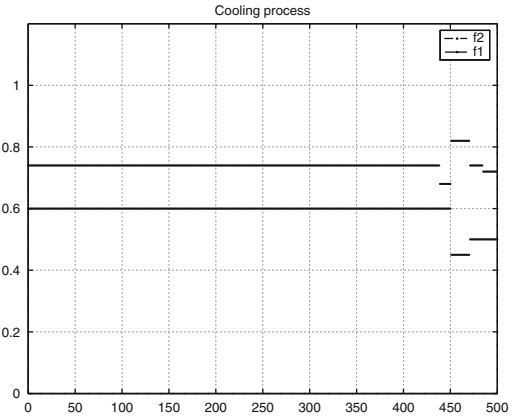

Fig. 2.10 Two objective values by random simulation-based PSA

2.6 Application to Chinese Fruit Beverage

2.6.1 Background Statement

As we know, the DC location problem involves how to select locations of distribution centers from a potential set and how to transport products from the manufactorers to each selected distribution center and also from the selected distribution center to each customer so that the total relevant cost is minimized and customer satisfaction is maximized. However, in practice, it is hard to describe the problem parameters as known due to the complexity of the social and economic environment as well as unpredictable factors such as weather. For instance, since the changing gasoline price often results in a scarcity of precise data, the transport cost from one facility to one DC (or from one DC to a customer) is usually a normal distributed variable with an expected value μ. For some seasonal products, their sales also vary along with the changing season, and usually follow a normal stochastic distribution. Therefore, there does exist a situation where shipping costs and the customer demand where may be random variables. In this situation, we can use random variables to deal with these uncertain parameters. Considering company managers' objectives, we minimize the total relevant costs (including transport costs and fixed costs) of logistic distribution centers, and maximize customer satisfaction in terms of an acceptable delivery time. Hence, we formulate a single-product, multi-objective logistic distribution centers location problem with consumers' random demands and transport costs, and the following assumptions are considered:

1. The number of customers and suppliers are known.
2. The number of potential DCs and their maximum capacities are known.
3. Customers are supplied products from a single DC.
4. The amount of demand on the products and the transport costs are regarded as random variables.

5. Since the manager of this company wants to reduce costs and find an optimal network strategy, inventory is considered as zero in this example.

For understanding this problem easily, we can consult Fig. 2.1.

2.6.2 Notations

The mathematic notation and formulations are as follows:

1. Indices

Suppose that there are I plants, J DCs and K customers. The task is to transfer the products from the plants to the DCs and from DCs to customers to satisfy customer demand.

$i \in \{1, 2, \ldots, I\}$ is the index for plants;

$j \in \{1, 2, \ldots, J\}$ is the index for DCs;

$k \in \{1, 2, \ldots, K\}$ is the index for customers.

2. Parameters

$\widetilde{b_k}$ which is a random variable denoting the demand of customer k;

$\widetilde{pd}_{ij}$ which is a random variable denoting the transport cost of a unit product transported from plant i to distribution center j;

$\widetilde{dc}_{jk}$ which is a random variable denoting the transport cost of a unit product transported from distribution center j to customer k;

t_{jk} denotes the transport time from j to k;

a_i is the capacity of plant i;

f_j denotes the fixed cost of opening distribution center j

m_j denotes the capacity of distribution center j;

3. Decision variables

x_{ij} denotes the quantity transported from plant i to distribution center j.

y_{jk} denotes the quantity transported from distribution center j to customer k.

2.6.3 Modelling and Analysis

For this problem, we need to select the distribution centers from the potential set J. We use the binary z_j to denote whether the distribution center j is selected or not:

$$z_j = \begin{cases} 1, \text{ if distribution center } j \text{ is open} \\ 0, \text{ otherwise} \end{cases}$$

And in this section, we assume that customers are supplied products from a single DC, and we use v_{jk} to denote whether DC j serves customer k.

$$v_{jk} = \begin{cases} 1, \text{ if distribution center } j \text{ servers customer } k \\ 0, \text{ otherwise} \end{cases}$$

