Theory and Practice of Uncertain Programming

Second Edition

Baoding Liu
Uncertainty Theory Laboratory
Department of Mathematical Sciences
Tsinghua University
Beijing 100084, China
liu@tsinghua.edu.cn
http://orsc.edu.cn/liu

Reference to this book should be made as follows:
Liu B, *Theory and Practice of Uncertain Programming*,
2nd ed., http://orsc.edu.cn/liu/up.pdf

Contents

Preface

Real-life decisions are usually made in the state of uncertainty. How do we model optimization problems in uncertain environments? How do we solve these models? The main purpose of the book is just to provide uncertain programming theory to answer these questions.

By uncertain programming we mean the optimization theory in uncertain environments. Stochastic programming, fuzzy programming and hybrid programming are subtopics of uncertain programming.

This book provides a self-contained, comprehensive and up-to-date presentation of uncertain programming theory, including numerous modeling ideas and applications in system reliability design, project scheduling problem, vehicle routing problem, facility location problem, and machine scheduling problem.

Numerous intelligent algorithms such as genetic algorithms and neural networks have been developed by researchers of different backgrounds. A natural idea is to integrate these intelligent algorithms to produce more effective and powerful algorithms. In order to solve uncertain programming models, a spectrum of hybrid intelligent algorithms are documented in the book. The author also maintains a website at http://orsc.edu.cn/liu to post the C++ source files of simulations, genetic algorithms, neural networks, and hybrid intelligent algorithms.

It is assumed that readers are familiar with the basic concepts of mathematical programming, and elementary knowledge of C++ language. In order to make the book more readable, some background topics that will be useful in reading the book are also presented. The book is suitable for researchers, engineers, and students in the field of operations research, information science, management science, system science, computer science, and engineering. The readers will learn numerous new modeling ideas and effective algorithms, and find this work a stimulating and useful reference.

Baoding Liu
Tsinghua University
http://orsc.edu.cn/liu

March 26, 2008

Chapter 1

Mathematical Programming

As one of the most widely used techniques in operations research, *mathematical programming* is defined as a means of maximizing a quantity known as *objective function*, subject to a set of constraints represented by equations and inequalities. Some known subtopics of mathematical programming are linear programming, nonlinear programming, multiobjective programming, goal programming, dynamic programming, and multilevel programming.

It is impossible to cover in a single chapter every concept of mathematical programming. This chapter introduces only the basic concepts and techniques of mathematical programming such that readers gain an understanding of them throughout the book.

1.1 Single-Objective Programming

The general form of single-objective programming (SOP) is written as follows,

$$\begin{cases} \max \ f(\boldsymbol{x}) \\ \text{subject to:} \\ \qquad g_j(\boldsymbol{x}) \leq 0, \quad j = 1, 2, \cdots, p \end{cases} \tag{1.1}$$

which maximizes a real-valued function f of $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ subject to a set of constraints.

Definition 1.1 *In SOP (1.1), we call $\boldsymbol{x}$ a decision vector, and $x_1, x_2, \cdots, x_n$ decision variables. The function f is called the objective function. The set*

$$S = \left\{ \boldsymbol{x} \in \Re^n \;\middle|\; g_j(\boldsymbol{x}) \leq 0, \, j = 1, 2, \cdots, p \right\} \tag{1.2}$$

is called the feasible set. An element $\boldsymbol{x}$ in S is called a feasible solution.

Definition 1.2 *A feasible solution $\boldsymbol{x}^*$ is called the optimal solution of SOP (1.1) if and only if*

$$f(\boldsymbol{x}^*) \geq f(\boldsymbol{x}) \tag{1.3}$$

for any feasible solution $\boldsymbol{x}$.

One of the outstanding contributions to mathematical programming was known as the Kuhn-Tucker conditions. In order to introduce them, let us give some definitions. An inequality constraint $g_j(\boldsymbol{x}) \leq 0$ is said to be active at a point $\boldsymbol{x}^*$ if $g_j(\boldsymbol{x}^*) = 0$. A point $\boldsymbol{x}^*$ satisfying $g_j(\boldsymbol{x}^*) \leq 0$ is said to be regular if the gradient vectors $\nabla g_j(\boldsymbol{x})$ of all active constraints are linearly independent.

Let $\boldsymbol{x}^*$ be a regular point of the constraints of SOP (1.1) and assume that all the functions $f(\boldsymbol{x})$ and $g_j(\boldsymbol{x}), j = 1, 2, \cdots, p$ are differentiable. If $\boldsymbol{x}^*$ is a local optimal solution, then there exist Lagrange multipliers $\lambda_j, j = 1, 2, \cdots, p$ such that the following Kuhn-Tucker conditions hold,

$$\begin{cases} \nabla f(\boldsymbol{x}^*) - \sum\limits_{j=1}^{p} \lambda_j \nabla g_j(\boldsymbol{x}^*) = 0 \\ \lambda_j g_j(\boldsymbol{x}^*) = 0, \quad j = 1, 2, \cdots, p \\ \lambda_j \geq 0, \quad j = 1, 2, \cdots, p. \end{cases} \tag{1.4}$$

If all the functions $f(\boldsymbol{x})$ and $g_j(\boldsymbol{x}), j = 1, 2, \cdots, p$ are convex and differentiable, and the point $\boldsymbol{x}^*$ satisfies the Kuhn-Tucker conditions (1.4), then it has been proved that the point $\boldsymbol{x}^*$ is a global optimal solution of SOP (1.1).

Linear Programming

If the functions $f(\boldsymbol{x}), g_j(\boldsymbol{x}), j = 1, 2, \cdots, p$ are all linear, then SOP (1.1) is called a *linear programming*.

The feasible set of linear programming is always convex. A point $\boldsymbol{x}$ is called an extreme point of convex set S if $\boldsymbol{x} \in S$ and $\boldsymbol{x}$ cannot be expressed as a convex combination of two points in S. It has been shown that the optimal solution to linear programming corresponds to an extreme point of its feasible set provided that the feasible set S is bounded. This fact is the basis of the *simplex algorithm* which was developed by Dantzig [53] as a very efficient method for solving linear programming.

Roughly speaking, the simplex algorithm examines only the extreme points of the feasible set, rather than all feasible points. At first, the simplex algorithm selects an extreme point as the initial point. The successive extreme point is selected so as to improve the objective function value. The procedure is repeated until no improvement in objective function value can be made. The last extreme point is the optimal solution.

Nonlinear Programming

If at least one of the functions $f(\boldsymbol{x}), g_j(\boldsymbol{x}), j = 1, 2, \cdots, p$ is nonlinear, then SOP (1.1) is called a *nonlinear programming*.

A large number of classical optimization methods have been developed to treat special-structural nonlinear programming based on the mathematical theory concerned with analyzing the structure of problems.

Now we consider a nonlinear programming which is confronted solely with maximizing a real-valued function with domain $\Re^n$. Whether derivatives are available or not, the usual strategy is first to select a point in $\Re^n$ which is thought to be the most likely place where the maximum exists. If there is no information available on which to base such a selection, a point is chosen at random. From this first point an attempt is made to construct a sequence of points, each of which yields an improved objective function value over its predecessor. The next point to be added to the sequence is chosen by analyzing the behavior of the function at the previous points. This construction continues until some termination criterion is met. Methods based upon this strategy are called *ascent methods*, which can be classified as *direct methods*, *gradient methods*, and *Hessian methods* according to the information about the behavior of objective function f. Direct methods require only that the function can be evaluated at each point. Gradient methods require the evaluation of first derivatives of f. Hessian methods require the evaluation of second derivatives. In fact, there is no superior method for all problems. The efficiency of a method is very much dependent upon the objective function.

Integer Programming

Integer programming is a special mathematical programming in which all of the variables are assumed to be only integer values. When there are not only integer variables but also conventional continuous variables, we call it *mixed integer programming*. If all the variables are assumed either 0 or 1, then the problem is termed a *zero-one programming*. Although integer programming can be solved by an *exhaustive enumeration* theoretically, it is impractical to solve realistically sized integer programming problems. The most successful algorithm so far found to solve integer programming is called the *branch-and-bound enumeration* developed by Balas (1965) and Dakin (1965). The other technique to integer programming is the *cutting plane method* developed by Gomory (1959).

1.2 Multiobjective Programming

SOP is related to maximizing a single function subject to a number of constraints. However, it has been increasingly recognized that many real-world

decision-making problems involve multiple, noncommensurable, and conflicting objectives which should be considered simultaneously. As an extension, *multiobjective programming* (MOP) is defined as a means of optimizing multiple objective functions subject to a number of constraints, i.e.,

$$\begin{cases} \max\ [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x})] \\ \text{subject to:} \\ \qquad g_j(\boldsymbol{x}) \leq 0,\ j = 1, 2, \cdots, p \end{cases} \tag{1.5}$$

where $f_i(\boldsymbol{x})$ are objective functions, $i = 1, 2, \cdots, m$, and $g_j(\boldsymbol{x}) \leq 0$ are system constraints, $j = 1, 2, \cdots, p$.

When the objectives are in conflict, there is no optimal solution that simultaneously maximizes all the objective functions. For this case, we employ a concept of *Pareto solution*, which means that it is impossible to improve any one objective without sacrificing on one or more of the other objectives.

Definition 1.3 *A feasible solution $\boldsymbol{x}^*$ is said to be a Pareto solution if there is no feasible solution $\boldsymbol{x}$ such that*

$$f_i(\boldsymbol{x}) \geq f_i(\boldsymbol{x}^*), \quad i = 1, 2, \cdots, m \tag{1.6}$$

and $f_j(\boldsymbol{x}) > f_j(\boldsymbol{x}^*)$ for at least one index j.

If the decision maker has a real-valued *preference function* aggregating the m objective functions, then we may maximize the aggregating preference function subject to the same set of constraints. This model is referred to as a *compromise model* whose solution is called a *compromise solution*.

The first well-known compromise model is set up by weighting the objective functions, i.e.,

$$\begin{cases} \max \sum_{i=1}^{m} \lambda_i f_i(\boldsymbol{x}) \\ \text{subject to:} \\ \qquad g_j(\boldsymbol{x}) \leq 0,\ j = 1, 2, \cdots, p \end{cases} \tag{1.7}$$

where the weights $\lambda_1, \lambda_2, \cdots, \lambda_m$ are nonnegative numbers with $\lambda_1 + \lambda_2 + \cdots + \lambda_m = 1$. Note that the solution of (1.7) must be a Pareto solution of the original one.

The second way is related to minimizing the *distance function* from a solution $(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x}))$ to an ideal vector $(f_1^*, f_2^*, \cdots, f_m^*)$, where f_i^* are the optimal values of the ith objective functions without considering other objectives, $i = 1, 2, \cdots, m$, respectively, i.e.,

$$\begin{cases} \min \sqrt{(f_1(\boldsymbol{x}) - f_1^*)^2 + \cdots + (f_m(\boldsymbol{x}) - f_m^*)^2} \\ \text{subject to:} \\ \qquad g_j(\boldsymbol{x}) \leq 0,\ j = 1, 2, \cdots, p. \end{cases} \tag{1.8}$$

By the third way a compromise solution can be found via an *interactive approach* consisting of a sequence of decision phases and computation phases. Various interactive approaches have been developed in the past literature.

1.3 Goal Programming

Goal programming (GP) was developed by Charnes and Cooper [39] and subsequently studied by many researchers. GP can be regarded as a special compromise model for multiobjective programming and has been applied in a wide variety of real-world problems.

In multiobjective decision-making problems, we assume that the decision-maker is able to assign a target level for each goal and the key idea is to minimize the deviations (positive, negative, or both) from the target levels. In the real-world situation, the goals are achievable only at the expense of other goals and these goals are usually incompatible. Therefore, there is a need to establish a hierarchy of importance among these incompatible goals so as to satisfy as many goals as possible in the order specified. The general form of GP is written as follows,

$$\begin{cases} \min \sum_{j=1}^{l} P_j \sum_{i=1}^{m} (u_{ij} d_i^+ \vee 0 + v_{ij} d_i^- \vee 0) \\ \text{subject to:} \\ \qquad f_i(\boldsymbol{x}) - b_i = d_i^+, \quad i = 1, 2, \cdots, m \\ \qquad b_i - f_i(\boldsymbol{x}) = d_i^-, \quad i = 1, 2, \cdots, m \\ \qquad g_j(\boldsymbol{x}) \leq 0, \qquad\quad j = 1, 2, \cdots, p \end{cases} \tag{1.9}$$

where P_j is the preemptive priority factor which expresses the relative importance of various goals, $P_j \gg P_{j+1}$, 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, $d_i^+ \vee 0$ is the positive deviation from the target of goal i, $d_i^- \vee 0$ is the negative deviation from the target of goal i, 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. Sometimes, the objective function of GP (1.9) is written as

$$\text{lexmin} \left\{ \sum_{i=1}^{m} (u_{i1} d_i^+ \vee 0 + v_{i1} d_i^- \vee 0), \cdots, \sum_{i=1}^{m} (u_{il} d_i^+ \vee 0 + v_{il} d_i^- \vee 0) \right\}$$

where lexmin represents lexicographically minimizing the objective vector.

Linear GP can be successfully solved by the simplex goal method. The approaches of nonlinear GP are summarized by Saber and Ravindran [270] and the efficiency of these approaches varies. They are classified as follows:

(a) simplex-based approach, whose main idea lies in converting the nonlinear GP into a set of approximation linear GPs which can be handled by the simplex goal method; (b) direct search approach [50], in which the given nonlinear GP is translated into a set of SOPs, and then the SOPs are solved by the direct search methods; (c) gradient-based approach [154][270], which utilizes the gradient of constraints to identify a feasible direction and then solves the GP based on the feasible direction method; (d) interactive approach [309][225], which can yield a satisfactory solution in a relatively few iterations since the decision-maker is involved in the solution process; and (e) genetic algorithm [88], which can deal with complex nonlinear GP but have to spend more CPU time.

1.4 Dynamic Programming

Let us denote a multistage decision process by $[\boldsymbol{a}, T(\boldsymbol{a}, \boldsymbol{x})]$, where $\boldsymbol{a}$ is called *state*, $T(\boldsymbol{a}, \boldsymbol{x})$ is called a *state transition function*, and $\boldsymbol{x}$ is called *decision vector*. It is clear that the state transition function depends on both state $\boldsymbol{a}$ and decision vector $\boldsymbol{x}$. We suppose that we have sufficient influence over the process so that at each stage we can choose a decision vector $\boldsymbol{x}$ from the allowable set. Let $\boldsymbol{x}_i$ be the decision vector at the ith stage. Then we have the following sequence,

$$\begin{aligned} &\boldsymbol{a}_1 = \boldsymbol{a}_0, \quad \text{(an initial state)} \\ &\boldsymbol{a}_{i+1} = T(\boldsymbol{a}_i, \boldsymbol{x}_i), \quad i = 1, 2, \cdots \end{aligned}$$

We are concerned with processes in which the decision vectors $\boldsymbol{x}_i$'s are chosen so as to maximize a *criterion function* $R(\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_N; \boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N)$. A decision is called *optimal* if it maximizes the criterion function.

In view of the general nature of the criterion function R, the decision vectors $\boldsymbol{x}_i$'s are dependent upon the current state of the system as well as the past and future states and decisions. However, there are some criterion functions which have some special structures so that the decision is dependent only on the current state. In this special but extremely important case, the optimal policy is characterized by Bellman's principle of optimality: *An optimal policy has the property that whatever the initial state and initial decision are, the remaining decision must constitute an optimal policy with regard to the state resulting from the first decision.*

Fortunately, many important criteria have the vital property of divorcing the past from the present. In general, it is easy to predict this property from the nature of the original multistage decision process. For example, let us consider a problem of maximizing the following special-structured function

$$R(\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_N; \boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N) = \sum_{i=1}^{N} r_i(\boldsymbol{a}_i, \boldsymbol{x}_i) \tag{1.10}$$

subject to $g_i(\boldsymbol{a}_i, \boldsymbol{x}_i) \leq 0$ for $i = 1, 2, \cdots, N$. Let $f_n(\boldsymbol{a})$ be the maximum values of criterion function R, starting in state $\boldsymbol{a}$ at the stage n, $n = 1, 2, \cdots, N$, respectively. Then by Bellman's principle of optimality, we have

$$\begin{cases} f_N(\boldsymbol{a}) = \max\limits_{g_N(\boldsymbol{a},\boldsymbol{x})\leq 0} r_N(\boldsymbol{a}, \boldsymbol{x}) \\ f_{N-1}(\boldsymbol{a}) = \max\limits_{g_{N-1}(\boldsymbol{a},\boldsymbol{x})\leq 0} \big\{r_{N-1}(\boldsymbol{a}, \boldsymbol{x}) + f_N(T(\boldsymbol{a}, \boldsymbol{x}))\big\} \\ \qquad \cdots \\ f_1(\boldsymbol{a}) = \max\limits_{g_1(\boldsymbol{a},\boldsymbol{x})\leq 0} \big\{r_1(\boldsymbol{a}, \boldsymbol{x}) + f_2(T(\boldsymbol{a}, \boldsymbol{x}))\big\}. \end{cases} \tag{1.11}$$

Please mention that,

$$\max_{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N} R(\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_N; \boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N) = f_1(\boldsymbol{a}_0). \tag{1.12}$$

The system of equations (1.11) is called *dynamic programming* (DP) by Richard Bellman [12] which can be simply written as

$$\begin{cases} f_N(\boldsymbol{a}) = \max\limits_{g_N(\boldsymbol{a},\boldsymbol{x})\leq 0} r_N(\boldsymbol{a}, \boldsymbol{x}) \\ f_n(\boldsymbol{a}) = \max\limits_{g_n(\boldsymbol{a},\boldsymbol{x})\leq 0} \big\{r_n(\boldsymbol{a}, \boldsymbol{x}) + f_{n+1}(T(\boldsymbol{a}, \boldsymbol{x}))\big\} \\ \qquad n \leq N - 1. \end{cases} \tag{1.13}$$

In order to obtain the optimal solutions in reasonable time for real practical problems, we should develop effectively computational algorithms for DP. To explore the general DP algorithms, readers may consult the book by Bertsekas and Tsitsiklis [16] in which numerous different ways to solve DP problems have been suggested.

1.5 Multilevel Programming

Multilevel programming (MLP) offers a means of studying decentralized decision systems in which we assume that the leader and followers may have their own decision variables and objective functions, and the leader can only influence the reactions of followers through his own decision variables, while the followers have full authority to decide how to optimize their own objective functions in view of the decisions of the leader and other followers.

We now assume that in a decentralized two-level decision system there is one leader and m followers. Let $\boldsymbol{x}$ and $\boldsymbol{y}_i$ be the decision vectors of the leader and the ith followers, $i = 1, 2, \cdots, m$, respectively. We also assume that the objective functions of the leader and ith followers are $F(\boldsymbol{x}, \boldsymbol{y}_1, \cdots, \boldsymbol{y}_m)$ and $f_i(\boldsymbol{x}, \boldsymbol{y}_1, \cdots, \boldsymbol{y}_m)$, $i = 1, 2, \cdots, m$, respectively.

In addition, let the feasible set of control vector $\boldsymbol{x}$ of the leader be determined by

$$G(\boldsymbol{x}) \leq 0 \tag{1.14}$$

where G is a vector-valued function of decision vector $\boldsymbol{x}$ and 0 is a vector with zero components. Then for each decision $\boldsymbol{x}$ chosen by the leader, the feasibility of decision vectors $\boldsymbol{y}_i$ of the ith followers should be dependent on not only $\boldsymbol{x}$ but also $\boldsymbol{y}_1, \cdots, \boldsymbol{y}_{i-1}, \boldsymbol{y}_{i+1}, \cdots, \boldsymbol{y}_m$, and generally represented by

$$g_i(\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m) \leq 0 \tag{1.15}$$

where g_i are vector-valued functions, $i = 1, 2, \cdots, m$, respectively.

Assume that the leader first chooses his decision vector $\boldsymbol{x}$, and the followers determine their decision array $(\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m)$ after that. In order to find the optimal decision vector of the leader, we have to use the following bilevel programming,

$$\begin{cases} \max\limits_{\boldsymbol{x}} F(\boldsymbol{x}, \boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*) \\ \text{subject to:} \\ \quad G(\boldsymbol{x}) \leq 0 \\ \quad (\boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*) \text{ solves problems } (i = 1, 2, \cdots, m) \\ \quad \begin{cases} \max\limits_{\boldsymbol{y_i}} f_i(\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m) \\ \text{subject to:} \\ \quad g_i(\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m) \leq 0. \end{cases} \end{cases} \tag{1.16}$$

Definition 1.4 *Let $\boldsymbol{x}$ be a fixed decision vector of the leader. A Nash equilibrium of followers with respect to $\boldsymbol{x}$ is the feasible array $(\boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*)$ such that*

$$f_i(\boldsymbol{x}, \boldsymbol{y}_1^*, \cdots, \boldsymbol{y}_{i-1}^*, \boldsymbol{y}_i, \boldsymbol{y}_{i+1}^*, \cdots, \boldsymbol{y}_m^*) \leq f_i(\boldsymbol{x}, \boldsymbol{y}_1^*, \cdots, \boldsymbol{y}_{i-1}^*, \boldsymbol{y}_i^*, \boldsymbol{y}_{i+1}^*, \cdots, \boldsymbol{y}_m^*)$$

for any feasible array $(\boldsymbol{y}_1^*, \cdots, \boldsymbol{y}_{i-1}^*, \boldsymbol{y}_i, \boldsymbol{y}_{i+1}^*, \cdots, \boldsymbol{y}_m^*)$ and $i = 1, 2, \cdots, m$.

Definition 1.5 *Suppose that $\boldsymbol{x}^*$ is a feasible decision vector of the leader and $(\boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*)$ is a Nash equilibrium of followers with respect to $\boldsymbol{x}^*$. We call $(\boldsymbol{x}^*, \boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*)$ a Stackelberg-Nash equilibrium to MLP (1.16) if and only if*

$$F(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}_1, \overline{\boldsymbol{y}}_2, \cdots, \overline{\boldsymbol{y}}_m) \leq F(\boldsymbol{x}^*, \boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*) \tag{1.17}$$

for any feasible $\overline{\boldsymbol{x}}$ and Nash equilibrium $(\overline{\boldsymbol{y}}_1, \overline{\boldsymbol{y}}_2, \cdots, \overline{\boldsymbol{y}}_m)$ with respect to $\overline{\boldsymbol{x}}$.

Ben-Ayed and Blair [14] showed that MLP is an NP-hard problem. In order to solve MLP, a lot of numerical algorithms have been developed, for example, implicit enumeration scheme (Candler and Townsley [34]), the kth best algorithm (Bialas and Karwan [18]), parametric complementary pivot algorithm (Bialas and Karwan [18]), one-dimensional grid search algorithm (Bard [8][10]), branch-and-bound algorithm (Bard and Moore [9]), the steepest-descent direction (Savard and Gauvin [277]), and genetic algorithm (Liu [171]).

Chapter 2

Genetic Algorithms

Genetic algorithm (GA) is a stochastic search method for optimization problems based on the mechanics of natural selection and natural genetics (i.e., survival of the fittest). GA has demonstrated considerable success in providing good solutions to many complex optimization problems and received more and more attentions during the past three decades. When the objective functions to be optimized in the optimization problems are multimodal or the search spaces are particularly irregular, algorithms need to be highly robust in order to avoid getting stuck at a local optimal solution. The advantage of GA is just able to obtain the global optimal solution fairly. In addition, GA does not require the specific mathematical analysis of optimization problems, which makes GA easily coded by users who are not necessarily good at mathematics and algorithms.

One of the important technical terms in GA is *chromosome*, which is usually a string of symbols or numbers. A chromosome is a coding of a solution of an optimization problem, not necessarily the solution itself. GA starts with an initial set of randomly generated chromosomes called a *population*. The number of individuals in the population is a predetermined integer and is called *population size*. All chromosomes are evaluated by the so-called *evaluation function*, which is some measure of *fitness*. A new population will be formed by a *selection process* using some *sampling mechanism* based on the fitness values. The cycle from one population to the next one is called a *generation*. In each new generation, all chromosomes will be updated by the *crossover* and *mutation* operations. The revised chromosomes are also called *offspring*. The selection process selects chromosomes to form a new population and the genetic system enters a new generation. After performing the genetic system a given number of cycles, we decode the best chromosome into a solution which is regarded as the optimal solution of the optimization problem.

GA has been well-documented in the literature, such as in Holland [98], Goldberg [91], Michalewicz [229], Fogel [71], Koza [139][140], Liu [182], and

have been applied to a wide variety of problems. The aim of this section is to introduce an effective GA for solving complex optimization problems. Moreover, we design this algorithm for solving not only single-objective optimization but also multiobjective programming, goal programming, and multilevel programming. Finally, we illustrate the effectiveness of GA by some numerical examples.

2.1 Representation Structure

A key problem of GA is how to encode a solution $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ into a chromosome $V = (v_1, v_2, \cdots, v_m)$. That is, we must construct a link between a solution space and a coding space. The mapping from the solution space to coding space is called *encoding*. The mapping from the coding space to solution space is called *decoding*.

It is clear that the representation structure is problem-dependent. For example, let (x_1, x_2, x_3) be a solution vector in the solution space

$$\begin{cases} x_1 + x_2^2 + x_3^3 = 1 \\ x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0. \end{cases} \tag{2.1}$$

We may encode the solution by a chromosome (v_1, v_2, v_3) in the coding space

$$v_1 \geq 0, \quad v_2 \geq 0, \quad v_3 \geq 0. \tag{2.2}$$

Then the encoding and decoding processes are determined by the link

$$x_1 = \frac{v_1}{v_1 + v_2 + v_3}, \quad x_2 = \sqrt{\frac{v_2}{v_1 + v_2 + v_3}}, \quad x_3 = \sqrt[3]{\frac{v_3}{v_1 + v_2 + v_3}}. \tag{2.3}$$

2.2 Handling Constraints

In mathematical programming, if there are some equality constraints, for example, $h_k(\boldsymbol{x}) = 0$, $k = 1, 2, \cdots, q$, we should eliminate the q equality constraints by replacing q variables of them with the representation of the remaining variables, where the representation is obtained by solving the system of equalities in the constraints.

If we cannot do so, we may eliminate the equality constraints by Lagrangian method based on the idea of transforming a constrained problem into an unconstrained one.

2.3 Initialization Process

We define an integer *pop_size* as the number of chromosomes and initialize *pop_size* chromosomes randomly. Usually, it is difficult for complex optimization problems to produce feasible chromosomes explicitly.

Assume that the decision-maker can predetermine a region which contains the optimal solution (not necessarily the whole feasible set). Such a region is also problem-dependent. At any rate, the decision-maker can provide such a region, only it may be a bit too large. Usually, this region will be designed to have a nice sharp, for example, a hypercube, because the computer can easily sample points from a hypercube.

We generate a random point from the hypercube and check the feasibility of this point. If it is feasible, then it will be accepted as a chromosome. If not, then we regenerate a point from the hypercube randomly until a feasible one is obtained. We can make *pop_size* initial feasible chromosomes $V_1, V_2, \cdots, V_{pop\_size}$ by repeating the above process *pop_size* times.

2.4 Evaluation Function

Evaluation function, denoted by $Eval(V)$, is to assign a probability of reproduction to each chromosome V so that its likelihood of being selected is proportional to its fitness relative to the other chromosomes in the population. That is, the chromosomes with higher fitness will have more chance to produce offspring by using *roulette wheel selection*.

Let $V_1, V_2, \cdots, V_{pop\_size}$ be the *pop_size* chromosomes at the current generation. One well-known evaluation function is based on allocation of reproductive trials according to rank rather than actual objective values. No matter what type of mathematical programming it is, it is reasonable to assume that the decision-maker can give an order relationship among the *pop_size* chromosomes $V_1, V_2, \cdots, V_{pop\_size}$ such that the *pop_size* chromosomes can be rearranged from good to bad (i.e., the better the chromosome is, the smaller the ordinal number it has). For example, for a single-objective maximizing problem, a chromosome with larger objective value is better; for a multiobjective programming, we may define a preference function to evaluate the chromosomes; for a goal programming, we have the following order relationship for the chromosomes: for any two chromosomes, if the higher-priority objectives are equal, then, in the current priority level, the one with minimal objective value is better. If two different chromosomes have the same objective values at every level, then we are indifferent between them. For this case, we rearrange them randomly.

Now let a parameter $a \in (0, 1)$ in the genetic system be given. We can define the *rank-based evaluation function* as follows,

$$Eval(V_i) = a(1-a)^{i-1}, \qquad i = 1, 2, \cdots, pop\_size. \tag{2.4}$$

Note that $i = 1$ means the best individual, $i = pop\_size$ the worst one.

2.5 Selection Process

The selection process is based on spinning the roulette wheel *pop_size* times. Each time we select a single chromosome for a new population. The roulette wheel is a fitness-proportional selection. No matter what type of evaluation function is employed, the selection process is always stated as follows:

Algorithm 2.1 (Selection Process)
Step 1. Calculate the cumulative probability q_i for each chromosome V_i,

$$q_0 = 0, \quad q_i = \sum_{j=1}^{i} Eval(V_j), \quad i = 1, 2, \cdots, pop\_size.$$

Step 2. Generate a random number r in $(0, q_{pop\_size}]$.
Step 3. Select the chromosome V_i such that $q_{i-1} < r \leq q_i$.
Step 4. Repeat the second and third steps *pop_size* times and obtain *pop_size* copies of chromosome.

Please note that in the above selection process we do not require the condition $q_{pop\_size} = 1$. In fact, if we want, we can divide all q_i's, $i = 1, 2, \cdots, pop\_size$, by q_{pop_size} such that $q_{pop\_size} = 1$ and the new probabilities are also proportional to the fitnesses. However, it does not exert any influence on the genetic process.

2.6 Crossover Operation

We define a parameter P_c of a genetic system as the probability of crossover. This probability gives us the expected number $P_c \cdot pop\_size$ of chromosomes undergoing the crossover operation.

In order to determine the parents for crossover operation, let us do the following process repeatedly from $i = 1$ to *pop_size*: generating a random number r from the interval $[0, 1]$, the chromosome V_i is selected as a parent if $r < P_c$. We denote the selected parents by $V_1', V_2', V_3', \cdots$ and divide them into the following pairs:

$$(V_1', V_2'), \quad (V_3', V_4'), \quad (V_5', V_6'), \quad \cdots$$

Let us illustrate the crossover operator on each pair by (V_1', V_2'). At first, we generate a random number c from the open interval $(0, 1)$, then the crossover operator on V_1' and V_2' will produce two children X and Y as follows:

$$X = c \cdot V_1' + (1 - c) \cdot V_2', \qquad Y = (1 - c) \cdot V_1' + c \cdot V_2'. \tag{2.5}$$

If the feasible set is convex, this crossover operation ensures that both children are feasible if both parents are. However, in many cases, the feasible set is not

necessarily convex, nor is it hard to verify the convexity. Thus we must check the feasibility of each child before accepting it. If both children are feasible, then we replace the parents with them. If not, we keep the feasible one if it exists, and then redo the crossover operator by regenerating a random number c until two feasible children are obtained or a given number of cycles is finished. In this case, we only replace the parents with the feasible children.

2.7 Mutation Operation

We define a parameter P_m of a genetic system as the probability of mutation. This probability gives us the expected number of $P_m \cdot pop\_size$ of chromosomes undergoing the mutation operations.

In a similar manner to the process of selecting parents for crossover operation, we repeat the following steps from $i = 1$ to pop_size: generating a random number r from the interval $[0, 1]$, the chromosome V_i is selected as a parent for mutation if $r < P_m$.

For each selected parent, denoted by $V = (v_1, v_2, \cdots, v_m)$, we mutate it in the following way. Let M be an appropriate large positive number. We choose a mutation direction $\boldsymbol{d}$ in $\Re^m$ randomly. If $V + M \cdot \boldsymbol{d}$ is not feasible, then we set M as a random number between 0 and M until it is feasible. If the above process cannot find a feasible solution in a predetermined number of iterations, then we set $M = 0$. Anyway, we replace the parent V with its child

$$X = V + M \cdot \boldsymbol{d}. \tag{2.6}$$

2.8 General Procedure

Following selection, crossover, and mutation, the new population is ready for its next evaluation. GA will terminate after a given number of cyclic repetitions of the above steps or a suitable solution has been found. We now summarize the GA for optimization problems as follows.

Algorithm 2.2 (Genetic Algorithm)
Step 1. Initialize pop_size chromosomes at random.
Step 2. Update the chromosomes by crossover and mutation operations.
Step 3. Calculate the objective values for all chromosomes.
Step 4. Compute the fitness of each chromosome via the objective values.
Step 5. Select the chromosomes by spinning the roulette wheel.
Step 6. Repeat the second to fifth steps for a given number of cycles.
Step 7. Report the best chromosome as the optimal solution.

Remark 2.1: It is well-known that the best chromosome does not necessarily appear in the last generation. Thus we have to keep the best one from the

beginning. If we find a better one in the new population, then we replace the old one with it. This chromosome will be reported as the optimal solution after finishing the evolution.

2.9 Numerical Experiments

Example 2.1: Now we use GA to solve the following maximization problem,

$$\begin{cases} \max \sqrt{x_1} + \sqrt{x_2} + \sqrt{x_3} \\ \text{subject to:} \\ \quad x_1^2 + 2x_2^2 + 3x_3^2 \leq 1 \\ \quad x_1, x_2, x_3 \geq 0. \end{cases} \tag{2.7}$$

We may encode a solution $\boldsymbol{x} = (x_1, x_2, x_3)$ into a chromosome $V = (v_1, v_2, v_3)$, and decode the chromosome into the solution in the following way,

$$x_1 = v_1, \quad x_2 = v_2, \quad x_3 = v_3.$$

It is easy to know that the feasible coding space is contained in the following hypercube

$$\mathcal{V} = \left\{ (v_1, v_2, v_3) \;\middle|\; 0 \leq v_1 \leq 1,\ 0 \leq v_2 \leq 1,\ 0 \leq v_3 \leq 1 \right\}$$

which is simple for the computer because we can easily sample points from it. We may take

$$v_1 = \mathcal{U}(0,1), \quad v_2 = \mathcal{U}(0,1), \quad v_3 = \mathcal{U}(0,1) \tag{2.8}$$

where the function $\mathcal{U}(a,b)$ generates uniformly distributed variables on the interval $[a,b]$ and will be discussed in detail later. If this chromosome is infeasible, then we reject it and regenerate one by (2.8). If the generated chromosome is feasible, then we accept it as one in the population. After finite times, we can obtain 30 feasible chromosomes. A run of GA with 400 generations shows that the optimal solution is

$$(x_1^*, x_2^*, x_3^*) = (0.636, 0.395, 0.307)$$

whose objective value is 1.980.

Example 2.2: GA is also able to solve the following nonlinear goal programming,

$$\begin{cases} \text{lexmin} \left\{ d_1^- \vee 0, d_2^- \vee 0, d_3^- \vee 0 \right\} \\ \text{subject to:} \\ \quad 3 - \sqrt{x_1} = d_1^- \\ \quad 4 - \sqrt{x_1 + 2x_2} = d_2^- \\ \quad 5 - \sqrt{x_1 + 2x_2 + 3x_3} = d_3^- \\ \quad x_1^2 + x_2^2 + x_3^2 \leq 100 \\ \quad x_1, x_2, x_3 \geq 0. \end{cases}$$

We may encode a solution $\boldsymbol{x} = (x_1, x_2, x_3)$ into a chromosome $V = (v_1, v_2, v_3)$, and decode the chromosome into the solution in the following way,

$$x_1 = v_1, \quad x_2 = v_2, \quad x_3 = v_3.$$

Since the feasible coding space is contained in the following hypercube

$$\mathcal{V} = \left\{(v_1, v_2, v_3) \;\middle|\; 0 \le v_1 \le 10,\, 0 \le v_2 \le 10,\, 0 \le v_3 \le 10\right\},$$

we may take $v_1 = \mathcal{U}(0, 10)$, $v_2 = \mathcal{U}(0, 10)$, and $v_3 = \mathcal{U}(0, 10)$, and accept it as a chromosome if it is feasible. It is clear that we can make 30 feasible chromosomes in finite times. A run of GA with 2000 generations shows that the optimal solution is

$$(x_1^*, x_2^*, x_3^*) = (9.000, 3.500, 2.597)$$

which satisfies the first two goals, but the last objective is 0.122.

Example 2.3: For the following bilevel programming model,

$$\left\{\begin{array}{l} \max\limits_{\boldsymbol{x}} F(\boldsymbol{x}, \boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*) \\ \text{subject to:} \\ \quad G(\boldsymbol{x}) \le 0 \\ \quad (\boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*) \text{ solves problems } (i = 1, 2, \cdots, m) \\ \quad \left\{\begin{array}{l} \max\limits_{\boldsymbol{y_i}} f_i(\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m) \\ \text{subject to:} \\ \quad\quad g_i(\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m) \le 0, \end{array}\right. \end{array}\right. \tag{2.9}$$

we define symbols

$$\boldsymbol{y}_{-i} = (\boldsymbol{y}_1, \cdots, \boldsymbol{y}_{i-1}, \boldsymbol{y}_{i+1}, \cdots, \boldsymbol{y}_m), \quad i = 1, 2, \cdots, m. \tag{2.10}$$

For any decision $\boldsymbol{x}$ revealed by the leader, if the ith follower knows the strategies $\boldsymbol{y}_{-i}$ of other followers, then the optimal reaction of the ith follower is represented by a mapping $\boldsymbol{y}_i = r_i(\boldsymbol{y}_{-i})$, which should solve the subproblem

$$\left\{\begin{array}{l} \max\limits_{\boldsymbol{y_i}} f_i(\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m) \\ \text{subject to:} \\ \quad g_i(\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m) \le 0. \end{array}\right. \tag{2.11}$$

In order to search for the Stackelberg-Nash equilibrium, Liu [171] designed a GA to solve multilevel programming. We first compute the Nash equilibrium with respect to any decision revealed by the leader. It is clear that the Nash equilibrium of the m followers will be the solution $(\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m)$ of the system of equations

$$\boldsymbol{y}_i = r_i(\boldsymbol{y}_{-i}), \quad i = 1, 2, \cdots, m. \tag{2.12}$$

In other words, we should find a fixed point of the vector-valued function $(r_1, r_2, \cdots, r_m)$. In order to solve the system of equations (2.12), we should design some efficient algorithms. The argument breaks down into three cases.

(a) If we have explicit expressions of all functions r_i, $i = 1, 2, \cdots, m$, then we might get an analytic solution to the system (2.12). Unfortunately, it is almost impossible to do this in practice.

(b) The system (2.12) might be solved by some iterative method that generates a sequence of points $\boldsymbol{y}^k = (\boldsymbol{y}_1^k, \boldsymbol{y}_2^k, \cdots, \boldsymbol{y}_m^k)$, $k = 0, 1, 2, \cdots$ via the iteration formula

$$\boldsymbol{y}_i^{k+1} = r_i(\boldsymbol{y}_{-i}^k), \quad i = 1, 2, \cdots, m \tag{2.13}$$

where $\boldsymbol{y}_{-i}^k = (\boldsymbol{y}_1^k, \cdots, \boldsymbol{y}_{i-1}^k, \boldsymbol{y}_{i+1}^k, \cdots, \boldsymbol{y}_m^k)$. However, generally speaking, it is not easy to verify the conditions on the convergence of the iterative method for practical problems.

(c) If the iterative method fails to find a fixed point, we may employ GA to solve the following minimization problem,

$$\min R(\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m) = \sum_{i=1}^{m} \|\boldsymbol{y}_i - r_i(\boldsymbol{y}_{-i})\| \tag{2.14}$$

If an array $(\boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*)$ makes $R(\boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*) = 0$, then $\boldsymbol{y}_i^* = r_i(\boldsymbol{y}_{-i}^*)$, $i = 1, 2, \cdots, m$ and $(\boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*)$ must be a solution of (2.12). If not, then the system of equations (2.12) is inconsistent. In other words, there is no Nash equilibrium of followers in the given bilevel programming. Although this method can deal with general problem, it is a slow way to find a Nash equilibrium.

After obtaining the Nash equilibrium for each given decision vector of the leader, we may compute the objective value of the leader for each given control vector according to the Nash equilibrium. Hence we may employ the GA to search for the Stackelberg-Nash equilibrium.