3. Objective functions.

The objective F_1 is to minimize the total costs comprised of the fixed costs of DCs $\sum_{j\in J} f_j z_j$, the transport costs from plants to DCs $\sum_{i\in I}\sum_{j\in J} \widetilde{pd}_{ij} x_{ij}$ and from DCs to customers $\sum_{j\in J}\sum_{k\in K} \widetilde{dc}_{jk} y_{jk}$. Further more, the objective F_2 is to maximize customer satisfaction. To measure customer satisfaction, we employ the membership function of fuzzy due time $\zeta_{jk}(t_{jk})$ to specify it, which is described as follows,

$$\zeta_{jk}(t_{jk}) = \begin{cases} 0 & t_{jk} \le ET_k \\ \dfrac{(t_{jk} - ET_k)}{(ET_k^d - ET_k)} & ET_k < t_{jk} < ET_k^d \\ 1 & ET_k^d \le t_{jk} \le LT_k^d \\ \dfrac{LT_k - t_{jk}}{LT_k - LT_k^d} & LT_k^d < t_{jk} < LT_k \\ 0 & t_{jk} \ge LT_k \end{cases} \tag{2.88}$$

where (ET_k, LT_k): the max time range for customer k to tolerant; (ET_k^d, LT_k^d): the expected time range of customer k. Figure 2.11 presents the description of this function.

Thus, we can develop mathematical formulations of objectives as follows,

$$\min F_1 = \sum_{j\in J} f_j z_j + \sum_{i\in I}\sum_{j\in J} \widetilde{pd}_{ij} x_{ij} + \sum_{j\in J}\sum_{k\in K} \widetilde{dc}_{jk} y_{jk},$$
$$\max F_2 = \sum_{j\in J}\sum_{k\in K} \zeta_{jk}(t_{jk}) v_{jk}.$$

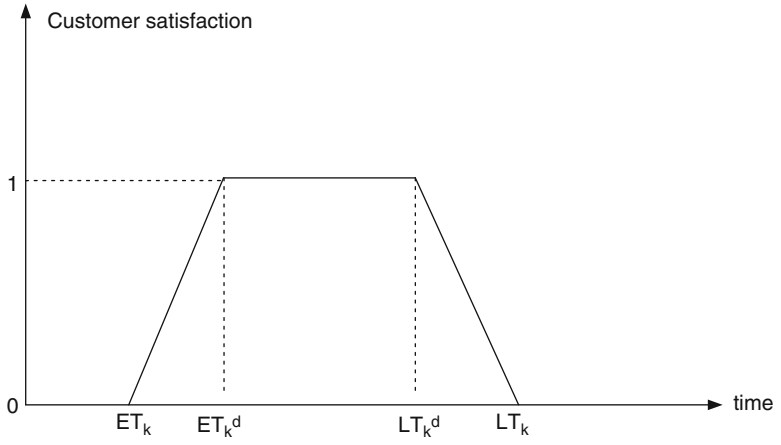

Fig. 2.11 The membership function of $\zeta_{jk}(t_{jk})$

5. Constraints

Since one customer can be serviced by only one DC, thus we employ the constraint

$$\sum_{j \in J} v_{jk} = 1, \qquad \forall k \tag{2.89}$$

With a similar process of dealing with binary variables, we can obtain

$$z_j \in \{0, 1\}, \qquad \forall k \tag{2.90}$$

$$v_{jk} \in \{0, 1\}, \qquad \forall j, k \tag{2.91}$$

And, the number of opened DCs must be not larger than the maximum number M for the DCs. Thus, the constraint

$$\sum_{j \in J} z_j \leq M \tag{2.92}$$

In addition, products transported from the plant i can not exceed its supply capacity a_i, thus we employ the following constraint

$$\sum_{j \in J} x_{ij} \leq a_i, \qquad \forall i \tag{2.93}$$

to specify it. Since there is an assumption of no inventory, the quantity of input is equal to the output at the DC and the quantity of products transported from the DC to the customer is equal to the number of products customers demand. Thus

$$\sum_{i \in I} x_{ij} = \sum_{k \in K} y_{jk}, \qquad \forall j \tag{2.94}$$

$$y_{jk} = \widetilde{b}_k v_{jk}, \qquad \forall j \tag{2.95}$$

Capacity constraints for opened DCs are that the quantity of customers demand on products must be not larger than the quantity of annual throughput of the opened DCs and the quantity of products transported from plants to DCs should be not larger than the DCs' capacity constraint.