Now we consider a bilevel programming with three followers in which the leader has a decision vector (x_1, x_2, x_3) and the three followers have decision

vector $(y_{11}, y_{12}, y_{21}, y_{22}, y_{31}, y_{32})$,

$$\begin{cases} \max\limits_{x_1,x_2,x_3} y_{11}^* y_{12}^* \sin x_1 + 2y_{21}^* y_{22}^* \sin x_2 + 3y_{31}^* y_{32}^* \sin x_3 \\ \text{subject to:} \\ \quad x_1 + x_2 + x_3 \le 10,\ x_1 \ge 0,\ x_2 \ge 0,\ x_3 \ge 0 \\ \quad (y_{11}^*, y_{12}^*, y_{21}^*, y_{22}^*, y_{31}^*, y_{32}^*) \text{ solves the problems} \\ \quad \begin{cases} \max\limits_{y_{11},y_{12}} y_{11} \sin y_{12} + y_{12} \sin y_{11} \\ \text{subject to:} \\ \quad\quad y_{11} + y_{12} \le x_1,\ y_{11} \ge 0,\ y_{12} \ge 0 \end{cases} \\ \quad \begin{cases} \max\limits_{y_{21},y_{22}} y_{21} \sin y_{22} + y_{22} \sin y_{21} \\ \text{subject to:} \\ \quad\quad y_{21} + y_{22} \le x_2,\ y_{21} \ge 0,\ y_{22} \ge 0 \end{cases} \\ \quad \begin{cases} \max\limits_{y_{31},y_{32}} y_{31} \sin y_{32} + y_{32} \sin y_{31} \\ \text{subject to:} \\ \quad\quad y_{31} + y_{32} \le x_3,\ y_{31} \ge 0,\ y_{32} \ge 0. \end{cases} \end{cases}$$

A run of GA with 1000 generations shows that the Stackelberg-Nash equilibrium is

$$\begin{aligned} (x_1^*, x_2^*, x_3^*) &= (0.000, 1.936, 8.064), \\ (y_{11}^*, y_{12}^*) &= (0.000, 0.000), \\ (y_{21}^*, y_{22}^*) &= (0.968, 0.968), \\ (y_{31}^*, y_{32}^*) &= (1.317, 6.747) \end{aligned}$$

with optimal objective values

$$\begin{aligned} y_{11}^* y_{12}^* \sin x_1^* + 2y_{21}^* y_{22}^* \sin x_2^* + 3y_{31}^* y_{32}^* \sin x_3^* &= 27.822, \\ y_{11}^* \sin y_{12}^* + y_{12}^* \sin y_{11}^* &= 0.000, \\ y_{21}^* \sin y_{22}^* + y_{22}^* \sin y_{21}^* &= 1.595, \\ y_{31}^* \sin y_{32}^* + y_{32}^* \sin y_{31}^* &= 7.120. \end{aligned}$$

Chapter 3

Neural Networks

Neural network (NN), inspired by the current understanding of biological NN, is a class of adaptive systems consisting of a number of simple processing elements, called neurons, that are interconnected to each other in a feedforward way. Although NN can perform some human brain-like tasks, there is still a huge gap between biological and artificial NN. An important contribution of NN is the ability to learn to perform operations, not only for inputs exactly like the training data, but also for new data that may be incomplete or noisy. NN has also the benefit of easy modification by retraining with an updated data set. For our purpose, the significant advantage of NN is the speed of operation after it is trained.

3.1 Basic Concepts

The artificial neuron simulates the behavior of the biological neuron to make a simple operation of a weighted sum of the incoming signals as

$$y = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n \tag{3.1}$$

where $x_1, x_2, \cdots, x_n$ are inputs, $w_0, w_1, w_2, \cdots, w_n$ are weights, and y is output. Figure 3.1 illustrates a neuron.

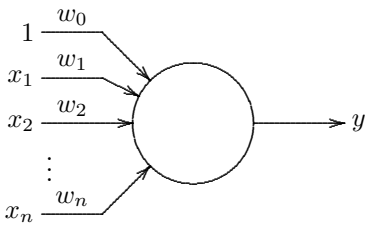

Figure 3.1: An Artificial Neuron

In most applications, we define a memoryless nonlinear function σ as an activation function to change the output to

$$y = \sigma\left(w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n\right). \tag{3.2}$$

The choice of the activation functions depends on the application area. In this book we employ the sigmoid function defined as

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.3}$$

whose derivative is

$$\sigma'(x) = \frac{e^{-x}}{(1 + e^{-x})^2}. \tag{3.4}$$

They are shown in Figure 3.2.

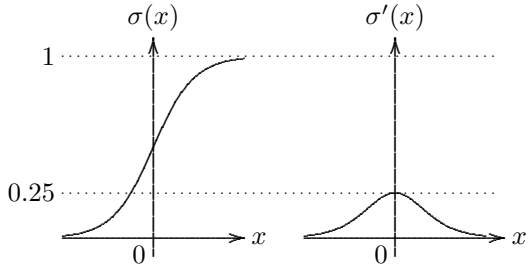

Figure 3.2: Sigmoid Function and Derivative

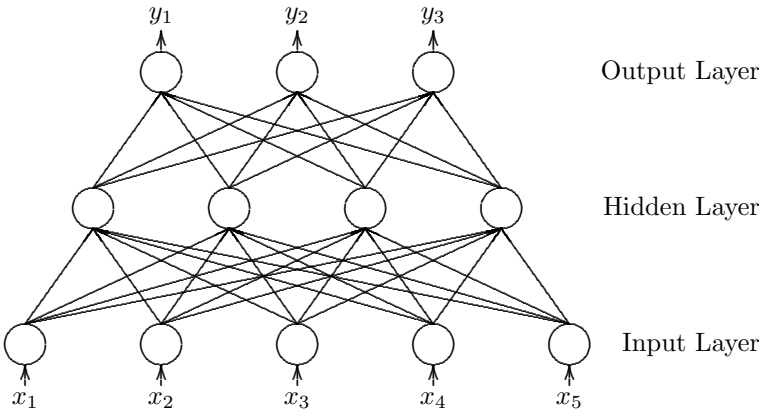

Figure 3.3: A Neural Network

Let us consider an NN with one hidden layer, in which there are n neurons in the input layer, m neurons in the output layer, and p neurons in the hidden

layer which is pictured in Figure 3.3. Then the outputs of the neurons in the hidden layer are

$$x_i^1 = \sigma\left(\sum_{j=1}^{n} w_{ij}^0 x_j + w_{i0}^0\right), \quad i = 1, 2, \cdots, p. \tag{3.5}$$

Thus the outputs of the neurons in the output layer are

$$y_i = \sum_{j=1}^{p} w_{ij}^1 x_j^1 + w_{i0}^1, \quad i = 1, 2, \cdots, m. \tag{3.6}$$

The coefficients $w_{ij}^0, i = 1, 2, \cdots, p, j = 0, 1, \cdots, n$ in (3.5) and $w_{ij}^1, i = 1, 2, \cdots, m, j = 0, 1, \cdots, p$ in (3.6) are called the network weights.

3.2 Function Approximation

NN is clearly a nonlinear mapping from the input space to the output space. It has been proved that any continuous nonlinear function can be approximated arbitrarily well over a compact set by an NN consisting of one hidden layer provided that there are infinite neurons in the hidden layer.

Assume that $f : \Re^n \to \Re^m$ is a continuous nonlinear function. We hope to train an NN to approximate the function $f(\boldsymbol{x})$. For an NN with a fixed number of neurons and architecture, the network weights may be arranged into a vector $\boldsymbol{w}$. Let $F(\boldsymbol{x}, \boldsymbol{w})$ be the output of mapping implemented by the NN.

The training process is to find an appropriate weight vector $\boldsymbol{w}$ that provides the best possible approximation of $f(\boldsymbol{x})$. Let $\{(\boldsymbol{x}_i^*, \boldsymbol{y}_i^*)|i = 1, 2, \cdots, N\}$ be a set of training input-output data on $f(\boldsymbol{x})$. We wish to choose a weight vector $\boldsymbol{w}$ so that the output $F(\boldsymbol{x}, \boldsymbol{w})$ is "close" to the desired output $\boldsymbol{y}_i^*$ for the input $\boldsymbol{x}_i^*$. That is, the training process is to find the weight vector $\boldsymbol{w}$ that minimizes the following error function,

$$Err(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{N} \|F(\boldsymbol{x}_i^*, \boldsymbol{w}) - \boldsymbol{y}_i^*\|^2. \tag{3.7}$$

3.3 Neuron Number Determination

Since the function f is a mapping from $\Re^n$ to $\Re^m$, the number of input neurons is always n, and the number of output neurons is always m. Thus the main problem is to determine the best number of hidden neurons.

Although any continuous function can be approximated with an arbitrary accuracy by an NN with infinite neurons in the hidden layer, it is practically impossible to have infinite hidden neurons. On the one hand, too few hidden

neurons make the NN lack of generalization ability. On the other hand, too many hidden neurons increase the training time and response time of the trained NN. A lot of methods have been proposed for determining the number of hidden neurons, some add hidden neurons, and other delete hidden neurons during the training process.

3.4 Backpropagation Algorithm

The values of the weights represent all memory of the NN. During the training phase of an NN, the values of weights are continuously updated by the training process until some termination criterion is met. In other words, learning in NN is a modification process of the values of weights so as to bring the mapping implemented by the NN as close as possible to a desired mapping. It may also be viewed as an optimization problem of selecting weights to minimize the error between the target output and the actual output.

Backpropagation algorithm is an effective learning algorithm. It is essentially a gradient method. Here we introduce the backpropagation algorithm for the NN with one hidden layer. Assume that there are N samples $(x^*_{k1}, x^*_{k2}, \cdots, x^*_{kn}; y^*_{k1}, y^*_{k2}, \cdots, y^*_{km})$ for $k = 1, 2, \cdots, N$.

We first initialize the weight vector $\boldsymbol{w}$ at random, set $\Delta w^1_{ij} = 0$ for $i = 1, 2, \cdots, m$, $j = 0, 1, \cdots, p$, $\Delta w^0_{ij} = 0$ for $i = 1, 2, \cdots, p$ and $j = 0, 1, \cdots, n$, and the adaptive parameter $\lambda = 1$. Then we adjust the weights by an on-line learning process. When the k-th sample is used, the outputs of the hidden neurons are

$$x^1_{ki} = \sigma\left(\sum_{j=1}^{n} w^0_{ij} x^*_{kj} + w^0_{i0}\right), \quad i = 1, 2, \cdots, p,$$

and the outputs of the NN are

$$y_{ki} = \sum_{j=1}^{p} w^1_{ij} x^1_{kj} + w^1_{i0}, \quad i = 1, 2, \cdots, m.$$

In order to speed up the learning process, we use an improved error function

$$E_k = \frac{1}{2} \sum_{i=1}^{m} \left[\lambda (y^*_{ki} - y_{ki})^2 + (1 - \lambda)\Phi(y^*_{ki} - y_{ki})\right] \tag{3.8}$$

where $\Phi(x) = \ln(\cosh(\beta x))/\beta$ and β is a constant, for example, $\beta = 4/3$.

Thus the equations for weight change are given as follows: For the hidden-output weights $w^1_{ij}, i = 1, 2, \cdots, m, j = 0, 1, \cdots, p$, we have

$$\Delta w^1_{ij} \leftarrow -\alpha \frac{\partial E_k}{\partial w^1_{ij}} + \eta \Delta w^1_{ij} = \alpha C^1_i x^1_{kj} + \eta \Delta w^1_{ij} \tag{3.9}$$

where

$$C_i^1 = \lambda(y_{ki}^* - y_{ki}) + (1-\lambda)\tanh(\beta(y_{ki}^* - y_{ki})), \quad x_{k0}^1 = 1.$$

For the input-hidden weights $w_{ij}^0, i = 1, 2, \cdots, p, j = 0, 1, \cdots, n$, we have

$$\Delta w_{ij}^0 \leftarrow -\alpha \frac{\partial E_k}{\partial w_{ij}^0} + \eta \Delta w_{ij}^0 = \alpha C_i^0 x_{kj}^* + \eta \Delta w_{ij}^0 \tag{3.10}$$

where

$$C_i^0 = \left(1 - \left(x_{ki}^1\right)^2\right) \sum_{l=1}^{m} C_l^1 w_{li}^1, \quad x_{k0}^* = 1,$$

α and η are numbers between 0 and 1, for example, $\alpha = 0.05$, $\eta = 0.01$.

After training the NN one time over all input-output data, we calculate the total error $E = E_1 + E_2 + \cdots + E_N$. If the error E is less than a predetermined precision E_0 (for example, 0.05), then the NN is considered trained well. Otherwise, we set $\lambda = \exp(-1/E^2)$ and repeat the learning process until $E < E_0$.

Algorithm 3.1 (Backpropagation Algorithm)
Step 1. Initialize weight vector $\boldsymbol{w}$, and set $\lambda = 1$ and $k = 0$.
Step 2. $k \leftarrow k + 1$.
Step 3. Adjust the weight vector $\boldsymbol{w}$ according to (3.9) and (3.10).
Step 4. Calculate the error E_k according to (3.8).
Step 5. If $k < N$, go to Step 2.
Step 6. Set $E = E_1 + E_2 + \cdots + E_N$.
Step 7. If $E > E_0$, then $k = 0$, $\lambda = \exp(-1/E^2)$ and go to Step 2.
Step 8. End.

3.5 Numerical Experiments

The NN architecture used in this section is the network with one hidden layer which is pictured in Figure 3.3. The NN will be used for approximating some continuous functions.

Example 3.1: Let us design an NN to approximate the continuous function,

$$f(x_1, x_2, x_3, x_4) = \sin x_1 + \sin x_2 + \sin x_3 + \sin x_4$$

defined on $[0, 2\pi]^4$. In order to approximate the function $f(x_1, x_2, x_3, x_4)$ by an NN, we generate 3000 input-output data. Then we train an NN (4 input neurons, 10 hidden neurons, 1 output neuron) by the backpropagation algorithm. The sum-squared error of the trained NN is 0.51, and the average error is 0.01.

Example 3.2: Consider the vector-valued continuous function,

$$f(x_1, x_2, x_3, x_4, x_5, x_6) = \begin{pmatrix} x_1 \ln x_2 + x_2 \ln x_3 \\ x_3 \ln x_4 + x_4 \ln x_5 \\ x_5 \ln x_6 + x_6 \ln x_1 \end{pmatrix}$$

defined on the region $[1, 5]^6$. We generate 3000 training data for the function $f(\boldsymbol{x})$. Then we train an NN (6 input neurons, 15 hidden neurons, 3 output neurons) to approximate the function. A run of backpropagation algorithm shows that the sum-squared error of the trained NN is 6.16, and the average error is 0.05.

Example 3.3: Consider the function

$$f(x_1, x_2, x_3, x_4) = \frac{x_1}{1 + x_1} + \frac{x_2}{1 + x_2} + \frac{x_3}{1 + x_3} + \frac{x_4}{1 + x_4}$$

defined on $[0, 2]$. Assume that the input-output data for the function $f(\boldsymbol{x})$ are randomly generated on $[0, 2]$ with a uniformly distributed noise $\mathcal{U}(-a, a)$, where $\mathcal{U}(-a, a)$ represents the uniformly distributed variable on the interval $[-a, a]$. That is, for each input $\boldsymbol{x}$, the output $y = f(\boldsymbol{x}) + \mathcal{U}(-a, a)$.

We produce 2000 input-output data $\{(\boldsymbol{x}_i^*, y_i^*) | i = 1, 2, \cdots, 2000\}$ with noise $\mathcal{U}(-a, a)$ for the function $f(\boldsymbol{x})$. The backpropagation algorithm may obtain an NN (4 input neurons, 6 hidden neurons, 1 output neuron) to approximate the function $f(\boldsymbol{x})$ according to the noise data.

Let $F(\boldsymbol{x}, \boldsymbol{w}^*)$ be the output of mapping implemented by the NN. We generate 1000 test noise data $\{(\boldsymbol{x}_i^*, y_i^*) | i = 2001, 2002, \cdots, 3000\}$, then we have the errors shown in Table 3.1.

Table 3.1: Sum-Squared Errors

Noise	$\frac{1}{2}\sum_{i=2001}^{3000} \lvert F(\boldsymbol{x}_i^*, \boldsymbol{w}^*) - f(\boldsymbol{x}_i^*)\rvert^2$	$\frac{1}{2}\sum_{i=2001}^{3000} \lvert y_i^* - f(\boldsymbol{x}_i^*)\rvert^2$
$\mathcal{U}(-0.05, 0.05)$	0.362	0.389
$\mathcal{U}(-0.10, 0.10)$	1.333	1.643
$\mathcal{U}(-0.20, 0.20)$	4.208	6.226
$\mathcal{U}(-0.30, 0.30)$	7.306	14.01
$\mathcal{U}(-0.40, 0.40)$	14.74	24.90

Note that the errors in the first column are less than that in the second column. This means that the trained NN can compensate for the error of the noise training data.

Chapter 4

Stochastic Programming

With the requirement of considering randomness, different types of stochastic programming have been developed to suit the different purposes of management. The first type of stochastic programming is the *expected value model*, which optimizes the expected objective functions subject to some expected constraints. The second, *chance-constrained programming*, was pioneered by Charnes and Cooper [38] as a means of handling uncertainty by specifying a confidence level at which it is desired that the stochastic constraint holds. After that, Liu [175] generalized chance-constrained programming to the case with not only stochastic constraints but also stochastic objectives. In practice, there usually exist multiple events in a complex stochastic decision system. Sometimes the decision-maker wishes to maximize the chance functions of satisfying these events. In order to model this type of problem, Liu [167] provided a theoretical framework of the third type of stochastic programming, called *dependent-chance programming*.

This chapter will give some basic concepts of probability theory and introduce stochastic programming. A hybrid intelligent algorithm is also documented.

4.1 Random Variables

Definition 4.1 *Let Ω be a nonempty set, and $\mathcal{A}$ a σ-algebra of subsets (called events) of Ω. The set function* Pr *is called a probability measure if*
Axiom 1. *(Normality)* $\Pr\{\Omega\} = 1$*;*
Axiom 2. *(Nonnegativity)* $\Pr\{A\} \geq 0$ *for any* $A \in \mathcal{A}$*;*
Axiom 3. *(Countable Additivity) For every countable sequence of mutually disjoint events* $\{A_i\}$*, we have*

$$\Pr\left\{\bigcup_{i=1}^{\infty} A_i\right\} = \sum_{i=1}^{\infty} \Pr\{A_i\}. \tag{4.1}$$

Example 4.1: Let $\Omega = \{\omega_1, \omega_2, \cdots\}$, and let $\mathcal{A}$ be the power set of Ω. Assume that $p_1, p_2, \cdots$ are nonnegative numbers such that $p_1 + p_2 + \cdots = 1$. Define a set function on $\mathcal{A}$ as

$$\Pr\{A\} = \sum_{\omega_i \in A} p_i, \quad A \in \mathcal{A}. \tag{4.2}$$

Then Pr is a probability measure.

Example 4.2: Let $\Omega = [0, 1]$ and let $\mathcal{A}$ be the Borel algebra over Ω. If Pr is the Lebesgue measure, then Pr is a probability measure.

Theorem 4.1 *Let Ω be a nonempty set, $\mathcal{A}$ a σ-algebra over Ω, and* Pr *a probability measure. Then* $\Pr\{\emptyset\} = 0$ *and* $0 \leq \Pr\{A\} \leq 1$ *for any event A.*

Definition 4.2 *Let Ω be a nonempty set, $\mathcal{A}$ a σ-algebra of subsets of Ω, and* Pr *a probability measure. Then the triplet $(\Omega, \mathcal{A}, \Pr)$ is called a probability space.*

Definition 4.3 *A random variable is a measurable function from a probability space $(\Omega, \mathcal{A}, \Pr)$ to the set of real numbers, i.e., for any Borel set B of real numbers, the set*

$$\{\xi \in B\} = \{\omega \in \Omega \mid \xi(\omega) \in B\} \tag{4.3}$$

is an event.

An n-dimensional random vector is defined as a measurable function from a probability space $(\Omega, \mathcal{A}, \Pr)$ to the set of n-dimensional real vectors. It has been proved that $(\xi_1, \xi_2, \cdots, \xi_n)$ is a random vector if and only if $\xi_1, \xi_2, \cdots, \xi_n$ are random variables.

Definition 4.4 *Let $f : \Re^n \to \Re$ be a measurable function, and $\xi_1, \xi_2, \cdots, \xi_n$ random variables defined on the probability space $(\Omega, \mathcal{A}, \Pr)$. Then $\xi = f(\xi_1, \xi_2, \cdots, \xi_n)$ is a random variable defined by*

$$\xi(\omega) = f(\xi_1(\omega), \xi_2(\omega), \cdots, \xi_n(\omega)), \quad \forall \omega \in \Omega. \tag{4.4}$$

Probability Distribution

Definition 4.5 *The probability distribution Φ: $\Re \to [0, 1]$ of a random variable ξ is defined by*

$$\Phi(x) = \Pr\left\{\omega \in \Omega \mid \xi(\omega) \leq x\right\}. \tag{4.5}$$

That is, $\Phi(x)$ is the probability that the random variable ξ takes a value less than or equal to x.

Definition 4.6 *The probability density function ϕ: $\Re \to [0, +\infty)$ of a random variable ξ is a function such that*

$$\Phi(x) = \int_{-\infty}^{x} \phi(y)\mathrm{d}y \tag{4.6}$$

holds for all $x \in \Re$, where Φ is the probability distribution of the random variable ξ.

Uniform Distribution: A random variable ξ has a uniform distribution if its probability density function is

$$\phi(x) = \begin{cases} \dfrac{1}{b-a}, & \text{if } a \le x \le b \\ 0, & \text{otherwise} \end{cases} \tag{4.7}$$

denoted by $\mathcal{U}(a, b)$, where a and b are given real numbers with $a < b$.

Exponential Distribution: A random variable ξ has an exponential distribution if its probability density function is

$$\phi(x) = \begin{cases} \dfrac{1}{\beta} \exp\left(-\dfrac{x}{\beta}\right), & \text{if } x \ge 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.8}$$

denoted by $\mathcal{EXP}(\beta)$, where β is a positive number.

Normal Distribution: A random variable ξ has a normal distribution if its probability density function is

$$\phi(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \quad x \in \Re \tag{4.9}$$

denoted by $\mathcal{N}(\mu, \sigma^2)$, where μ and σ are real numbers.

Theorem 4.2 *Let ξ be a random variable whose probability density function ϕ exists. Then for any Borel set B of $\Re$, we have*

$$\Pr\{\xi \in B\} = \int_B \phi(y)\mathrm{d}y. \tag{4.10}$$

Independence

Definition 4.7 *The random variables $\xi_1, \xi_2, \cdots, \xi_m$ are said to be independent if*

$$\Pr\{\xi_i \in B_i, i = 1, 2, \cdots, m\} = \prod_{i=1}^{m} \Pr\{\xi_i \in B_i\} \tag{4.11}$$

for any Borel sets $B_1, B_2, \cdots, B_m$ of real numbers.

Let $\xi_1, \xi_2, \cdots, \xi_m$ be independent random variables with probability density functions $\phi_1, \phi_2, \cdots, \phi_m$, respectively, and $f : \Re^m \to \Re$ a measurable function. Then for any Borel set B of real numbers, the probability $\Pr\{f(\xi_1, \xi_2, \cdots, \xi_m) \in B\}$ is

$$\iint \cdots \int_{f(x_1,x_2,\cdots,x_m)\in B} \phi_1(x_1)\phi_2(x_2)\cdots\phi_m(x_m)\mathrm{d}x_1\mathrm{d}x_2\cdots\mathrm{d}x_m.$$

Expected Value

Definition 4.8 *Let ξ be a random variable. Then the expected value of ξ is defined by*

$$E[\xi] = \int_0^{+\infty} \Pr\{\xi \geq r\}\mathrm{d}r - \int_{-\infty}^{0} \Pr\{\xi \leq r\}\mathrm{d}r \tag{4.12}$$

provided that at least one of the two integrals is finite.

Let ξ and η be random variables with finite expected values. For any numbers a and b, it has been proved that $E[a\xi + b\eta] = aE[\xi] + bE[\eta]$. That is, the expected value operator has the linearity property.

Theorem 4.3 *Let ξ be a random variable whose probability density function ϕ exists. If the Lebesgue integral*

$$\int_{-\infty}^{+\infty} x\phi(x)\mathrm{d}x$$

is finite, then we have

$$E[\xi] = \int_{-\infty}^{+\infty} x\phi(x)\mathrm{d}x. \tag{4.13}$$

Example 4.3: Let ξ be a uniformly distributed random variable on the interval $[a, b]$. Then its expected value is $(a + b)/2$.

Example 4.4: Let ξ be an exponentially distributed random variable $\mathcal{EXP}(\beta)$. Then its expected value is β.

Example 4.5: Let ξ be a normally distributed random variable $\mathcal{N}(\mu, \sigma^2)$. Then its expected value is μ.

Critical Values

Definition 4.9 *Let ξ be a random variable, and $\alpha \in (0, 1]$. Then*

$$\xi_{\sup}(\alpha) = \sup\left\{r \;\middle|\; \Pr\{\xi \geq r\} \geq \alpha\right\} \tag{4.14}$$

is called the α-optimistic value of ξ; and

$$\xi_{\inf}(\alpha) = \inf\left\{r \;\middle|\; \Pr\{\xi \leq r\} \geq \alpha\right\} \tag{4.15}$$

is called the α-pessimistic value of ξ.

This means that the random variable ξ will reach upwards of the α-optimistic value $\xi_{\sup}(\alpha)$ at least α of time, and will be below the α-pessimistic value $\xi_{\inf}(\alpha)$ at least α of time.

Ranking Criteria

Let ξ and η be two random variables. Different from the situation of real numbers, there does not exist a natural ordership in a random world. Thus an important problem appearing in this area is how to rank random variables. Here we give four ranking criteria.

Expected Value Criterion: We say $\xi > \eta$ if and only if $E[\xi] > E[\eta]$, where E is the expected value operator of random variables.

Optimistic Value Criterion: We say $\xi > \eta$ if and only if, for some predetermined confidence level $\alpha \in (0, 1]$, we have $\xi_{\sup}(\alpha) > \eta_{\sup}(\alpha)$, where $\xi_{\sup}(\alpha)$ and $\eta_{\sup}(\alpha)$ are the α-optimistic values of ξ and η, respectively.

Pessimistic Value Criterion: We say $\xi > \eta$ if and only if, for some predetermined confidence level $\alpha \in (0, 1]$, we have $\xi_{\inf}(\alpha) > \eta_{\inf}(\alpha)$, where $\xi_{\inf}(\alpha)$ and $\eta_{\inf}(\alpha)$ are the α-pessimistic values of ξ and η, respectively.

Probability Criterion: We say $\xi > \eta$ if and only if $\Pr\{\xi \geq \overline{r}\} > \Pr\{\eta \geq \overline{r}\}$ for some predetermined level $\overline{r}$.

Random Number Generation

Random number generation is a very important issue in Monte Carlo simulation. Generally, let ξ be a random variable with a probability distribution $\Phi(\cdot)$. Since $\Phi(\cdot)$ is an increasing function, the inverse function $\Phi^{-1}(\cdot)$ is defined on $[0, 1]$. Assume that u is a uniformly distributed random variable on the interval $[0, 1]$. Then we have

$$\Pr\left\{\Phi^{-1}(u) \leq y\right\} = \Pr\{u \leq \Phi(y)\} = \Phi(y) \tag{4.16}$$

which proves that the variable $\xi = \Phi^{-1}(u)$ has the probability distribution $\Phi(\cdot)$. In order to get a random variable ξ with probability distribution $\Phi(\cdot)$, we can produce a uniformly distributed random variable u from the interval $[0, 1]$, and ξ is assigned to be $\Phi^{-1}(u)$. The above process is called the *inverse transform method*. But for the main known distributions, instead of using the inverse transform method, we have direct generating processes. For detailed expositions, the interested readers may consult Fishman [69], Law and Kelton [149], Bratley et al. [23], Rubinstein [268], and Liu [182]. Here we give some generating methods for probability distributions frequently used in this book.

The subfunction of generating pseudorandom numbers has been provided by the C library for any type of computer, defined as

int rand(void)

which produces a pseudorandom integer between 0 and RAND_MAX, where RAND_MAX is defined in stdlib.h as $2^{15} - 1$. Thus the uniform distribution, exponential distribution, and normal distribution can be generated by the following way:

Algorithm 4.1 (Uniform Distribution $\mathcal{U}(a, b)$)
Step 1. $u = \text{rand}(\,)$.
Step 2. $u \leftarrow u/\text{RAND\_MAX}$.
Step 3. Return $a + u(b - a)$.

Algorithm 4.2 (Exponential Distribution $\mathcal{EXP}(\beta)$)
Step 1. Generate u from $\mathcal{U}(0, 1)$.
Step 2. Return $-\beta \ln(u)$.

Algorithm 4.3 (Normal Distribution $\mathcal{N}(\mu, \sigma^2)$)
Step 1. Generate μ_1 and μ_2 from $\mathcal{U}(0, 1)$.
Step 2. $y = [-2\ln(\mu_1)]^{\frac{1}{2}} \sin(2\pi\mu_2)$.
Step 3. Return $\mu + \sigma y$.

4.2 Expected Value Model

The first type of stochastic programming is the so-called *expected value model* (EVM), which optimizes some expected objective function subject to some expected constraints, for example, minimizing expected cost, maximizing expected profit, and so forth.

Now let us recall the well-known newsboy problem 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 $\$c$/newspaper every day. It is known that the selling price is $\$a$/newspaper. However, if the newspapers are not sold at the end of the day, then the newspapers have a small value of $\$b$/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 $x \leq \xi$. Thus the profit of the newsboy should be

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

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 $E[f(x,\xi)]$. 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.,

$$\begin{cases} \max\ E[f(x,\xi)] \\ \text{subject to:} \\ \qquad x \geq 0, \quad \text{integer.} \end{cases}$$

This is a typical example of EVM. Generally, if we want to find a decision with maximum expected return subject to some expected constraints, then we have the following EVM,

$$\begin{cases} \max\ E[f(\boldsymbol{x},\boldsymbol{\xi})] \\ \text{subject to:} \\ \qquad E[g_j(\boldsymbol{x},\boldsymbol{\xi})] \leq 0, \quad j=1,2,\cdots,p \end{cases} \tag{4.17}$$

where $\boldsymbol{x}$ is a decision vector, $\boldsymbol{\xi}$ is a stochastic vector, $f(\boldsymbol{x},\boldsymbol{\xi})$ is the return function, $g_j(\boldsymbol{x},\boldsymbol{\xi})$ are stochastic constraint functions for $j=1,2,\cdots,p$.

Definition 4.10 *A solution $\boldsymbol{x}$ is feasible if and only if $E[g_j(\boldsymbol{x},\boldsymbol{\xi})] \leq 0$ for $j=1,2,\cdots,p$. A feasible solution $\boldsymbol{x}^*$ is an optimal solution to EVM (4.17) if $E[f(\boldsymbol{x}^*,\boldsymbol{\xi})] \geq E[f(\boldsymbol{x},\boldsymbol{\xi})]$ for any feasible solution $\boldsymbol{x}$.*

In many cases, there are multiple objectives. Thus we have to employ the following expected value multiobjective programming (EVMOP),

$$\begin{cases} \max\ \big[E[f_1(\boldsymbol{x},\boldsymbol{\xi})], E[f_2(\boldsymbol{x},\boldsymbol{\xi})], \cdots, E[f_m(\boldsymbol{x},\boldsymbol{\xi})]\big] \\ \text{subject to:} \\ \qquad E[g_j(\boldsymbol{x},\boldsymbol{\xi})] \leq 0, \quad j=1,2,\cdots,p \end{cases} \tag{4.18}$$

where $f_i(\boldsymbol{x},\boldsymbol{\xi})$ are return functions for $i=1,2,\cdots,m$.

Definition 4.11 *A feasible solution $\boldsymbol{x}^*$ is said to be a Pareto solution to EVMOP (4.18) if there is no feasible solution $\boldsymbol{x}$ such that*

$$E[f_i(\boldsymbol{x},\boldsymbol{\xi})] \geq E[f_i(\boldsymbol{x}^*,\boldsymbol{\xi})], \quad i=1,2,\cdots,m \tag{4.19}$$

and $E[f_j(\boldsymbol{x},\boldsymbol{\xi})] > E[f_j(\boldsymbol{x}^*,\boldsymbol{\xi})]$ for at least one index j.

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

$$\begin{cases} \min \sum_{j=1}^{l} P_j \sum_{i=1}^{m} (u_{ij} d_i^+ \vee 0 + v_{ij} d_i^- \vee 0) \\ \text{subject to:} \\ \qquad E[f_i(\boldsymbol{x},\boldsymbol{\xi})] - b_i = d_i^+, \quad i=1,2,\cdots,m \\ \qquad b_i - E[f_i(\boldsymbol{x},\boldsymbol{\xi})] = d_i^-, \quad i=1,2,\cdots,m \\ \qquad E[g_j(\boldsymbol{x},\boldsymbol{\xi})] \leq 0, \qquad\quad j=1,2,\cdots,p \end{cases} \tag{4.20}$$

where P_j is the preemptive priority factor which expresses the relative importance of various goals, $P_j \gg P_{j+1}$, 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, $d_i^+ \vee 0$ is the positive deviation from the target of goal i, $d_i^- \vee 0$ is the negative deviation from the target of goal i, 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.

4.3 Chance-Constrained Programming

As the second type of stochastic programming developed by Charnes and Cooper [38], chance-constrained programming (CCP) offers a powerful means of modeling stochastic decision systems with assumption that the stochastic constraints will hold at least α of time, where α is referred to as the *confidence level* provided as an appropriate safety margin by the decision-maker. After that, Liu [175] generalized CCP to the case with not only stochastic constraints but also stochastic objectives.

Assume that $\boldsymbol{x}$ is a decision vector, $\boldsymbol{\xi}$ is a stochastic vector, $f(\boldsymbol{x}, \boldsymbol{\xi})$ is a return function, and $g_j(\boldsymbol{x}, \boldsymbol{\xi})$ are stochastic constraint functions, $j = 1, 2, \cdots, p$. Since the stochastic constraints $g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, j = 1, 2, \cdots, p$ do not define a deterministic feasible set, it is desired that the stochastic constraints hold with a confidence level α. Thus we have a chance constraint as follows,

$$\Pr\{g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, j = 1, 2, \cdots, p\} \geq \alpha \tag{4.21}$$

which is called a joint chance constraint.

Definition 4.12 *A point $\boldsymbol{x}$ is called feasible if and only if the probability measure of the event $\{g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, j = 1, 2, \cdots, p\}$ is at least α.*

In other words, the constraints will be violated at most $(1 - \alpha)$ of time. Sometimes, the joint chance constraint is separately considered as

$$\Pr\{g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0\} \geq \alpha_j, \quad j = 1, 2, \cdots, p \tag{4.22}$$

which is referred to as a separate chance constraint.

Maximax Chance-Constrained Programming

In a stochastic environment, in order to maximize the optimistic return with a given confidence level subject to some chance constraint, Liu [175] gave the

following CCP:

$$\begin{cases} \max\limits_{\boldsymbol{x}} \max\limits_{\overline{f}} \overline{f} \\ \text{subject to:} \\ \quad \Pr\left\{f(\boldsymbol{x},\boldsymbol{\xi}) \geq \overline{f}\right\} \geq \beta \\ \quad \Pr\left\{g_j(\boldsymbol{x},\boldsymbol{\xi}) \leq 0, j=1,2,\cdots,p\right\} \geq \alpha \end{cases} \tag{4.23}$$

where α and β are the predetermined confidence levels, and $\max \overline{f}$ is the β-optimistic return.

In practice, we may have multiple objectives. Thus we have to employ the following chance-constrained multiobjective programming (CCMOP),

$$\begin{cases} \max\limits_{\boldsymbol{x}} \left[\max\limits_{\overline{f}_1} \overline{f}_1, \max\limits_{\overline{f}_2} \overline{f}_2, \cdots, \max\limits_{\overline{f}_m} \overline{f}_m\right] \\ \text{subject to:} \\ \quad \Pr\left\{f_i(\boldsymbol{x},\boldsymbol{\xi}) \geq \overline{f}_i\right\} \geq \beta_i, \quad i=1,2,\cdots,m \\ \quad \Pr\left\{g_j(\boldsymbol{x},\boldsymbol{\xi}) \leq 0\right\} \geq \alpha_j, \quad j=1,2,\cdots,p \end{cases} \tag{4.24}$$

where $\alpha_1, \alpha_2, \cdots, \alpha_p, \beta_1, \beta_2, \cdots, \beta_m$ are the predetermined confidence levels, and $\max \overline{f}_i$ are the β_i-optimistic values to the ith return functions $f_i(\boldsymbol{x},\boldsymbol{\xi})$, $i=1,2,\cdots,m$, respectively.

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

$$\begin{cases} \min \sum\limits_{j=1}^{l} P_j \sum\limits_{i=1}^{m} \left(u_{ij}\left(\min\limits_{d_i^+} d_i^+ \vee 0\right) + v_{ij}\left(\min\limits_{d_i^-} d_i^- \vee 0\right)\right) \\ \text{subject to:} \\ \quad \Pr\left\{f_i(\boldsymbol{x},\boldsymbol{\xi}) - b_i \leq d_i^+\right\} \geq \beta_i^+, \quad i=1,2,\cdots,m \\ \quad \Pr\left\{b_i - f_i(\boldsymbol{x},\boldsymbol{\xi}) \leq d_i^-\right\} \geq \beta_i^-, \quad i=1,2,\cdots,m \\ \quad \Pr\left\{g_j(\boldsymbol{x},\boldsymbol{\xi}) \leq 0\right\} \geq \alpha_j, \quad j=1,2,\cdots,p \end{cases} \tag{4.25}$$

where P_j is the preemptive priority factor which expresses the relative importance of various goals, $P_j \gg P_{j+1}$, 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, $\min d_i^+ \vee 0$ is the β_i^+-optimistic positive deviation from the target of goal i, $\min d_i^- \vee 0$ is the β_i^--optimistic negative deviation from the target of goal i, 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 4.1: In a deterministic goal programming, at most one of positive deviation and negative deviation takes a positive value. However, for a CCGP, it is possible that both of them are positive.

Minimax Chance-Constrained Programming

In a stochastic environment, in order to maximize the pessimistic return with a given confidence level subject to some chance constraint, Liu [182] provided the following minimax CCP model:

$$\begin{cases} \max\limits_{\boldsymbol{x}} \min\limits_{\overline{f}} \overline{f} \\ \text{subject to:} \\ \quad \Pr\left\{f(\boldsymbol{x}, \boldsymbol{\xi}) \leq \overline{f}\right\} \geq \beta \\ \quad \Pr\left\{g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, j = 1, 2, \cdots, p\right\} \geq \alpha \end{cases} \tag{4.26}$$

where α and β are the given confidence levels, and $\min \overline{f}$ is the β-pessimistic return.

If there are multiple objectives, then we may employ the following minimax CCMOP,

$$\begin{cases} \max\limits_{\boldsymbol{x}} \left[\min\limits_{\overline{f}_1} \overline{f}_1,\ \min\limits_{\overline{f}_2} \overline{f}_2, \cdots, \min\limits_{\overline{f}_m} \overline{f}_m\right] \\ \text{subject to:} \\ \quad \Pr\left\{f_i(\boldsymbol{x}, \boldsymbol{\xi}) \leq \overline{f}_i\right\} \geq \beta_i, \quad i = 1, 2, \cdots, m \\ \quad \Pr\left\{g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0\right\} \geq \alpha_j, \quad j = 1, 2, \cdots, p \end{cases} \tag{4.27}$$

where α_j and β_i are confidence levels, and $\min \overline{f}_i$ are the β_i-pessimistic values to the return functions $f_i(\boldsymbol{x}, \boldsymbol{\xi})$, $i = 1, 2, \cdots, m$, $j = 1, 2, \cdots, p$, respectively.

We can also formulate a stochastic decision system as a minimax CCGP according to the priority structure and target levels set by the decision-maker:

$$\begin{cases} \min\limits_{\boldsymbol{x}} \sum\limits_{j=1}^{l} P_j \sum\limits_{i=1}^{m} \left[u_{ij} \left(\max\limits_{d_i^+} d_i^+ \vee 0\right) + v_{ij} \left(\max\limits_{d_i^-} d_i^- \vee 0\right)\right] \\ \text{subject to:} \\ \quad \Pr\left\{f_i(\boldsymbol{x}, \boldsymbol{\xi}) - b_i \geq d_i^+\right\} \geq \beta_i^+, \quad i = 1, 2, \cdots, m \\ \quad \Pr\left\{b_i - f_i(\boldsymbol{x}, \boldsymbol{\xi}) \geq d_i^-\right\} \geq \beta_i^-, \quad i = 1, 2, \cdots, m \\ \quad \Pr\left\{g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0\right\} \geq \alpha_j, \qquad j = 1, 2, \cdots, p \end{cases} \tag{4.28}$$

where P_j is the preemptive priority factor which expresses the relative importance of various goals, $P_j \gg P_{j+1}$, 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, $\max d_i^+ \vee 0$ is the β_i^+-pessimistic positive deviation from the target of goal i, $\max d_i^- \vee 0$ is the β_i^--pessimistic negative deviation from the target of goal i, 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.