$$\sum_{k \in K} \widetilde{b}_k v_{jk} \leq m_j z_j, \qquad \forall j \tag{2.96}$$

$$\sum_{i \in I} x_{ij} \leq m_j z_j, \qquad \forall j \tag{2.97}$$

From the above discussion, by integration of (2.89)–(2.97), we can formulate a random multi-objective programming model as follows:

$$\begin{cases} \min F_1 = \sum\limits_{j\in J} f_j z_j + \sum\limits_{i\in I}\sum\limits_{j\in J} \widetilde{pd}_{ij} x_{ij} + \sum\limits_{j\in J}\sum\limits_{k\in K} \widetilde{dc}_{jk} y_{jk} \\ \max F_2 = \sum\limits_{j\in J}\sum\limits_{k\in K} \zeta_{jk}(t_{jk}) v_{jk} \\ \text{s.t.} \begin{cases} \sum\limits_{j\in J} v_{jk} = 1, \quad \forall k \\ \sum\limits_{j\in J} z_j \le M \\ \sum\limits_{j\in J} x_{ij} \le a_i, \quad \forall i \\ \sum\limits_{i\in I} x_{ij} = \sum\limits_{k\in K} y_{jk}, \quad \forall j \\ y_{jk} = \widetilde{b}_k v_{jk}, \quad \forall j \\ \sum\limits_{k\in K} \widetilde{b}_k v_{jk} \le m_j z_j, \quad \forall j \\ \sum\limits_{i\in I} x_{ij} \le m_j z_j, \quad \forall j \\ z_j = \{0,1\}, \quad \forall k \\ v_{jk} = \{0,1\}, \quad \forall j,k \\ x_{ij} \ge 0, \quad \forall i,j \\ y_{jk} \ge 0, \quad \forall j,k \end{cases} \end{cases} \tag{2.98}$$

Then using the proposed technique in the above sections, let's consider the following real-life problem. Luzhou Xin orchards Co., Ltd. is one of the producers of Chinese fruit beverages in China. Now, the task for a decision-maker is to establish a distribution network at four DC locations which are Shanghai, Beijing, Chengdu and Guangzhou. The objectives, as given in the mathematical model, are the minimization of overall transport costs and the maximization of customer satisfaction. At the beginning of this task, the decision-maker needs to obtain basic data, such as demand quantity, and the per unit transport cost. In fact, since the transport plan is made in advance, we generally cannot exact data. In this condition, the usual way is to obtain random data by means of an experienced evaluation or expert advice. In this example, the notations $\widetilde{b}_k$, $\widetilde{pd}_{ij}$ and $\widetilde{dc}_{jk}$ are employed to denote the demand quantity, transport cost from plant i to DC j and from DC j to the customer respectively. Since it will take more space, we can not give all customers and we give an example of a small sized problem of 5 customers to specify the efficiency of the random simulation-based SA. The information regarding the capacity of the plants and DCs, fixed cost of DCs is given in Table 2.7, and corresponding random data are listed from Tables 2.8 to 2.10.

Transport time from DC j to customer k is in list Table 2.11, the maximum time range for customer tolerance, and the expected time range for the customer can not be directly gained. We assume transport time t_k required by the customer k are independent random variables with a normal distribution: $\mathcal{N}(\mu_k, \sigma_k^2)$, the max

Table 2.7 Capacities and fixed costs for plants and distribution centers

Plants	Capacity(ton/year)	DCs	Capacity (ton/year)	Fixed cost (ten thousand yuan/year)
Jiangy	10,000	Shangh	7,604	48
Longm	10,000	Beij	8,751	44
Luxian	20,000	Chengd	17,600	49
		Wuh	15,800	51

Table 2.8 Shipping cost from plants to DCs (yuan/ton)

Plants	Shipping cost	DCs				σ_i
		Beij	Shangh	Chengd	Wuh	
Jiangy	μ_{1j}^d	550	650	200	350	3
Longm	μ_{2j}^d	550	550	200	350	4
Luxian	μ_{1j}^d	500	500	200	300	3.5