Deterministic Equivalents

The traditional solution methods require conversion of the chance constraints to their respective deterministic equivalents. As we know, this process is usually hard to perform and only successful for some special cases. Let us consider the following form of chance constraint,

$$\Pr\{g(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0\} \geq \alpha. \tag{4.29}$$

It is clear that

(a) the chance constraints (4.22) are a set of form (4.29);

(b) the stochastic objective constraint $\Pr\{f(\boldsymbol{x}, \boldsymbol{\xi}) \geq \overline{f}\} \geq \beta$ coincides with the form (4.29) by defining $g(\boldsymbol{x}, \boldsymbol{\xi}) = \overline{f} - f(\boldsymbol{x}, \boldsymbol{\xi})$;

(c) the stochastic objective constraint $\Pr\{f(\boldsymbol{x}, \boldsymbol{\xi}) \leq \overline{f}\} \geq \beta$ coincides with the form (4.29) by defining $g(\boldsymbol{x}, \boldsymbol{\xi}) = f(\boldsymbol{x}, \boldsymbol{\xi}) - \overline{f}$;

(d) the stochastic goal constraints $\Pr\{b - f(\boldsymbol{x}, \boldsymbol{\xi}) \leq d^-\} \geq \beta$ and $\Pr\{f(\boldsymbol{x}, \boldsymbol{\xi}) - b \leq d^+\} \geq \beta$ coincide with the form (4.29) by defining $g(\boldsymbol{x}, \boldsymbol{\xi}) = b - f(\boldsymbol{x}, \boldsymbol{\xi}) - d^-$ and $g(\boldsymbol{x}, \boldsymbol{\xi}) = f(\boldsymbol{x}, \boldsymbol{\xi}) - b - d^+$, respectively; and

(e) the stochastic goal constraints $\Pr\{b - f(\boldsymbol{x}, \boldsymbol{\xi}) \geq d^-\} \geq \beta$ and $\Pr\{f(\boldsymbol{x}, \boldsymbol{\xi}) - b \geq d^+\} \geq \beta$ coincide with the form (4.29) by defining $g(\boldsymbol{x}, \boldsymbol{\xi}) = f(\boldsymbol{x}, \boldsymbol{\xi}) + d^- - b$ and $g(\boldsymbol{x}, \boldsymbol{\xi}) = b - f(\boldsymbol{x}, \boldsymbol{\xi}) + d^+$, respectively.

This section summarizes some known results.

Theorem 4.4 *Assume that the stochastic vector $\boldsymbol{\xi}$ degenerates to a random variable ξ with probability distribution Φ, and the function $g(\boldsymbol{x}, \boldsymbol{\xi})$ has the form $g(\boldsymbol{x}, \boldsymbol{\xi}) = h(\boldsymbol{x}) - \xi$. Then $\Pr\{g(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0\} \geq \alpha$ if and only if $h(\boldsymbol{x}) \leq K_\alpha$, where K_α is the maximal number such that $\Pr\{K_\alpha \leq \xi\} \geq \alpha$.*

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

$$\Pr\{h(\boldsymbol{x}) \leq \xi\} \geq \alpha. \tag{4.30}$$

For each given confidence level α ($0 < \alpha \leq 1$), let K_α be the maximal number (maybe multiple or $+\infty$) such that

$$\Pr\{K_\alpha \leq \xi\} \geq \alpha. \tag{4.31}$$

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

Remark 4.2: For a continuous random variable ξ, the equation $\Pr\{K_\alpha \leq \xi\} = 1 - \Phi(K_\alpha)$ always holds, and we have, by (4.31),

$$K_\alpha = \Phi^{-1}(1-\alpha) \tag{4.32}$$

where Φ^{-1} is the inverse function of Φ.

Example 4.6: Assume that we have the following chance constraint,

$$\begin{cases} \Pr\{3x_1 + 4x_2 \leq \xi_1\} \geq 0.8 \\ \Pr\left\{x_1^2 - x_2^3 \leq \xi_2\right\} \geq 0.9 \end{cases} \tag{4.33}$$

where ξ_1 is an exponentially distributed variable $\mathcal{EXP}(2)$ whose probability distribution is denoted by Φ_1, and ξ_2 is a normally distributed variable $\mathcal{N}(2,1)$ whose probability distribution is denoted by Φ_2. It follows from Theorem 4.4 that the chance constraint (4.33) is equivalent to

$$\begin{cases} 3x_1 + 4x_2 \leq \Phi_1^{-1}(1-0.8) = 0.446 \\ x_1^2 - x_2^3 \leq \Phi_2^{-1}(1-0.9) = 0.719. \end{cases}$$

Theorem 4.5 *Assume that the stochastic vector $\boldsymbol{\xi} = (a_1, a_2, \cdots, a_n, b)$ 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*

$$\sum_{i=1}^{n} E[a_i]x_i + \Phi^{-1}(\alpha)\sqrt{\sum_{i=1}^{n} V[a_i]x_i^2 + V[b]} \leq E[b] \tag{4.34}$$

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,

$$\Pr\left\{\sum_{i=1}^{n} a_ix_i \leq b\right\} \geq \alpha. \tag{4.35}$$

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

$$y = \sum_{i=1}^{n} a_ix_i - b$$

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

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

We note that

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

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

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

the chance constraint (4.35) is equivalent to

$$\Pr\left\{\eta \leq -\frac{\sum_{i=1}^{n} E[a_i]x_i - E[b]}{\sqrt{\sum_{i=1}^{n} V[a_i]x_i^2 + V[b]}}\right\} \geq \alpha \tag{4.36}$$

where η is the standardized normally distributed variable. Then the chance constraint (4.36) holds if and only if

$$\Phi^{-1}(\alpha) \leq -\frac{\sum_{i=1}^{n} E[a_i]x_i - E[b]}{\sqrt{\sum_{i=1}^{n} V[a_i]x_i^2 + V[b]}}. \tag{4.37}$$

That is, the deterministic equivalent of chance constraint is (4.34). The theorem is proved.

Example 4.7: Suppose that the chance constraint set has the following form,

$$\Pr\{a_1x_1 + a_2x_2 + a_2x_3 \leq b\} \geq 0.95 \tag{4.38}$$

where a_1, a_2, a_3, and b are normally distributed variables $\mathcal{N}(1,1)$, $\mathcal{N}(2,1)$, $\mathcal{N}(3,1)$, and $\mathcal{N}(4,1)$, respectively. Then the formula (4.34) yields the deterministic equivalent of (4.38) as follows,

$$x_1 + 2x_2 + 3x_3 + 1.645\sqrt{x_1^2 + x_2^2 + x_3^2 + 1} \leq 4$$

by the fact that $\Phi^{-1}(0.95) = 1.645$.

4.4 Dependent-Chance Programming

In practice, there usually exist multiple events in a complex stochastic decision system. Sometimes, the decision-maker wishes to maximize the probabilities of meeting these events. In order to model this type of stochastic decision system, Liu [167] provided the third type of stochastic programming, called *dependent-chance programming* (DCP), in which the underlying philosophy is based on selecting the decision with maximal chance to meet the event.

DCP theory breaks the concept of feasible set and replaces it with uncertain environment. Roughly speaking, DCP involves maximizing chance functions of events in an uncertain environment. In deterministic model, EVM, and CCP, 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 DCP theory never assumes that the feasible set is deterministic. In fact, DCP is constructed in an uncertain environment. This special feature of DCP is very different from the other existing types of stochastic programming. However, such problems do exist in the real world.

In this section, we introduce the concepts of uncertain environment, event, and chance function, and discuss the principle of uncertainty, thus offering a spectrum of DCP models. We will take a supply system, represented by Figure 4.1 as the background.

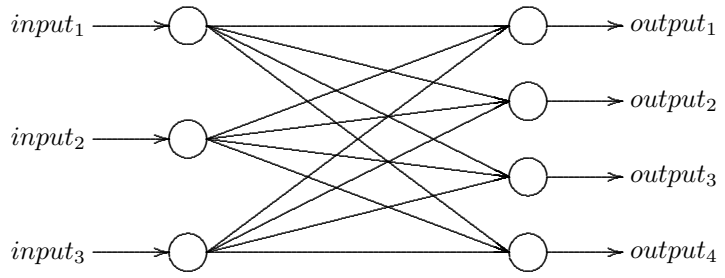

Figure 4.1: A Supply System

As an illustrative example, in Figure 4.1 there are 3 inputs representing 3 locations of resources and 4 outputs representing the demands of 4 users. We must answer the following supply problem: What is the appropriate combination of resources such that certain goals of supply are achieved?

In order to obtain the appropriate combination of resources for the supply problem, we use 12 decision variables $x_1, x_2, \cdots, x_{12}$ to represent an action, where x_1, x_2, x_3, x_4 are quantities ordered from $input_1$ to outputs 1,2,3,4 respectively; x_5, x_6, x_7, x_8 from $input_2$; $x_9, x_{10}, x_{11}, x_{12}$ from $input_3$. In prac-

tice, some variables may vanish due to some physical constraints.

We note that the inputs are available outside resources. Thus they have their own properties. For example, the capacities of resources are finite. Let ξ_1, ξ_2, ξ_3 be the maximum quantities supplied by the three resources. Then we have the following constraint,

$$\begin{cases} x_1^+ + x_2^+ + x_3^+ + x_4^+ \leq \xi_1 \\ x_5^+ + x_6^+ + x_7^+ + x_8^+ \leq \xi_2 \\ x_9^+ + x_{10}^+ + x_{11}^+ + x_{12}^+ \leq \xi_3 \\ x_i \geq 0, \quad i = 1, 2, \cdots, 12 \end{cases} \tag{4.39}$$

which represents that the quantities ordered from the resources are nonnegative and cannot exceed the maximum quantities, where x_i^+ represents x_i if x_i takes positive value, and vanishes otherwise. This means that the decision variable $x_i = 0$ must be able to perform for any realization of stochastic resources.

If at least one of ξ_1, ξ_2, and ξ_3 is really stochastic, then the constraint (4.39) is uncertain because we cannot make a decision such that it can be performed certainly before knowing the realization of ξ_1, ξ_2, and ξ_3. We will call this type of constraint the uncertain environment, and in this case the stochastic environment.

Definition 4.13 *By uncertain environment we mean the following stochastic constraint,*

$$g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, \quad j = 1, 2, \cdots, p, \tag{4.40}$$

where $\boldsymbol{x}$ is a decision vector, and $\boldsymbol{\xi}$ is a stochastic vector.

In the supply system, we should satisfy the demands of the 4 users, marked by c_1, c_2, c_3, and c_4. Then we have the following four events:

$$\begin{aligned} x_1 + x_5 + x_9 = c_1, \quad & x_2 + x_6 + x_{10} = c_2, \\ x_3 + x_7 + x_{11} = c_3, \quad & x_4 + x_8 + x_{12} = c_4. \end{aligned}$$

These equalities mean that the decision should satisfy the demands of users. Generally, an event is defined as follows.

Definition 4.14 *By event we mean a system of stochastic inequalities,*

$$h_k(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, \quad k = 1, 2, \cdots, q \tag{4.41}$$

where $\boldsymbol{x}$ is a decision vector, and $\boldsymbol{\xi}$ is a stochastic vector.

In view of the uncertainty of this system, we are not sure whether a decision can be performed before knowing the realization of stochastic variables.

Thus we wish to employ the following chance functions to evaluate these four events,

$$f_1(\boldsymbol{x}) = \Pr\{x_1 + x_5 + x_9 = c_1\}, \quad f_2(\boldsymbol{x}) = \Pr\{x_2 + x_6 + x_{10} = c_2\},$$
$$f_3(\boldsymbol{x}) = \Pr\{x_3 + x_7 + x_{11} = c_3\}, \quad f_4(\boldsymbol{x}) = \Pr\{x_4 + x_8 + x_{12} = c_4\},$$

subject to the uncertain environment (4.39).

Definition 4.15 *The chance function of an event $\mathcal{E}$ characterized by (4.41) is defined as the probability measure of the event, i.e.,*

$$f(\boldsymbol{x}) = \Pr\{h_k(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, k = 1, 2, \cdots, q\} \tag{4.42}$$

subject to the uncertain environment (4.40).

Usually, we hope to maximize the four chance functions $f_1(\boldsymbol{x})$, $f_2(\boldsymbol{x})$, $f_3(\boldsymbol{x})$ and $f_4(\boldsymbol{x})$. Here we remind the reader once more that the events like $x_1 + x_5 + x_9 = c_1$ do possess uncertainty because they are in an uncertain environment. *Any event is uncertain if it is in an uncertain environment*! This is an important law in the uncertain world. In fact, the randomness of the event is caused by the stochastic parameters ξ_1, ξ_2, ξ_3, and ξ_4 in the uncertain environment.

Until now we have formulated a stochastic programming model for the supply problem in an uncertain environment as follows,

$$\begin{cases} \max f_1(\boldsymbol{x}) = \Pr\{x_1 + x_5 + x_9 = c_1\} \\ \max f_2(\boldsymbol{x}) = \Pr\{x_2 + x_6 + x_{10} = c_2\} \\ \max f_3(\boldsymbol{x}) = \Pr\{x_3 + x_7 + x_{11} = c_3\} \\ \max f_4(\boldsymbol{x}) = \Pr\{x_4 + x_8 + x_{12} = c_4\} \\ \text{subject to:} \\ \quad x_1^+ + x_2^+ + x_3^+ + x_4^+ \leq \xi_1 \\ \quad x_5^+ + x_6^+ + x_7^+ + x_8^+ \leq \xi_2 \\ \quad x_9^+ + x_{10}^+ + x_{11}^+ + x_{12}^+ \leq \xi_3 \\ \quad x_i \geq 0, \quad i = 1, 2, \cdots, 12 \end{cases} \tag{4.43}$$

where ξ_1, ξ_2, and ξ_3 are stochastic variables. In this stochastic programming model, some variables (for example, x_1, x_2, x_3, x_4) are stochastically dependent because they share a common uncertain resource ξ_1. This also implies that the chance functions are stochastically dependent. We will call the stochastic programming (4.43) *dependent-chance programming* (DCP).

Principle of Uncertainty

How do we compute the chance function of an event $\mathcal{E}$ in an uncertain environment? In order to answer this question, we first give some definitions.

Definition 4.16 *Let $r(x_1, x_2, \cdots, x_n)$ be an n-dimensional function. The ith decision variable x_i is said to be degenerate if*

$$r(x_1, \cdots, x_{i-1}, x_i', x_{i+1}, \cdots, x_n) = r(x_1, \cdots, x_{i-1}, x_i'', x_{i+1}, \cdots, x_n)$$

for any x_i' and x_i''; otherwise it is nondegenerate.

For example, $r(x_1, x_2, x_3, x_4, x_5) = (x_1 + x_3)/x_4$ is a 5-dimensional function. The variables x_1, x_3, x_4 are nondegenerate, but x_2 and x_5 are degenerate.

Definition 4.17 *Let $\mathcal{E}$ be an event $h_k(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0$, $k = 1, 2, \cdots, q$. The support of the event $\mathcal{E}$, denoted by $\mathcal{E}^*$, is defined as the set consisting of all nondegenerate decision variables of functions $h_k(\boldsymbol{x}, \boldsymbol{\xi})$, $k = 1, 2, \cdots, q$.*

For example, let $\boldsymbol{x} = (x_1, x_2, \cdots, x_{12})$ be a decision vector, and let $\mathcal{E}$ be an event characterized by $x_1 + x_5 + x_9 = c_1$ and $x_2 + x_6 + x_{10} = c_2$. It is clear that x_1, x_5, x_9 are nondegenerate variables of the function $x_1 + x_5 + x_9$, and x_2, x_6, x_{10} are nondegenerate variables of the function $x_2 + x_6 + x_{10}$. Thus the support $\mathcal{E}^*$ of the event $\mathcal{E}$ is $\{x_1, x_2, x_5, x_6, x_9, x_{10}\}$.

Definition 4.18 *The jth constraint $g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0$ is called an active constraint of the event $\mathcal{E}$ if the set of nondegenerate decision variables of $g_j(\boldsymbol{x}, \boldsymbol{\xi})$ and the support $\mathcal{E}^*$ have nonempty intersection; otherwise it is inactive.*

Definition 4.19 *Let $\mathcal{E}$ be an event $h_k(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0$, $k = 1, 2, \cdots, q$ in the uncertain environment $g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, j = 1, 2, \cdots, p$. The dependent support of the event $\mathcal{E}$, denoted by $\mathcal{E}^{**}$, is defined as the set consisting of all nondegenerate decision variables of $h_k(\boldsymbol{x}, \boldsymbol{\xi}), k = 1, 2, \cdots, q$ and $g_j(\boldsymbol{x}, \boldsymbol{\xi})$ in the active constraints to the event $\mathcal{E}$.*

Remark 4.3: It is obvious that $\mathcal{E}^* \subset \mathcal{E}^{**}$ holds.

Definition 4.20 *The jth constraint $g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0$ is called a dependent constraint of the event $\mathcal{E}$ if the set of nondegenerate decision variables of $g_j(\boldsymbol{x}, \boldsymbol{\xi})$ and the dependent support $\mathcal{E}^{**}$ have nonempty intersection; otherwise it is independent.*

Remark 4.4: An active constraint must be a dependent constraint.

Definition 4.21 *Let $\mathcal{E}$ be an event $h_k(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0$, $k = 1, 2, \cdots, q$ in the uncertain environment $g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, j = 1, 2, \cdots, p$. For each decision $\boldsymbol{x}$ and realization $\boldsymbol{\xi}$, the event $\mathcal{E}$ is said to be consistent in the uncertain environment if the following two conditions hold: (i) $h_k(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0$, $k = 1, 2, \cdots, q$; and (ii) $g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0$, $j \in J$, where J is the index set of all dependent constraints.*

Intuitively, an event can be met by a decision provided that the decision meets both the event itself and the dependent constraints. We conclude it with the following principle of uncertainty.

Principle of Uncertainty: *The chance of a random event is the probability that the event is consistent in the uncertain environment.*

Assume that there are m events $\mathcal{E}_i$ characterized by $h_{ik}(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, k = 1, 2, \cdots, q_i$ for $i = 1, 2, \cdots, m$ in the uncertain environment $g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, j = 1, 2, \cdots, p$. The principle of uncertainty implies that the chance function of the ith event $\mathcal{E}_i$ in the uncertain environment is

$$f_i(\boldsymbol{x}) = \Pr\left\{\begin{array}{l} h_{ik}(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, k = 1, 2, \cdots, q_i \\ g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, j \in J_i \end{array}\right\} \tag{4.44}$$

where J_i are defined by

$$J_i = \left\{j \in \{1, 2, \cdots, p\} \;\middle|\; g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0 \text{ is a dependent constraint of } \mathcal{E}_i\right\}$$

for $i = 1, 2, \cdots, m$.

Remark 4.5: The principle of uncertainty is the basis of solution procedure of DCP that we shall encounter throughout the remainder of the book. However, the principle of uncertainty does not apply in all cases. For example, consider an event $x_1 \geq 6$ in the uncertain environment $x_1 - x_2 \leq \xi_1, x_2 - x_3 \leq \xi_2, x_3 \leq \xi_3$. It follows from the principle of uncertainty that the chance of the event is $\Pr\{x_1 \geq 6, x_1 - x_2 \leq \xi_1, x_2 - x_3 \leq \xi_2\}$, which is clearly wrong because the realization of $x_3 \leq \xi_3$ must be considered. Fortunately, such a case does not exist in real-life problems.

General Models

In this section, we consider the single-objective DCP. A typical DCP is represented as maximizing the chance function of an event subject to an uncertain environment,

$$\left\{\begin{array}{l} \max \Pr\left\{h_k(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, k = 1, 2, \cdots, q\right\} \\ \text{subject to:} \\ \quad g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, \quad j = 1, 2, \cdots, p \end{array}\right. \tag{4.45}$$

where $\boldsymbol{x}$ is an n-dimensional decision vector, $\boldsymbol{\xi}$ is a random vector of parameters, the system $h_k(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0$, $k = 1, 2, \cdots, q$ represents an event $\mathcal{E}$, and the constraints $g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0$, $j = 1, 2, \cdots, p$ are an uncertain environment.

DCP (4.45) reads as "maximizing the probability of the random event $h_k(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, k = 1, 2, \cdots, q$ subject to the uncertain environment $g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, j = 1, 2, \cdots, p$".

We now go back to the supply system. Assume that there is only one event $\mathcal{E}$ that satisfies the demand c_1 of $output_1$ (i.e., $x_1 + x_5 + x_9 = c_1$). If

we want to find a decision $\boldsymbol{x}$ with maximum probability to meet the event $\mathcal{E}$, then we have the following DCP model,

$$\begin{cases} \max \Pr\{x_1 + x_5 + x_9 = c_1\} \\ \text{subject to:} \\ \quad x_1^+ + x_2^+ + x_3^+ + x_4^+ \leq \xi_1 \\ \quad x_5^+ + x_6^+ + x_7^+ + x_8^+ \leq \xi_2 \\ \quad x_9^+ + x_{10}^+ + x_{11}^+ + x_{12}^+ \leq \xi_3 \\ \quad x_i \geq 0, i = 1, 2, \cdots, 12. \end{cases} \tag{4.46}$$

It is clear that the support of the event $\mathcal{E}$ is $\mathcal{E}^* = \{x_1, x_5, x_9\}$. If $x_1 \neq 0, x_5 \neq 0, x_9 \neq 0$, then the uncertain environment is

$$\begin{cases} x_1 + x_2 + x_3 + x_4 \leq \xi_1 \\ x_5 + x_6 + x_7 + x_8 \leq \xi_2 \\ x_9 + x_{10} + x_{11} + x_{12} \leq \xi_3 \\ x_i \geq 0, i = 1, 2, \cdots, 12. \end{cases}$$

Thus the dependent support $\mathcal{E}^{**} = \{x_1, x_2, \cdots, x_{12}\}$, and all constraints are dependent constraints. It follows from the principle of uncertainty that the chance function of the event $\mathcal{E}$ is

$$f(\boldsymbol{x}) = \Pr \begin{Bmatrix} x_1 + x_5 + x_9 = c_1 \\ x_1 + x_2 + x_3 + x_4 \leq \xi_1 \\ x_5 + x_6 + x_7 + x_8 \leq \xi_2 \\ x_9 + x_{10} + x_{11} + x_{12} \leq \xi_3 \\ x_i \geq 0, i = 1, 2, \cdots, 12 \end{Bmatrix}.$$

If $x_1 = 0, x_5 \neq 0, x_9 \neq 0$, then the uncertain environment is

$$\begin{cases} 0 + x_2 + x_3 + x_4 \leq \xi_1 \\ x_5 + x_6 + x_7 + x_8 \leq \xi_2 \\ x_9 + x_{10} + x_{11} + x_{12} \leq \xi_3 \\ x_i \geq 0, i = 1, 2, \cdots, 12. \end{cases}$$

Thus the dependent support $\mathcal{E}^{**} = \{x_5, x_6, \cdots, x_{12}\}$. It follows from the principle of uncertainty that the chance function of the event $\mathcal{E}$ is

$$f(\boldsymbol{x}) = \Pr \begin{Bmatrix} x_1 + x_5 + x_9 = c_1 \\ x_5 + x_6 + x_7 + x_8 \leq \xi_2 \\ x_9 + x_{10} + x_{11} + x_{12} \leq \xi_3 \\ x_i \geq 0, i = 5, 6, \cdots, 12 \end{Bmatrix}.$$

Similarly, if $x_1 \neq 0, x_5 = 0, x_9 \neq 0$, then the chance function of the event $\mathcal{E}$ is

$$f(\boldsymbol{x}) = \Pr \begin{Bmatrix} x_1 + x_5 + x_9 = c_1 \\ x_1 + x_2 + x_3 + x_4 \leq \xi_1 \\ x_9 + x_{10} + x_{11} + x_{12} \leq \xi_3 \\ x_i \geq 0, i = 1, 2, 3, 4, 9, 10, 11, 12 \end{Bmatrix}.$$

If $x_1 \neq 0, x_5 \neq 0, x_9 = 0$, then the chance function of the event $\mathcal{E}$ is

$$f(\boldsymbol{x}) = \Pr\begin{Bmatrix} x_1 + x_5 + x_9 = c_1 \\ x_1 + x_2 + x_3 + x_4 \leq \xi_1 \\ x_5 + x_6 + x_7 + x_8 \leq \xi_2 \\ x_i \geq 0, i = 1, 2, \cdots, 8 \end{Bmatrix}.$$

If $x_1 = 0, x_5 = 0, x_9 \neq 0$, then the chance function of the event $\mathcal{E}$ is

$$f(\boldsymbol{x}) = \Pr\begin{Bmatrix} x_1 + x_5 + x_9 = c_1 \\ x_9 + x_{10} + x_{11} + x_{12} \leq \xi_3 \\ x_i \geq 0, i = 9, 10, \cdots, 12 \end{Bmatrix}.$$

If $x_1 = 0, x_5 \neq 0, x_9 = 0$, then the chance function of the event $\mathcal{E}$ is

$$f(\boldsymbol{x}) = \Pr\begin{Bmatrix} x_1 + x_5 + x_9 = c_1 \\ x_5 + x_6 + x_7 + x_8 \leq \xi_2 \\ x_i \geq 0, i = 5, 6, \cdots, 8 \end{Bmatrix}.$$

If $x_1 \neq 0, x_5 = 0, x_9 = 0$, then the chance function of the event $\mathcal{E}$ is

$$f(\boldsymbol{x}) = \Pr\begin{Bmatrix} x_1 + x_5 + x_9 = c_1 \\ x_1 + x_2 + x_3 + x_4 \leq \xi_1 \\ x_i \geq 0, i = 1, 2, \cdots, 4 \end{Bmatrix}.$$

Note that the case $x_1 = x_5 = x_9 = 0$ is impossible because $c_1 \neq 0$. It follows that DCP (4.46) is equivalent to the unconstrained model "$\max f(\boldsymbol{x})$".

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 multiobjective programming (DCMOP) is represented as maximizing multiple chance functions subject to an uncertain environment,

$$\begin{cases} \max \begin{bmatrix} \Pr\{h_{1k}(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, k = 1, 2, \cdots, q_1\} \\ \Pr\{h_{2k}(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, k = 1, 2, \cdots, q_2\} \\ \quad \cdots \\ \Pr\{h_{mk}(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, k = 1, 2, \cdots, q_m\} \end{bmatrix} \\ \text{subject to:} \\ \quad g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, \quad j = 1, 2, \cdots, p \end{cases} \tag{4.47}$$

where $h_{ik}(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, k = 1, 2, \cdots, q_i$ represent events $\mathcal{E}_i$ for $i = 1, 2, \cdots, m$, respectively.

It follows from the principle of uncertainty that we can construct a relationship between decision vectors and chance functions, thus calculating the chance functions by stochastic simulations or traditional methods. Then we can solve DCMOP 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.

When some management targets are given, 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 dependent-chance goal programming (DCGP),

$$\begin{cases} \min \sum_{j=1}^{l} P_j \sum_{i=1}^{m} (u_{ij} d_i^+ \vee 0 + v_{ij} d_i^- \vee 0) \\ \text{subject to:} \\ \quad \Pr\{h_{ik}(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, k = 1, 2, \cdots, q_i\} - b_i = d_i^+, & i = 1, 2, \cdots, m \\ \quad b_i - \Pr\{h_{ik}(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, k = 1, 2, \cdots, q_i\} = d_i^-, & i = 1, 2, \cdots, m \\ \quad g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, & j = 1, 2, \cdots, p \end{cases}$$

where P_j is the preemptive priority factor, 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, $d_i^+ \vee 0$ is the positive deviation from the target of goal i, $d_i^- \vee 0$ 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.

4.5 Hybrid Intelligent Algorithm

From the mathematical viewpoint, there is no difference between deterministic mathematical programming and stochastic programming except for the fact that there exist uncertain functions in the latter. If the uncertain functions can be converted to their deterministic forms, then we can obtain equivalent deterministic models. However, generally speaking, we cannot do so. It is thus more convenient to deal with them by stochastic simulations. Essentially, there are three types of uncertain functions in stochastic programming as follows:

$$\begin{aligned} & U_1 : \boldsymbol{x} \to E[f(\boldsymbol{x}, \boldsymbol{\xi})], \\ & U_2 : \boldsymbol{x} \to \Pr\{g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, j = 1, 2, \cdots, p\}, \\ & U_3 : \boldsymbol{x} \to \max\left\{\overline{f} \mid \Pr\left\{f(\boldsymbol{x}, \boldsymbol{\xi}) \geq \overline{f}\right\} \geq \alpha\right\}. \end{aligned} \tag{4.48}$$

In order to compute the uncertain function $U_1(\boldsymbol{x})$, we generate ω_k from Ω according to the probability measure Pr and produce $\boldsymbol{\xi}(\omega_k)$ for $k = 1, 2, \cdots, N$. *Equivalently, we generate random vectors* $\boldsymbol{\xi}(\omega_k)$ *according to the probability distribution* Φ *for* $k = 1, 2, \cdots, N$. It follows from the strong law of large

numbers that

$$\frac{\sum_{k=1}^{N} f(\boldsymbol{x}, \boldsymbol{\xi}(\omega_k))}{N} \longrightarrow U_1(\boldsymbol{x}), \quad \text{a.s.} \tag{4.49}$$

as $N \to \infty$. Therefore, the value $U_1(\boldsymbol{x})$ is estimated by $\frac{1}{N}\sum_{k=1}^{N} f(\boldsymbol{x}, \boldsymbol{\xi}(\omega_k))$ provided that N is sufficiently large.

Algorithm 4.4 (Stochastic Simulation for $U_1(\boldsymbol{x})$)
Step 1. Set $e = 0$.
Step 2. Generate ω from Ω according to the probability measure Pr. Equivalently, we generate a random vector $\boldsymbol{\xi}(\omega)$ according to its probability distribution.
Step 3. $e \leftarrow e + f(\boldsymbol{x}, \boldsymbol{\xi}(\omega))$.
Step 4. Repeat the second and third steps N times.
Step 5. $U_1(\boldsymbol{x}) = e/N$.

In order to compute the uncertain function $U_2(\boldsymbol{x})$, we generate ω_k from Ω according to the probability measure Pr and produce $\boldsymbol{\xi}(\omega_k)$ for $k = 1, 2, \cdots, N$. Let N' denote the number of occasions on which $g_j(\boldsymbol{x}, \boldsymbol{\xi}(\omega_k)) \leq 0, j = 1, 2, \cdots, p$ for $k = 1, 2, \cdots, N$ (i.e., the number of random vectors satisfying the system of inequalities). Let us define

$$h(\boldsymbol{x}, \boldsymbol{\xi}(\omega_k)) = \begin{cases} 1, & \text{if } g_j(\boldsymbol{x}, \boldsymbol{\xi}(\omega_k)) \leq 0, j = 1, 2, \cdots, p \\ 0, & \text{otherwise.} \end{cases}$$

Then we have $E[h(\boldsymbol{x}, \boldsymbol{\xi}(\omega_k))] = U_2(\boldsymbol{x})$ for all k, and $N' = \sum_{k=1}^{N} h(\boldsymbol{x}, \boldsymbol{\xi}(\omega_k))$. It follows from the strong law of large numbers that

$$\frac{N'}{N} = \frac{\sum_{k=1}^{N} h(\boldsymbol{x}, \boldsymbol{\xi}(\omega_k))}{N}$$

converges a.s. to $U_2(\boldsymbol{x})$. Thus $U_2(\boldsymbol{x})$ can be estimated by N'/N provided that N is sufficiently large.

Algorithm 4.5 (Stochastic Simulation for $U_2(\boldsymbol{x})$)
Step 1. Set $N' = 0$.
Step 2. Generate ω from Ω according to the probability measure Pr. Equivalently, we generate a random vector $\boldsymbol{\xi}(\omega)$ according to its probability distribution.
Step 3. If $g_j(\boldsymbol{x}, \boldsymbol{\xi}(\omega)) \leq 0$ for $j = 1, 2, \cdots, p$, then $N' \leftarrow N' + 1$.
Step 4. Repeat the second and third steps N times.
Step 5. $U_2(\boldsymbol{x}) = N'/N$.

In order to compute the uncertain function $U_3(\boldsymbol{x})$, we generate ω_k from Ω according to the probability measure Pr and produce $\boldsymbol{\xi}(\omega_k)$ for $k = 1, 2, \cdots, N$. Now we define

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

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

$$\frac{\sum_{k=1}^{N} h(\boldsymbol{x}, \boldsymbol{\xi}(\omega_k))}{N} \longrightarrow \alpha, \quad \text{a.s.}$$

as $N \to \infty$. Note that the sum $\sum_{k=1}^{N} h(\boldsymbol{x}, \boldsymbol{\xi}(\omega_k))$ is just the number of $\boldsymbol{\xi}(\omega_k)$ satisfying $f(\boldsymbol{x}, \boldsymbol{\xi}(\omega_k)) \geq \overline{f}$ for $k = 1, 2, \cdots, N$. Thus $\overline{f}$ is just the N'th largest element in the sequence $\{f(\boldsymbol{x}, \boldsymbol{\xi}(\omega_1)), f(\boldsymbol{x}, \boldsymbol{\xi}(\omega_2)), \cdots, f(\boldsymbol{x}, \boldsymbol{\xi}(\omega_N))\}$, where N' is the integer part of αN.

Algorithm 4.6 (Stochastic Simulation for $U_3(\boldsymbol{x})$)

Step 1. Generate $\omega_1, \omega_2, \cdots, \omega_N$ from Ω according to the probability measure Pr. Equivalently, we generate random vectors $\boldsymbol{\xi}(\omega_1), \boldsymbol{\xi}(\omega_2), \cdots, \boldsymbol{\xi}(\omega_N)$ according to its probability distribution.

Step 2. Set $f_i = f(\boldsymbol{x}, \boldsymbol{\xi}(\omega_k))$ for $k = 1, 2, \cdots, N$.

Step 3. Set N' as the integer part of βN.

Step 4. Return the N'th largest element in $\{f_1, f_2, \cdots, f_N\}$ as $U_3(\boldsymbol{x})$.

Although stochastic simulations are able to compute the uncertain functions, we need relatively simple functions to approximate the uncertain functions because the stochastic simulations are a time-consuming process. In order to speed up the solution process, neural network (NN) is employed to approximate uncertain functions due to the following reasons: (i) NN has the ability to approximate the uncertain functions by using the training data; (ii) NN can compensate for the error of training data (all input-output data obtained by stochastic simulation are clearly not precise); and (iii) NN has the high speed of operation after they are trained.

Liu [182] integrated stochastic simulation, NN and GA to produce a hybrid intelligent algorithm for solving stochastic programming models.

Algorithm 4.7 (Hybrid Intelligent Algorithm)

Step 1. Generate training input-output data for uncertain functions like

$$U_1 : \boldsymbol{x} \to E[f(\boldsymbol{x}, \boldsymbol{\xi})],$$
$$U_2 : \boldsymbol{x} \to \Pr\{g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, j = 1, 2, \cdots, p\},$$
$$U_3 : \boldsymbol{x} \to \max\left\{\overline{f} \mid \Pr\left\{f(\boldsymbol{x}, \boldsymbol{\xi}) \geq \overline{f}\right\} \geq \alpha\right\}$$

by the stochastic simulation.

Step 2. Train a neural network to approximate the uncertain functions according to the generated training input-output data.

Step 3. Initialize *pop_size* chromosomes whose feasibility may be checked by the trained neural network.

Step 4. Update the chromosomes by crossover and mutation operations in which the feasibility of offspring may be checked by the trained neural network.

Step 5. Calculate the objective values for all chromosomes by the trained neural network.

Step 6. Compute the fitness of each chromosome according to the objective values.

Step 7. Select the chromosomes by spinning the roulette wheel.

Step 8. Repeat the fourth to seventh steps for a given number of cycles.

Step 9. Report the best chromosome as the optimal solution.

4.6 Numerical Experiments

In order to illustrate its effectiveness, a set of numerical examples has been done, and the results are successful. Here we give some numerical examples which are all performed on a personal computer with the following parameters: the population size is 30, the probability of crossover P_c is 0.3, the probability of mutation P_m is 0.2, and the parameter a in the rank-based evaluation function is 0.05.

Example 4.8: Now we consider the following EVM,

$$\begin{cases} \min E\left[\sqrt{(x_1-\xi_1)^2+(x_2-\xi_2)^2+(x_3-\xi_3)^2}\right] \\ \text{subject to:} \\ \quad x_1^2+x_2^2+x_3^2 \leq 10 \end{cases}$$

where ξ_1 is a uniformly distributed variable $\mathcal{U}(1,2)$, ξ_2 is a normally distributed variable $\mathcal{N}(3,1)$, and ξ_3 is an exponentially distributed variable $\mathcal{EXP}(4)$.

In order to solve this model, we generate input-output data for the uncertain function

$$U : (x_1, x_2, x_3) \to E\left[\sqrt{(x_1-\xi_1)^2+(x_2-\xi_2)^2+(x_3-\xi_3)^2}\right]$$

by stochastic simulation. Then we train an NN (3 input neurons, 5 hidden neurons, 1 output neuron) to approximate the uncertain function U. After that, the trained NN is embedded into a GA to produce a hybrid intelligent algorithm.

A run of the hybrid intelligent algorithm (3000 cycles in simulation, 2000 data in NN, 300 generations in GA) shows that the optimal solution is

$$(x_1^*, x_2^*, x_3^*) = (1.1035, 2.1693, 2.0191)$$

whose objective value is 3.56.

Example 4.9: Let us consider the following CCP in which there are three decision variables and nine stochastic parameters,

$$\begin{cases} \max \ \overline{f} \\ \text{subject to:} \\ \quad \Pr\left\{\xi_1 x_1 + \xi_2 x_2 + \xi_3 x_3 \geq \overline{f}\right\} \geq 0.90 \\ \quad \Pr\left\{\eta_1 x_1^2 + \eta_2 x_2^2 + \eta_3 x_3^2 \leq 8\right\} \geq 0.80 \\ \quad \Pr\left\{\tau_1 x_1^3 + \tau_2 x_2^3 + \tau_3 x_3^3 \leq 15\right\} \geq 0.85 \\ \quad x_1, x_2, x_3 \geq 0 \end{cases}$$

where ξ_1, η_1, and τ_1 are uniformly distributed variables $\mathcal{U}(1,2)$, $\mathcal{U}(2,3)$, and $\mathcal{U}(3,4)$, respectively, ξ_2, η_2, and τ_2 are normally distributed variables $\mathcal{N}(1,1)$, $\mathcal{N}(2,1)$, and $\mathcal{N}(3,1)$, respectively, and ξ_3, η_3, and τ_3 are exponentially distributed variables $\mathcal{EXP}(1)$, $\mathcal{EXP}(2)$, and $\mathcal{EXP}(3)$, respectively,

We employ stochastic simulation to generate input-output data for the uncertain function $U : \boldsymbol{x} \to (U_1(\boldsymbol{x}), U_2(\boldsymbol{x}), U_3(\boldsymbol{x}))$, where

$$\begin{aligned} U_1(\boldsymbol{x}) &= \max\left\{\overline{f} \mid \Pr\left\{\xi_1 x_1 + \xi_2 x_2 + \xi_3 x_3 \geq \overline{f}\right\} \geq 0.90\right\}, \\ U_2(\boldsymbol{x}) &= \Pr\left\{\eta_1 x_1^2 + \eta_2 x_2^2 + \eta_3 x_3^2 \leq 8\right\}, \\ U_3(\boldsymbol{x}) &= \Pr\left\{\tau_1 x_1^3 + \tau_2 x_2^3 + \tau_3 x_3^3 \leq 15\right\}. \end{aligned}$$

Then we train an NN (3 input neurons, 15 hidden neurons, 3 output neurons) to approximate the uncertain function U. Finally, we integrate the trained NN and GA to produce a hybrid intelligent algorithm.

A run of the hybrid intelligent algorithm (5000 cycles in simulation, 3000 training data in NN, 1000 generations in GA) shows that the optimal solution is

$$(x_1^*, x_2^*, x_3^*) = (1.458,\ 0.490,\ 0.811)$$

with objective value $\overline{f}^* = 2.27$.

Example 4.10: Let us now turn our attention to the following DCGP,

$$\begin{cases} \text{lexmin } \left\{d_1^- \vee 0, d_2^- \vee 0, d_3^- \vee 0\right\} \\ \text{subject to:} \\ \quad 0.92 - \Pr\{x_1 + x_4^2 = 4\} = d_1^- \\ \quad 0.85 - \Pr\{x_2^2 + x_6 = 3\} = d_2^- \\ \quad 0.85 - \Pr\{x_3^2 + x_5^2 + x_7^2 = 2\} = d_3^- \\ \quad x_1 + x_2 + x_3 \leq \xi_1 \\ \quad x_4 + x_5 \leq \xi_2 \\ \quad x_6 + x_7 \leq \xi_3 \\ \quad x_i \geq 0, \quad i = 1, 2, \cdots, 7 \end{cases}$$

where ξ_1, ξ_2, and ξ_3 are uniformly distributed variable $\mathcal{U}[3, 5]$, normally distributed variable $\mathcal{N}(3.5, 1)$, and exponentially distributed variable $\mathcal{EXP}(9)$, respectively.

In the first priority level, there is only one event $\mathcal{E}_1$ which will be fulfilled by $x_1 + x_4^2 = 4$. It is clear that the support $\mathcal{E}_1^* = \{x_1, x_4\}$ and the dependent support $\mathcal{E}_1^{**} = \{x_1, x_2, x_3, x_4, x_5\}$. Thus the dependent constraints of $\mathcal{E}_1$ are

$$x_1 + x_2 + x_3 \leq \xi_1, \quad x_4 + x_5 \leq \xi_2, \quad x_1, x_2, x_3, x_4, x_5 \geq 0.$$

It follows from the principle of uncertainty that the chance function $f_1(\boldsymbol{x})$ of $\mathcal{E}_1$ is

$$f_1(\boldsymbol{x}) = \Pr\left\{\begin{array}{l} x_1 + x_4^2 = 4 \\ x_1 + x_2 + x_3 \leq \xi_1 \\ x_4 + x_5 \leq \xi_2 \\ x_1, x_2, x_3, x_4, x_5 \geq 0 \end{array}\right\}.$$

At the second priority level, there is an event $\mathcal{E}_2$ which will be fulfilled by $x_2^2 + x_6 = 3$. The support $\mathcal{E}_2^* = \{x_2, x_6\}$ and the dependent support $\mathcal{E}_2^{**} = \{x_1, x_2, x_3, x_6, x_7\}$. Thus the dependent constraints of $\mathcal{E}_2$ are

$$x_1 + x_2 + x_3 \leq \xi_1, \quad x_6 + x_7 \leq \xi_3, \quad x_1, x_2, x_3, x_6, x_7 \geq 0.$$

The principle of uncertainty implies that the chance function $f_2(\boldsymbol{x})$ of the event $\mathcal{E}_2$ is

$$f_2(\boldsymbol{x}) = \Pr\left\{\begin{array}{l} x_2^2 + x_6 = 3 \\ x_1 + x_2 + x_3 \leq \xi_1 \\ x_6 + x_7 \leq \xi_3 \\ x_1, x_2, x_3, x_6, x_7 \geq 0 \end{array}\right\}.$$

At the third priority level, there is an event $\mathcal{E}_3$ which will be fulfilled by $x_3^2 + x_5^2 + x_7^2 = 2$. The support $\mathcal{E}_3^* = \{x_3, x_5, x_7\}$ and the dependent support $\mathcal{E}_3^{**}$ includes all decision variables. Thus all constraints are dependent

constraints of $\mathcal{E}_3$. It follows from the principle of uncertainty that the chance function $f_3(\boldsymbol{x})$ of the event $\mathcal{E}_3$ is

$$f_3(\boldsymbol{x}) = \Pr\left\{\begin{array}{l} x_3^2 + x_5^2 + x_7^2 = 2 \\ x_1 + x_2 + x_3 \leq \xi_1 \\ x_4 + x_5 \leq \xi_2 \\ x_6 + x_7 \leq \xi_3 \\ x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0 \end{array}\right\}.$$

We encode a solution into a chromosome $V = (v_1, v_2, v_3, v_4)$, and decode a chromosome into a feasible solution in the following way,

$$\begin{array}{llll} x_1 = v_1, & x_2 = v_2, & x_3 = v_3, & x_4 = \sqrt{4 - v_1}, \\ x_5 = v_4, & x_6 = 3 - v_2^2, & x_7 = \sqrt{2 - v_3^2 - v_4^2}. & \end{array}$$

We first employ stochastic simulation to generate input-output data for the uncertain function $U : (v_1, v_2, v_3, v_4) \to (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), f_3(\boldsymbol{x}))$. Then we train an NN (4 input neurons, 10 hidden neurons, 3 output neurons) to approximate it. Finally, we embed the trained NN into a GA to produce a hybrid intelligent algorithm.

A run of the hybrid intelligent algorithm (6000 cycles in simulation, 3000 data in NN, 1000 generations in GA) shows that the optimal solution is

$$\boldsymbol{x}^* = (0.1180, 1.7320, 0.1491, 1.9703, 0.0000, 0.0000, 1.4063)$$

which can satisfy the first two goals, but the third objective is 0.05.

Chapter 5

Fuzzy Programming

Fuzzy programming offers a powerful means of handling optimization problems with fuzzy parameters. Fuzzy programming has been used in different ways in the past. Liu and Liu [185] presented a concept of expected value operator of fuzzy variable and provided a spectrum of fuzzy expected value models which optimize the expected objective functions subject to some expected constraints. In addition, Liu and Iwamura [169][170] introduced a spectrum of fuzzy maximax chance-constrained programming, and Liu [172] constructed a spectrum of fuzzy minimax chance-constrained programming in which we assume that the fuzzy constraints will hold with a credibility level. Liu [173] provided a fuzzy dependent-chance programming theory in order to maximize the chance functions of satisfying some events.

5.1 Fuzzy Variables

The concept of fuzzy set was initialized by Zadeh [328] in 1965. Fuzzy set theory has been well developed and applied in a wide variety of real problems. In order to measure a fuzzy event, Zadeh [331] proposed the concept of possibility measure in 1978. Although possibility measure has been widely used, it is not self-dual. However, a self-dual measure is absolutely needed in both theory and practice. In order to define a self-dual measure, Liu and Liu [185] presented the concept of credibility measure in 2002. In addition, a sufficient and necessary condition for credibility measure was given by Li and Liu [162]. Credibility theory was founded by Liu [187] in 2004 and refined by Liu [190] in 2007 as a branch of mathematics for studying the behavior of fuzzy phenomena.

Let Θ be a nonempty set, and $\mathcal{P}$ the power set of Θ. Each element in $\mathcal{P}$ is called an event. In order to present an axiomatic definition of credibility, it is necessary to assign to each event A a number $\mathrm{Cr}\{A\}$ which indicates the credibility that A will occur. In order to ensure that the number $\mathrm{Cr}\{A\}$ has

certain mathematical properties which we intuitively expect a credibility to have, we accept the following four axioms:

Axiom 1. *(Normality)* $\mathrm{Cr}\{\Theta\} = 1$.

Axiom 2. *(Monotonicity)* $\mathrm{Cr}\{A\} \leq \mathrm{Cr}\{B\}$ *whenever* $A \subset B$.

Axiom 3. *(Self-Duality)* $\mathrm{Cr}\{A\} + \mathrm{Cr}\{A^c\} = 1$ *for any event* A.

Axiom 4. *(Maximality)* $\mathrm{Cr}\{\cup_i A_i\} = \sup_i \mathrm{Cr}\{A_i\}$ *for any events* $\{A_i\}$ *with* $\sup_i \mathrm{Cr}\{A_i\} < 0.5$.

Definition 5.1 *(Liu and Liu [185]) The set function* Cr *is called a credibility measure if it satisfies the normality, monotonicity, self-duality, and maximality axioms.*

Example 5.1: Let $\Theta = \{\theta_1, \theta_2\}$. For this case, there are only four events: $\emptyset, \{\theta_1\}, \{\theta_2\}, \Theta$. Define $\mathrm{Cr}\{\emptyset\} = 0$, $\mathrm{Cr}\{\theta_1\} = 0.7$, $\mathrm{Cr}\{\theta_2\} = 0.3$, and $\mathrm{Cr}\{\Theta\} = 1$. Then the set function Cr is a credibility measure because it satisfies the first four axioms.

Example 5.2: Let Θ be a nonempty set. Define $\mathrm{Cr}\{\emptyset\} = 0$, $\mathrm{Cr}\{\Theta\} = 1$ and $\mathrm{Cr}\{A\} = 1/2$ for any subset A (excluding $\emptyset$ and Θ). Then the set function Cr is a credibility measure.

Theorem 5.1 *Let* Cr *be a credibility measure. Then* $\mathrm{Cr}\{\emptyset\} = 0$ *and* $0 \leq \mathrm{Cr}\{A\} \leq 1$ *for any* $A \in \mathcal{P}$.

Definition 5.2 *Let* Θ *be a nonempty set,* $\mathcal{P}$ *the power set of* Θ*, and* Cr *a credibility measure. Then the triplet* $(\Theta, \mathcal{P}, \mathrm{Cr})$ *is called a credibility space.*

Definition 5.3 *A fuzzy variable is defined as a (measurable) function from a credibility space* $(\Theta, \mathcal{P}, \mathrm{Cr})$ *to the set of real numbers.*

An n-dimensional fuzzy vector is defined as a function from a credibility space $(\Theta, \mathcal{P}, \mathrm{Cr})$ to the set of n-dimensional real vectors. It has been proved that $(\xi_1, \xi_2, \cdots, \xi_n)$ is a fuzzy vector if and only if $\xi_1, \xi_2, \cdots, \xi_n$ are fuzzy variables.

Definition 5.4 *Let* $f : \Re^n \to \Re$ *be a function, and* $\xi_1, \xi_2, \cdots, \xi_n$ *fuzzy variables on the credibility space* $(\Theta, \mathcal{P}, \mathrm{Cr})$*. Then* $\xi = f(\xi_1, \xi_2, \cdots, \xi_n)$ *is a fuzzy variable defined as*

$$\xi(\theta) = f(\xi_1(\theta), \xi_2(\theta), \cdots, \xi_n(\theta)) \tag{5.1}$$

for any $\theta \in \Theta$.

Membership Function

Definition 5.5 *Let ξ be a fuzzy variable defined on the credibility space $(\Theta, \mathcal{P}, \mathrm{Cr})$. Then its membership function is derived from the credibility measure by*

$$\mu(x) = (2\mathrm{Cr}\{\xi = x\}) \wedge 1, \quad x \in \Re. \tag{5.2}$$

Example 5.3: By an *equipossible fuzzy variable* on $[a, b]$ we mean the fuzzy variable whose membership function is given by

$$\mu(x) = \begin{cases} 1, & \text{if } a \leq x \leq b \\ 0, & \text{otherwise.} \end{cases}$$

Example 5.4: By a *triangular fuzzy variable* we mean the fuzzy variable fully determined by the triplet (r_1, r_2, r_3) of crisp numbers with $r_1 < r_2 < r_3$, whose membership function is given by

$$\mu(x) = \begin{cases} \dfrac{x - r_1}{r_2 - r_1}, & \text{if } r_1 \leq x \leq r_2 \\ \dfrac{x - r_3}{r_2 - r_3}, & \text{if } r_2 \leq x \leq r_3 \\ 0, & \text{otherwise.} \end{cases}$$

Example 5.5: By a *trapezoidal fuzzy variable* we mean the fuzzy variable fully determined by quadruplet (r_1, r_2, r_3, r_4) of crisp numbers with $r_1 < r_2 < r_3 < r_4$, whose membership function is given by

$$\mu(x) = \begin{cases} \dfrac{x - r_1}{r_2 - r_1}, & \text{if } r_1 \leq x \leq r_2 \\ 1, & \text{if } r_2 \leq x \leq r_3 \\ \dfrac{x - r_4}{r_3 - r_4}, & \text{if } r_3 \leq x \leq r_4 \\ 0, & \text{otherwise.} \end{cases}$$

Theorem 5.2 *(Credibility Inversion Theorem) Let ξ be a fuzzy variable with membership function μ. Then for any set B of real numbers, we have*

$$\mathrm{Cr}\{\xi \in B\} = \frac{1}{2}\left(\sup_{x \in B} \mu(x) + 1 - \sup_{x \in B^c} \mu(x)\right). \tag{5.3}$$

Independence

The independence of fuzzy variables has been discussed by many authors from different angles, for example, Zadeh [331], Nahmias [240], Yager [318], Liu [187], and Liu and Gao [204]. A lot of equivalence conditions of independence are presented. Here we use the condition given by Liu and Gao [204].

Definition 5.6 *(Liu and Gao [204]) The fuzzy variables $\xi_1, \xi_2, \cdots, \xi_m$ are said to be independent if*

$$\mathrm{Cr}\left\{\bigcap_{i=1}^{m}\{\xi_i \in B_i\}\right\} = \min_{1\leq i\leq m} \mathrm{Cr}\{\xi_i \in B_i\} \tag{5.4}$$

for any sets $B_1, B_2, \cdots, B_m$ of real numbers.

Theorem 5.3 *(Extension Principle of Zadeh) Let $\xi_1, \xi_2, \cdots, \xi_n$ be independent fuzzy variables with membership functions $\mu_1, \mu_2, \cdots, \mu_n$, respectively, and $f : \Re^n \to \Re$ a function. Then the membership function μ of $\xi = f(\xi_1, \xi_2, \cdots, \xi_n)$ is derived from the membership functions $\mu_1, \mu_2, \cdots, \mu_n$ by*

$$\mu(x) = \sup_{x=f(x_1,x_2,\cdots,x_n)} \min_{1\leq i\leq n} \mu_i(x_i). \tag{5.5}$$

Example 5.6: By using Theorem 5.3, we may verify that the sum of independent equipossible fuzzy variables $\xi = (a_1, a_2)$ and $\eta = (b_1, b_2)$ is also an equipossible fuzzy variable, and

$$\xi + \eta = (a_1 + b_1, a_2 + b_2).$$

Their product is also an equipossible fuzzy variable, and

$$\xi \cdot \eta = \left(\min_{a_1\leq x\leq a_2, b_1\leq y\leq b_2} xy,\ \max_{a_1\leq x\leq a_2, b_1\leq y\leq b_2} xy\right).$$

Example 5.7: The sum of independent triangular fuzzy variables $\xi = (a_1, a_2, a_3)$ and $\eta = (b_1, b_2, b_3)$ is also a triangular fuzzy variable, and

$$\xi + \eta = (a_1 + b_1, a_2 + b_2, a_3 + b_3).$$

The product of a triangular fuzzy variable $\xi = (a_1, a_2, a_3)$ and a scalar number λ is

$$\lambda \cdot \xi = \begin{cases} (\lambda a_1, \lambda a_2, \lambda a_3), & \text{if } \lambda \geq 0 \\ (\lambda a_3, \lambda a_2, \lambda a_1), & \text{if } \lambda < 0. \end{cases}$$

That is, the product of a triangular fuzzy variable and a scalar number is also a triangular fuzzy variable. However, the product of two triangular fuzzy variables is not a triangular one.

Example 5.8: The sum of independent trapezoidal fuzzy variables $\xi = (a_1, a_2, a_3, a_4)$ and $\eta = (b_1, b_2, b_3, b_4)$ is also a trapezoidal fuzzy variable, and $\xi + \eta = (a_1 + b_1, a_2 + b_2, a_3 + b_3, a_4 + b_4)$. The product of a trapezoidal fuzzy variable $\xi = (a_1, a_2, a_3, a_4)$ and a scalar number λ is

$$\lambda \cdot \xi = \begin{cases} (\lambda a_1, \lambda a_2, \lambda a_3, \lambda a_4), & \text{if } \lambda \geq 0 \\ (\lambda a_4, \lambda a_3, \lambda a_2, \lambda a_1), & \text{if } \lambda < 0. \end{cases}$$

That is, the product of a trapezoidal fuzzy variable and a scalar number is also a trapezoidal fuzzy variable.

Example 5.9: Let $\xi_1, \xi_2, \cdots, \xi_n$ be independent fuzzy variables with membership functions $\mu_1, \mu_2, \cdots, \mu_n$, respectively, and $f : \Re^n \to \Re$ a function. Then for any set B of real numbers, the credibility $\mathrm{Cr}\{f(\xi_1, \xi_2, \cdots, \xi_n) \in B\}$ is

$$\frac{1}{2}\left(\sup_{f(x_1,x_2,\cdots,x_n)\in B} \min_{1\le i\le n} \mu_i(x_i) + 1 - \sup_{f(x_1,x_2,\cdots,x_n)\in B^c} \min_{1\le i\le n} \mu_i(x_i)\right).$$

Expected Value

For fuzzy variables, there are many ways to define an expected value operator, for example, Dubois and Prade [60], Heilpern [96], Campos and González [31], González [92] and Yager [315][321]. The most general definition of expected value operator of fuzzy variable was given by Liu and Liu [185]. This definition is not only applicable to continuous fuzzy variables but also discrete ones.

Definition 5.7 *(Liu and Liu [185]) Let ξ be a fuzzy variable. Then the expected value of ξ is defined by*

$$E[\xi] = \int_0^{+\infty} \mathrm{Cr}\{\xi \ge r\}\mathrm{d}r - \int_{-\infty}^0 \mathrm{Cr}\{\xi \le r\}\mathrm{d}r \tag{5.6}$$

provided that at least one of the two integrals is finite.

Example 5.10: Let ξ be an equipossible fuzzy variable on $[a, b]$. Then its expected value is $E[\xi] = (a + b)/2$.

Example 5.11: The triangular fuzzy variable $\xi = (a, b, c)$ has an expected value $E[\xi] = (a + 2b + c)/4$.

Example 5.12: The trapezoidal fuzzy variable $\xi = (a, b, c, d)$ has an expected value $E[\xi] = (a + b + c + d)/4$.

Example 5.13: Let ξ be a continuous nonnegative fuzzy variable with membership function μ. If μ is decreasing on $[0, +\infty)$, then $\mathrm{Cr}\{\xi \ge x\} = \mu(x)/2$ for any $x > 0$, and

$$E[\xi] = \frac{1}{2}\int_0^{+\infty} \mu(x)\mathrm{d}x.$$

Example 5.14: Let ξ be a continuous fuzzy variable with membership function μ. If its expected value exists, and there is a point x_0 such that $\mu(x)$ is

increasing on $(-\infty, x_0)$ and decreasing on $(x_0, +\infty)$, then

$$E[\xi] = x_0 + \frac{1}{2}\int_{x_0}^{+\infty} \mu(x)\mathrm{d}x - \frac{1}{2}\int_{-\infty}^{x_0} \mu(x)\mathrm{d}x.$$

Example 5.15: The definition of expected value operator is also applicable to discrete case. Assume that ξ is a simple fuzzy variable whose membership function is given by

$$\mu(x) = \begin{cases} \mu_1, & \text{if } x = x_1 \\ \mu_2, & \text{if } x = x_2 \\ \cdots & \\ \mu_m, & \text{if } x = x_m \end{cases} \tag{5.7}$$

where $x_1, x_2, \cdots, x_m$ are distinct numbers. Note that $\mu_1 \vee \mu_2 \vee \cdots \vee \mu_m = 1$. Definition 5.7 implies that the expected value of ξ is

$$E[\xi] = \sum_{i=1}^{m} w_i x_i \tag{5.8}$$

where the weights are given by

$$w_i = \frac{1}{2}\left(\max_{1\le j\le m}\{\mu_j | x_j \le x_i\} - \max_{1\le j\le m}\{\mu_j | x_j < x_i\} + \max_{1\le j\le m}\{\mu_j | x_j \ge x_i\} - \max_{1\le j\le m}\{\mu_j | x_j > x_i\}\right)$$

for $i = 1, 2, \cdots, m$. It is easy to verify that all $w_i \ge 0$ and the sum of all weights is just 1.

Example 5.16: Consider the fuzzy variable ξ defined by (5.7). Suppose $x_1 < x_2 < \cdots < x_m$. Then the expected value is determined by (5.8) and the weights are given by

$$w_i = \frac{1}{2}\left(\max_{1\le j\le i}\mu_j - \max_{1\le j<i}\mu_j + \max_{i\le j\le m}\mu_j - \max_{i<j\le m}\mu_j\right)$$

for $i = 1, 2, \cdots, m$.

Example 5.17: Consider the fuzzy variable ξ defined by (5.7). Suppose $x_1 < x_2 < \cdots < x_m$ and there exists an index k with $1 < k < m$ such that

$$\mu_1 \le \mu_2 \le \cdots \le \mu_k \quad \text{and} \quad \mu_k \ge \mu_{k+1} \ge \cdots \ge \mu_m.$$

Note that $\mu_k \equiv 1$. The expected value $E[\xi]$ is

$$\frac{\mu_1}{2}x_1 + \sum_{i=2}^{k-1}\frac{\mu_i - \mu_{i-1}}{2}x_i + \left(1 - \frac{\mu_{k-1} + \mu_{k+1}}{2}\right)x_k + \sum_{i=k+1}^{m-1}\frac{\mu_i - \mu_{i+1}}{2}x_i + \frac{\mu_m}{2}x_m.$$

Critical Values

Definition 5.8 *(Liu [182]) Let ξ be a fuzzy variable, and $\alpha \in (0,1]$. Then*

$$\xi_{\sup}(\alpha) = \sup\left\{r \mid \mathrm{Cr}\{\xi \geq r\} \geq \alpha\right\} \tag{5.9}$$

is called the α-optimistic value to ξ; and

$$\xi_{\inf}(\alpha) = \inf\left\{r \mid \mathrm{Cr}\{\xi \leq r\} \geq \alpha\right\} \tag{5.10}$$

is called the α-pessimistic value to ξ.

Example 5.18: Let ξ be an equipossible fuzzy variable on $[a,b]$. Then its α-optimistic and α-pessimistic values are

$$\xi_{\sup}(\alpha) = \begin{cases} b, & \text{if } \alpha \leq 0.5 \\ a, & \text{if } \alpha > 0.5, \end{cases} \qquad \xi_{\inf}(\alpha) = \begin{cases} a, & \text{if } \alpha \leq 0.5 \\ b, & \text{if } \alpha > 0.5. \end{cases}$$

Example 5.19: Let $\xi = (r_1, r_2, r_3)$ be a triangular fuzzy variable. Then its α-optimistic and α-pessimistic values are

$$\xi_{\sup}(\alpha) = \begin{cases} 2\alpha r_2 + (1-2\alpha)r_3, & \text{if } \alpha \leq 0.5 \\ (2\alpha-1)r_1 + (2-2\alpha)r_2, & \text{if } \alpha > 0.5, \end{cases}$$

$$\xi_{\inf}(\alpha) = \begin{cases} (1-2\alpha)r_1 + 2\alpha r_2, & \text{if } \alpha \leq 0.5 \\ (2-2\alpha)r_2 + (2\alpha-1)r_3, & \text{if } \alpha > 0.5. \end{cases}$$

Example 5.20: Let $\xi = (r_1, r_2, r_3, r_4)$ be a trapezoidal fuzzy variable. Then its α-optimistic and α-pessimistic values are

$$\xi_{\sup}(\alpha) = \begin{cases} 2\alpha r_3 + (1-2\alpha)r_4, & \text{if } \alpha \leq 0.5 \\ (2\alpha-1)r_1 + (2-2\alpha)r_2, & \text{if } \alpha > 0.5, \end{cases}$$

$$\xi_{\inf}(\alpha) = \begin{cases} (1-2\alpha)r_1 + 2\alpha r_2, & \text{if } \alpha \leq 0.5 \\ (2-2\alpha)r_3 + (2\alpha-1)r_4, & \text{if } \alpha > 0.5. \end{cases}$$

Ranking Criteria

Let ξ and η be two fuzzy variables. Different from the situation of real numbers, there does not exist a natural ordership in a fuzzy world. Thus an important problem appearing in this area is how to rank fuzzy variables. Here we give four ranking criteria.

Expected Value Criterion: We say $\xi > \eta$ if and only if $E[\xi] > E[\eta]$, where E is the expected value operator of fuzzy variable.

Optimistic Value Criterion: We say $\xi > \eta$ if and only if, for some predetermined confidence level $\alpha \in (0, 1]$, we have $\xi_{\sup}(\alpha) > \eta_{\sup}(\alpha)$, where $\xi_{\sup}(\alpha)$ and $\eta_{\sup}(\alpha)$ are the α-optimistic values of ξ and η, respectively.

Pessimistic Value Criterion: We say $\xi > \eta$ if and only if, for some predetermined confidence level $\alpha \in (0, 1]$, we have $\xi_{\inf}(\alpha) > \eta_{\inf}(\alpha)$, where $\xi_{\inf}(\alpha)$ and $\eta_{\inf}(\alpha)$ are the α-pessimistic values of ξ and η, respectively.

Credibility Criterion: We say $\xi > \eta$ if and only if $\mathrm{Cr}\{\xi \geq \overline{r}\} > \mathrm{Cr}\{\eta \geq \overline{r}\}$ for some predetermined level $\overline{r}$.

5.2 Expected Value Model

Assume that $\boldsymbol{x}$ is a decision vector, $\boldsymbol{\xi}$ is a fuzzy vector, $f(\boldsymbol{x}, \boldsymbol{\xi})$ is a return function, and $g_j(\boldsymbol{x}, \boldsymbol{\xi})$ are constraint functions, $j = 1, 2, \cdots, p$. Let us examine the following "fuzzy programming",

$$\begin{cases} \max f(\boldsymbol{x}, \boldsymbol{\xi}) \\ \text{subject to:} \\ \quad g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, \quad j = 1, 2, \cdots, p. \end{cases} \tag{5.11}$$

Similar to stochastic programming, the model (5.11) is not well-defined because (i) we cannot maximize the fuzzy quantity $f(\boldsymbol{x}, \boldsymbol{\xi})$ (just like that we cannot maximize a random quantity), and (ii) the constraints $g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, j = 1, 2, \cdots, p$ do not produce a crisp feasible set.

Unfortunately, the form of fuzzy programming like (5.11) appears frequently in the literature. Fuzzy programming is a class of mathematical models. Different from fashion or building models, everyone should have the same understanding of the same mathematical model. In other words, a mathematical model must have an unambiguous explanation. The form (5.11) does not have mathematical meaning because it has different interpretations.

In order to obtain the decision with maximum expected return, Liu and Liu [185] provided a spectrum of fuzzy expected value model (EVM),

$$\begin{cases} \max E[f(\boldsymbol{x}, \boldsymbol{\xi})] \\ \text{subject to:} \\ \quad E[g_j(\boldsymbol{x}, \boldsymbol{\xi})] \leq 0, \quad j = 1, 2, \cdots, p \end{cases} \tag{5.12}$$

where $\boldsymbol{x}$ is a decision vector, $\boldsymbol{\xi}$ is a fuzzy vector, $f(\boldsymbol{x}, \boldsymbol{\xi})$ is the return function, and $g_j(\boldsymbol{x}, \boldsymbol{\xi})$ are the constraint functions, $j = 1, 2, \cdots, p$.

In many cases, we may have multiple return functions. Thus we have to employ fuzzy expected value multiobjective programming (EVMOP),

$$\begin{cases} \max [E[f_1(\boldsymbol{x}, \boldsymbol{\xi})], E[f_2(\boldsymbol{x}, \boldsymbol{\xi})], \cdots, E[f_m(\boldsymbol{x}, \boldsymbol{\xi})]] \\ \text{subject to:} \\ \quad E[g_j(\boldsymbol{x}, \boldsymbol{\xi})] \leq 0, \quad j = 1, 2, \cdots, p \end{cases} \tag{5.13}$$

where $f_i(\boldsymbol{x}, \boldsymbol{\xi})$ are return functions, $i = 1, 2, \cdots, m$.

In order to balance the multiple conflicting objectives, we may employ the following fuzzy expected value goal programming (EVGP),

$$\begin{cases} \min \sum_{j=1}^{l} P_j \sum_{i=1}^{m} (u_{ij} d_i^+ \vee 0 + v_{ij} d_i^- \vee 0) \\ \text{subject to:} \\ \quad E[f_i(\boldsymbol{x}, \boldsymbol{\xi})] - b_i = d_i^+, \quad i = 1, 2, \cdots, m \\ \quad b_i - E[f_i(\boldsymbol{x}, \boldsymbol{\xi})] = d_i^-, \quad i = 1, 2, \cdots, m \\ \quad E[g_j(\boldsymbol{x}, \boldsymbol{\xi})] \leq 0, \qquad j = 1, 2, \cdots, p \end{cases} \tag{5.14}$$

where P_j is the preemptive priority factor which expresses the relative importance of various goals, $P_j \gg P_{j+1}$, 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, $d_i^+ \vee 0$ is the positive deviation from the target of goal i, $d_i^- \vee 0$ is the negative deviation from the target of goal i, 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.

5.3 Chance-Constrained Programming

Assume that $\boldsymbol{x}$ is a decision vector, $\boldsymbol{\xi}$ is a fuzzy vector, $f(\boldsymbol{x}, \boldsymbol{\xi})$ is a return function, and $g_j(\boldsymbol{x}, \boldsymbol{\xi})$ are constraint functions, $j = 1, 2, \cdots, p$. Since the fuzzy constraints $g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, j = 1, 2, \cdots, p$ do not define a deterministic feasible set, a natural idea is to provide a confidence level α at which it is desired that the fuzzy constraints hold. Thus we have a chance constraint as follows,

$$\mathrm{Cr}\left\{g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, j = 1, 2, \cdots, p\right\} \geq \alpha. \tag{5.15}$$

Sometimes, we may employ the following separate chance constraints,

$$\mathrm{Cr}\left\{g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0\right\} \geq \alpha_j, \quad j = 1, 2, \cdots, p \tag{5.16}$$

where α_j are confidence levels for $j = 1, 2, \cdots, p$.

Maximax Chance-Constrained Programming

Liu and Iwamura [169][170] suggested a spectrum of fuzzy CCP. When we want to maximize the optimistic return, we have the following fuzzy CCP,

$$\begin{cases} \max_{\boldsymbol{x}} \max_{\overline{f}} \overline{f} \\ \text{subject to:} \\ \quad \mathrm{Cr}\left\{f(\boldsymbol{x}, \boldsymbol{\xi}) \geq \overline{f}\right\} \geq \beta \\ \quad \mathrm{Cr}\left\{g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, j = 1, 2, \cdots, p\right\} \geq \alpha \end{cases} \tag{5.17}$$

where α and β are the predetermined confidence levels, and $\max \overline{f}$ is the β-optimistic return.

If there are multiple objectives, then we have a chance-constrained multiobjective programming (CCMOP),

$$\begin{cases} \max\limits_{\boldsymbol{x}} \left[\max\limits_{\overline{f}_1} \overline{f}_1, \max\limits_{\overline{f}_2} \overline{f}_2, \cdots, \max\limits_{\overline{f}_m} \overline{f}_m \right] \\ \text{subject to:} \\ \quad \operatorname{Cr}\left\{ f_i(\boldsymbol{x}, \boldsymbol{\xi}) \geq \overline{f}_i \right\} \geq \beta_i, \quad i = 1, 2, \cdots, m \\ \quad \operatorname{Cr}\{ g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0 \} \geq \alpha_j, \quad j = 1, 2, \cdots, p \end{cases} \tag{5.18}$$

where $\alpha_1, \alpha_2, \cdots, \alpha_p, \beta_1, \beta_2, \cdots, \beta_m$ are the predetermined confidence levels, and $\max \overline{f}_i$ are the β_i-optimistic values to the return functions $f_i(\boldsymbol{x}, \boldsymbol{\xi})$, $i = 1, 2, \cdots, m$, respectively.

We can also formulate the fuzzy decision system as a minimin chance-constrained goal programming (CCGP) according to the priority structure and target levels set by the decision-maker:

$$\begin{cases} \min\limits_{\boldsymbol{x}} \sum\limits_{j=1}^{l} P_j \sum\limits_{i=1}^{m} \left(u_{ij} \left(\min\limits_{d_i^+} d_i^+ \vee 0 \right) + v_{ij} \left(\min\limits_{d_i^-} d_i^- \vee 0 \right) \right) \\ \text{subject to:} \\ \quad \operatorname{Cr}\left\{ f_i(\boldsymbol{x}, \boldsymbol{\xi}) - b_i \leq d_i^+ \right\} \geq \beta_i^+, \quad i = 1, 2, \cdots, m \\ \quad \operatorname{Cr}\left\{ b_i - f_i(\boldsymbol{x}, \boldsymbol{\xi}) \leq d_i^- \right\} \geq \beta_i^-, \quad i = 1, 2, \cdots, m \\ \quad \operatorname{Cr}\left\{ g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0 \right\} \geq \alpha_j, \quad j = 1, 2, \cdots, p \end{cases} \tag{5.19}$$

where P_j is the preemptive priority factor which expresses the relative importance of various goals, $P_j \gg P_{j+1}$, 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, $\min d_i^+ \vee 0$ is the β_i^+-optimistic positive deviation from the target of goal i, $\min d_i^- \vee 0$ is the β_i^--optimistic negative deviation from the target of goal i, 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, p is the number of real constraints.

Minimax Chance-Constrained Programming

In fact, maximax CCP models are essentially a type of optimistic models which maximize the maximum possible return. This section introduces a spectrum of minimax CCP constructed by Liu [172], which will select the alternative that provides the best of the worst possible return.

If we want to maximize the pessimistic return, then we have the following minimax CCP,

$$\begin{cases} \max\limits_{\boldsymbol{x}} \min\limits_{\overline{f}} \overline{f} \\ \text{subject to:} \\ \quad \operatorname{Cr}\left\{f(\boldsymbol{x},\boldsymbol{\xi}) \le \overline{f}\right\} \ge \beta \\ \quad \operatorname{Cr}\left\{g_j(\boldsymbol{x},\boldsymbol{\xi}) \le 0, j=1,2,\cdots,p\right\} \ge \alpha \end{cases} \tag{5.20}$$

where $\min \overline{f}$ is the β-pessimistic return.

If there are multiple objectives, we may employ the following minimax CCMOP,

$$\begin{cases} \max\limits_{\boldsymbol{x}} \left[\min\limits_{\overline{f}_1} \overline{f}_1,\ \min\limits_{\overline{f}_2} \overline{f}_2,\ \cdots,\ \min\limits_{\overline{f}_m} \overline{f}_m\right] \\ \text{subject to:} \\ \quad \operatorname{Cr}\left\{f_i(\boldsymbol{x},\boldsymbol{\xi}) \le \overline{f}_i\right\} \ge \beta_i, \quad i=1,2,\cdots,m \\ \quad \operatorname{Cr}\left\{g_j(\boldsymbol{x},\boldsymbol{\xi}) \le 0\right\} \ge \alpha_j, \quad j=1,2,\cdots,p \end{cases} \tag{5.21}$$

where α_j and β_i are confidence levels, and $\min \overline{f}_i$ are the β_i-pessimistic values to the return functions $f_i(\boldsymbol{x},\boldsymbol{\xi})$, $i=1,2,\cdots,m$, respectively.

According to the priority structure and target levels set by the decision-maker, the minimax CCGP is written as follows,

$$\begin{cases} \min\limits_{\boldsymbol{x}} \sum\limits_{j=1}^{l} P_j \sum\limits_{i=1}^{m} \left[u_{ij}\left(\max\limits_{d_i^+} d_i^+ \vee 0\right) + v_{ij}\left(\max\limits_{d_i^-} d_i^- \vee 0\right)\right] \\ \text{subject to:} \\ \quad \operatorname{Cr}\left\{f_i(\boldsymbol{x},\boldsymbol{\xi}) - b_i \ge d_i^+\right\} \ge \beta_i^+, \quad i=1,2,\cdots,m \\ \quad \operatorname{Cr}\left\{b_i - f_i(\boldsymbol{x},\boldsymbol{\xi}) \ge d_i^-\right\} \ge \beta_i^-, \quad i=1,2,\cdots,m \\ \quad \operatorname{Cr}\left\{g_j(\boldsymbol{x},\boldsymbol{\xi}) \le 0\right\} \ge \alpha_j, \qquad j=1,2,\cdots,p \end{cases} \tag{5.22}$$

where P_j is the preemptive priority factor which expresses the relative importance of various goals, $P_j \gg P_{j+1}$, 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, $\max d_i^+ \vee 0$ is the β_i^+-pessimistic positive deviation from the target of goal i, $\max d_i^- \vee 0$ is the β_i^--pessimistic 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.

Crisp Equivalents

One way of solving fuzzy CCP is to convert the chance constraint

$$\operatorname{Cr}\left\{g(\boldsymbol{x},\boldsymbol{\xi}) \le 0\right\} \ge \alpha \tag{5.23}$$

into its crisp equivalent and then solve the equivalent crisp model by the traditional solution process. Please note that

(a) the system constraints $\mathrm{Cr}\{g_j(\boldsymbol{x},\boldsymbol{\xi}) \le 0\} \ge \alpha_j$, $j = 1, 2, \cdots, p$ are a set of form (5.23);

(b) the objective constraint $\mathrm{Cr}\{f(\boldsymbol{x},\boldsymbol{\xi}) \ge \overline{f}\} \ge \beta$ coincides with the form (5.23) by defining $g(\boldsymbol{x},\boldsymbol{\xi}) = \overline{f} - f(\boldsymbol{x},\boldsymbol{\xi})$;

(c) the fuzzy constraint $\mathrm{Cr}\{f(\boldsymbol{x},\boldsymbol{\xi}) \le \overline{f}\} \ge \beta$ coincides with the form (5.23) by defining $g(\boldsymbol{x},\boldsymbol{\xi}) = f(\boldsymbol{x},\boldsymbol{\xi}) - \overline{f}$;

(d) $\mathrm{Cr}\{b - f(\boldsymbol{x},\boldsymbol{\xi}) \le d^-\} \ge \beta$ and $\mathrm{Cr}\{f(\boldsymbol{x},\boldsymbol{\xi}) - b \le d^+\} \ge \beta$ coincide with the form (5.23) by defining $g(\boldsymbol{x},\boldsymbol{\xi}) = b - f(\boldsymbol{x},\boldsymbol{\xi}) - d^-$ and $g(\boldsymbol{x},\boldsymbol{\xi}) = f(\boldsymbol{x},\boldsymbol{\xi}) - b - d^+$, respectively; and

(e) $\mathrm{Cr}\{b - f(\boldsymbol{x},\boldsymbol{\xi}) \ge d^-\} \ge \beta$ and $\mathrm{Cr}\{f(\boldsymbol{x},\boldsymbol{\xi}) - b \ge d^+\} \ge \beta$ coincide with the form (5.23) by defining $g(\boldsymbol{x},\boldsymbol{\xi}) = f(\boldsymbol{x},\boldsymbol{\xi}) + d^- - b$ and $g(\boldsymbol{x},\boldsymbol{\xi}) = b - f(\boldsymbol{x},\boldsymbol{\xi}) + d^+$, respectively.

This section presents some useful results.

Theorem 5.4 *Assume that the fuzzy vector $\boldsymbol{\xi}$ degenerates to a fuzzy variable ξ with continuous membership function μ, and the function $g(\boldsymbol{x},\boldsymbol{\xi})$ has the form $g(\boldsymbol{x},\boldsymbol{\xi}) = h(\boldsymbol{x}) - \xi$. Then $\mathrm{Cr}\{g(\boldsymbol{x},\boldsymbol{\xi}) \le 0\} \ge \alpha$ if and only if $h(\boldsymbol{x}) \le K_\alpha$, where*

$$K_\alpha = \begin{cases} \sup\left\{K | K = \mu^{-1}(2\alpha)\right\}, & \text{if } \alpha < 1/2 \\ \inf\left\{K | K = \mu^{-1}(2(1-\alpha))\right\}, & \text{if } \alpha \ge 1/2. \end{cases} \tag{5.24}$$

Proof: It is easy to verify the theorem by the relation between credibility measure and membership function.

Theorem 5.5 *Assume that the function $g(\boldsymbol{x},\boldsymbol{\xi})$ can be rewritten as,*

$$g(\boldsymbol{x},\boldsymbol{\xi}) = h_1(\boldsymbol{x})\xi_1 + h_2(\boldsymbol{x})\xi_2 + \cdots + h_t(\boldsymbol{x})\xi_t + h_0(\boldsymbol{x})$$

where ξ_k are trapezoidal fuzzy variables $(r_{k1}, r_{k2}, r_{k3}, r_{k4})$, $k = 1, 2, \cdots, t$, respectively. We define two functions $h_k^+(\boldsymbol{x}) = h_k(\boldsymbol{x}) \vee 0$ and $h_k^-(\boldsymbol{x}) = -(h_k(\boldsymbol{x}) \wedge 0)$ for $k = 1, 2, \cdots, t$. Then we have
(a) when $\alpha < 1/2$, $\mathrm{Cr}\{g(\boldsymbol{x},\boldsymbol{\xi}) \le 0\} \ge \alpha$ if and only if

$$\begin{aligned} (1-2\alpha) \sum_{k=1}^{t} \left[r_{k1} h_k^+(\boldsymbol{x}) - r_{k4} h_k^-(\boldsymbol{x})\right] \\ +2\alpha \sum_{k=1}^{t} \left[r_{k2} h_k^+(\boldsymbol{x}) - r_{k3} h_k^-(\boldsymbol{x})\right] + h_0(\boldsymbol{x}) \le 0; \end{aligned} \tag{5.25}$$

(b) when $\alpha \geq 1/2$, $\mathrm{Cr}\{g(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0\} \geq \alpha$ *if and only if*

$$\begin{aligned}(2-2\alpha)\sum_{k=1}^{t}\left[r_{k3}h_k^+(\boldsymbol{x})-r_{k2}h_k^-(\boldsymbol{x})\right]\\ +(2\alpha-1)\sum_{k=1}^{t}\left[r_{k4}h_k^+(\boldsymbol{x})-r_{k1}h_k^-(\boldsymbol{x})\right]+h_0(\boldsymbol{x})\leq 0.\end{aligned} \tag{5.26}$$

Proof: It is clear that the functions $h_k^+(\boldsymbol{x})$ and $h_k^-(\boldsymbol{x})$ are all nonnegative and $h_k(\boldsymbol{x}) = h_k^+(\boldsymbol{x}) - h_k^-(\boldsymbol{x})$. Thus we have

$$\begin{aligned} g(\boldsymbol{x}, \boldsymbol{\xi}) &= \sum_{k=1}^{t} h_k(\boldsymbol{x})\xi_k + h_0(\boldsymbol{x}) \\ &= \sum_{k=1}^{t}\left[h_k^+(\boldsymbol{x}) - h_k^-(\boldsymbol{x})\right]\xi_k + h_0(\boldsymbol{x}) \\ &= \sum_{k=1}^{t}\left[h_k^+(\boldsymbol{x})\xi_k + h_k^-(\boldsymbol{x})\xi_k'\right] + h_0(\boldsymbol{x}) \end{aligned}$$

where ξ_k' are also trapezoidal fuzzy variables,

$$\xi_k' = (-r_{k4}, -r_{k3}, -r_{k2}, -r_{k1}), \quad k = 1, 2, \cdots, t.$$

By the addition and multiplication operations of trapezoidal fuzzy variables, the function $g(\boldsymbol{x}, \boldsymbol{\xi})$ is also a trapezoidal fuzzy variable determined by the quadruple

$$g(\boldsymbol{x}, \boldsymbol{\xi}) = \left(\begin{array}{c} \sum_{k=1}^{t}\left[r_{k1}h_k^+(\boldsymbol{x}) - r_{k4}h_k^-(\boldsymbol{x})\right] + h_0(\boldsymbol{x}) \\ \sum_{k=1}^{t}\left[r_{k2}h_k^+(\boldsymbol{x}) - r_{k3}h_k^-(\boldsymbol{x})\right] + h_0(\boldsymbol{x}) \\ \sum_{k=1}^{t}\left[r_{k3}h_k^+(\boldsymbol{x}) - r_{k2}h_k^-(\boldsymbol{x})\right] + h_0(\boldsymbol{x}) \\ \sum_{k=1}^{t}\left[r_{k4}h_k^+(\boldsymbol{x}) - r_{k1}h_k^-(\boldsymbol{x})\right] + h_0(\boldsymbol{x}) \end{array}\right)^T.$$

It follows that the results hold.