Shipping cost from plants to DCs $\widetilde{pd}_{ij} \sim \mathcal{N}(\mu_{ij}^d, \sigma_i^2)$

Table 2.9 Shipping cost value from DCs to customers (yuan/ton)

DCs	Shipping cost	Customers					σ_i
		1	2	3	4	5	
Shangh	μ_{1j}^c	450	550	200	500	550	4
Beij	μ_{2j}^c	200	400	600	200	550	5
Chengd	μ_{3j}^c	500	200	600	500	400	4
Wuh	μ_{4j}^c	500	310	300	500	550	3

Shipping cost value from DCs to customers $\widetilde{dc}_{jk} \sim \mathcal{N}(\mu_{ij}^c, \sigma_i^2)$

Table 2.10 Customer demand (ton/year)

Customer demand	Customers					σ_j
	1	2	3	4	5	
μ_{1k}	5,950	5,860	5,288	5,520	7,310	15
μ_{2k}	6,000	6,280	7,094	7,760	7,755	20
μ_{3k}	6,050	6,700	8,900	10,000	8,200	16

Customer demand $\widetilde{b}_k \sim \mathcal{N}(\mu_{jk}, \sigma_j^2)$

Table 2.11 Transport time from DCs to customers (h)

Customers DCs	Shangh	Beij	Chengd	Wuh
1	23	32	29	28
2	24	30	27.5	27
3	24.5	30.5	32	30
4	24	33	28.5	30
5	26.5	27	25.5	31.5

time range for customer tolerance is $[\mu_k - \sigma_k, \mu_k + \sigma_k]$ and the expected time range of the customer is $[\mu_k - 0.5\sigma_k, \mu_k + 0.5\sigma_k]$, where μ_k is deemed equal to t_k in practice and σ_k is random from range [1, 5]. Then through (2.88), we have customer satisfaction shown in Table 2.12.

Table 2.12 Customer satisfaction

Customers DCs	μ_k	σ_k	(ET_{jk}, LT_{jk})	(ET^d_{jk}, LT^d_{jk})	Shangh	Beij	Chengd	Wuh
1	28	1.5	[26.5, 29.5]	[27.3, 28.8]	0	0	0.71	1
2	30	4.0	[26.0, 34.0]	[28.0, 32.0]	0	1	0.75	0.5
3	32	1.2	[30.8, 33.2]	[31.4, 32.6]	0	0	1	0
4	28	3.2	[24.8, 31.2]	[26.4, 29.6]	0	0	1	0.75
5	30	4.4	[25.6, 34.4]	[27.8, 32.2]	0.41	0.64	0	1

It is usually difficult for decision makers to justify where to set a distribution and how many products to provide under the condition that there are many uncertain coefficients due to the randomness. So we must apply the methods introduce in the above sections to deal with the uncertainty, i.e., expected value model, chance constraint model and dependent chance model.

Firstly, taking all the random coefficients into the model (2.98), we have the equivalent model by Theorem 2.2 as follows,

$$\begin{cases} \min F_1 = \sum\limits_{j=1}^{4} f_j z_j + \sum\limits_{i=1}^{3}\sum\limits_{j=1}^{4} \mu^d_{ij} x_{ij} + \sum\limits_{j=1}^{4}\sum\limits_{k=1}^{5} \mu^c_{jk} y_{jk} \\ \max F_2 = \sum\limits_{j=1}^{4}\sum\limits_{k=1}^{5} \frac{1}{4}(ET_{jk} + LT_{jk} + ET^d_{jk} + LT^d_{jk}) v_{jk} \\ \text{s.t.} \begin{cases} \sum\limits_{j=1}^{4} v_{jk} = 1, \quad \forall k \\ \sum\limits_{j=1}^{4} z_j \le M \\ \sum\limits_{j=1}^{4} x_{ij} \le a_i, \quad \forall i \\ \sum\limits_{i=1}^{3} x_{ij} = \sum\limits_{k=1}^{5} y_{jk}, \quad \forall j \\ y_{jk} = \mu_{jk} v_{jk}, \quad \forall j \\ \sum\limits_{k=1}^{5} \mu_{jk} v_{jk} \le m_j z_j, \quad \forall j \\ \sum\limits_{i=1}^{3} x_{ij} \le m_j z_j, \quad \forall j \\ z_j = \{0, 1\}, \quad \forall k \\ v_{jk} = \{0, 1\}, \quad \forall j, k \\ x_{ij} \ge 0, \quad \forall i, j \\ y_{jk} \ge 0, \quad \forall j, k \end{cases} \end{cases} \tag{2.99}$$

If the system requires the number of distribution centers cannot be more than 3, i.e., $M = 3$. Then we have the optimal solution as follows: $z* = (1, 1, 0, 1)^T$.