5.4 Dependent-Chance Programming

Liu [173] provided a fuzzy dependent-chance programming (DCP) theory in which the underlying philosophy is based on selecting the decision with maximum credibility to meet the event.

Principle of Uncertainty

Uncertain environment, event and chance function are key elements in the framework of DCP in a stochastic environment. Let us redefine them in fuzzy environments.

Definition 5.9 *By uncertain environment (in this case the fuzzy environment) we mean the fuzzy constraints represented by*

$$g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, \quad j = 1, 2, \cdots, p \tag{5.27}$$

where $\boldsymbol{x}$ is a decision vector, and $\boldsymbol{\xi}$ is a fuzzy vector.

Definition 5.10 *By event we mean a system of fuzzy inequalities,*

$$h_k(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, \quad k = 1, 2, \cdots, q \tag{5.28}$$

where $\boldsymbol{x}$ is a decision vector, and $\boldsymbol{\xi}$ is a fuzzy vector.

Definition 5.11 *The chance function of an event $\mathcal{E}$ characterized by (5.28) is defined as the credibility measure of the event $\mathcal{E}$, i.e.,*

$$f(\boldsymbol{x}) = \mathrm{Cr}\{h_k(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0,\, k = 1, 2, \cdots, q\} \tag{5.29}$$

subject to the uncertain environment (5.27).

The concepts of the support, dependent support, active constraint, and dependent constraint are the same with those in stochastic case. Thus, for each decision $\boldsymbol{x}$ and realization $\boldsymbol{\xi}$, an event $\mathcal{E}$ is said to be consistent in the uncertain environment if the following two conditions hold: (i) $h_k(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0$, $k = 1, 2, \cdots, q$; and (ii) $g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0$, $j \in J$, where J is the index set of all dependent constraints. In order to compute the chance function of a fuzzy event, we need the following principle of uncertainty.

Principle of Uncertainty: *The chance of a fuzzy event is the credibility that the event is consistent in the uncertain environment.*

Assume that there are m events $\mathcal{E}_i$ characterized by $h_{ik}(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, k = 1, 2, \cdots, q_i$ for $i = 1, 2, \cdots, m$ in the uncertain environment $g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, j = 1, 2, \cdots, p$. The principle of uncertainty implies that the chance function of the ith event $\mathcal{E}_i$ in the uncertain environment is

$$f_i(\boldsymbol{x}) = \mathrm{Cr}\left\{\begin{matrix} h_{ik}(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, k = 1, 2, \cdots, q_i \\ g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, j \in J_i \end{matrix}\right\} \tag{5.30}$$

where J_i are defined by

$$J_i = \left\{ j \in \{1, 2, \cdots, p\} \;\middle|\; g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0 \text{ is a dependent constraint of } \mathcal{E}_i \right\}$$

for $i = 1, 2, \cdots, m$.

General Models

A typical formulation of DCP in a fuzzy environment is given as follows:

$$\begin{cases} \max \operatorname{Cr}\{h_k(\boldsymbol{x},\boldsymbol{\xi}) \le 0, k=1,2,\cdots,q\} \\ \text{subject to:} \\ \quad g_j(\boldsymbol{x},\boldsymbol{\xi}) \le 0, \quad j=1,2,\cdots,p \end{cases} \tag{5.31}$$

where $\boldsymbol{x}$ is an n-dimensional decision vector, $\boldsymbol{\xi}$ is a fuzzy vector, the event $\mathcal{E}$ is characterized by $h_k(\boldsymbol{x},\boldsymbol{\xi}) \le 0, k=1,2,\cdots,q$, and the uncertain environment is described by the fuzzy constraints $g_j(\boldsymbol{x},\boldsymbol{\xi}) \le 0, j=1,2,\cdots,p$.

Fuzzy DCP (5.31) reads as "maximizing the credibility of the fuzzy event $h_k(\boldsymbol{x},\boldsymbol{\xi}) \le 0, k=1,2,\cdots,q$ subject to the uncertain environment $g_j(\boldsymbol{x},\boldsymbol{\xi}) \le 0, j=1,2,\cdots,p$".

Since a complex decision system usually undertakes multiple tasks, there undoubtedly exist multiple potential objectives. A typical formulation of fuzzy dependent-chance multiobjective programming (DCMOP) is given as follows,

$$\begin{cases} \max \begin{bmatrix} \operatorname{Cr}\{h_{1k}(\boldsymbol{x},\boldsymbol{\xi}) \le 0, k=1,2,\cdots,q_1\} \\ \operatorname{Cr}\{h_{2k}(\boldsymbol{x},\boldsymbol{\xi}) \le 0, k=1,2,\cdots,q_2\} \\ \cdots \\ \operatorname{Cr}\{h_{mk}(\boldsymbol{x},\boldsymbol{\xi}) \le 0, k=1,2,\cdots,q_m\} \end{bmatrix} \\ \text{subject to:} \\ \quad g_j(\boldsymbol{x},\boldsymbol{\xi}) \le 0, \quad j=1,2,\cdots,p \end{cases} \tag{5.32}$$

where $h_{ik}(\boldsymbol{x},\boldsymbol{\xi}) \le 0, k=1,2,\cdots,q_i$ represent events $\mathcal{E}_i$ for $i=1,2,\cdots,m$, respectively.

Dependent-chance goal programming (DCGP) in fuzzy environment may be considered as an extension of goal programming in a complex fuzzy decision system. When some management targets are given, the objective function may minimize the deviations, positive, negative, or both, with a certain priority structure. Thus we can formulate a fuzzy decision system as a DCGP according to the priority structure and target levels set by the decision-maker,

$$\begin{cases} \min \sum_{j=1}^{l} P_j \sum_{i=1}^{m} (u_{ij} d_i^+ \vee 0 + v_{ij} d_i^- \vee 0) \\ \text{subject to:} \\ \quad \operatorname{Cr}\{h_{ik}(\boldsymbol{x},\boldsymbol{\xi}) \le 0, k=1,2,\cdots,q_i\} - b_i = d_i^+, \quad i=1,2,\cdots,m \\ \quad b_i - \operatorname{Cr}\{h_{ik}(\boldsymbol{x},\boldsymbol{\xi}) \le 0, k=1,2,\cdots,q_i\} = d_i^-, \quad i=1,2,\cdots,m \\ \quad g_j(\boldsymbol{x},\boldsymbol{\xi}) \le 0, \qquad\qquad\qquad\qquad\qquad\qquad\quad j=1,2,\cdots,p \end{cases}$$

where P_j is the preemptive priority factor which expresses the relative importance of various goals, $P_j \gg P_{j+1}$, 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, $d_i^+ \vee 0$ is the positive deviation from the target of goal i, $d_i^- \vee 0$ is the negative deviation from the target of goal i, 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.

5.5 Hybrid Intelligent Algorithm

In order to solve general fuzzy programming models, we must deal with the following three types of uncertain function:

$$\begin{aligned} U_1 &: \boldsymbol{x} \to E[f(\boldsymbol{x}, \boldsymbol{\xi})], \\ U_2 &: \boldsymbol{x} \to \mathrm{Cr}\left\{g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, j = 1, 2, \cdots, p\right\}, \\ U_3 &: \boldsymbol{x} \to \max\left\{\overline{f} \mid \mathrm{Cr}\left\{f(\boldsymbol{x}, \boldsymbol{\xi}) \geq \overline{f}\right\} \geq \alpha\right\}. \end{aligned} \tag{5.33}$$

In order to compute the expected value $U_1(\boldsymbol{x})$, the following procedure may be used. We randomly generate θ_k from the credibility space $(\Theta, \mathcal{P}, \mathrm{Cr})$, write $\nu_k = (2\mathrm{Cr}\{\theta_k\}) \wedge 1$ and produce $\boldsymbol{\xi}(\theta_k)$, $k = 1, 2, \cdots, N$, respectively. *Equivalently, we randomly generate $\boldsymbol{\xi}(\theta_k)$ and write $\nu_k = \mu(\boldsymbol{\xi}(\theta_k))$ for $k = 1, 2, \cdots, N$, where μ is the membership function of $\boldsymbol{\xi}$.* Then for any number $r \geq 0$, the credibility $\mathrm{Cr}\{f(\boldsymbol{x}, \boldsymbol{\xi}) \geq r\}$ can be estimated by

$$\frac{1}{2}\left(\max_{1\leq k\leq N}\left\{\nu_k \mid f(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k)) \geq r\right\} + \min_{1\leq k\leq N}\left\{1 - \nu_k \mid f(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k)) < r\right\}\right)$$

and for any number $r < 0$, the credibility $\mathrm{Cr}\{f(\boldsymbol{x}, \boldsymbol{\xi}) \leq r\}$ can be estimated by

$$\frac{1}{2}\left(\max_{1\leq k\leq N}\left\{\nu_k \mid f(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k)) \leq r\right\} + \min_{1\leq k\leq N}\left\{1 - \nu_k \mid f(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k)) > r\right\}\right)$$

provided that N is sufficiently large. Thus $U_1(\boldsymbol{x})$ may be estimated by the following procedure.

Algorithm 5.1 (Fuzzy Simulation for $U_1(\boldsymbol{x})$)

Step 1. Set $e = 0$.

Step 2. Randomly generate θ_k from the credibility space $(\Theta, \mathcal{P}, \mathrm{Cr})$, write $\nu_k = (2\mathrm{Cr}\{\theta_k\}) \wedge 1$ and produce $\boldsymbol{\xi}(\theta_k)$, $k = 1, 2, \cdots, N$, respectively. *Equivalently, we randomly generate $\boldsymbol{\xi}(\theta_k)$ and write $\nu_k = \mu(\boldsymbol{\xi}(\theta_k))$ for $k = 1, 2, \cdots, N$, where μ is the membership function of $\boldsymbol{\xi}$.*

Step 3. Set two numbers $a = f(\boldsymbol{x}, \boldsymbol{\xi}(\theta_1)) \wedge f(\boldsymbol{x}, \boldsymbol{\xi}(\theta_2)) \wedge \cdots \wedge f(\boldsymbol{x}, \boldsymbol{\xi}(\theta_N))$ and $b = f(\boldsymbol{x}, \boldsymbol{\xi}(\theta_1)) \vee f(\boldsymbol{x}, \boldsymbol{\xi}(\theta_2)) \vee \cdots \vee f(\boldsymbol{x}, \boldsymbol{\xi}(\theta_N))$.

Step 4. Randomly generate r from $[a, b]$.

Step 5. If $r \geq 0$, then $e \leftarrow e + \mathrm{Cr}\{f(\boldsymbol{x}, \boldsymbol{\xi}) \geq r\}$.
Step 6. If $r < 0$, then $e \leftarrow e - \mathrm{Cr}\{f(\boldsymbol{x}, \boldsymbol{\xi}) \leq r\}$.
Step 7. Repeat the fourth to sixth steps for N times.
Step 8. $U_1(\boldsymbol{x}) = a \vee 0 + b \wedge 0 + e \cdot (b-a)/N$.

In order to compute the uncertain function $U_2(\boldsymbol{x})$, we randomly generate θ_k from the credibility space $(\Theta, \mathcal{P}, \mathrm{Cr})$, write $\nu_k = (2\mathrm{Cr}\{\theta_k\}) \wedge 1$ and produce $\boldsymbol{\xi}(\theta_k)$, $k = 1, 2, \cdots, N$, respectively. *Equivalently, we randomly generate $\boldsymbol{\xi}(\theta_k)$ and write $\nu_k = \mu(\boldsymbol{\xi}(\theta_k))$ for $k = 1, 2, \cdots, N$, where μ is the membership function of $\boldsymbol{\xi}$.* Then the credibility $U_2(\boldsymbol{x})$ can be estimated by the formula,

$$\frac{1}{2}\left(\max_{1\leq k\leq N}\left\{\nu_k \mid \begin{array}{r} g_j(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k)) \leq 0 \\ j = 1, 2, \cdots, p \end{array}\right\} + \min_{1\leq k\leq N}\left\{1 - \nu_k \mid \begin{array}{r} g_j(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k)) > 0 \\ \text{for some } j \end{array}\right\}\right).$$

Algorithm 5.2 (Fuzzy Simulation for $U_2(\boldsymbol{x})$)
Step 1. Randomly generate θ_k from the credibility space $(\Theta, \mathcal{P}, \mathrm{Cr})$, write $\nu_k = (2\mathrm{Cr}\{\theta_k\}) \wedge 1$ and produce $\boldsymbol{\xi}(\theta_k)$, $k = 1, 2, \cdots, N$, respectively. *Equivalently, we randomly generate $\boldsymbol{\xi}(\theta_k)$ and write $\nu_k = \mu(\boldsymbol{\xi}(\theta_k))$ for $k = 1, 2, \cdots, N$, where μ is the membership function of $\boldsymbol{\xi}$.*
Step 2. Return $U_2(\boldsymbol{x})$ via the estimation formula.

In order to compute the uncertain function $U_3(\boldsymbol{x})$, we randomly generate θ_k from the credibility space $(\Theta, \mathcal{P}, \mathrm{Cr})$, write $\nu_k = (2\mathrm{Cr}\{\theta_k\}) \wedge 1$ and produce $\boldsymbol{\xi}(\theta_k)$, $k = 1, 2, \cdots, N$, respectively. *Equivalently, we randomly generate $\boldsymbol{\xi}(\theta_k)$ and write $\nu_k = \mu(\boldsymbol{\xi}(\theta_k))$ for $k = 1, 2, \cdots, N$, where μ is the membership function of $\boldsymbol{\xi}$.* For any number r, we set

$$L(r) = \frac{1}{2}\left(\max_{1\leq k\leq N}\left\{\nu_k \mid f(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k)) \geq r\right\} + \min_{1\leq k\leq N}\left\{1 - \nu_k \mid f(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k)) < r\right\}\right).$$

It follows from monotonicity that we may employ bisection search to find the maximal value r such that $L(r) \geq \alpha$. This value is an estimation of $U_3(\boldsymbol{x})$. We summarize this process as follows.

Algorithm 5.3 (Fuzzy Simulation for $U_3(\boldsymbol{x})$)
Step 1. Randomly generate θ_k from the credibility space $(\Theta, \mathcal{P}, \mathrm{Cr})$, write $\nu_k = (2\mathrm{Cr}\{\theta_k\}) \wedge 1$ and produce $\boldsymbol{\xi}(\theta_k)$, $k = 1, 2, \cdots, N$, respectively. *Equivalently, we randomly generate $\boldsymbol{\xi}(\theta_k)$ and write $\nu_k = \mu(\boldsymbol{\xi}(\theta_k))$ for $k = 1, 2, \cdots, N$, where μ is the membership function of $\boldsymbol{\xi}$.*
Step 2. Find the maximal value r such that $L(r) \geq \alpha$ holds.
Step 3. Return r.

Now we integrate fuzzy simulation, NN and GA to produce a hybrid intelligent algorithm for solving fuzzy programming models.

Algorithm 5.4 (Hybrid Intelligent Algorithm)
Step 1. Generate training input-output data for uncertain functions like

$$U_1 : \boldsymbol{x} \to E[f(\boldsymbol{x}, \boldsymbol{\xi})],$$
$$U_2 : \boldsymbol{x} \to \mathrm{Cr}\left\{g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, j = 1, 2, \cdots, p\right\},$$
$$U_3 : \boldsymbol{x} \to \max\left\{\overline{f} \mid \mathrm{Cr}\left\{f(\boldsymbol{x}, \boldsymbol{\xi}) \geq \overline{f}\right\} \geq \alpha\right\}$$

by the fuzzy simulation.
Step 2. Train a neural network to approximate the uncertain functions according to the generated training input-output data.
Step 3. Initialize *pop_size* chromosomes whose feasibility may be checked by the trained neural network.
Step 4. Update the chromosomes by crossover and mutation operations and the trained neural network may be employed to check the feasibility of offsprings.
Step 5. Calculate the objective values for all chromosomes by the trained neural network.
Step 6. Compute the fitness of each chromosome by rank-based evaluation function based on the objective values.
Step 7. Select the chromosomes by spinning the roulette wheel.
Step 8. Repeat the fourth to seventh steps a given number of cycles.
Step 9. Report the best chromosome as the optimal solution.

5.6 Numerical Experiments

In order to illustrate its effectiveness, a set of numerical examples has been done, and the results are successful. Here we give some numerical examples which are all performed on a personal computer with the following parameters: the population size is 30, the probability of crossover P_c is 0.3, the probability of mutation P_m is 0.2, and the parameter a in the rank-based evaluation function is 0.05.

Example 5.21: Consider first the following fuzzy EVM,

$$\begin{cases} \max E\left[\sqrt{|x_1 + \xi_1| + |x_2 + \xi_2| + |x_3 + \xi_3|}\right] \\ \text{subject to:} \\ \quad x_1^2 + x_2^2 + x_3^2 \leq 10 \end{cases}$$

where ξ_1, ξ_2 and ξ_3 are triangular fuzzy variables $(1, 2, 3)$, $(2, 3, 4)$, and $(3, 4, 5)$, respectively.

In order to solve this model, we first generate input-output data for the

uncertain function

$$U : \boldsymbol{x} \to E\left[\sqrt{|x_1+\xi_1|+|x_2+\xi_2|+|x_3+\xi_3|}\right]$$

by fuzzy simulation. Then we train an NN (3 input neurons, 5 hidden neurons, 1 output neuron) to approximate the function $U(\boldsymbol{x})$. After that, the trained NN is embedded into a GA to produce a hybrid intelligent algorithm.

A run of the hybrid intelligent algorithm (6000 cycles in simulation, 2000 data in NN, 1000 generations in GA) shows that the optimal solution is

$$x_1^* = 1.8310, \quad x_2^* = 1.8417, \quad x_3^* = 1.8043$$

whose objective value is 3.80.

Example 5.22: Let us consider the following single-objective fuzzy CCP,

$$\begin{cases} \max \overline{f} \\ \text{subject to:} \\ \quad \mathrm{Cr}\left\{\sqrt{x_1+\xi_1}+\sqrt{x_2+\xi_2}+\sqrt{x_3+\xi_3} \geq \overline{f}\right\} \geq 0.9 \\ \quad \mathrm{Cr}\left\{\sqrt{(x_1+\xi_1)^2+(x_2+\xi_2)^2+(x_3+\xi_3)^2} \leq 6\right\} \geq 0.8 \\ \quad x_1, x_2, x_3 \geq 0 \end{cases} \tag{5.34}$$

where ξ_1, ξ_2 and ξ_3 are assumed to triangular fuzzy variables $(0,1,2)$, $(1,2,3)$ and $(2,3,4)$, respectively.

In order to solve this model, we generate training input-output data for the uncertain function $U : (\boldsymbol{x}) \to (U_1(\boldsymbol{x}), U_2(\boldsymbol{x}))$, where

$$U_1(\boldsymbol{x}) = \max\left\{\overline{f} \mid \mathrm{Cr}\left\{\sqrt{x_1+\xi_1}+\sqrt{x_2+\xi_2}+\sqrt{x_3+\xi_3} \geq \overline{f}\right\} \geq 0.9\right\},$$
$$U_2(\boldsymbol{x}) = \mathrm{Cr}\left\{\sqrt{(x_1+\xi_1)^2+(x_2+\xi_2)^2+(x_3+\xi_3)^2} \leq 6\right\}.$$

Then we train an NN (3 input neurons, 6 hidden neurons, 2 output neurons) to approximate the uncertain function U. Finally, we integrate the trained NN and GA to produce a hybrid intelligent algorithm.

A run of the hybrid intelligent algorithm (6000 cycles in simulation, 2000 training data in NN, 1500 generations in GA) shows that the optimal solution is

$$(x_1^*, x_2^*, x_3^*) = (1.9780, 0.6190, 0.0000)$$

with objective value $\overline{f}^* = 5.02$.

Example 5.23: We now consider the following CCGP model,

$$\begin{cases} \text{lexmin} \left\{ d_1^- \vee 0, d_2^- \vee 0, d_3^- \vee 0 \right\} \\ \text{subject to:} \\ \quad \text{Cr} \left\{ 3 - (x_1^2 \xi_1 + x_2 \tau_1 + x_3 \eta_1^2) \leq d_1^- \right\} \geq 0.90 \\ \quad \text{Cr} \left\{ 4 - (x_1 \xi_2 + x_2^2 \tau_2^2 + x_3 \eta_2) \leq d_2^- \right\} \geq 0.85 \\ \quad \text{Cr} \left\{ 6 - (x_1 \xi_3^2 + x_2 \tau_3 + x_3^2 \eta_3) \leq d_3^- \right\} \geq 0.80 \\ \quad x_1 + x_2 + x_3 = 1 \\ \quad x_1, x_2, x_3 \geq 0 \end{cases} \tag{5.35}$$

where ξ_1, ξ_2, ξ_3 are fuzzy variables with membership functions $\exp[-|x-1|]$, $\exp[-|x-2|]$, $\exp[-|x-3|]$, τ_1, τ_2, τ_3 are triangular fuzzy variables $(1,2,3)$, $(2,3,4)$, $(3,4,5)$, η_1, η_2, η_3 are trapezoidal fuzzy variables $(2,3,4,5)$, $(3,4,5,6)$, $(4,5,6,7)$, respectively.

In order to solve this problem, we employ fuzzy simulation to generate input-output data for the uncertain function $U : (\boldsymbol{x}) \to (U_1(\boldsymbol{x}), U_2(\boldsymbol{x}), U_3(\boldsymbol{x}))$, where

$$\begin{aligned} U_1(\boldsymbol{x}) &= \max \left\{ d \mid \text{Cr} \left\{ x_1^2 \xi_1 + x_2 \tau_1 + x_3 \eta_1^2 \geq d \right\} \geq 0.90 \right\}, \\ U_2(\boldsymbol{x}) &= \max \left\{ d \mid \text{Cr} \left\{ x_1 \xi_2 + x_2^2 \tau_2^2 + x_3 \eta_2 \geq d \right\} \geq 0.85 \right\}, \\ U_3(\boldsymbol{x}) &= \max \left\{ d \mid \text{Cr} \left\{ x_1 \xi_3^2 + x_2 \tau_3 + x_3^2 \eta_3 \geq d \right\} \geq 0.80 \right\}. \end{aligned}$$

Then we train an NN (3 input neurons, 8 hidden neurons, 3 output neurons) to approximate the uncertain function U. Note that

$$d_1^- = [3 - U_1(\boldsymbol{x})] \vee 0, \quad d_2^- = [4 - U_2(\boldsymbol{x})] \vee 0, \quad d_3^- = [6 - U_3(\boldsymbol{x})] \vee 0.$$

Finally, we integrate the trained NN and GA to produce a hybrid intelligent algorithm.

A run of the hybrid intelligent algorithm (5000 cycles in simulation, 3000 training data in NN, 3000 generations in GA) shows that the optimal solution is

$$(x_1^*, x_2^*, x_3^*) = (0.2910, 0.5233, 0.1857)$$

which can satisfy the first two goals, but the negative deviation of the third goal is 0.57.

Example 5.24: Let us now turn our attention to the following DCGP,

$$\begin{cases} \text{lexmin } \{d_1^- \vee 0, d_2^- \vee 0, d_3^- \vee 0\} \\ \text{subject to:} \\ \quad 0.95 - \text{Cr}\{x_1 + x_3^2 = 6\} = d_1^- \\ \quad 0.90 - \text{Cr}\{x_2 + x_5^2 = 5\} = d_2^- \\ \quad 0.85 - \text{Cr}\{x_4 + x_6^2 = 4\} = d_3^- \\ \quad x_1 + x_2 \leq \tilde{a} \\ \quad x_3 + x_4 \leq \tilde{b} \\ \quad x_5 \leq \tilde{c} \\ \quad x_6 \leq \tilde{d} \\ \quad x_i \geq 0, \quad i = 1, 2, \cdots, 6 \end{cases}$$

where $\tilde{a}, \tilde{b}, \tilde{c}$ are triangular fuzzy variables (3,4,5), (2,3,4), (0,1,2), respectively, and $\tilde{d}$ is a fuzzy variable with membership function $\mu_{\tilde{d}}(r) = 1/[1 + (r-1)^2]$.

In the first priority level, there is only one event denoted by $\mathcal{E}_1$ in the fuzzy environment, which should be fulfilled by $x_1 + x_3^2 = 6$. It is clear that the support $\mathcal{E}_1^* = \{x_1, x_3\}$ and the dependent support $\mathcal{E}_1^{**} = \{x_1, x_2, x_3, x_4\}$. It follows from the principle of uncertainty that the chance function $f_1(\boldsymbol{x})$ of the event $\mathcal{E}_1$ is

$$f_1(\boldsymbol{x}) = \text{Cr}\left\{\begin{array}{l} x_1 + x_3^2 = 6 \\ x_1 + x_2 \leq \tilde{a} \\ x_3 + x_4 \leq \tilde{b} \\ x_1, x_2, x_3, x_4 \geq 0 \end{array}\right\}.$$

At the second priority level, there is an event $\mathcal{E}_2$ which will be fulfilled by $x_2 + x_5^2 = 5$. The support $\mathcal{E}_2^* = \{x_2, x_5\}$ and the dependent support $\mathcal{E}_2^{**} = \{x_1, x_2, x_5\}$. It follows from the principle of uncertainty that the chance function $f_2(\boldsymbol{x})$ of the event $\mathcal{E}_2$ is

$$f_2(\boldsymbol{x}) = \text{Cr}\left\{\begin{array}{l} x_2 + x_5^2 = 5 \\ x_1 + x_2 \leq \tilde{a} \\ x_5 \leq \tilde{c} \\ x_1, x_2, x_5 \geq 0 \end{array}\right\}.$$

At the third priority level, there is an event $\mathcal{E}_3$ which will be fulfilled by $x_4 + x_6^2 = 4$. The support $\mathcal{E}_3^* = \{x_4, x_6\}$ and the dependent support $\mathcal{E}_3^{**} = \{x_3, x_4, x_6\}$. It follows from the principle of uncertainty that the chance function $f_3(\boldsymbol{x})$ of the event $\mathcal{E}_3$ is

$$f_3(\boldsymbol{x}) = \text{Cr}\left\{\begin{array}{l} x_4 + x_6^2 = 4 \\ x_3 + x_4 \leq \tilde{b} \\ x_6 \leq \tilde{d} \\ x_3, x_4, x_6 \geq 0 \end{array}\right\}.$$

We encode a solution by a chromosome $V = (v_1, v_2, v_3)$, and decode it into a feasible solution in the following way,

$$\begin{array}{lll} x_1 = v_1, & x_2 = v_2, & x_3 = \sqrt{6 - v_1} \\ x_4 = v_3, & x_5 = \sqrt{5 - v_2}, & x_6 = \sqrt{4 - v_3} \end{array}$$

which ensures that $x_1 + x_3^2 = 6$, $x_2 + x_5^2 = 5$ and $x_4 + x_6^2 = 4$.

At first, we employ fuzzy simulation to generate input-output data for the chance function

$$U : (v_1, v_2, v_3) \to (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), f_3(\boldsymbol{x})) .$$

Then we train an NN (3 input neurons, 8 hidden neurons, 3 output neurons) to approximate it. Finally, we embed the trained NN into a GA to produce a hybrid intelligent algorithm.

A run of the hybrid intelligent algorithm (5000 cycles in simulation, 2000 data in NN, 1000 generations in GA) shows that the optimal solution is

$$\boldsymbol{x}^* = (0.2097, 3.8263, 2.4063, 0.6407, 1.0833, 1.8328)$$

which can satisfy the first and second goals, but the third objective is 0.25.

Chapter 6

Hybrid Programming

In many cases, fuzziness and randomness simultaneously appear in a system. In order to describe this phenomena, a fuzzy random variable was introduced by Kwakernaak [144][145] as a random element taking "fuzzy variable" values. This concept was then developed by several researchers such as Puri and Ralescu [258], Kruse and Meyer [141], and Liu and Liu [197] according to different requirements of measurability. By fuzzy random programming we mean the optimization theory in fuzzy random environments. Liu and Liu [196] presented a spectrum of fuzzy random expected value model (EVM), Liu [180] initialized a general framework of fuzzy random chance-constrained programming (CCP), and Liu [181] introduced the concepts of uncertain environment and chance function for fuzzy random decision problems, and constructed a theoretical framework of fuzzy random dependent-chance programming (DCP).

A random fuzzy variable was proposed by Liu [182] as a fuzzy element taking "random variable" values. By random fuzzy programming we mean the optimization theory in random fuzzy environments. Liu and Liu [198] introduced a spectrum of random fuzzy EVM, Liu [182] proposed the random fuzzy CCP, and Liu [186] presented a spectrum of random fuzzy DCP in which the underlying philosophy is based on selecting the decision with maximum chance to meet the event.

More generally, a hybrid variable was introduced by Liu [188] as a measurable function from a chance space to the set of real numbers. Fuzzy random variable and random fuzzy variable are instances of hybrid variable. In order to measure hybrid events, a concept of chance measure was introduced by Li and Liu [163]. This chapter will assume the hybrid environment and introduce a spectrum of hybrid programming. In order to solve general hybrid programming, we will integrate hybrid simulation, neural network (NN) and genetic algorithm (GA) to produce a hybrid intelligent algorithm, and illustrate its effectiveness via some numerical examples.

6.1 Hybrid Variables

Definition 6.1 *(Liu [188]) Suppose that* $(\Theta, \mathcal{P}, \mathrm{Cr})$ *is a credibility space and* $(\Omega, \mathcal{A}, \mathrm{Pr})$ *is a probability space. The product* $(\Theta, \mathcal{P}, \mathrm{Cr}) \times (\Omega, \mathcal{A}, \mathrm{Pr})$ *is called a chance space.*

The universal set $\Theta \times \Omega$ is clearly the set of all ordered pairs of the form (θ, ω), where $\theta \in \Theta$ and $\omega \in \Omega$. What is the product σ-algebra $\mathcal{P} \times \mathcal{A}$? What is the product measure $\mathrm{Cr} \times \mathrm{Pr}$? Let us discuss these two basic problems.

What is the product σ-algebra $\mathcal{P} \times \mathcal{A}$?

Definition 6.2 *(Liu [190]) Let* $(\Theta, \mathcal{P}, \mathrm{Cr}) \times (\Omega, \mathcal{A}, \mathrm{Pr})$ *be a chance space. A subset* $\Lambda \subset \Theta \times \Omega$ *is called an event if*

$$\Lambda(\theta) = \left\{ \omega \in \Omega \;\middle|\; (\theta, \omega) \in \Lambda \right\} \in \mathcal{A} \tag{6.1}$$

for each $\theta \in \Theta$.

Example 6.1: Empty set $\emptyset$ and universal set $\Theta \times \Omega$ are clearly events.

Example 6.2: Let $X \in \mathcal{P}$ and $Y \in \mathcal{A}$. Then $X \times Y$ is a subset of $\Theta \times \Omega$. Since the set

$$(X \times Y)(\theta) = \begin{cases} Y, & \text{if } \theta \in X \\ \emptyset, & \text{if } \theta \in X^c \end{cases}$$

is in the σ-algebra $\mathcal{A}$ for each $\theta \in \Theta$, the rectangle $X \times Y$ is an event.

Theorem 6.1 *(Liu [190]) Let* $(\Theta, \mathcal{P}, \mathrm{Cr}) \times (\Omega, \mathcal{A}, \mathrm{Pr})$ *be a chance space. The class of all events is a* σ*-algebra over* $\Theta \times \Omega$*, and denoted by* $\mathcal{P} \times \mathcal{A}$.

What is the product measure $\mathrm{Cr} \times \mathrm{Pr}$?

Product probability is a probability measure, and product credibility is a credibility measure. What is the product measure $\mathrm{Cr} \times \mathrm{Pr}$? We will call it *chance measure* and define it as follows.

Definition 6.3 *(Li and Liu [163]) Let* $(\Theta, \mathcal{P}, \mathrm{Cr}) \times (\Omega, \mathcal{A}, \mathrm{Pr})$ *be a chance space. Then a chance measure of an event* Λ *is defined as*

$$\mathrm{Ch}\{\Lambda\} = \begin{cases} \sup\limits_{\theta \in \Theta} (\mathrm{Cr}\{\theta\} \wedge \mathrm{Pr}\{\Lambda(\theta)\}), \\ \qquad \textit{if } \sup\limits_{\theta \in \Theta} (\mathrm{Cr}\{\theta\} \wedge \mathrm{Pr}\{\Lambda(\theta)\}) < 0.5 \\ 1 - \sup\limits_{\theta \in \Theta} (\mathrm{Cr}\{\theta\} \wedge \mathrm{Pr}\{\Lambda^c(\theta)\}), \\ \qquad \textit{if } \sup\limits_{\theta \in \Theta} (\mathrm{Cr}\{\theta\} \wedge \mathrm{Pr}\{\Lambda(\theta)\}) \geq 0.5. \end{cases} \tag{6.2}$$

It has been proved that $\mathrm{Ch}\{\emptyset\} = 0$ and $\mathrm{Ch}\{\Theta \times \Omega\} = 1$. Furthermore, we have $\mathrm{Ch}\{X \times Y\} = \mathrm{Cr}\{X\} \wedge \mathrm{Pr}\{Y\}$ for any event $X \times Y$.

Hybrid Variables

Recall that a random variable is a measurable function from a probability space to the set of real numbers, and a fuzzy variable is a function from a credibility space to the set of real numbers. In order to describe a quantity with both fuzziness and randomness, we introduce the concept of hybrid variable as follows.

Definition 6.4 *(Liu [188]) A hybrid variable is a measurable function from a chance space* $(\Theta, \mathcal{P}, \mathrm{Cr}) \times (\Omega, \mathcal{A}, \mathrm{Pr})$ *to the set of real numbers, i.e., for any Borel set* B *of real numbers, the set*

$$\{\xi \in B\} = \{(\theta, \omega) \in \Theta \times \Omega \,\big|\, \xi(\theta, \omega) \in B\} \tag{6.3}$$

is an event.

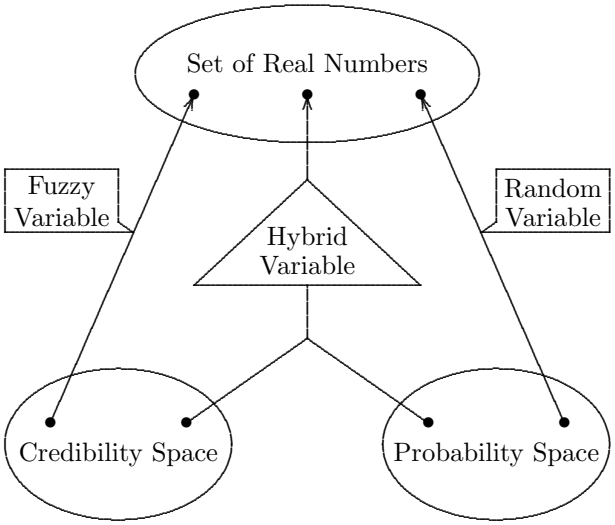

Figure 6.1: Graphical Representation of Hybrid Variable

Remark 6.1: A hybrid variable degenerates to a fuzzy variable if the value of $\xi(\theta, \omega)$ does not vary with ω. For example,

$$\xi(\theta, \omega) = \theta, \quad \xi(\theta, \omega) = \theta^2 + 1, \quad \xi(\theta, \omega) = \sin\theta.$$

Remark 6.2: A hybrid variable degenerates to a random variable if the value of $\xi(\theta, \omega)$ does not vary with θ. For example,

$$\xi(\theta, \omega) = \omega, \quad \xi(\theta, \omega) = \omega^2 + 1, \quad \xi(\theta, \omega) = \sin\omega.$$

Remark 6.3: A hybrid variable $\xi(\theta,\omega)$ may also be regarded as a function from a credibility space $(\Theta,\mathcal{P},\mathrm{Cr})$ to the set $\{\xi(\theta,\cdot)|\theta\in\Theta\}$ of random variables. Thus ξ is a *random fuzzy variable* defined by Liu [182].

Remark 6.4: A hybrid variable $\xi(\theta,\omega)$ may be regarded as a function from a probability space $(\Omega,\mathcal{A},\mathrm{Pr})$ to the set $\{\xi(\cdot,\omega)|\omega\in\Omega\}$ of fuzzy variables. If $\mathrm{Cr}\{\xi(\cdot,\omega)\in B\}$ is a measurable function of ω for any Borel set B of real numbers, then ξ is a *fuzzy random variable* in the sense of Liu and Liu [197].

Model I

If $\tilde{a}$ is a fuzzy variable and η is a random variable, then the sum $\xi=\tilde{a}+\eta$ is a hybrid variable. The product $\xi=\tilde{a}\cdot\eta$ is also a hybrid variable. Generally speaking, if $f:\Re^2\to\Re$ is a measurable function, then

$$\xi=f(\tilde{a},\eta) \tag{6.4}$$

is a hybrid variable. Suppose that $\tilde{a}$ has a membership function μ, and η has a probability density function ϕ. Then for any Borel set B of real numbers, Qin and Liu [259] proved the following formula,

$$\mathrm{Ch}\{f(\tilde{a},\eta)\in B\}=\begin{cases}\displaystyle\sup_x\left(\frac{\mu(x)}{2}\wedge\int_{f(x,y)\in B}\phi(y)\mathrm{d}y\right), \\ \quad\text{if } \displaystyle\sup_x\left(\frac{\mu(x)}{2}\wedge\int_{f(x,y)\in B}\phi(y)\mathrm{d}y\right)<0.5 \\ \displaystyle 1-\sup_x\left(\frac{\mu(x)}{2}\wedge\int_{f(x,y)\in B^c}\phi(y)\mathrm{d}y\right), \\ \quad\text{if } \displaystyle\sup_x\left(\frac{\mu(x)}{2}\wedge\int_{f(x,y)\in B}\phi(y)\mathrm{d}y\right)\geq 0.5.\end{cases}$$

More generally, let $\tilde{a}_1,\tilde{a}_2,\cdots,\tilde{a}_m$ be fuzzy variables, and let $\eta_1,\eta_2,\cdots,\eta_n$ be random variables. If $f:\Re^{m+n}\to\Re$ is a measurable function, then

$$\xi=f(\tilde{a}_1,\tilde{a}_2,\cdots,\tilde{a}_m;\eta_1,\eta_2,\cdots,\eta_n) \tag{6.5}$$

is a hybrid variable. The chance $\mathrm{Ch}\{f(\tilde{a}_1,\tilde{a}_2,\cdots,\tilde{a}_m;\eta_1,\eta_2,\cdots,\eta_n)\in B\}$ may be calculated in a similar way provided that μ is the joint membership function and ϕ is the joint probability density function.