Secondly, if decision makers want to control the cost budget and satisfaction levels to a certain extent, for example, under probability $\alpha = 0.9$ and with credibility $\beta = 0.95$, we can apply the chance constraint operator to deal with it. Taking all the

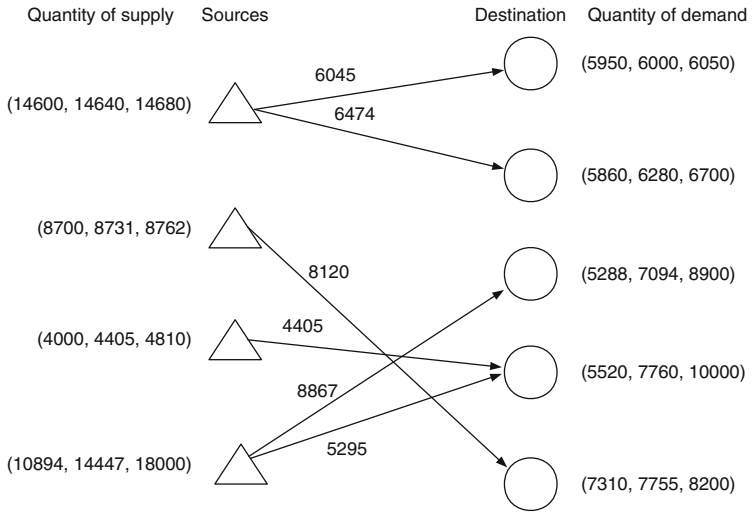

Fig. 2.12 Illustration of the optimal transportation plan

numbers into the model (2.98), the chance constraint model is as follows,

$$
\begin{cases}
\min[\bar{f_1}, -\bar{f_2}] \\
\text{s.t.} \begin{cases}
Pr\left\{ \sum\limits_{j=1}^{4} f_j z_j + \sum\limits_{i=1}^{3} \sum\limits_{j=1}^{4} \widetilde{pd}_{ij} x_{ij} + \sum\limits_{j=1}^{4} \sum\limits_{k=1}^{5} \widetilde{dc}_{jk} y_{jk} \le \bar{f_1} \right\} \ge 0.9 \\
\sum\limits_{j=1}^{4} v_{jk} = 1, \quad \forall k \\
\sum\limits_{j=1}^{4} z_j \le M \\
\sum\limits_{j=1}^{4} x_{ij} \le a_i, \quad \forall i \\
\sum\limits_{i=1}^{3} x_{ij} = \sum\limits_{k=1}^{5} y_{jk}, \quad \forall j \\
y_{jk} = \mu_{jk} v_{jk}, \quad \forall j \\
\sum\limits_{k=1}^{5} \mu_{jk} v_{jk} \le m_j z_j, \quad \forall j \\
\sum\limits_{i=1}^{3} x_{ij} \le m_j z_j, \quad \forall j \\
z_j = \{0, 1\}, \quad \forall k \\
v_{jk} = \{0, 1\}, \quad \forall j, k \\
x_{ij} \ge 0, \quad \forall i, j \\
y_{jk} \ge 0, \quad \forall j, k
\end{cases}
\end{cases}
\tag{2.100}
$$

Then by Theorem 2.6, we can get the equivalent of (2.100) and by the proposed method in the above section we get the optimal transport plan as shown in Fig. 2.12.

http://www.springer.com/978-3-642-17999-0

Random-Like Multiple Objective Decision Making

Xu, J.; Yao, L.

2011, XX, 436 p. 97 illus., Softcover
ISBN: 978-3-642-17999-0