Model II

Let $\tilde{a}_1, \tilde{a}_2, \cdots, \tilde{a}_m$ be fuzzy variables, and let $p_1, p_2, \cdots, p_m$ be nonnegative numbers with $p_1 + p_2 + \cdots + p_m = 1$. Then

$$\xi = \begin{cases} \tilde{a}_1 \text{ with probability } p_1 \\ \tilde{a}_2 \text{ with probability } p_2 \\ \cdots \\ \tilde{a}_m \text{ with probability } p_m \end{cases} \tag{6.6}$$

is clearly a hybrid variable. If $\tilde{a}_1, \tilde{a}_2, \cdots, \tilde{a}_m$ have membership functions $\mu_1, \mu_2, \cdots, \mu_m$, respectively, then for any Borel set B of real numbers, we have ([259])

$$\mathrm{Ch}\{\xi \in B\} = \begin{cases} \displaystyle\sup_{x_1, x_2 \cdots, x_m} \left(\left(\min_{1 \le i \le m} \frac{\mu_i(x_i)}{2} \right) \wedge \sum_{i=1}^{m} \{p_i \,|\, x_i \in B\} \right), \\ \quad \text{if } \displaystyle\sup_{x_1, x_2 \cdots, x_m} \left(\left(\min_{1 \le i \le m} \frac{\mu_i(x_i)}{2} \right) \wedge \sum_{i=1}^{m} \{p_i \,|\, x_i \in B\} \right) < 0.5 \\ 1 - \displaystyle\sup_{x_1, x_2 \cdots, x_m} \left(\left(\min_{1 \le i \le m} \frac{\mu_i(x_i)}{2} \right) \wedge \sum_{i=1}^{m} \{p_i \,|\, x_i \in B^c\} \right), \\ \quad \text{if } \displaystyle\sup_{x_1, x_2 \cdots, x_m} \left(\left(\min_{1 \le i \le m} \frac{\mu_i(x_i)}{2} \right) \wedge \sum_{i=1}^{m} \{p_i \,|\, x_i \in B\} \right) \ge 0.5. \end{cases}$$

Model III

Let $\eta_1, \eta_2, \cdots, \eta_m$ be random variables, and let $u_1, u_2, \cdots, u_m$ be nonnegative numbers with $u_1 \vee u_2 \vee \cdots \vee u_m = 1$. Then

$$\xi = \begin{cases} \eta_1 \text{ with membership degree } u_1 \\ \eta_2 \text{ with membership degree } u_2 \\ \quad \cdots \\ \eta_m \text{ with membership degree } u_m \end{cases} \tag{6.7}$$

is clearly a hybrid variable. If $\eta_1, \eta_2, \cdots, \eta_m$ have probability density functions $\phi_1, \phi_2, \cdots, \phi_m$, respectively, then for any Borel set B of real numbers,

we have ([259])

$$\mathrm{Ch}\{\xi \in B\} = \begin{cases} \max\limits_{1\le i\le m}\left(\dfrac{u_i}{2} \wedge \displaystyle\int_B \phi_i(x)\mathrm{d}x\right), \\ \qquad \text{if } \max\limits_{1\le i\le m}\left(\dfrac{u_i}{2} \wedge \displaystyle\int_B \phi_i(x)\mathrm{d}x\right) < 0.5 \\ 1 - \max\limits_{1\le i\le m}\left(\dfrac{u_i}{2} \wedge \displaystyle\int_{B^c} \phi_i(x)\mathrm{d}x\right), \\ \qquad \text{if } \max\limits_{1\le i\le m}\left(\dfrac{u_i}{2} \wedge \displaystyle\int_B \phi_i(x)\mathrm{d}x\right) \ge 0.5. \end{cases}$$

Model IV

In many statistics problems, the probability density function is completely known except for the values of one or more parameters. For example, it might be known that the lifetime ξ of a modern engine is an exponentially distributed random variable with an unknown expected value β. Usually, there is some relevant information in practice. It is thus possible to specify an interval in which the value of β is likely to lie, or to give an approximate estimate of the value of β. It is typically not possible to determine the value of β exactly. If the value of β is provided as a fuzzy variable, then ξ is a hybrid variable. More generally, suppose that ξ has a probability density function

$$\phi(x; \tilde{a}_1, \tilde{a}_2, \cdots, \tilde{a}_m), \quad x \in \Re \tag{6.8}$$

in which the parameters $\tilde{a}_1, \tilde{a}_2, \cdots, \tilde{a}_m$ are fuzzy variables rather than crisp numbers. Then ξ is a hybrid variable provided that $\phi(x; y_1, y_2, \cdots, y_m)$ is a probability density function for any $(y_1, y_2, \cdots, y_m)$ that $(\tilde{a}_1, \tilde{a}_2, \cdots, \tilde{a}_m)$ may take. If $\tilde{a}_1, \tilde{a}_2, \cdots, \tilde{a}_m$ have membership functions $\mu_1, \mu_2, \cdots, \mu_m$, respectively, then for any Borel set B of real numbers, the chance $\mathrm{Ch}\{\xi \in B\}$ is ([259])

$$\begin{cases} \sup\limits_{y_1, y_2 \cdots, y_m}\left(\left(\min\limits_{1\le i\le m}\dfrac{\mu_i(y_i)}{2}\right) \wedge \displaystyle\int_B \phi(x; y_1, y_2, \cdots, y_m)\mathrm{d}x\right), \\ \qquad \text{if } \sup\limits_{y_1, y_2, \cdots, y_m}\left(\left(\min\limits_{1\le i\le m}\dfrac{\mu_i(y_i)}{2}\right) \wedge \displaystyle\int_B \phi(x; y_1, y_2, \cdots, y_m)\mathrm{d}x\right) < 0.5 \\ 1 - \sup\limits_{y_1, y_2 \cdots, y_m}\left(\left(\min\limits_{1\le i\le m}\dfrac{\mu_i(y_i)}{2}\right) \wedge \displaystyle\int_{B^c} \phi(x; y_1, y_2, \cdots, y_m)\mathrm{d}x\right), \\ \qquad \text{if } \sup\limits_{y_1, y_2, \cdots, y_m}\left(\left(\min\limits_{1\le i\le m}\dfrac{\mu_i(y_i)}{2}\right) \wedge \displaystyle\int_B \phi(x; y_1, y_2, \cdots, y_m)\mathrm{d}x\right) \ge 0.5. \end{cases}$$

Model V

Suppose a fuzzy variable ξ has a normal membership function with unknown expected value e and variance σ. If e and σ are provided as random variables, then ξ is a hybrid variable. More generally, suppose that ξ has a membership function

$$\mu(x; \eta_1, \eta_2, \cdots, \eta_m), \quad x \in \Re \tag{6.9}$$

in which the parameters $\eta_1, \eta_2, \cdots, \eta_m$ are random variables rather than deterministic numbers. Then ξ is a hybrid variable if $\mu(x; y_1, y_2, \cdots, y_m)$ is a membership function for any $(y_1, y_2, \cdots, y_m)$ that $(\eta_1, \eta_2, \cdots, \eta_m)$ may take.

Hybrid Vectors

Definition 6.5 *An n-dimensional hybrid vector is a measurable function from a chance space $(\Theta, \mathcal{P}, \mathrm{Cr}) \times (\Omega, \mathcal{A}, \mathrm{Pr})$ to the set of n-dimensional real vectors, i.e., for any Borel set B of $\Re^n$, the set*

$$\{\boldsymbol{\xi} \in B\} = \left\{(\theta, \omega) \in \Theta \times \Omega \,\middle|\, \boldsymbol{\xi}(\theta, \omega) \in B\right\} \tag{6.10}$$

is an event.

It has been proved that $(\xi_1, \xi_2, \cdots, \xi_n)$ is a hybrid vector if and only if $\xi_1, \xi_2, \cdots, \xi_n$ are hybrid variables.

Hybrid Arithmetic

Definition 6.6 *Let $f : \Re^n \to \Re$ be a measurable function, and $\xi_1, \xi_2, \cdots, \xi_n$ hybrid variables on the chance space $(\Theta, \mathcal{P}, \mathrm{Cr}) \times (\Omega, \mathcal{A}, \mathrm{Pr})$. Then $\xi = f(\xi_1, \xi_2, \cdots, \xi_n)$ is a hybrid variable defined as*

$$\xi(\theta, \omega) = f(\xi_1(\theta, \omega), \xi_2(\theta, \omega), \cdots, \xi_n(\theta, \omega)), \quad \forall (\theta, \omega) \in \Theta \times \Omega. \tag{6.11}$$

Example 6.3: Let ξ_1 and ξ_2 be two hybrid variables defined as follows,

$$\xi_1 \sim \begin{cases} \mathcal{N}(u_1, \sigma_1^2) \text{ with membership degree } 0.7 \\ \mathcal{N}(u_2, \sigma_2^2) \text{ with membership degree } 1.0, \end{cases}$$

$$\xi_2 \sim \begin{cases} \mathcal{N}(u_3, \sigma_3^2) \text{ with membership degree } 1.0 \\ \mathcal{N}(u_4, \sigma_4^2) \text{ with membership degree } 0.8. \end{cases}$$

Then the sum of ξ_1 and ξ_2 is

$$\xi \sim \begin{cases} \mathcal{N}(u_1 + u_3, \sigma_1^2 + \sigma_3^2) \text{ with membership degree } 0.7 \\ \mathcal{N}(u_1 + u_4, \sigma_1^2 + \sigma_4^2) \text{ with membership degree } 0.7 \\ \mathcal{N}(u_2 + u_3, \sigma_2^2 + \sigma_3^2) \text{ with membership degree } 1.0 \\ \mathcal{N}(u_2 + u_4, \sigma_2^2 + \sigma_4^2) \text{ with membership degree } 0.8. \end{cases}$$

Example 6.4: Let ξ_1 and ξ_2 be two hybrid variables defined as follows,

$$\xi_1 = \begin{cases} (a_1, a_2, a_3, a_4) \text{ with probability } 0.3 \\ (b_1, b_2, b_3, b_4) \text{ with probability } 0.7, \end{cases}$$

$$\xi_2 = \begin{cases} (c_1, c_2, c_3, c_4) \text{ with probability } 0.6 \\ (d_1, d_2, d_3, d_4) \text{ with probability } 0.4. \end{cases}$$

Then the sum of ξ_1 and ξ_2 is

$$\xi_1 + \xi_2 = \begin{cases} (a_1 + c_1, a_2 + c_2, a_3 + c_3, a_4 + c_4) \text{ with probability } 0.18 \\ (a_1 + d_1, a_2 + d_2, a_3 + d_3, a_4 + d_4) \text{ with probability } 0.12 \\ (b_1 + c_1, b_2 + c_2, b_3 + c_3, b_4 + c_4) \text{ with probability } 0.42 \\ (b_1 + d_1, b_2 + d_2, b_3 + d_3, b_4 + d_4) \text{ with probability } 0.28. \end{cases}$$

Expected Value

Expected value has been defined in several ways. For example, Kwakernaak [144], Puri and Ralescu [258], Kruse and Meyer [141], and Liu and Liu [197] gave different expected value operators of fuzzy random variables. Liu and Liu [198] presented an expected value operator of random fuzzy variables. Li and Liu [163]) suggested the following definition of expected value operator of hybrid variables.

Definition 6.7 *Let ξ be a hybrid variable. Then the expected value of ξ is defined by*

$$E[\xi] = \int_0^{+\infty} \mathrm{Ch}\{\xi \geq r\}\mathrm{d}r - \int_{-\infty}^{0} \mathrm{Ch}\{\xi \leq r\}\mathrm{d}r \tag{6.12}$$

provided that at least one of the two integrals is finite.

Example 6.5: If a hybrid variable ξ degenerates to a random variable η, then

$$\mathrm{Ch}\{\xi \leq x\} = \mathrm{Pr}\{\eta \leq x\}, \quad \mathrm{Ch}\{\xi \geq x\} = \mathrm{Pr}\{\eta \geq x\}, \quad \forall x \in \Re.$$

It follows from (6.12) that $E[\xi] = E[\eta]$. In other words, the expected value operator of hybrid variable coincides with that of random variable.

Example 6.6: If a hybrid variable ξ degenerates to a fuzzy variable $\tilde{a}$, then

$$\mathrm{Ch}\{\xi \leq x\} = \mathrm{Cr}\{\tilde{a} \leq x\}, \quad \mathrm{Ch}\{\xi \geq x\} = \mathrm{Cr}\{\tilde{a} \geq x\}, \quad \forall x \in \Re.$$

It follows from (6.12) that $E[\xi] = E[\tilde{a}]$. In other words, the expected value operator of hybrid variable coincides with that of fuzzy variable.

Example 6.7: Let $\tilde{a}$ be a fuzzy variable and η a random variable with finite expected values. Then the hybrid variable $\xi = \tilde{a} + \eta$ has expected value $E[\xi] = E[\tilde{a}] + E[\eta]$.

Critical Values

In order to rank fuzzy random variables, Liu [180] defined two critical values: optimistic value and pessimistic value. Analogously, Liu [182] gave the concepts of critical values of random fuzzy variables. Li and Liu [163] presented the following definition of critical values of hybrid variables.

Definition 6.8 *Let ξ be a hybrid variable, and $\alpha \in (0, 1]$. Then*

$$\xi_{\sup}(\alpha) = \sup\left\{r \mid \mathrm{Ch}\{\xi \geq r\} \geq \alpha\right\} \tag{6.13}$$

is called the α-optimistic value to ξ, and

$$\xi_{\inf}(\alpha) = \inf\left\{r \mid \mathrm{Ch}\{\xi \leq r\} \geq \alpha\right\} \tag{6.14}$$

is called the α-pessimistic value to ξ.

The hybrid variable ξ reaches upwards of the α-optimistic value $\xi_{\sup}(\alpha)$, and is below the α-pessimistic value $\xi_{\inf}(\alpha)$ with chance α.

Example 6.8: If a hybrid variable ξ degenerates to a random variable η, then

$$\mathrm{Ch}\{\xi \leq x\} = \mathrm{Pr}\{\eta \leq x\}, \quad \mathrm{Ch}\{\xi \geq x\} = \mathrm{Pr}\{\eta \geq x\}, \quad \forall x \in \Re.$$

It follows from the definition of critical values that

$$\xi_{\sup}(\alpha) = \eta_{\sup}(\alpha), \quad \xi_{\inf}(\alpha) = \eta_{\inf}(\alpha), \quad \forall \alpha \in (0, 1].$$

In other words, the critical values of hybrid variable coincide with that of random variable.

Example 6.9: If a hybrid variable ξ degenerates to a fuzzy variable $\tilde{a}$, then

$$\mathrm{Ch}\{\xi \leq x\} = \mathrm{Cr}\{\tilde{a} \leq x\}, \quad \mathrm{Ch}\{\xi \geq x\} = \mathrm{Cr}\{\tilde{a} \geq x\}, \quad \forall x \in \Re.$$

It follows from the definition of critical values that

$$\xi_{\sup}(\alpha) = \tilde{a}_{\sup}(\alpha), \quad \xi_{\inf}(\alpha) = \tilde{a}_{\inf}(\alpha), \quad \forall \alpha \in (0, 1].$$

In other words, the critical values of hybrid variable coincide with that of fuzzy variable.

Ranking Criteria

Let ξ and η be two hybrid variables. Different from the situation of real numbers, there does not exist a natural ordership in a hybrid world. Thus an important problem appearing in this area is how to rank hybrid variables. Here we give four ranking criteria.

Expected Value Criterion: We say $\xi > \eta$ if and only if $E[\xi] > E[\eta]$.

Optimistic Value Criterion: We say $\xi > \eta$ if and only if, for some predetermined confidence level $\alpha \in (0, 1]$, we have $\xi_{\sup}(\alpha) > \eta_{\sup}(\alpha)$, where $\xi_{\sup}(\alpha)$ and $\eta_{\sup}(\alpha)$ are the α-optimistic values of ξ and η, respectively.

Pessimistic Value Criterion: We say $\xi > \eta$ if and only if, for some predetermined confidence level $\alpha \in (0, 1]$, we have $\xi_{\inf}(\alpha) > \eta_{\inf}(\alpha)$, where $\xi_{\inf}(\alpha)$ and $\eta_{\inf}(\alpha)$ are the α-pessimistic values of ξ and η, respectively.

Chance Criterion: We say $\xi > \eta$ if and only if, for some predetermined levels $\overline{r}$, we have $\mathrm{Ch}\{\xi \geq \overline{r}\} > \mathrm{Ch}\{\eta \geq \overline{r}\}$.

6.2 Expected Value Model

In order to obtain the decision with maximum expected return subject to expected constraints, Liu and Liu [196] introduced fuzzy random EVM, and Liu and Liu [198] introduced random fuzzy EVM. This section provides the following hybrid EVM,

$$\begin{cases} \max E[f(\boldsymbol{x}, \boldsymbol{\xi})] \\ \text{subject to:} \\ \quad E[g_j(\boldsymbol{x}, \boldsymbol{\xi})] \leq 0, \ j = 1, 2, \cdots, p \end{cases} \tag{6.15}$$

where $\boldsymbol{x}$ is a decision vector, $\boldsymbol{\xi}$ is a hybrid vector, f is the objective function, and g_j are the constraint functions for $j = 1, 2, \cdots, p$.

In practice, a decision maker may want to optimize multiple objectives. Thus we have the following hybrid expected value multiobjective programming (EVMOP),

$$\begin{cases} \max [E[f_1(\boldsymbol{x}, \boldsymbol{\xi})], E[f_2(\boldsymbol{x}, \boldsymbol{\xi})], \cdots, E[f_m(\boldsymbol{x}, \boldsymbol{\xi})]] \\ \text{subject to:} \\ \quad E[g_j(\boldsymbol{x}, \boldsymbol{\xi})] \leq 0, \ j = 1, 2, \cdots, p \end{cases} \tag{6.16}$$

where $f_i(\boldsymbol{x}, \boldsymbol{\xi})$ are objective functions for $i = 1, 2, \cdots, m$, and $g_j(\boldsymbol{x}, \boldsymbol{\xi})$ are constraint functions for $j = 1, 2, \cdots, p$.

In order to balance the multiple conflicting objectives, a decision-maker may establish a hierarchy of importance among these incompatible goals so as to satisfy as many goals as possible in the order specified. Thus we have a hybrid expected value goal programming (EVGP),

$$\begin{cases} \min \displaystyle\sum_{j=1}^{l} P_j \sum_{i=1}^{m} (u_{ij} d_i^+ \vee 0 + v_{ij} d_i^- \vee 0) \\ \text{subject to:} \\ \quad E[f_i(\boldsymbol{x}, \boldsymbol{\xi})] - b_i = d_i^+, \quad i = 1, 2, \cdots, m \\ \quad b_i - E[f_i(\boldsymbol{x}, \boldsymbol{\xi})] = d_i^-, \quad i = 1, 2, \cdots, m \\ \quad E[g_j(\boldsymbol{x}, \boldsymbol{\xi})] \leq 0, \qquad j = 1, 2, \cdots, p \end{cases} \tag{6.17}$$

where P_j is the preemptive priority factor which expresses the relative importance of various goals, $P_j \gg P_{j+1}$, 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, $d_i^+ \vee 0$ is the positive deviation from the target of goal i, $d_i^- \vee 0$ is the negative deviation from the target of goal i, 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.

6.3 Chance-Constrained Programming

Fuzzy random CCP was initialized by Liu [180], and random fuzzy CCP was proposed by Liu [182]. This section introduces hybrid CCP.

Chance Constraints

Assume that $\boldsymbol{x}$ is a decision vector, $\boldsymbol{\xi}$ is a hybrid vector, $f(\boldsymbol{x}, \boldsymbol{\xi})$ is a return function, and $g_j(\boldsymbol{x}, \boldsymbol{\xi})$ are constraint functions, $j = 1, 2, \cdots, p$. Since the hybrid constraints $g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, j = 1, 2, \cdots, p$ do not define a deterministic feasible set, it is naturally desired that the hybrid constraints hold with chance α, where α is a specified confidence level. Then we have a chance constraint as follows,

$$\mathrm{Ch}\left\{g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, j = 1, 2, \cdots, p\right\} \geq \alpha. \tag{6.18}$$

Maximax Chance-Constrained Programming

If we want to maximize the optimistic value to the hybrid return subject to some chance constraints, we have the following hybrid maximax CCP,

$$\begin{cases} \max\limits_{\boldsymbol{x}} \max\limits_{\overline{f}} \overline{f} \\ \text{subject to:} \\ \quad \mathrm{Ch}\left\{f(\boldsymbol{x}, \boldsymbol{\xi}) \geq \overline{f}\right\} \geq \beta \\ \quad \mathrm{Ch}\left\{g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0\right\} \geq \alpha_j, \quad j = 1, 2, \cdots, p \end{cases} \tag{6.19}$$

where α_j and β are specified confidence levels for $j = 1, 2, \cdots, p$, and $\max \overline{f}$ is the β-optimistic return.

In practice, we may have multiple objectives. We thus have the following

hybrid maximax chance-constrained multiobjective programming (CCMOP),

$$\begin{cases} \max\limits_{\boldsymbol{x}} \left[\max\limits_{\overline{f}_1} \overline{f}_1, \max\limits_{\overline{f}_2} \overline{f}_2, \cdots, \max\limits_{\overline{f}_m} \overline{f}_m \right] \\ \text{subject to:} \\ \quad \text{Ch}\left\{ f_i(\boldsymbol{x}, \boldsymbol{\xi}) \geq \overline{f}_i \right\} \geq \beta_i, \quad i = 1, 2, \cdots, m \\ \quad \text{Ch}\left\{ g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0 \right\} \geq \alpha_j, \quad j = 1, 2, \cdots, p \end{cases} \tag{6.20}$$

where β_i are predetermined confidence levels for $i = 1, 2, \cdots, m$, and $\max \overline{f}_i$ are the β-optimistic values to the return functions $f_i(\boldsymbol{x}, \boldsymbol{\xi})$, $i = 1, 2, \cdots, m$, respectively.

If the priority structure and target levels are set by the decision-maker, then we have a minimin chance-constrained goal programming (CCGP),

$$\begin{cases} \min\limits_{\boldsymbol{x}} \sum\limits_{j=1}^{l} P_j \sum\limits_{i=1}^{m} \left(u_{ij} \left(\min\limits_{d_i^+} d_i^+ \vee 0 \right) + v_{ij} \left(\min\limits_{d_i^-} d_i^- \vee 0 \right) \right) \\ \text{subject to:} \\ \quad \text{Ch}\left\{ f_i(\boldsymbol{x}, \boldsymbol{\xi}) - b_i \leq d_i^+ \right\} \geq \beta_i^+, \quad i = 1, 2, \cdots, m \\ \quad \text{Ch}\left\{ b_i - f_i(\boldsymbol{x}, \boldsymbol{\xi}) \leq d_i^- \right\} \geq \beta_i^-, \quad i = 1, 2, \cdots, m \\ \quad \text{Ch}\left\{ g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0 \right\} \geq \alpha_j, \qquad j = 1, 2, \cdots, p \end{cases} \tag{6.21}$$

where P_j is the preemptive priority factor which expresses the relative importance of various goals, $P_j \gg P_{j+1}$, 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, $\min d_i^+ \vee 0$ is the β_i^+-optimistic positive deviation from the target of goal i, $\min d_i^- \vee 0$ is the β_i^--optimistic negative deviation from the target of goal i, b_i is the target value according to goal i, and l is the number of priorities.

Minimax Chance-Constrained Programming

If we want to maximize the pessimistic value subject to some chance constraints, Liu [180] presented fuzzy random minimax CCP, and Liu [182] proposed random fuzzy minimax CCP. This section introduces the following hybrid minimax CCP,

$$\begin{cases} \max\limits_{\boldsymbol{x}} \min\limits_{\overline{f}} \overline{f} \\ \text{subject to:} \\ \quad \text{Ch}\left\{ f(\boldsymbol{x}, \boldsymbol{\xi}) \leq \overline{f} \right\} \geq \beta \\ \quad \text{Ch}\left\{ g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0 \right\} \geq \alpha_j, \quad j = 1, 2, \cdots, p \end{cases} \tag{6.22}$$

where α_j and β are specified confidence levels for $j = 1, 2, \cdots, p$, and $\min \overline{f}$ is the β-pessimistic return.

If there are multiple objectives, then we have the following hybrid minimax CCMOP,

$$\begin{cases} \max\limits_{\boldsymbol{x}} \left[\min\limits_{\overline{f}_1} \overline{f}_1,\ \min\limits_{\overline{f}_2} \overline{f}_2, \cdots, \min\limits_{\overline{f}_m} \overline{f}_m \right] \\ \text{subject to:} \\ \quad \text{Ch}\left\{ f_i(\boldsymbol{x}, \boldsymbol{\xi}) \le \overline{f}_i \right\} \ge \beta_i, \quad i = 1, 2, \cdots, m \\ \quad \text{Ch}\left\{ g_j(\boldsymbol{x}, \boldsymbol{\xi}) \le 0 \right\} \ge \alpha_j, \quad j = 1, 2, \cdots, p \end{cases} \tag{6.23}$$

where $\min \overline{f}_i$ are the β_i-pessimistic values to the return functions $f_i(\boldsymbol{x}, \boldsymbol{\xi})$, $i = 1, 2, \cdots, m$, respectively.

We can also formulate a hybrid decision system as a hybrid minimax CCGP according to the priority structure and target levels set by the decision-maker:

$$\begin{cases} \min\limits_{\boldsymbol{x}} \sum\limits_{j=1}^{l} P_j \sum\limits_{i=1}^{m} \left[u_{ij} \left(\max\limits_{d_i^+} d_i^+ \vee 0 \right) + v_{ij} \left(\max\limits_{d_i^-} d_i^- \vee 0 \right) \right] \\ \text{subject to:} \\ \quad \text{Ch}\left\{ f_i(\boldsymbol{x}, \boldsymbol{\xi}) - b_i \ge d_i^+ \right\} \ge \beta_i^+, \quad i = 1, 2, \cdots, m \\ \quad \text{Ch}\left\{ b_i - f_i(\boldsymbol{x}, \boldsymbol{\xi}) \ge d_i^- \right\} \ge \beta_i^-, \quad i = 1, 2, \cdots, m \\ \quad \text{Ch}\left\{ g_j(\boldsymbol{x}, \boldsymbol{\xi}) \le 0 \right\} \ge \alpha_j, \quad j = 1, 2, \cdots, p \end{cases} \tag{6.24}$$

where P_j is the preemptive priority factor which expresses the relative importance of various goals, $P_j \gg P_{j+1}$, 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, $\max d_i^+ \vee 0$ is the β_i^+-pessimistic positive deviation from the target of goal i, $\max d_i^- \vee 0$ is the β_i^--pessimistic negative deviation from the target of goal i, b_i is the target value according to goal i, and l is the number of priorities.

6.4 Dependent-Chance Programming

Liu [181] introduced a theoretical framework of fuzzy random DCP, and Liu [186] presented a spectrum of random fuzzy DCP. This section provides hybrid DCP in which the underlying philosophy is based on selecting the decision with maximum chance to meet the event.

Uncertain environment and chance function are key elements in DCP. Let us redefine them in hybrid decision systems, and introduce the principle of uncertainty.

By uncertain environment (in this case the hybrid environment) we mean the hybrid constraints represented by

$$g_j(\boldsymbol{x}, \boldsymbol{\xi}) \le 0, \quad j = 1, 2, \cdots, p \tag{6.25}$$

where $\boldsymbol{x}$ is a decision vector, and $\boldsymbol{\xi}$ is a hybrid vector. By event we mean the system of inequalities

$$h_k(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, \quad k = 1, 2, \cdots, q. \tag{6.26}$$

The chance function of an event $\mathcal{E}$ characterized by (6.26) is defined as the chance measure of the event $\mathcal{E}$, i.e.,

$$f(\boldsymbol{x}) = \mathrm{Ch}\{h_k(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0,\ k = 1, 2, \cdots, q\} \tag{6.27}$$

subject to the uncertain environment (6.25).

For each decision $\boldsymbol{x}$ and realization $\boldsymbol{\xi}$, an event $\mathcal{E}$ is said to be consistent in the uncertain environment if the following two conditions hold: (i) $h_k(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0,\ k = 1, 2, \cdots, q$; and (ii) $g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0,\ j \in J$, where J is the index set of all dependent constraints.

Principle of Uncertainty: *The chance of a hybrid event is the chance that the event is consistent in the uncertain environment.*

Assume that there are m events $\mathcal{E}_i$ characterized by $h_{ik}(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, k = 1, 2, \cdots, q_i$ for $i = 1, 2, \cdots, m$ in the uncertain environment $g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, j = 1, 2, \cdots, p$. The principle of uncertainty implies that the chance function of the ith event $\mathcal{E}_i$ in the uncertain environment is

$$f_i(\boldsymbol{x}) = \mathrm{Ch}\left\{\begin{matrix} h_{ik}(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, k = 1, 2, \cdots, q_i \\ g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, j \in J_i \end{matrix}\right\} \tag{6.28}$$

where J_i are defined by

$$J_i = \left\{j \in \{1, 2, \cdots, p\} \;\middle|\; g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0 \text{ is a dependent constraint of } \mathcal{E}_i\right\}$$

for $i = 1, 2, \cdots, m$.

We then formulate a hybrid DCP as follows:

$$\begin{cases} \max \mathrm{Ch}\left\{h_k(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, k = 1, 2, \cdots, q\right\} \\ \text{subject to:} \\ \quad g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, \quad j = 1, 2, \cdots, p \end{cases} \tag{6.29}$$

where $\boldsymbol{x}$ is an n-dimensional decision vector, $\boldsymbol{\xi}$ is a hybrid vector, the event $\mathcal{E}$ is characterized by $h_k(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, k = 1, 2, \cdots, q$, and the uncertain environment is described by the hybrid constraints $g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, j = 1, 2, \cdots, p$.

If there are multiple events in the uncertain environment, then we have the following hybrid dependent-chance multiobjective programming (DCMOP),

$$\begin{cases} \max \left[\begin{matrix} \mathrm{Ch}\left\{h_{1k}(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, k = 1, 2, \cdots, q_1\right\} \\ \mathrm{Ch}\left\{h_{2k}(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, k = 1, 2, \cdots, q_2\right\} \\ \cdots \\ \mathrm{Ch}\left\{h_{mk}(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, k = 1, 2, \cdots, q_m\right\} \end{matrix}\right] \\ \text{subject to:} \\ \quad g_j(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0, \quad j = 1, 2, \cdots, p \end{cases} \tag{6.30}$$

where the events $\mathcal{E}_i$ are characterized by $h_{ik}(\boldsymbol{x},\boldsymbol{\xi}) \leq 0, k = 1,2,\cdots,q_i$, $i = 1,2,\cdots,m$, respectively.

Hybrid dependent-chance goal programming (DCGP) is employed to formulate hybrid decision systems according to the priority structure and target levels set by the decision-maker,

$$\begin{cases} \min \sum_{j=1}^{l} P_j \sum_{i=1}^{m} (u_{ij} d_i^+ \vee 0 + v_{ij} d_i^- \vee 0) \\ \text{subject to:} \\ \quad \text{Ch}\left\{h_{ik}(\boldsymbol{x},\boldsymbol{\xi}) \leq 0, k = 1,2,\cdots,q_i\right\} - b_i = d_i^+, \quad i = 1,2,\cdots,m \\ \quad b_i - \text{Ch}\left\{h_{ik}(\boldsymbol{x},\boldsymbol{\xi}) \leq 0, k = 1,2,\cdots,q_i\right\} = d_i^-, \quad i = 1,2,\cdots,m \\ \quad g_j(\boldsymbol{x},\boldsymbol{\xi}) \leq 0, \qquad\qquad\qquad\qquad\qquad\qquad\quad j = 1,2,\cdots,p \end{cases}$$

where P_j is the preemptive priority factor which expresses the relative importance of various goals, $P_j \gg P_{j+1}$, 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, $d_i^+ \vee 0$ is the positive deviation from the target of goal i, $d_i^- \vee 0$ is the negative deviation from the target of goal i, 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.

6.5 Hybrid Intelligent Algorithm

Liu [190] first designed hybrid simulations to estimate the uncertain functions like

$$\begin{aligned} &U_1 : \boldsymbol{x} \to E[f(\boldsymbol{x},\boldsymbol{\xi})], \\ &U_2 : \boldsymbol{x} \to \text{Ch}\left\{g_j(\boldsymbol{x},\boldsymbol{\xi}) \leq 0, j = 1,2,\cdots,p\right\}, \\ &U_3 : \boldsymbol{x} \to \max\left\{\overline{f} \;\middle|\; \text{Ch}\left\{f(\boldsymbol{x},\boldsymbol{\xi}) \geq \overline{f}\right\} \geq \alpha\right\}. \end{aligned} \tag{6.31}$$

In order to compute $U_1(\boldsymbol{x})$, we randomly generate $\theta_1, \theta_2, \cdots, \theta_N$ from the credibility space $(\Theta, \mathcal{P}, \text{Cr})$, and $\omega_1, \omega_2, \cdots, \omega_N$ from the probability space $(\Omega, \mathcal{A}, \text{Pr})$. For any number $r \geq 0$, the value $\text{Ch}\{f(\boldsymbol{x},\boldsymbol{\xi}) \geq r\}$ is

$$\begin{cases} \max\limits_{1\leq k\leq N} \text{Cr}\{\theta_k\} \wedge \text{Pr}\{f(\boldsymbol{x},\boldsymbol{\xi}(\theta_k,\cdot)) \geq r\}, \\ \qquad \text{if } \max\limits_{1\leq k\leq N} \text{Cr}\{\theta_k\} \wedge \text{Pr}\{f(\boldsymbol{x},\boldsymbol{\xi}(\theta_k,\cdot)) \geq r\} < 0.5 \\ 1 - \max\limits_{1\leq k\leq N} \text{Cr}\{\theta_k\} \wedge \text{Pr}\{f(\boldsymbol{x},\boldsymbol{\xi}(\theta_k,\cdot)) < r\}, \\ \qquad \text{if } \max\limits_{1\leq k\leq N} \text{Cr}\{\theta_k\} \wedge \text{Pr}\{f(\boldsymbol{x},\boldsymbol{\xi}(\theta_k,\cdot)) \geq r\} \geq 0.5 \end{cases} \tag{6.32}$$

and for any number $r < 0$, the value $\mathrm{Ch}\{f(\boldsymbol{x}, \boldsymbol{\xi}) \le r\}$ is

$$\begin{cases} \max\limits_{1\le k\le N} \mathrm{Cr}\{\theta_k\} \wedge \mathrm{Pr}\{f(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k, \cdot)) \le r\}, \\ \quad \text{if } \max\limits_{1\le k\le N} \mathrm{Cr}\{\theta_k\} \wedge \mathrm{Pr}\{f(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k, \cdot)) \le r\} < 0.5 \\ 1 - \max\limits_{1\le k\le N} \mathrm{Cr}\{\theta_k\} \wedge \mathrm{Pr}\{f(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k, \cdot)) > r\}, \\ \quad \text{if } \max\limits_{1\le k\le N} \mathrm{Cr}\{\theta_k\} \wedge \mathrm{Pr}\{f(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k, \cdot)) \le r\} \ge 0.5. \end{cases} \tag{6.33}$$

The expected value is then obtained by the following process.

Algorithm 6.1 (Hybrid Simulation for $U_1(\boldsymbol{x})$)
Step 1. Set $e = 0$.
Step 2. Generate $\theta_1, \theta_2, \cdots, \theta_N$ from the credibility space $(\Theta, \mathcal{P}, \mathrm{Cr})$.
Step 3. Generate $\omega_1, \omega_2, \cdots, \omega_N$ from the probability space $(\Omega, \mathcal{A}, \mathrm{Pr})$.
Step 4. $a = \min\limits_{1\le k\le N, 1\le j\le N} f(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k, \omega_j))$, $b = \max\limits_{1\le k\le N, 1\le j\le N} f(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k, \omega_j))$.
Step 5. Randomly generate r from $[a, b]$.
Step 6. If $r \ge 0$, then $e \leftarrow e + \mathrm{Ch}\{f(\boldsymbol{x}, \boldsymbol{\xi}) \ge r\}$ by using (6.32).
Step 7. If $r < 0$, then $e \leftarrow e - \mathrm{Ch}\{f(\boldsymbol{x}, \boldsymbol{\xi}) \le r\}$ by using (6.33).
Step 8. Repeat the fifth to seventh steps for N times.
Step 9. $E[f(\boldsymbol{\xi})] = a \vee 0 + b \wedge 0 + e \cdot (b - a)/N$.

In order to compute the uncertain function $U_2(\boldsymbol{x})$, we randomly generate $\theta_1, \theta_2, \cdots, \theta_N$ from the credibility space $(\Theta, \mathcal{P}, \mathrm{Cr})$, and $\omega_1, \omega_2, \cdots, \omega_N$ from the probability space $(\Omega, \mathcal{A}, \mathrm{Pr})$. For each θ_k, we estimate $\mathrm{Pr}\{g_j(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k, \cdot)) \le 0 \text{ for all } j\}$ and $\mathrm{Pr}\{g_j(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k, \cdot)) > 0 \text{ for some } j\}$ by stochastic simulation via the samples $\omega_1, \omega_2, \cdots, \omega_N$. Then $U_2(\boldsymbol{x})$ is

$$\begin{cases} \max\limits_{1\le k\le N} \mathrm{Cr}\{\theta_k\} \wedge \mathrm{Pr}\{g_j(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k, \cdot)) \le 0 \text{ for all } j\}, \\ \quad \text{if } \max\limits_{1\le k\le N} \mathrm{Cr}\{\theta_k\} \wedge \mathrm{Pr}\{g_j(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k, \cdot)) \le 0 \text{ for all } j\} < 0.5 \\ 1 - \max\limits_{1\le k\le N} \mathrm{Cr}\{\theta_k\} \wedge \mathrm{Pr}\{g_j(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k, \cdot)) > 0 \text{ for some } j\}, \\ \quad \text{if } \max\limits_{1\le k\le N} \mathrm{Cr}\{\theta_k\} \wedge \mathrm{Pr}\{g_j(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k, \cdot)) \le 0 \text{ for all } j\} \ge 0.5. \end{cases} \tag{6.34}$$

We summarize this process as follows.

Algorithm 6.2 (Hybrid Simulation for $U_2(\boldsymbol{x})$)
Step 1. Generate $\theta_1, \theta_2, \cdots, \theta_N$ from the credibility space $(\Theta, \mathcal{P}, \mathrm{Cr})$.
Step 2. Generate $\omega_1, \omega_2, \cdots, \omega_N$ from the probability space $(\Omega, \mathcal{A}, \mathrm{Pr})$.
Step 3. Employ stochastic simulation to estimate $\mathrm{Pr}\{g_j(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k, \cdot)) \le 0 \text{ for all } j\}$ and $\mathrm{Pr}\{g_j(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k, \cdot)) > 0 \text{ for some } j\}$ via the samples $\omega_1, \omega_2, \cdots, \omega_N$.

Step 4. Output the chance $U_2(\boldsymbol{x})$ via (6.34).

In order to compute $U_3(\boldsymbol{x})$, we randomly generate $\theta_1, \theta_2, \cdots, \theta_N$ from the credibility space $(\Theta, \mathcal{P}, \mathrm{Cr})$, and $\omega_1, \omega_2, \cdots, \omega_N$ from the probability space $(\Omega, \mathcal{A}, \Pr)$. For any number r, we set

$$L(r) = \begin{cases} \max\limits_{1\le k\le N} \mathrm{Cr}\{\theta_k\} \wedge \Pr\{f(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k, \cdot)) \ge r\}, \\ \quad \text{if } \max\limits_{1\le k\le N} \mathrm{Cr}\{\theta_k\} \wedge \Pr\{f(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k, \cdot)) \ge r\} < 0.5 \\ 1 - \max\limits_{1\le k\le N} \mathrm{Cr}\{\theta_k\} \wedge \Pr\{f(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k, \cdot)) < r\}, \\ \quad \text{if } \max\limits_{1\le k\le N} \mathrm{Cr}\{\theta_k\} \wedge \Pr\{f(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k, \cdot)) \ge r\} \ge 0.5. \end{cases} \tag{6.35}$$

It follows from the monotonicity of $L(r)$ that we may employ bisection search to find the maximal value r such that $L(r) \ge \alpha$. This value is an estimation of $\overline{f}$. We summarize this process as follows.

Algorithm 6.3 (Hybrid Simulation for $U_3(\boldsymbol{x})$)
Step 1. Generate $\theta_1, \theta_2, \cdots, \theta_N$ from the credibility space $(\Theta, \mathcal{P}, \mathrm{Cr})$.
Step 2. Generate $\omega_1, \omega_2, \cdots, \omega_N$ from the probability space $(\Omega, \mathcal{A}, \Pr)$.
Step 3. $a = \min\limits_{1\le k\le N, 1\le j\le N} f(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k, \omega_j))$, $b = \max\limits_{1\le k\le N, 1\le j\le N} f(\boldsymbol{x}, \boldsymbol{\xi}(\theta_k, \omega_j))$.
Step 4. Set $r = (a + b)/2$.
Step 5. Compute $L(r)$ by (6.35).
Step 6. If $L(r) > \alpha$, then set $a = r$. Otherwise, set $b = r$.
Step 7. If $|a - b| > \varepsilon$ (a predetermined precision), then go to Step 4.
Step 8. Output r as the critical value.

In order to solve hybrid programming models, we may integrate hybrid simulation, NN and GA to produce a hybrid intelligent algorithm as follows,

Algorithm 6.4 (Hybrid Intelligent Algorithm)
Step 1. Generate training input-output data for uncertain functions like

$$\begin{aligned} &U_1 : \boldsymbol{x} \to E[f(\boldsymbol{x}, \boldsymbol{\xi})], \\ &U_2 : \boldsymbol{x} \to \mathrm{Ch}\left\{g_j(\boldsymbol{x}, \boldsymbol{\xi}) \le 0, j = 1, 2, \cdots, p\right\}, \\ &U_3 : \boldsymbol{x} \to \max\left\{\overline{f} \;\middle|\; \mathrm{Ch}\left\{f(\boldsymbol{x}, \boldsymbol{\xi}) \ge \overline{f}\right\} \ge \alpha\right\} \end{aligned}$$

by the hybrid simulation.
Step 2. Train a neural network to approximate the uncertain functions according to the generated training input-output data.
Step 3. Initialize *pop_size* chromosomes whose feasibility may be checked by the trained neural network.

Step 4. Update the chromosomes by crossover and mutation operations in which the feasibility of offspring may be checked by the trained neural network.
Step 5. Calculate the objective values for all chromosomes by the trained neural network.
Step 6. Compute the fitness of each chromosome according to the objective values.
Step 7. Select the chromosomes by spinning the roulette wheel.
Step 8. Repeat the fourth to seventh steps for a given number of cycles.
Step 9. Report the best chromosome as the optimal solution.

6.6 Numerical Experiments

We now provide some numerical examples to illustrate the effectiveness of hybrid intelligent algorithm.

Example 6.10: Consider the following hybrid EVM

$$\begin{cases} \min E\left[\sqrt{(x_1-\xi_1)^2+(x_2-\xi_2)^2+(x_3-\xi_3)^2}\right] \\ \text{subject to:} \\ \quad |x_1|+|x_2|+|x_3| \le 4 \end{cases}$$

where ξ_1, ξ_2 and ξ_3 are hybrid variables defined as

$$\begin{aligned} &\xi_1 \sim \mathcal{U}(\rho-1,\rho), \text{ with } \rho=(-2,-1,0), \\ &\xi_2 \sim \mathcal{U}(\rho,\rho+1), \text{ with } \rho=(-1,0,1), \\ &\xi_3 \sim \mathcal{U}(\rho+1,\rho+2), \text{ with } \rho=(0,1,2). \end{aligned}$$

In order to solve this model, we first generate input-output data for the uncertain function

$$U: \boldsymbol{x} \to E\left[\sqrt{(x_1-\xi_1)^2+(x_2-\xi_2)^2+(x_3-\xi_3)^2}\right]$$

by hybrid simulation. Then we train an NN (3 input neurons, 5 hidden neurons, 1 output neuron) to approximate the uncertain function U. Lastly, the trained NN is embedded into a GA to produce a hybrid intelligent algorithm.

A run of the hybrid intelligent algorithm (1000 cycles in hybrid simulation, 2000 data in NN, 400 generations in GA) shows that the optimal solution is

$$\boldsymbol{x}^* = (-1.3140, 0.2198, 2.4662)$$

whose objective value is 1.5059.

Example 6.11: Let us consider the following hybrid CCP,

$$\begin{cases} \max \overline{f} \\ \text{subject to:} \\ \quad \text{Ch}\left\{\xi_1 x_1 x_3 + \xi_2 x_2 x_4 \geq \overline{f}\right\} \geq 0.90 \\ \quad \text{Ch}\left\{(\xi_3 + x_1 + x_2)(\xi_4 + x_3 + x_4) \leq 30\right\} \geq 0.85 \\ \quad x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

where $\xi_1, \xi_2, \xi_3, \xi_4$ are hybrid variables defined as

$$\begin{aligned} &\xi_1 \sim \mathcal{N}(\rho, 1), \text{ with } \mu_\rho(x) = [1 - (x-1)^2] \vee 0, \\ &\xi_2 \sim \mathcal{N}(\rho, 1), \text{ with } \mu_\rho(x) = [1 - (x-2)^2] \vee 0, \\ &\xi_3 \sim \mathcal{N}(\rho, 1), \text{ with } \mu_\rho(x) = [1 - (x-3)^2] \vee 0, \\ &\xi_4 \sim \mathcal{N}(\rho, 1), \text{ with } \mu_\rho(x) = [1 - (x-4)^2] \vee 0. \end{aligned}$$

In order to solve this model, we produce input-output data for the uncertain function $U : \boldsymbol{x} \to (U_1(\boldsymbol{x}), U_2(\boldsymbol{x}))$, where

$$\begin{aligned} U_1(\boldsymbol{x}) &= \max\left\{\overline{f} \;\middle|\; \text{Ch}\left\{\xi_1 x_1 x_2 + \xi_2 x_3 x_4 \geq \overline{f}\right\} \geq 0.90\right\}, \\ U_2(\boldsymbol{x}) &= \text{Ch}\left\{(\xi_3 + x_1 + x_2)(\xi_4 + x_3 + x_4) \leq 30\right\}, \end{aligned}$$

by the hybrid simulation. Based on the input-output data, we train an NN (4 input neurons, 8 hidden neurons, 2 output neurons) to approximate the uncertain function U. After that, the trained NN is embedded into a GA to produce a hybrid intelligent algorithm.

A run of the hybrid intelligent algorithm (5000 cycles in simulation, 3000 training data in NN, 600 generations in GA) shows that the optimal solution is

$$(x_1^*, x_2^*, x_3^*, x_4^*) = (0.000, 1.303, 0.000, 1.978)$$

whose objective value is 2.85. Moreover, we have

$$\begin{aligned} &\text{Ch}\left\{\xi_1 x_1^* x_3^* + \xi_2 x_2^* x_4^* \geq 2.85\right\} \approx 0.90, \\ &\text{Ch}\left\{(\xi_3 + x_1^* + x_2^*)(\xi_4 + x_3^* + x_4^*) \leq 30\right\} \approx 0.85. \end{aligned}$$

Example 6.12: This is a hybrid DCP which maximizes the chance of an uncertain event subject to a deterministic constraint,

$$\begin{cases} \max \text{Ch}\left\{\xi_1 x_1 + \xi_2 x_2 + \xi_3 x_3 \geq 5\right\} \\ \text{subject to:} \\ \quad x_1^2 + x_2^2 + x_3^2 \leq 4 \end{cases}$$

where ξ_1, ξ_2, ξ_3 are hybrid variables defined as

$$\begin{aligned} &\xi_1 \sim \mathcal{N}(\rho, 1), \text{ with } \mu_\rho(x) = 1/[1 + (x-1)^2], \\ &\xi_2 \sim \mathcal{N}(\rho, 1), \text{ with } \mu_\rho(x) = 1/[1 + (x-2)^2], \\ &\xi_3 \sim \mathcal{N}(\rho, 1), \text{ with } \mu_\rho(x) = 1/[1 + (x-3)^2]. \end{aligned}$$

We produce a set of input-output data for the uncertain function

$$U : (x_1, x_2, x_3) \to \mathrm{Ch}\left\{\xi_1 x_1 + \xi_2 x_2 + \xi_3 x_3 \geq 5\right\}$$

by the hybrid simulation. According to the generated data, we train a feed-forward NN (3 input neurons, 5 hidden neurons, 1 output neuron) to approximate the uncertain function U. Then we integrate the trained NN and GA to produce a hybrid intelligent algorithm.

A run of the hybrid intelligent algorithm (6000 cycles in simulation, 2000 data in NN, 500 generations in GA) shows that the optimal solution is

$$(x_1^*, x_2^*, x_3^*) = (0.6847, 1.2624, 1.3919)$$

whose chance is 0.9.

Example 6.13: We consider the following hybrid DCGP,

$$\begin{cases} \text{lexmin}\left\{d_1^- \vee 0, d_2^- \vee 0, d_3^- \vee 0\right\} \\ \text{subject to:} \\ \quad 0.88 - \mathrm{Ch}\left\{x_1 + x_5 = 1\right\} = d_1^- \\ \quad 0.85 - \mathrm{Ch}\left\{x_2 + x_3 = 2\right\} = d_2^- \\ \quad 0.82 - \mathrm{Ch}\left\{x_4 + x_6 = 3\right\} = d_3^- \\ \quad x_1^2 \leq \xi_1 \\ \quad x_2^2 + x_3^2 + x_4^2 \leq \xi_2 \\ \quad x_5^2 + x_6^2 \leq \xi_3 \end{cases}$$

where $\xi_1, \xi_2, \xi_3, \xi_4$ are hybrid variables defined as follows,

$$\begin{aligned} &\xi_1 \sim \mathcal{EXP}(\rho), \text{ with } \mu_\rho(x) = [1 - (x-6)^2] \vee 0, \\ &\xi_2 \sim \mathcal{EXP}(\rho), \text{ with } \mu_\rho(x) = [1 - (x-30)^2] \vee 0, \\ &\xi_3 \sim \mathcal{EXP}(\rho), \text{ with } \mu_\rho(x) = [1 - (x-18)^2] \vee 0. \end{aligned}$$

At the first priority level, there is one event denoted by $\mathcal{E}_1$, which will be fulfilled by $x_1 + x_5 = 1$. It is clear that the support $\mathcal{E}_1^* = \{x_1, x_5\}$ and the dependent support $\mathcal{E}_1^{**} = \{x_1, x_5, x_6\}$. It follows from the principle of uncertainty that the chance function of the event $\mathcal{E}_1$ is

$$f_1(\boldsymbol{x}) = \mathrm{Ch}\left\{\begin{matrix} x_1 + x_5 = 1 \\ x_1^2 \leq \xi_1 \\ x_5^2 + x_6^2 \leq \xi_3 \end{matrix}\right\}.$$

At the second priority level, there is an event $\mathcal{E}_2$ which will be fulfilled by $x_2 + x_3 = 2$. The support $\mathcal{E}_2^* = \{x_2, x_3\}$ and the dependent support $\mathcal{E}_2^{**} = \{x_2, x_3, x_4\}$. It follows from the principle of uncertainty that the chance function of the event $\mathcal{E}_2$ is

$$f_2(\boldsymbol{x}) = \mathrm{Ch}\left\{\begin{matrix} x_2 + x_3 = 2 \\ x_2^2 + x_3^2 + x_4^2 \leq \xi_2 \end{matrix}\right\}.$$

At the third priority level, there is an event $\mathcal{E}_3$ which will be fulfilled by $x_4 + x_6 = 3$. The support $\mathcal{E}_3^* = \{x_4, x_6\}$ and the dependent support $\mathcal{E}_3^{**} = \{x_2, x_3, x_4, x_5, x_6\}$. It follows from the principle of uncertainty that the chance function of the event $\mathcal{E}_3$ is

$$f_3(\boldsymbol{x}) = \mathrm{Ch}\left\{\begin{array}{l} x_4 + x_6 = 3 \\ x_2^2 + x_3^2 + x_4^2 \leq \xi_2 \\ x_5^2 + x_6^2 \leq \xi_3 \end{array}\right\}.$$

In order to solve the hybrid DCGP model, we encode a solution by a chromosome $V = (v_1, v_2, v_3)$. Thus a chromosome can be converted into a solution by

$$x_1 = v_1, \quad x_2 = v_2, \quad x_3 = 2 - v_2, \quad x_4 = v_3, \quad x_5 = 1 - v_1, \quad x_6 = 3 - v_3.$$

We generate a set of input-output data for the uncertain function $U : (v_1, v_2, v_3) \to (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), f_3(\boldsymbol{x}))$ by the hybrid simulation. Then we train a feedforward NN to approximate the uncertain function U. After that, the trained NN is embedded into a GA to produce a hybrid intelligent algorithm. A run of the hybrid intelligent algorithm (6000 cycles in simulation, 3000 data in NN, 1000 generations in GA) shows that the optimal solution is

$$\boldsymbol{x}^* = (0.4005, 1.0495, 0.9505, 1.7574, 0.5995, 1.2427)$$

which can satisfy the first two goals, but the third objective is 0.05. In fact, we also have

$$f_1(\boldsymbol{x}^*) \approx 0.88, \quad f_2(\boldsymbol{x}^*) \approx 0.85, \quad f_3(\boldsymbol{x}^*) \approx 0.77.$$

Chapter 7

System Reliability Design

One of the approaches to improve system reliability is to provide redundancy for components in a system. There are two ways to provide component redundancy: parallel redundancy and standby redundancy. In parallel redundancy, all redundant elements are required to operate simultaneously. This method is usually used when element replacements are not permitted during the system operation. In standby redundancy, one of the redundant elements begins to work only when the active element fails. This method is usually employed when the replacement is allowable and can be finished immediately.

The system reliability design problem is to determine the optimal number of redundant elements for each component so as to optimize some system performance.

7.1 Problem Description

Assume that a system consists of n components, and the ith components consist of x_i redundant elements, $i = 1, 2, \cdots, n$, respectively. For example, Figure 7.1 shows a bridge system in which we suppose that redundant elements are in standby for the first and second components, and are in parallel for the third to fifth components.

The first problem is how to estimate the system lifetime when the value of the vector $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ is determined. For such a given decision vector $\boldsymbol{x}$, suppose that the redundant elements j operating in components i have lifetimes ξ_{ij}, $j = 1, 2, \cdots, x_i$, $i = 1, 2, \cdots, n$, respectively. For convenience, we use the vector

$$\boldsymbol{\xi} = (\xi_{11}, \xi_{12}, \cdots, \xi_{1x_1}, \xi_{21}, \xi_{22}, \cdots, \xi_{2x_2}, \cdots, \xi_{n1}, \xi_{n2}, \cdots, \xi_{nx_n})$$

to denote the lifetimes of all redundant elements in the system. For parallel redundancy components, the lifetimes are

$$T_i(\boldsymbol{x}, \boldsymbol{\xi}) = \max_{1 \le j \le x_i} \xi_{ij}, \quad i = 1, 2, \cdots, n.$$

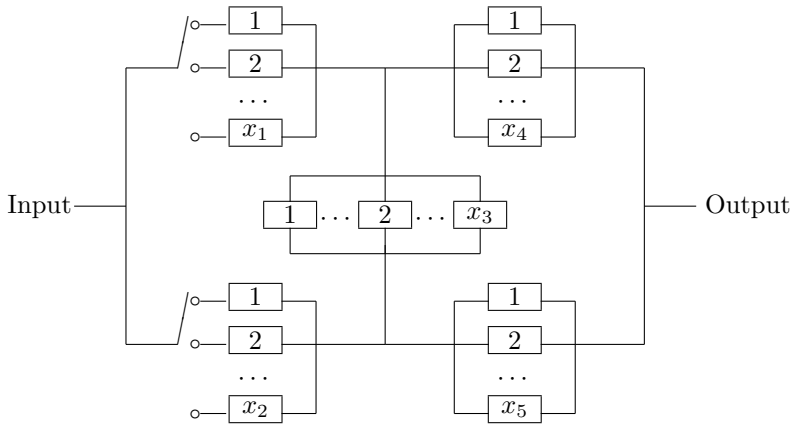

Figure 7.1: A Bridge System

For standby redundancy components, the lifetimes are

$$T_i(\boldsymbol{x}, \boldsymbol{\xi}) = \sum_{j=1}^{x_i} \xi_{ij}, \quad i = 1, 2, \cdots, n.$$

How do we calculate the system lifetime $T(\boldsymbol{x}, \boldsymbol{\xi})$? It is problem-dependent. For the bridge system shown in Figure 7.1, since the system works if and only if there is a path of working components from the input of the system to the output, the system lifetime is

$$T(\boldsymbol{x}, \boldsymbol{\xi}) = \max \left\{ \begin{array}{c} T_1(\boldsymbol{x}, \boldsymbol{\xi}) \wedge T_4(\boldsymbol{x}, \boldsymbol{\xi}) \\ T_2(\boldsymbol{x}, \boldsymbol{\xi}) \wedge T_5(\boldsymbol{x}, \boldsymbol{\xi}) \\ T_2(\boldsymbol{x}, \boldsymbol{\xi}) \wedge T_3(\boldsymbol{x}, \boldsymbol{\xi}) \wedge T_4(\boldsymbol{x}, \boldsymbol{\xi}) \\ T_1(\boldsymbol{x}, \boldsymbol{\xi}) \wedge T_3(\boldsymbol{x}, \boldsymbol{\xi}) \wedge T_5(\boldsymbol{x}, \boldsymbol{\xi}) \end{array} \right\}.$$

7.2 Stochastic Models

In practice, the lifetime $\boldsymbol{\xi}$ of elements is usually a random vector. Thus the component lifetimes and system lifetime are also random variables.

Stochastic Expected Lifetime Maximization Model

One of system performances is the *expected lifetime* $E[T(\boldsymbol{x}, \boldsymbol{\xi})]$. It is obvious that the greater the expected lifetime $E[T(\boldsymbol{x}, \boldsymbol{\xi})]$, the better the decision $\boldsymbol{x}$.

Let us consider the bridge system shown in Figure 7.1. For simplicity, we suppose that there is only one type of element to be selected for each

component. The lifetimes of elements are assumed to be exponentially distributed variables $\mathcal{EXP}(\beta)$ shown in Table 7.1. The decision vector may be represented by $\boldsymbol{x} = (x_1, x_2, \cdots, x_5)$, where x_i denote the numbers of the i-th types of elements selected, $i = 1, 2, \cdots, 5$, respectively.

Table 7.1: Random Lifetimes and Prices of Elements

Type	1	2	3	4	5
Lifetime	$\mathcal{EXP}(20)$	$\mathcal{EXP}(30)$	$\mathcal{EXP}(40)$	$\mathcal{EXP}(50)$	$\mathcal{EXP}(60)$
Price	50	60	70	80	90

Another important problem is to compute the cost spent for the system. It follows from Table 7.1 that the the total cost

$$C(\boldsymbol{x}) = 50x_1 + 60x_2 + 70x_3 + 80x_4 + 90x_5.$$

If the total capital available is 600, then we have a constraint $C(\boldsymbol{x}) \leq 600$.

For the redundancy system, since we wish to maximize the expected lifetime $E[T(\boldsymbol{x}, \boldsymbol{\xi})]$ subject to the cost constraint, we have the following stochastic expected lifetime maximization model,

$$\begin{cases} \max E[T(\boldsymbol{x}, \boldsymbol{\xi})] \\ \text{subject to:} \\ \quad C(\boldsymbol{x}) \leq 600 \\ \quad \boldsymbol{x} \geq 1, \text{ integer vector.} \end{cases} \tag{7.1}$$

Hybrid Intelligent Algorithm

In order to solve this type of model, we may employ the hybrid intelligent algorithm documented in Chapter 4 provided that the initialization, crossover and mutation operations are revised as follows.

Generally speaking, we use an integer vector $V = (x_1, x_2, \cdots, x_n)$ as a chromosome to represent a solution $\boldsymbol{x}$, where x_i are positive integers, $i = 1, 2, \cdots, n$. First we set all genes x_i as 1, $i = 1, 2, \cdots, n$, and form a chromosome V. Then we randomly sample an integer i between 1 and n, and the gene x_i of V is replaced with $x_i + 1$. We repeat this process until the chromosome V is proven to be infeasible. We take the last feasible chromosome as an initial chromosome.

We do the crossover operation on V_1 and V_2 in the following way. Write

$$V_1 = \left(x_1^{(1)}, x_2^{(1)}, \cdots, x_n^{(1)}\right), \quad V_2 = \left(x_1^{(2)}, x_2^{(2)}, \cdots, x_n^{(2)}\right)$$

and randomly generate two integers between 1 and n as the crossover points denoted by n_1 and n_2 such that $n_1 < n_2$. Then we exchange the genes of

the chromosomes V_1 and V_2 between n_1 and n_2 and produce two children as follows,

$$V_1' = \left(x_1^{(1)}, x_2^{(1)}, \cdots, x_{n_1-1}^{(1)}, x_{n_1}^{(2)}, \cdots, x_{n_2}^{(2)}, x_{n_2+1}^{(1)}, \cdots, x_n^{(1)}\right),$$

$$V_2' = \left(x_1^{(2)}, x_2^{(2)}, \cdots, x_{n_1-1}^{(2)}, x_{n_1}^{(1)}, \cdots, x_{n_2}^{(1)}, x_{n_2+1}^{(2)}, \cdots, x_n^{(2)}\right).$$

If the child V_1' is infeasible, then we use the following strategy to repair it and make it feasible. At first, we randomly sample an integer i between 1 and n, and then replace the gene x_i of V_1' with $x_i - 1$ provided that $x_i \geq 2$. Repeat this process until the revised chromosome V_1' is feasible. If the child V_1' is proven to be feasible, then we revise it in the following way. We randomly sample an integer i between 1 and n, and the gene x_i of V_1' will be replaced with x_i+1. We repeat this process until the revised chromosome is infeasible, and take the last feasible chromosome as V_1'. A similar revising process will be made on V_2'.

For each selected parent $V = (x_1, x_2, \cdots, x_n)$, we mutate it by the following way. We randomly choose two mutation positions n_1 and n_2 between 1 and n such that $n_1 < n_2$, then we set all genes x_j of V as 1 for $j = n_1, n_1 + 1, \cdots, n_2$, and form a new one

$$V' = (x_1, \cdots, x_{n_1-1}, 1, \cdots, 1, x_{n_2+1}, \cdots, x_n).$$

We will modify V' by the following process. We randomly sample an integer i between n_1 and n_2, and the gene x_i of V' will be replaced with $x_i + 1$. We repeat this process until the revised chromosome is infeasible. Finally, we replace the parent V' with the last feasible chromosome.

In order to solve the stochastic expected lifetime maximization model, we deal with the uncertain function $U : \boldsymbol{x} \to E[T(\boldsymbol{x}, \boldsymbol{\xi})]$ by stochastic simulation. Then, we embed the stochastic simulation into a GA and produce a hybrid intelligent algorithm. A run of the hybrid intelligent algorithm (5000 cycles in simulation and 1000 generations in GA) shows that the optimal solution is

$$\boldsymbol{x}^* = (1, 3, 1, 1, 2)$$

whose expected system lifetime is $E[T(\boldsymbol{x}^*, \boldsymbol{\xi})] = 62.5$, and total cost is $C(\boldsymbol{x}^*) = 560$.

Stochastic α-Lifetime Maximization Model

One type of system performance is the α-*lifetime* defined as the largest value $\overline{T}$ satisfying $\Pr\left\{T(\boldsymbol{x}, \boldsymbol{\xi}) \geq \overline{T}\right\} \geq \alpha$, where α is a predetermined confidence level.

This section will model redundancy optimization under this criterion. Consider the bridge system shown in Figure 7.1. The aim is to determine

the optimal numbers of the redundant elements so as to maximize the α-lifetime under the cost constraint. Zhao and Liu [333] presented the following stochastic α-lifetime maximization model,

$$\begin{cases} \max \overline{T} \\ \text{subject to:} \\ \quad \Pr\left\{T(\boldsymbol{x}, \boldsymbol{\xi}) \geq \overline{T}\right\} \geq \alpha \\ \quad C(\boldsymbol{x}) \leq 600 \\ \quad \boldsymbol{x} \geq 1, \text{ integer vector.} \end{cases} \tag{7.2}$$

For each observational vector $\boldsymbol{\xi}$ of lifetimes of elements, we may estimate the system lifetime $T(\boldsymbol{x}, \boldsymbol{\xi})$. We use the stochastic simulation to deal with the uncertain function $U(\boldsymbol{x}) = \max\left\{\overline{T} \mid \Pr\left\{T(\boldsymbol{x}, \boldsymbol{\xi}) \geq \overline{T}\right\} \geq \alpha\right\}$ based on the system structure function. Then the stochastic simulation is embedded into a GA to form a hybrid intelligent algorithm.

When $\alpha = 0.9$, a run of the hybrid intelligent algorithm (5000 cycles in simulation and 1000 generations in GA) shows that the optimal solution is

$$\boldsymbol{x}^* = (3, 2, 1, 2, 1)$$

with 0.9-system lifetime $\overline{T}^* = 25.7$, and $C(\boldsymbol{x}^*) = 590$.

Stochastic System Reliability Model

The third type of system performance is the *system reliability* $\Pr\{T(\boldsymbol{x}, \boldsymbol{\xi}) \geq T^0\}$, which is the probability that the system lifetime is greater than or equal to the given time T^0.

If one wants to maximize the system reliability under a cost constraint, then use the following stochastic system reliability model:

$$\begin{cases} \max \Pr\{T(\boldsymbol{x}, \boldsymbol{\xi}) \geq T^0\} \\ \text{subject to:} \\ \quad C(\boldsymbol{x}) \leq 600 \\ \quad \boldsymbol{x} \geq 1, \text{ integer vector.} \end{cases} \tag{7.3}$$

In order to solve this model, we deal with the uncertain function $U: \boldsymbol{x} \to \Pr\{T(\boldsymbol{x}, \boldsymbol{\xi}) \geq T^0\}$ by stochastic simulation. Then we embed the stochastic simulation into a GA to produce a hybrid intelligent algorithm.

A run of the hybrid intelligent algorithm (15000 cycles in simulation and 300 generations in GA) shows that the optimal solution is

$$\boldsymbol{x}^* = (4, 1, 1, 2, 1)$$

with $\Pr\{T(\boldsymbol{x}^*, \boldsymbol{\xi}) \geq T^0\} = 0.85$, and the total cost $C(x^*) = 580$.

7.3 Fuzzy Models

Although stochastic programming has been successfully applied in redundancy optimization, many problems require subjective judgment either due to the lack of data or due to the extreme complexity of the system. This fact motives us to apply fuzzy programming to redundancy optimization problems in which the lifetimes of elements are treated as fuzzy variables.

Fuzzy Expected Lifetime Maximization Model

Let us reconsider the bridge system shown in Figure 7.1. The lifetimes of elements are assumed to be triangular fuzzy variables shown in Table 7.2. The decision vector is represented by $\boldsymbol{x} = (x_1, x_2, \cdots, x_5)$, where x_i denote the numbers of the i-th types of elements selected, $i = 1, 2, \cdots, 5$, respectively. It follows from Table 7.2 that the the total cost

$$C(\boldsymbol{x}) = 50x_1 + 60x_2 + 70x_3 + 80x_4 + 90x_5.$$

If the total capital available is 600, then we have a constraint $C(\boldsymbol{x}) \leq 600$.

Table 7.2: Fuzzy Lifetimes and Prices of Elements

Type	1	2	3	4	5
Lifetime	$(10, 20, 40)$	$(20, 30, 50)$	$(30, 40, 60)$	$(40, 50, 60)$	$(50, 60, 80)$
Price	50	60	70	80	90

For such a standby redundancy system, we define *expected lifetime* as $E[T(\boldsymbol{x}, \boldsymbol{\xi})]$. If we wish to maximize the expected lifetime $E[T(\boldsymbol{x}, \boldsymbol{\xi})]$, then we have the following fuzzy expected lifetime maximization model (Zhao and Liu [336]),

$$\begin{cases} \max E[T(\boldsymbol{x}, \boldsymbol{\xi})] \\ \text{subject to:} \\ \quad C(\boldsymbol{x}) \leq 600 \\ \quad \boldsymbol{x} \geq 1, \text{ integer vector.} \end{cases} \tag{7.4}$$

A run of the hybrid intelligent algorithm (15000 cycles in simulation, 5000 data in NN, 300 generations in GA) shows that the optimal solution is

$$\boldsymbol{x}^* = (2, 4, 1, 1, 1)$$

whose expected system lifetime is $E[T(\boldsymbol{x}^*, \boldsymbol{\xi})] = 60.7$, and the total cost $C(x^*) = 580$.

Fuzzy α-Lifetime Maximization Model

By *α-lifetime* we mean the largest value $\overline{T}$ satisfying $\mathrm{Cr}\left\{T(\boldsymbol{x},\boldsymbol{\xi}) \geq \overline{T}\right\} \geq \alpha$, where α is a predetermined confidence level.

If the decision maker wants to maximize the α-lifetime subject to the cost constraint, then we have the following fuzzy α-lifetime maximization model (Zhao and Liu [336]),

$$\begin{cases} \max \overline{T} \\ \text{subject to:} \\ \quad \mathrm{Cr}\left\{T(\boldsymbol{x},\boldsymbol{\xi}) \geq \overline{T}\right\} \geq \alpha \\ \quad C(\boldsymbol{x}) \leq 600 \\ \quad \boldsymbol{x} \geq 1, \text{ integer vector.} \end{cases} \tag{7.5}$$

When $\alpha = 0.8$, a run of the hybrid intelligent algorithm (15000 cycles in simulation and 300 generations in GA) shows that the optimal solution is

$$\boldsymbol{x}^* = (2, 4, 1, 1, 1)$$

whose 0.8-system lifetime $\overline{T}^* = 53.1$, and the total cost $C(\boldsymbol{x}^*) = 580$.

Fuzzy System Reliability Model

By *system reliability* we mean $\mathrm{Cr}\{T(\boldsymbol{x},\boldsymbol{\xi}) \geq T^0\}$, i.e., the credibility that the system lifetime is greater than or equal to the given time T^0.

If one wants to maximize the system reliability under a cost constraint, then use the following fuzzy system reliability model (Zhao and Liu [336]):

$$\begin{cases} \max \mathrm{Cr}\{T(\boldsymbol{x},\boldsymbol{\xi}) \geq T^0\} \\ \text{subject to:} \\ \quad C(\boldsymbol{x}) \leq 600 \\ \quad \boldsymbol{x} \geq 1, \text{ integer vector.} \end{cases} \tag{7.6}$$

When $T^0 = 50$, a run of the hybrid intelligent algorithm (15000 cycles in simulation and 300 generations in GA) shows that the optimal solution is

$$\boldsymbol{x}^* = (2, 4, 1, 1, 1)$$

with $\mathrm{Cr}\{T(\boldsymbol{x}^*,\boldsymbol{\xi}) \geq T^0\} = 0.95$ and the total cost $C(\boldsymbol{x}^*) = 580$.

7.4 Hybrid Models

In a classical system reliability design problem, the element lifetimes are assumed to be random variables or fuzzy variables. Although this assumption has been accepted and accorded with the facts in widespread cases, it is not

appropriate in a vast range of situations. In many practical situations, the fuzziness and randomness of the element lifetimes are often mixed up with each other. For example, the element lifetimes are assumed to be exponentially distributed variables with unknown parameters. In this case, fuzziness and randomness of the element lifetimes are required to be considered simultaneously and the effectiveness of the classical redundancy optimization theory is lost.

Hybrid Expected Lifetime Maximization Model

Now we suppose that the element lifetimes are hybrid variables. If we wish to maximize the expected lifetime $E[T(\boldsymbol{x}, \boldsymbol{\xi})]$, then we have the following hybrid expected lifetime maximization model,

$$\begin{cases} \max E[T(\boldsymbol{x}, \boldsymbol{\xi})] \\ \text{subject to:} \\ \quad C(\boldsymbol{x}) \leq 600 \\ \quad \boldsymbol{x} \geq 1, \text{ integer vector.} \end{cases} \tag{7.7}$$

Hybrid α-Lifetime Maximization Model

We define the largest value $\overline{T}$ satisfying $\mathrm{Ch}\left\{T(\boldsymbol{x}, \boldsymbol{\xi}) \geq \overline{T}\right\} \geq \alpha$ as the α-*lifetime* of the system, where α is a predetermined confidence level. If the decision maker wants to maximize the α-lifetime subject to the cost constraint, then we have the following hybrid α-lifetime maximization model,

$$\begin{cases} \max \overline{T} \\ \text{subject to:} \\ \quad \mathrm{Ch}\left\{T(\boldsymbol{x}, \boldsymbol{\xi}) \geq \overline{T}\right\} \geq \alpha \\ \quad C(\boldsymbol{x}) \leq 600 \\ \quad \boldsymbol{x} \geq 1, \text{ integer vector.} \end{cases} \tag{7.8}$$

Hybrid System Reliability Model

By *system reliability* we mean $\mathrm{Ch}\{T(\boldsymbol{x}, \boldsymbol{\xi}) \geq T^0\}$, i.e., the chance that the system lifetime is greater than or equal to the given time T^0. If one wants to maximize the system reliability under a cost constraint, then use the following hybrid system reliability model:

$$\begin{cases} \max \mathrm{Ch}\{T(\boldsymbol{x}, \boldsymbol{\xi}) \geq T^0\} \\ \text{subject to:} \\ \quad C(\boldsymbol{x}) \leq 600 \\ \quad \boldsymbol{x} \geq 1, \text{ integer vector.} \end{cases} \tag{7.9}$$

Problem 7.1. Design a hybrid intelligent algorithm to solve hybrid models for system reliability design problem.

Chapter 8

Project Scheduling Problem

Project scheduling problem is to determine the schedule of allocating resources so as to balance the total cost and the completion time. Uncertainty always exists in project scheduling problem, due to the vagueness of project activity duration times. Freeman [73][74] first introduced probability theory into project scheduling problem in 1960. After that, project scheduling problem has been studied widely. This chapter will introduce some optimization models for stochastic and fuzzy project scheduling problems proposed by Ke and Liu [125][127].

8.1 Problem Description

Project scheduling is usually represented by a directed acyclic graph where nodes correspond to milestones, and arcs to activities which are basically characterized by the times and costs consumed.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ be a directed acyclic graph, where $\mathcal{V} = \{1, 2, \cdots, n, n+1\}$ is the set of nodes, $\mathcal{A}$ is the set of arcs, $(i, j) \in \mathcal{A}$ is the arc of the graph $\mathcal{G}$ from nodes i to j. It is well-known that we can rearrange the indexes of the nodes in $\mathcal{V}$ such that $i < j$ for all $(i, j) \in \mathcal{A}$.

Before we begin to study project scheduling problem with stochastic activity duration times, we first make some assumptions: (a) all of the costs needed are obtained via loans with some given interest rate; and (b) each activity can be processed only if the loan needed is allocated and all the foregoing activities are finished.

In order to model the project scheduling problem, we first introduce the following indices and parameters:

ξ_{ij}: uncertain duration time of activity (i, j) in $\mathcal{A}$;

c_{ij}: cost of activity (i, j) in $\mathcal{A}$;

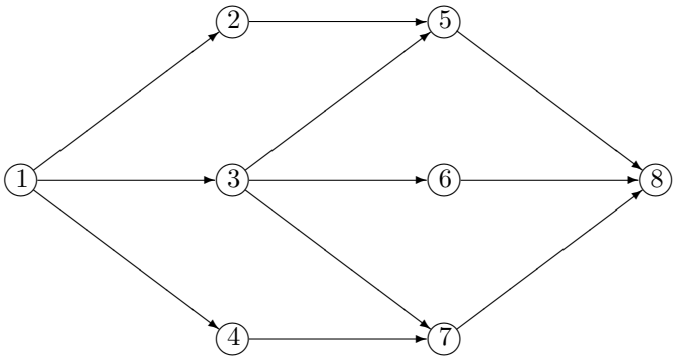

Figure 8.1: A Project Graph

r: the interest rate;

x_i: integer decision variable representing the allocating time of all loans needed for all activities (i, j) in $\mathcal{A}$.

For simplicity, we also write $\boldsymbol{\xi} = \{\xi_{ij} : (i, j) \in \mathcal{A}\}$, $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$. Denote $T_i(\boldsymbol{x}, \boldsymbol{\xi})$ as the starting time of all activities (i, j) in $\mathcal{A}$. According to the assumptions, the starting time of the total project should be

$$T_1(\boldsymbol{x}, \boldsymbol{\xi}) = x_1. \tag{8.1}$$

The starting time of activities (i, j) in $\mathcal{A}$ should be

$$T_i(\boldsymbol{x}, \boldsymbol{\xi}) = x_i \vee \max_{(k,i)\in\mathcal{A}} \{T_k(\boldsymbol{x}, \boldsymbol{\xi}) + \xi_{ki}\}, \quad i = 2, 3, \cdots, n. \tag{8.2}$$

The completion time of the total project is

$$T(\boldsymbol{x}, \boldsymbol{\xi}) = \max_{(k,n+1)\in\mathcal{A}} \{T_k(\boldsymbol{x}, \boldsymbol{\xi}) + \xi_{k,n+1}\}. \tag{8.3}$$

Therefore, the total cost of the project can be written as

$$C(\boldsymbol{x}, \boldsymbol{\xi}) = \sum_{(i,j)\in\mathcal{A}} c_{ij} \, (1 + r)^{\lceil (T(\boldsymbol{x},\boldsymbol{\xi}) - x_i \rceil} \tag{8.4}$$

where $\lceil a \rceil$ represents the minimal integer greater than or equal to a.

8.2 Stochastic Models

In this section, we assume that all duration times are all random variables, and introduce stochastic expected cost minimization model, α-cost minimization model and probability maximization model.

Stochastic Expected Cost Minimization Model

If we want to minimize the expected cost of the project under the expected completion time constraint, we may construct the following stochastic expected cost minimization model (Ke and Liu [125]),

$$\begin{cases} \min E[C(\boldsymbol{x}, \boldsymbol{\xi})] \\ \text{subject to:} \\ \quad E[T(\boldsymbol{x}, \boldsymbol{\xi})] \leq T^0 \\ \quad \boldsymbol{x} \geq 0, \text{ integer vector} \end{cases} \tag{8.5}$$

where T^0 is the due date of the project, $T(\boldsymbol{x}, \boldsymbol{\xi})$ and $C(\boldsymbol{x}, \boldsymbol{\xi})$ are the completion time and total cost defined by (8.3) and (8.4), respectively.

Example 8.1: Now let us consider a project scheduling problem shown in Figure 8.1. The duration times and the costs needed for the relevant activities in the project are presented in Table 8.1, and the monthly interest rate $r = 0.006$ according to some practical case. Note that the activity duration times are assumed to be normally distributed random variables $\mathcal{N}(\mu, \sigma^2)$.

Table 8.1: Random Duration Times and Costs of Activities

Arc	Duration Time (Month)	Cost	Arc	Duration Time (Month)	Cost
(1,2)	$\mathcal{N}(9,3)$	1500	(1,3)	$\mathcal{N}$(5,2)	1800
(1,4)	$\mathcal{N}$(10,3)	430	(2,5)	$\mathcal{N}(6,2)$	1600
(3,5)	$\mathcal{N}(8,2)$	800	(3,6)	$\mathcal{N}(8,2)$	500
(3,7)	$\mathcal{N}(9,3)$	2000	(4,7)	$\mathcal{N}(6,1)$	2100
(5,8)	$\mathcal{N}(10,2)$	550	(6,8)	$\mathcal{N}$(15,3)	530
(7,8)	$\mathcal{N}(11,2)$	630			

Now we are requested to finish the project within 32 months, i.e., $T^0 = 32$ (month). A run of the hybrid intelligent algorithm (5000 cycles in simulation, 4000 generations in GA) shows that the expected cost $E[C(\boldsymbol{x}^*, \boldsymbol{\xi})] = 14345$, the expected completion time $E[T(\boldsymbol{x}^*, \boldsymbol{\xi})] = 30.4$, and the optimal schedule for the project is shown in Table 8.2.

Stochastic α-Cost Minimization Model

The α-cost of a project is defined as $\min\left\{\bar{C} \middle| \Pr\{C(\boldsymbol{x}, \boldsymbol{\xi}) \leq \bar{C}\} \geq \alpha\right\}$, where α is a predetermined confidence level. If we want to minimize the α-cost

Table 8.2: Schedule of Expected Cost Model

Date	1	6	12	13	17	18
Node	1	3	2,4	6	7	5
Loan	3730	3300	3700	530	630	550

of the project under the completion time chance constraint with a predetermined confidence level β, we have the following stochastic α-cost minimization model (Ke and Liu [125]),

$$\begin{cases} \min \bar{C} \\ \text{subject to:} \\ \quad \Pr\{C(\boldsymbol{x}, \boldsymbol{\xi}) \leq \bar{C}\} \geq \alpha \\ \quad \Pr\{T(\boldsymbol{x}, \boldsymbol{\xi}) \leq T^0\} \geq \beta \\ \quad \boldsymbol{x} \geq 0, \text{ integer vector} \end{cases} \tag{8.6}$$

where T^0 is the due date of the project, $T(\boldsymbol{x}, \boldsymbol{\xi})$ and $C(\boldsymbol{x}, \boldsymbol{\xi})$ are the completion time and total cost defined by (8.3) and (8.4), respectively.

Example 8.2: If $\alpha = 0.9$, $\beta = 0.9$ and $T^0 = 32$ (month), then a run of the hybrid intelligent algorithm (5000 cycles in simulation, 4000 generations in GA) shows that the α-cost $C^* = 14502$, $\Pr\{C(\boldsymbol{x}^*, \boldsymbol{\xi}) \leq C^*\} = 0.901$, $\Pr\{T(\boldsymbol{x}^*, \boldsymbol{\xi}) \leq T^0\} = 0.905$, and the optimal schedule is presented in Table 8.3.

Table 8.3: Schedule of α-Cost Minimization Model

Date	1	5	12	13	18
Node	1	3	2,4	6	5,7
Loan	3730	3300	3700	530	1180

Probability Maximization Model

If we want to maximize the probability that the total cost should not exceed a predetermined level C^0 subject to the chance constraint $\Pr\{T(\boldsymbol{x}, \boldsymbol{\xi}) \leq T^0\} \geq \beta$, then we have the following probability maximization model (Ke and Liu [125]),

$$\begin{cases} \max \Pr\left\{C(\boldsymbol{x}, \boldsymbol{\xi}) \leq C^0\right\} \\ \text{subject to:} \\ \quad \Pr\{T(\boldsymbol{x}, \boldsymbol{\xi}) \leq T^0\} \geq \beta \\ \quad \boldsymbol{x} \geq 0, \text{ integer vector} \end{cases} \tag{8.7}$$

where $T(\boldsymbol{x}, \boldsymbol{\xi})$ and $C(\boldsymbol{x}, \boldsymbol{\xi})$ are the completion time and total cost defined by (8.3) and (8.4), respectively.

Example 8.3: If $C^0 = 14530$, $T^0 = 32$ and $\beta = 0.9$, then a run of the hybrid intelligent algorithm (5000 cycles in simulation, 4000 generations in GA) shows that the probability $\Pr\{C(\boldsymbol{x}^*, \boldsymbol{\xi}) \leq C^0\} = 0.912$, $\Pr\{T(\boldsymbol{x}^*, \boldsymbol{\xi}) \leq T^0\} = 0.906$ and the optimal schedule is presented in Table 8.4.

Table 8.4: Schedule of Probability Maximization Model

Date	1	6	10	11	13	17	18
Node	1	3	2	4	6	7	5
Loan	3730	3300	1600	2100	530	630	550

8.3 Fuzzy Models

This section will assume that the activity duration times are all fuzzy variables rather than random ones, and introduce fuzzy expected cost minimization model, α-cost minimization model and credibility maximization model.

Fuzzy Expected Cost Minimization Model

If we want to minimize the expected cost of the project under the expected completion time constraint, we may construct the following fuzzy expected cost minimization model (Ke and Liu [127]),

$$\begin{cases} \min E[C(\boldsymbol{x}, \boldsymbol{\xi})] \\ \text{subject to:} \\ \quad E[T(\boldsymbol{x}, \boldsymbol{\xi})] \leq T^0 \\ \quad \boldsymbol{x} \geq 0, \text{ integer vector} \end{cases} \tag{8.8}$$

where T^0 is the due date of the project, $T(\boldsymbol{x}, \boldsymbol{\xi})$ and $C(\boldsymbol{x}, \boldsymbol{\xi})$ are the completion time and total cost defined by (8.3) and (8.4), respectively.

Example 8.4: Now let us consider a project scheduling problem shown in Figure 8.1. The duration times and the costs needed for the relevant activities in the project are presented in Table 8.5, and the monthly interest rate $r = 0.006$ according to some practical case. Note that the activity duration times are assumed to be triangular fuzzy variables.

Now we are requested to finish the project within 32 months, i.e., $T^0 = 32$ (month). A run of the hybrid intelligent algorithm (5000 cycles in simulation, 4000 generations in GA) shows that the expected cost $E[C(\boldsymbol{x}^*, \boldsymbol{\xi})] = 14286$, the expected completion time $E[T(\boldsymbol{x}^*, \boldsymbol{\xi})] = 30.4$, and the optimal schedule for the project is shown in Table 8.6.

Table 8.5: Fuzzy Duration Times and Costs of Activities

Arc	Duration Time (Month)	Cost	Arc	Duration Time (Month)	Cost
(1,2)	$(7, 9, 12)$	1500	(1,3)	(3,5,7)	1800
(1,4)	(7,10,12)	430	(2,5)	(4,6,9)	1600
(3,5)	(6,8,11)	800	(3,6)	(6,8,10)	500
(3,7)	(6,9,12)	2000	(4,7)	(5,6,8)	2100
(5,8)	(8,10,12)	550	(6,8)	(13,16,18)	530
(7,8)	(9,11,13)	630			

Table 8.6: Schedule of Expected Cost Model

Date	1	7	12	13	14	18	19
Node	1	3	6	4	2	7	5
Loan	3730	3300	530	2100	1600	630	550

Fuzzy α-Cost Minimization Model

The α-cost of a project is defined as $\min\left\{\bar{C}|\mathrm{Cr}\{C(\boldsymbol{x},\boldsymbol{\xi})\leq\bar{C}\}\geq\alpha\right\}$, where α is a predetermined confidence level. If we want to minimize the α-cost of the project under the completion time chance constraint with a predetermined confidence level β, we have the following fuzzy α-cost minimization model (Ke and Liu [127]),

$$\begin{cases} \min \bar{C} \\ \text{subject to:} \\ \quad \mathrm{Cr}\{C(\boldsymbol{x},\boldsymbol{\xi})\leq\bar{C}\}\geq\alpha \\ \quad \mathrm{Cr}\{T(\boldsymbol{x},\boldsymbol{\xi})\leq T^0\}\geq\beta \\ \quad \boldsymbol{x}\geq 0, \text{ integer vector} \end{cases} \tag{8.9}$$

where T^0 is the due date of the project, $T(\boldsymbol{x},\boldsymbol{\xi})$ and $C(\boldsymbol{x},\boldsymbol{\xi})$ are the completion time and total cost defined by (8.3) and (8.4), respectively.

Example 8.5: If $\alpha = 0.9$, $\beta = 0.9$ and $T^0 = 32$ (month), then a run of the hybrid intelligent algorithm (5000 cycles in simulation, 4000 generations in GA) shows that the α-cost $C^* = 14331$, $\mathrm{Cr}\{C(\boldsymbol{x}^*,\boldsymbol{\xi})\leq C^*\} = 0.913$, $\mathrm{Cr}\{T(\boldsymbol{x}^*,\boldsymbol{\xi})\leq T^0\} = 0.917$, and the optimal schedule is presented in Table 8.7.

Table 8.7: Schedule of α-Cost Minimization Model

Date	1	6	13	14	15	20	21
Node	1	3	4	2	6	7	5
Loan	3730	3300	2100	1600	530	630	550

Credibility Maximization Model

If we want to maximize the credibility that the total cost should not exceed a predetermined level C^0 subject to the chance constraint $\mathrm{Cr}\{T(\boldsymbol{x},\boldsymbol{\xi}) \le T^0\} \ge \beta$, then we have the following credibility maximization model (Ke and Liu [127]),

$$\begin{cases} \max \mathrm{Cr}\left\{C(\boldsymbol{x},\boldsymbol{\xi}) \le C^0\right\} \\ \text{subject to:} \\ \quad \mathrm{Cr}\{T(\boldsymbol{x},\boldsymbol{\xi}) \le T^0\} \ge \beta \\ \quad \boldsymbol{x} \ge 0, \text{ integer vector} \end{cases} \tag{8.10}$$

where $T(\boldsymbol{x},\boldsymbol{\xi})$ and $C(\boldsymbol{x},\boldsymbol{\xi})$ are the completion time and total cost defined by (8.3) and (8.4), respectively.

Example 8.6: If $C^0 = 14370$, $T^0 = 32$ and $\beta = 0.9$, then a run of the hybrid intelligent algorithm (5000 cycles in simulation, 4000 generations in GA) shows that the credibility $\mathrm{Cr}\{C(\boldsymbol{x}^*,\boldsymbol{\xi}) \le C^0\} = 0.95$, $\mathrm{Cr}\{T(\boldsymbol{x}^*,\boldsymbol{\xi}) \le T^0\} = 0.95$ and the optimal schedule is presented in Table 8.8.

Table 8.8: Schedule of Credibility Maximization Model

Date	1	6	13	14	19
Node	1	3	4,6	2	5,7
Loan	3730	3300	2630	1600	1180

8.4 Hybrid Models

We suppose that the duration times are hybrid variables, and introduce hybrid expected cost minimization model, α-cost minimization model and chance maximization model.

Hybrid Expected Cost Minimization Model

If we want to minimize the expected cost of the project under the expected completion time constraint, we may construct the following hybrid expected

cost minimization model,

$$\begin{cases} \min E[C(\boldsymbol{x}, \boldsymbol{\xi})] \\ \text{subject to:} \\ \quad E[T(\boldsymbol{x}, \boldsymbol{\xi})] \leq T^0 \\ \quad \boldsymbol{x} \geq 0, \text{ integer vector} \end{cases} \tag{8.11}$$

where T^0 is the due date of the project, $T(\boldsymbol{x}, \boldsymbol{\xi})$ and $C(\boldsymbol{x}, \boldsymbol{\xi})$ are the completion time and total cost defined by (8.3) and (8.4), respectively.

Hybrid α-Cost Minimization Model

The α-cost of a project is defined as $\min\left\{\bar{C} | \text{Ch}\{C(\boldsymbol{x}, \boldsymbol{\xi}) \leq \bar{C}\} \geq \alpha\right\}$, where α is a predetermined confidence level. If we want to minimize the α-cost of the project under the completion time chance constraint with predetermined confidence level β, we have the following hybrid α-cost minimization model (Ke and Liu [126]),

$$\begin{cases} \min \bar{C} \\ \text{subject to:} \\ \quad \text{Ch}\{C(\boldsymbol{x}, \boldsymbol{\xi}) \leq \bar{C}\} \geq \alpha \\ \quad \text{Ch}\{T(\boldsymbol{x}, \boldsymbol{\xi}) \leq T^0\} \geq \beta \\ \quad \boldsymbol{x} \geq 0, \text{ integer vector} \end{cases} \tag{8.12}$$

where T^0 is the due date of the project, $T(\boldsymbol{x}, \boldsymbol{\xi})$ and $C(\boldsymbol{x}, \boldsymbol{\xi})$ are the completion time and total cost defined by (8.3) and (8.4), respectively.

Chance Maximization Model

If we want to maximize the chance that the total cost should not exceed a predetermined level C^0 subject to the chance constraint $\text{Ch}\{T(\boldsymbol{x}, \boldsymbol{\xi}) \leq T^0\} \geq \alpha$, then we have the following chance maximization model,

$$\begin{cases} \max \text{Ch}\left\{C(\boldsymbol{x}, \boldsymbol{\xi}) \leq C^0\right\} \\ \text{subject to:} \\ \quad \text{Ch}\{T(\boldsymbol{x}, \boldsymbol{\xi}) \leq T^0\} \geq \alpha \\ \quad \boldsymbol{x} \geq 0, \text{ integer vector} \end{cases} \tag{8.13}$$

where $T(\boldsymbol{x}, \boldsymbol{\xi})$ and $C(\boldsymbol{x}, \boldsymbol{\xi})$ are the completion time and total cost defined by (8.3) and (8.4), respectively.

Problem 8.1. Design a hybrid intelligent algorithm to solve hybrid models for project scheduling problem.

Chapter 9

Vehicle Routing Problem

Vehicle routing problem (VRP) is concerned with finding efficient routes, beginning and ending at a central depot, for a fleet of vehicles to serve a number of customers. See Figure 9.1.

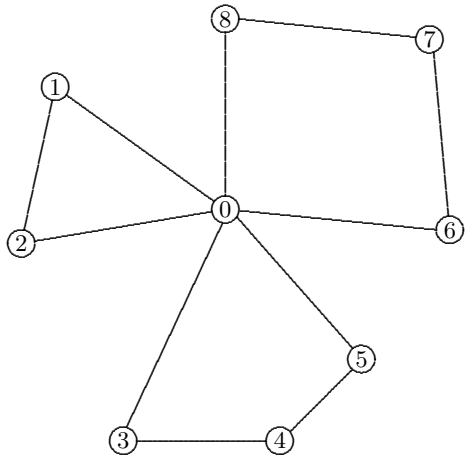

Figure 9.1: A Vehicle Routing Graph

Due to its wide applicability and economic importance, VRP has been extensively studied. Practically, there are uncertain factors in VRP. Waters [308] listed various potential uncertainties, such as demands of customers, travel times between customers, customers to be visited, locations of customers, capacities of vehicles, and number of vehicles available. This fact provides a motivation to study uncertain VRP. Bertsimas and Simchi-Levi [17] and Gendreau et al. [90] surveyed the new developments in the research area of stochastic VRP. Dror et al. [57], Bastian and Rinnooy Kan [11], Laporte et al. [148], and Liu and Lai [184] presented several models for stochastic

VRP. When the travel times are provided as fuzzy variables rather than random variables, Zheng and Liu [339] suggested a fuzzy optimization model for VRP.

9.1 Problem Description

We assume that: (a) a vehicle will be assigned for only one route on which there may be more than one customer; (b) a customer will be visited by one and only one vehicle; (c) each route begins and ends at the depot; and (d) each customer specifies its time window within which the delivery is permitted or preferred to start.

Let us first introduce the following indices and model parameters:

$i = 0$: depot;

$i = 1, 2, \cdots, n$: customers;

$k = 1, 2, \cdots, m$: vehicles;

D_{ij}: the travel distance from customers i to j, $i, j = 0, 1, 2, \cdots, n$;

T_{ij}: the uncertain travel time from customers i to j, $i, j = 0, 1, 2, \cdots, n$;

S_i: the unloading time at customer i, $i = 1, 2, \cdots, n$;

$[a_i, b_i]$: the time window of customer i, where a_i and b_i are the beginning and end of the time window, $i = 1, 2, \cdots, n$, respectively.

In this book, the operational plan is represented by Liu's formulation [182] via three decision vectors $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{t}$, where

$\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$: integer decision vector representing n customers with $1 \le x_i \le n$ and $x_i \ne x_j$ for all $i \ne j$, $i, j = 1, 2, \cdots, n$. That is, the sequence $\{x_1, x_2, \cdots, x_n\}$ is a rearrangement of $\{1, 2, \cdots, n\}$;

$\boldsymbol{y} = (y_1, y_2, \cdots, y_{m-1})$: integer decision vector with $y_0 \equiv 0 \le y_1 \le y_2 \le \cdots \le y_{m-1} \le n \equiv y_m$;

$\boldsymbol{t} = (t_1, t_2, \cdots, t_m)$: each t_k represents the starting time of vehicle k at the depot, $k = 1, 2, \cdots, m$.

We note that the operational plan is fully determined by the decision vectors $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{t}$ in the following way. For each k ($1 \le k \le m$), if $y_k = y_{k-1}$, then vehicle k is not used; if $y_k > y_{k-1}$, then vehicle k is used and starts from the depot at time t_k, and the tour of vehicle k is $0 \to x_{y_{k-1}+1} \to x_{y_{k-1}+2} \to \cdots \to x_{y_k} \to 0$. Thus the tours of all vehicles are as follows:

$$\begin{aligned}
&\text{Vehicle 1: } 0 \to x_{y_0+1} \to x_{y_0+2} \to \cdots \to x_{y_1} \to 0;\\
&\text{Vehicle 2: } 0 \to x_{y_1+1} \to x_{y_1+2} \to \cdots \to x_{y_2} \to 0;\\
&\quad \cdots\\
&\text{Vehicle } m\text{: } 0 \to x_{y_{m-1}+1} \to x_{y_{m-1}+2} \to \cdots \to x_{y_m} \to 0.
\end{aligned} \tag{9.1}$$

It is clear that this type of representation is intuitive, and the total number of decision variables is $n + 2m - 1$. We also note that the above decision variables $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{t}$ ensure that: (a) each vehicle will be used at most one

time; (b) all tours begin and end at the depot; (c) each customer will be visited by one and only one vehicle; and (d) there is no subtour.

Let $f_i(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t})$ be the arrival time function of some vehicles at customers i for $i = 1, 2, \cdots, n$. We remind readers that $f_i(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t})$ are determined by the decision variables $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{t}$, $i = 1, 2, \cdots, n$. Since unloading can start either immediately, or later, when a vehicle arrives at a customer, the calculation of $f_i(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t})$ is heavily dependent on the operational strategy. Here we assume that the customer does not permit a delivery earlier than the time window. That is, the vehicle will wait to unload until the beginning of the time window if it arrives before the time window. If a vehicle arrives at a customer after the beginning of the time window, unloading will start immediately. For each k with $1 \leq k \leq m$, if vehicle k is used (i.e., $y_k > y_{k-1}$), then we have

$$f_{x_{y_{k-1}+1}}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t}) = t_k + T_{0x_{y_{k-1}+1}} \tag{9.2}$$

and

$$\begin{aligned} f_{x_{y_{k-1}+j}}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t}) = \ & f_{x_{y_{k-1}+j-1}}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t}) \vee a_{x_{y_{k-1}+j-1}} \\ & + S_{x_{y_{k-1}+j-1}} + T_{x_{y_{k-1}+j-1}x_{y_{k-1}+j}} \end{aligned} \tag{9.3}$$

for $2 \leq j \leq y_k - y_{k-1}$, where $\vee$ denotes the maximum operator. It follows from the uncertainty of travel times T_{ij}'s that the arrival times $f_i(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t})$, $i = 1, 2, \cdots, n$ are uncertain variables fully determined by (9.2) and (9.3).

Let $g(\boldsymbol{x}, \boldsymbol{y})$ be the total travel distance of all vehicles. Then we have

$$g(\boldsymbol{x}, \boldsymbol{y}) = \sum_{k=1}^{m} g_k(\boldsymbol{x}, \boldsymbol{y}) \tag{9.4}$$

where

$$g_k(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} D_{0x_{y_{k-1}+1}} + \sum\limits_{j=y_{k-1}+1}^{y_k-1} D_{x_j x_{j+1}} + D_{x_{y_k}0}, & \text{if } y_k > y_{k-1} \\ 0, & \text{if } y_k = y_{k-1} \end{cases}$$

for $k = 1, 2, \cdots, m$.

9.2 Stochastic Models

Now we assume that the travel times are random variables, and introduce stochastic distance minimization model and probability maximization model.

Stochastic Distance Minimization Model

If we hope that all customers are visited within their time windows with a confidence level α, then we have the following chance constraint,

$$\Pr\{a_i \leq f_i(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t}) \leq b_i, i = 1, 2, \cdots, n\} \geq \alpha. \tag{9.5}$$

If we want to minimize the total travel distance of all vehicles subject to the time window constraint, then we have the following stochastic distance minimization model (Liu and Lai [184]),

$$\begin{cases} \min g(\boldsymbol{x}, \boldsymbol{y}) \\ \text{subject to:} \\ \quad \Pr\{a_i \le f_i(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t}) \le b_i, i = 1, 2, \cdots, n\} \ge \alpha \\ \quad 1 \le x_i \le n, \quad i = 1, 2, \cdots, n \\ \quad x_i \ne x_j, \quad i \ne j, \ i, j = 1, 2, \cdots, n \\ \quad 0 \le y_1 \le y_2 \le \cdots \le y_{m-1} \le n \\ \quad x_i, y_j, \quad i = 1, 2, \cdots, n, \quad j = 1, 2, \cdots, m-1, \quad \text{integers.} \end{cases} \tag{9.6}$$

Hybrid Intelligent Algorithm

In order to solve the stochastic models, we may employ the hybrid intelligent algorithm documented in Chapter 4 provided that the representation structure, initialization, crossover and mutation operations are revised as follows.

We represent an operational plan by the chromosome $V = (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t})$, where the genes $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t}$ are the same as the decision vectors. Without loss of generality, we also assume that the time window at the depot is $[a, b]$. This means that the gene $\boldsymbol{t}$ will be restricted in the hypercube $[a, b]^m$.

Let us show how to initialize a chromosome randomly. For gene $\boldsymbol{x}$, we define a sequence $\{x_1, x_2, \cdots, x_n\}$ with $x_i = i$, $i = 1, 2, \cdots, n$, and repeat the following process from $j = 1$ to n: generating a random position n' between j and n, and exchanging the values of x_j and $x_{n'}$. It is clear that $\{x_1, x_2, \cdots, x_n\}$ is just a random rearrangement of $\{1, 2, \cdots, n\}$. Then we obtain a gene $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$. For each i with $1 \le i \le m-1$, we set y_i as a random integer between 0 and n. Then we rearrange the sequence $\{y_1, y_2, \cdots, y_{m-1}\}$ from small to large. We thus have a gene $\boldsymbol{y} = (y_1, y_2, \cdots, y_{m-1})$. Finally, for each i with $1 \le i \le m$, we set t_i as a random number on the time window $[a, b]$. Then we get a gene $\boldsymbol{t} = (t_1, t_2, \cdots, t_m)$. If the generated chromosome $V = (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t})$ is proven to be feasible, then it is accepted as a chromosome; otherwise we repeat the above process until a feasible chromosome is obtained.

Let us illustrate the crossover operator on the pair V_1 and V_2. We denote $V_1 = (\boldsymbol{x}_1, \boldsymbol{y}_1, \boldsymbol{t}_1)$ and $V_2 = (\boldsymbol{x}_2, \boldsymbol{y}_2, \boldsymbol{t}_2)$. First, we generate a random number c from the open interval $(0, 1)$ and define

$$\boldsymbol{t}_1' = c \cdot \boldsymbol{t}_1 + (1 - c) \cdot \boldsymbol{t}_2, \quad \boldsymbol{t}_2' = (1 - c) \cdot \boldsymbol{t}_1 + c \cdot \boldsymbol{t}_2.$$

The two children V_1' and V_2' are produced by the crossover operation as follows: $V_1' = (\boldsymbol{x}_1, \boldsymbol{y}_2, \boldsymbol{t}_1')$ and $V_2' = (\boldsymbol{x}_2, \boldsymbol{y}_1, \boldsymbol{t}_2')$.

We mutate the chromosome $V = (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t})$ in the following way. For the gene $\boldsymbol{x}$, we randomly generate two mutation positions n_1 and n_2 between 1

and n, and rearrange the sequence $\{x_{n_1}, x_{n_1+1}, \cdots, x_{n_2}\}$ at random to form a new sequence $\{x'_{n_1}, x'_{n_1+1}, \cdots, x'_{n_2}\}$. We thus obtain a new gene

$$\boldsymbol{x}' = (x_1, \cdots, x_{n_1-1}, x'_{n_1}, x'_{n_1+1}, \cdots, x'_{n_2}, x_{n_2+1}, \cdots, x_n).$$

Similarly, for gene $\boldsymbol{y}$, we generate two random mutation positions n_1 and n_2 between 1 and $m-1$, and set y_i as a random integer number y'_i between 0 and n for $i = n_1, n_1+1, \cdots, n_2$. We then rearrange the sequence

$$\{y_1, \cdots, y_{n_1-1}, y'_{n_1}, y'_{n_1+1}, \cdots, y'_{n_2}, y_{n_2+1}, \cdots, y_{m-1}\}$$

from small to large and obtain a new gene $\boldsymbol{y}'$. For the gene $\boldsymbol{t}$, we choose a mutation direction $\boldsymbol{d}$ in $\Re^m$ randomly. If $\boldsymbol{t} + M \cdot \boldsymbol{d}$ is not in the time window $[a, b]^m$, then we set M as a random number between 0 and M until it is in $[a, b]^m$, where M is a predetermined step length. If the above process cannot yield a gene $\boldsymbol{t}$ in $[a, b]^m$ in a predetermined number of iterations, then we set $M = 0$. We replace the parent gene $\boldsymbol{t}$ with its child $\boldsymbol{t}' = \boldsymbol{t} + M \cdot \boldsymbol{d}$.

Example 9.1: We assume that there are 8 customers labeled "$1, 2, \cdots, 8$" in a company and one depot labeled "0". We assume that the travel distances among the depot and customers are listed in Table 9.1.

Table 9.1: Travel Distance Matrix

LCTs	0	1	2	3	4	5	6	7
1	18							
2	14	20						
3	14	34	15					
4	21	55	41	28				
5	17	49	43	36	21			
6	21	57	55	51	36	16		
7	18	49	52	51	43	22	13	
8	14	22	35	44	55	41	43	32

The travel times among the depot and customers are all normally distributed variables $\mathcal{N}(\mu, \sigma^2)$, which are given in Table 9.2.

The time windows of customers are shown in Table 9.3.

We suppose that the unloading times $(S_i, i = 1, 2, \cdots, 8)$ at locations are all 15 minutes.

We assign a confidence level $\alpha = 0.80$ at which all customers are visited within their time windows. If we want to minimize the total travel distance of all vehicles subject to the chance constraint, then we have a stochastic distance minimization model. A run of the hybrid intelligent algorithm (10000 cycles in simulation, 5000 generations in GA) shows that the best operational plan is:

Table 9.2: Random Travel Time Matrix (μ, σ^2)

LCTs	0	1	2	3
1	(50,25)			
2	(10,5)	(40,20)		
3	(50,25)	(10,5)	(40,20)	
4	(50,25)	(35,17)	(35,17)	(30,15)
5	(50,25)	(15,7)	(40,20)	(5,2)
6	(15,7)	(40,20)	(10,5)	(45,22)
7	(50,25)	(15,7)	(45,22)	(10,5)
8	(50,25)	(10,5)	(35,17)	(30,15)
LCTs	4	5	6	7
5	(30,15)			
6	(35,17)	(40,20)		
7	(30,15)	(10,5)	(40,20)	
8	(10,5)	(30,15)	(35,17)	(35,17)

Table 9.3: Time Windows of Customers

i	$[a_i, b_i]$	i	$[a_i, b_i]$	i	$[a_i, b_i]$
1	[09 : 30, 14 : 10]	2	[09 : 20, 11 : 00]	3	[09 : 40, 11 : 10]
4	[09 : 20, 13 : 00]	5	[09 : 10, 15 : 20]	6	[08 : 20, 10 : 00]
7	[09 : 40, 12 : 10]	8	[09 : 20, 10 : 00]		

Vehicle 1: depot$\rightarrow 6 \rightarrow 7 \rightarrow$depot, starting time = 8:45;

Vehicle 2: depot$\rightarrow 3 \rightarrow$depot, starting time = 9:17;

Vehicle 3: depot$\rightarrow 8 \rightarrow 1 \rightarrow 2 \rightarrow 5 \rightarrow 4 \rightarrow$depot, starting time = 8:35.

The total travel distance of the three vehicles is 221. Furthermore, when the obtained operational plan is performed, we have

$$\Pr\{a_i \leq f_i(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{t}^*) \leq b_i, i = 1, 2, \cdots, 8\} = 0.85.$$

Probability Maximization Model

If we hope that total travel distance does not exceed a fixed number $\overline{g}$, then we have a distance constraint $g(\boldsymbol{x}, \boldsymbol{y}) \leq \overline{g}$. If we want to maximize the probability that all customers are visited within their time windows subject to the distance constraint, then we have the following probability maximization

model,

$$\begin{cases} \max \Pr\{a_i \le f_i(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t}) \le b_i, i = 1, 2, \cdots, n\} \\ \text{subject to:} \\ \quad g(\boldsymbol{x}, \boldsymbol{y}) \le \overline{g} \\ \quad 1 \le x_i \le n, \quad i = 1, 2, \cdots, n \\ \quad x_i \ne x_j, \quad i \ne j, \ i, j = 1, 2, \cdots, n \\ \quad 0 \le y_1 \le y_2 \le \cdots \le y_{m-1} \le n \\ \quad x_i, y_j, \quad i = 1, 2, \cdots, n, \quad j = 1, 2, \cdots, m-1, \quad \text{integers.} \end{cases} \tag{9.7}$$

Example 9.2: We set $\overline{g} = 240$. A run of the hybrid intelligent algorithm (10000 cycles in simulation, 5000 generations in GA) shows that the best operational plan is:

Vehicle 1: depot$\to 1 \to 3 \to 2 \to$depot, starting time = 8:51;

Vehicle 2: depot$\to 6 \to 5 \to 7 \to 4 \to$depot, starting time = 9:04;

Vehicle 3: depot$\to 8 \to$depot, starting time = 8:58.

When the obtained operational plan is performed, the total travel distance is 232, and

$$\Pr\{a_i \le f_i(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{t}^*) \le b_i, i = 1, 2, \cdots, 8\} = 0.88.$$

9.3 Fuzzy Models

Here we assume that the travel times are fuzzy variables instead of stochastic variables. Since the travel times are fuzzy variables, every customer will be visited at a fuzzy time.

Fuzzy Distance Minimization Model

If we hope that all customers are visited within their time windows with a confidence level α, then we have the following chance constraint,

$$\mathrm{Cr}\{a_i \le f_i(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t}) \le b_i, i = 1, 2, \cdots, n\} \ge \alpha. \tag{9.8}$$

If we want to minimize the total distance traveled of all vehicles subject to time window constraints, then we have the following fuzzy distance

minimization model (Zheng and Liu [339]),

$$\begin{cases} \min g(\boldsymbol{x}, \boldsymbol{y}) \\ \text{subject to:} \\ \quad \text{Cr}\left\{a_i \le f_i(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t}) \le b_i, i = 1, 2, \cdots, n\right\} \ge \alpha \\ \quad 1 \le x_i \le n, \quad i = 1, 2, \cdots, n \\ \quad x_i \ne x_j, \quad i \ne j, \ i, j = 1, 2, \cdots, n \\ \quad 0 \le y_1 \le y_2 \le \cdots \le y_{m-1} \le n \\ \quad x_i, y_j, \quad i = 1, 2, \cdots, n, \quad j = 1, 2, \cdots, m-1, \quad \text{integers}. \end{cases}$$

Table 9.4: Fuzzy Travel Time Matrix

LCTs	0	1	2	3
1	(25,50,75)			
2	(5,10,15)	(20,40,60)		
3	(25,50,75)	(5,10,15)	(20,40,60)	
4	(25,50,75)	(17,35,53)	(17,35,53)	(15,30,45)
5	(25,50,75)	(7,15,23)	(20,40,60)	(2,5,8)
6	(7,15,23)	(20,40,60)	(5,10,15)	(22,45,68)
7	(25,50,75)	(7,15,23)	(22,45,68)	(5,10,15)
8	(25,50,75)	(5,10,15)	(17,35,53)	(15,30,45)
LCTs	4	5	6	7
5	(15,30,45)			
6	(17,35,53)	(20,40,60)		
7	(15,30,45)	(5,10,15)	(20,40,60)	
8	(5,10,15)	(15,30,45)	(17,35,53)	(17,35,53)

Example 9.3: Let us consider a fuzzy vehicle routing problem shown in Figure 9.1. We assume that the distance matrix is listed in Table 9.1 and the time windows customers are given in Table 9.3. We also assume that the travel times among the deport and customers are all triangular fuzzy variables as shown in Table 9.4. Finally, we suppose that the unloading times at the 8 locations are all 15 minutes.

If the confidence level α is 0.80, then a run of the hybrid intelligent algorithm (10000 cycles in simulation, 5000 generations in GA) shows that the best operational plan is:

Vehicle 1: depot$\to 6 \to 7 \to 4 \to 5 \to$depot, starting time = 9:44;

Vehicle 2: depot$\to 8 \to$depot, starting time = 8:48;

Vehicle 3: depot$\to 2 \to 3 \to 1 \to$depot, starting time = 9:21.

The total distance travelled by the three vehicles is 224. Furthermore, when

the operational plan is performed, we have

$$\mathrm{Cr}\left\{a_i \le f_i(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{t}^*) \le b_i, i = 1, 2, \cdots, 8\right\} = 0.87.$$

Credibility Maximization Model

If we hope that total travel distance does not exceed a fixed number $\overline{g}$, then we have a distance constraint $g(\boldsymbol{x}, \boldsymbol{y}) \le \overline{g}$. If we want to maximize the credibility that all customers are visited within their time windows subject to the distance constraint, then we have the following credibility maximization model,

$$\begin{cases} \max \mathrm{Cr}\left\{a_i \le f_i(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t}) \le b_i, i = 1, 2, \cdots, n\right\} \\ \text{subject to:} \\ \quad g(\boldsymbol{x}, \boldsymbol{y}) \le \overline{g} \\ \quad 1 \le x_i \le n, \quad i = 1, 2, \cdots, n \\ \quad x_i \ne x_j, \quad i \ne j, \ i, j = 1, 2, \cdots, n \\ \quad 0 \le y_1 \le y_2 \le \cdots \le y_{m-1} \le n \\ \quad x_i, y_j, \quad i = 1, 2, \cdots, n, \quad j = 1, 2, \cdots, m-1, \quad \text{integers.} \end{cases} \tag{9.9}$$

Example 9.4: We set $\overline{g} = 240$. A run of the hybrid intelligent algorithm (10000 cycles in simulation, 5000 generations in GA) shows that the best operational plan is

Vehicle 1: depot$\to 8 \to 1 \to 3 \to$depot, starting time = 8:55;
Vehicle 2: depot$\to 6 \to 5 \to 4 \to 7 \to$depot, starting time = 8:51;
Vehicle 3: depot$\to 2 \to$depot, starting time = 9:21.

When the operational plan is performed, the total travel distance is 231, and

$$\mathrm{Cr}\left\{a_i \le f_i(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{t}^*) \le b_i, i = 1, 2, \cdots, 8\right\} = 0.96.$$

9.4 Hybrid Models

Now we suppose that the the travel times are hybrid variables, and introduce hybrid distance minimization model and chance maximization model.

Hybrid Distance Minimization Model

If we hope that all customers are visited within their time windows with confidence level α, then we have the following chance constraint,

$$\mathrm{Ch}\left\{a_i \le f_i(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t}) \le b_i, i = 1, 2, \cdots, n\right\} \ge \alpha. \tag{9.10}$$

If we want to minimize the total travel distance of all vehicles subject to the time window constraint, then we have the following hybrid distance minimization model,

$$\begin{cases} \min g(\boldsymbol{x}, \boldsymbol{y}) \\ \text{subject to:} \\ \quad \text{Ch}\left\{a_i \le f_i(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t}) \le b_i, i = 1, 2, \cdots, n\right\} \ge \alpha \\ \quad 1 \le x_i \le n, \quad i = 1, 2, \cdots, n \\ \quad x_i \ne x_j, \quad i \ne j, \ i, j = 1, 2, \cdots, n \\ \quad 0 \le y_1 \le y_2 \le \cdots \le y_{m-1} \le n \\ \quad x_i, y_j, \quad i = 1, 2, \cdots, n, \quad j = 1, 2, \cdots, m-1, \quad \text{integers.} \end{cases} \tag{9.11}$$

Chance Maximization Model

If we hope that total travel distance does not exceed a fixed number $\overline{g}$, then we have a distance constraint $g(\boldsymbol{x}, \boldsymbol{y}) \le \overline{g}$. If we want to maximize the chance that all customers are visited within their time windows subject to the distance constraint, then we have the following chance maximization model,

$$\begin{cases} \max \text{Ch}\left\{a_i \le f_i(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t}) \le b_i, i = 1, 2, \cdots, n\right\} \\ \text{subject to:} \\ \quad g(\boldsymbol{x}, \boldsymbol{y}) \le \overline{g} \\ \quad 1 \le x_i \le n, \quad i = 1, 2, \cdots, n \\ \quad x_i \ne x_j, \quad i \ne j, \ i, j = 1, 2, \cdots, n \\ \quad 0 \le y_1 \le y_2 \le \cdots \le y_{m-1} \le n \\ \quad x_i, y_j, \quad i = 1, 2, \cdots, n, \quad j = 1, 2, \cdots, m-1, \quad \text{integers.} \end{cases} \tag{9.12}$$

Problem 9.1. Design a hybrid intelligent algorithm to solve hybrid models for vehicle routing problem.

Chapter 10

Facility Location Problem

Facility location problem is to find locations for new facilities such that the conveying cost from facilities to customers is minimized. Facility location problem has been studied for half a century because of its widely practical application backgrounds.

Facility location problem was first proposed by Cooper [51] in 1963. In practice, some factors such as demands, allocations, even locations of customers and facilities are usually changing. The uncapacitated facility location problem with stochastic demand was considered by Logendran and Terrell [207] in 1988, in which the customers are supplied by the nearest factory. However, in a capacitated problem, the customers may not be supplied by the nearest factory only. In order to solve this type of problem, Zhou and Liu [340] presented three types of stochastic model for capacitated facility location problem with stochastic demands. When the demands are given as fuzzy variables rather than random ones, Zhou and Liu [342] presented three types of fuzzy optimization model.

10.1 Problem Description

In order to model facility location problem, we use the following indices, parameters, and decision variables:

$i = 1, 2, \cdots, n$: facilities;

$j = 1, 2, \cdots, m$: customers;

(a_j, b_j): location of customer j, $1 \leq j \leq m$;

ξ_j: uncertain demand of customer j, $1 \leq j \leq m$;

s_i: capacity of facility i, $1 \leq i \leq n$;

(x_i, y_i): decision vector representing the location of facility i, $1 \leq i \leq n$;

z_{ij}: quantity supplied to customer j by facility i after the uncertain demands are realized, $1 \leq i \leq n$, $1 \leq j \leq m$.

10.2 Stochastic Models

We write the demand vector $\boldsymbol{\xi} = (\xi_1, \xi_2, \cdots, \xi_m)$. For convenience, we also write

$$(\boldsymbol{x}, \boldsymbol{y}) = \begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \cdots & \cdots \\ x_n & y_n \end{pmatrix}, \quad \boldsymbol{z} = \begin{pmatrix} z_{11} & z_{12} & \cdots & z_{1m} \\ z_{21} & z_{22} & \cdots & z_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ z_{n1} & z_{n2} & \cdots & z_{nm} \end{pmatrix}. \tag{10.1}$$

For each $\omega \in \Omega$, $\boldsymbol{\xi}(\omega)$ is a realization of uncertain vector $\boldsymbol{\xi}$. An allocation $\boldsymbol{z}$ is said to be feasible with respect to ω if and only if

$$\begin{cases} z_{ij} \geq 0, \ i = 1, 2, \cdots, n, \ j = 1, 2, \cdots, m \\ \sum\limits_{i=1}^{n} z_{ij} = \xi_j(\omega), \ j = 1, 2, \cdots, m \\ \sum\limits_{j=1}^{m} z_{ij} \leq s_i, \ i = 1, 2, \cdots, n. \end{cases} \tag{10.2}$$

We denote the feasible allocation set by

$$Z(\omega) = \left\{ \boldsymbol{z} \;\middle|\; \begin{array}{l} z_{ij} \geq 0, \ i = 1, 2, \cdots, n, \ j = 1, 2, \cdots, m \\ \sum\limits_{i=1}^{n} z_{ij} = \xi_j(\omega), \ j = 1, 2, \cdots, m \\ \sum\limits_{j=1}^{m} z_{ij} \leq s_i, \ i = 1, 2, \cdots, n \end{array} \right\}. \tag{10.3}$$

Note that $Z(\omega)$ may be an empty set for some ω.

For each $\omega \in \Omega$, the minimal cost is the one associated with the best allocation $\boldsymbol{z}$, i.e.,

$$C(\boldsymbol{x}, \boldsymbol{y}|\omega) = \min_{\boldsymbol{z} \in Z(\omega)} \sum_{i=1}^{n} \sum_{j=1}^{m} z_{ij} \sqrt{(x_i - a_j)^2 + (y_i - b_j)^2} \tag{10.4}$$

whose optimal solution $\boldsymbol{z}^*$ is called the optimal allocation. If $Z(\omega) = \emptyset$, then the demands of some customers are impossible to be met. As a penalty, we define

$$C(\boldsymbol{x}, \boldsymbol{y}|\omega) = \sum_{j=1}^{m} \max_{1 \leq i \leq n} \xi_j(\omega) \sqrt{(x_i - a_j)^2 + (y_i - b_j)^2}. \tag{10.5}$$

Stochastic Expected Cost Minimization Model

Since the demands are stochastic variables, the conveying cost $C(\boldsymbol{x}, \boldsymbol{y}|\omega)$ is also a stochastic variable. In order to evaluate the location design, we use its expected cost

$$E[C(\boldsymbol{x}, \boldsymbol{y}|\omega)] = \int_0^{\infty} \Pr\left\{\omega \in \boldsymbol{\Omega} \;\middle|\; C(\boldsymbol{x}, \boldsymbol{y}|\omega) \geq r\right\} \mathrm{d}r. \tag{10.6}$$

In order to minimize the expected cost, Zhou and Liu [340] presented the following expected cost minimization model for stochastic facility location problem,

$$\begin{cases} \min\limits_{\boldsymbol{x},\boldsymbol{y}} \displaystyle\int_0^{\infty} \Pr\left\{\omega \in \boldsymbol{\Omega} \mid C(\boldsymbol{x},\boldsymbol{y}|\omega) \geq r\right\} \mathrm{d}r \\ \text{subject to:} \\ \quad g_j(\boldsymbol{x},\boldsymbol{y}) \leq 0,\ j = 1,2,\cdots,p \end{cases} \tag{10.7}$$

where $g_j(\boldsymbol{x},\boldsymbol{y}) \leq 0,\ j = 1,2,\cdots,p$ represent the potential region of locations of new facilities and $C(\boldsymbol{x},\boldsymbol{y}|\omega)$ is defined by equation (10.4).

This model is different from traditional stochastic programming models because there is a sub-problem in it, i.e.,

$$\begin{cases} \min \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{m} z_{ij}\sqrt{(x_i-a_j)^2+(y_i-b_j)^2} \\ \text{subject to:} \\ \quad \displaystyle\sum_{i=1}^{n} z_{ij} = \xi_j(\omega),\ \ j = 1,2,\cdots,m \\ \quad \displaystyle\sum_{j=1}^{m} z_{ij} \leq s_i,\ \ i = 1,2,\cdots,n \\ \quad z_{ij} \geq 0,\ \ i = 1,2,\cdots,n,\ \ j = 1,2,\cdots,m. \end{cases} \tag{10.8}$$

Note that in (10.8) the parameters x_i, y_i and $\xi_j(\omega)$ are fixed real numbers for $i = 1,2,\cdots,n$, $j = 1,2,\cdots,m$. It is clearly a linear programming which may be solved by the simplex algorithm.

Example 10.1: Assume that there are 3 new facilities whose capacities are $(s_1, s_2, s_3) = (70, 80, 90)$, and 8 customers whose demands are uniformly distributed variables. The locations (a_j, b_j) and demands $\mathcal{U}(l_i, u_i),\ j = 1,2,\cdots,8$ of customers are given in Table 10.1.

Table 10.1: Locations and Random Demands of Customers

j	$(a_j,\ b_j)$	ξ_j	j	$(a_j,\ b_j)$	ξ_j
1	(28, 42)	(14,17)	5	(70, 18)	(21,26)
2	(18, 50)	(13,18)	6	(72, 98)	(24,28)
3	(74, 34)	(12,16)	7	(60, 50)	(13,16)
4	(74, 6)	(17,20)	8	(36, 40)	(12,17)

A run of the hybrid intelligent algorithm (5000 cycles in simulation, 300 generations in GA) shows that the optimal locations of the 3 facilities are

$$\begin{pmatrix} x_1^*, y_1^* \\ x_2^*, y_2^* \\ x_3^*, y_3^* \end{pmatrix} = \begin{pmatrix} 29.92, 43.19 \\ 70.04, 17.99 \\ 72.02, 98.02 \end{pmatrix}$$

whose expected conveying cost is 1259.

Stochastic α-Cost Minimization Model

Now we define the α-cost as the minimum number $\overline{C}$ such that

$$\Pr\{C(\boldsymbol{x}, \boldsymbol{y}|\omega) \leq \overline{C}\} \geq \alpha. \tag{10.9}$$

If we want to minimize the α-cost, then we have the following stochastic α-cost minimization model (Zhou and Liu [340]),

$$\begin{cases} \min\limits_{\boldsymbol{x},\boldsymbol{y}} \overline{C} \\ \text{subject to:} \\ \quad \Pr\left\{\omega \in \Omega \;\middle|\; C(\boldsymbol{x}, \boldsymbol{y}|\omega) \leq \overline{C}\right\} \geq \alpha \\ \quad g_j(\boldsymbol{x}, \boldsymbol{y}) \leq 0, \;\; j = 1, 2, \cdots, p \end{cases} \tag{10.10}$$

where $\overline{f}$ is the α-cost and $C(\boldsymbol{x}, \boldsymbol{y}|\omega)$ is defined by equation (10.4).

Example 10.2: Here we suppose that the 0.9-cost is to be minimized. A run of the hybrid intelligent algorithm (5000 cycles in simulations and 300 generations in GA) shows that the optimal locations of the 3 facilities are

$$\begin{pmatrix} x_1^*, y_1^* \\ x_2^*, y_2^* \\ x_3^*, y_3^* \end{pmatrix} = \begin{pmatrix} 31.00, 43.08 \\ 70.04, 17.92 \\ 71.98, 97.99 \end{pmatrix}$$

whose 0.9-cost is 1313.

Probability Maximization Model

If we hope to maximize the probability that the conveying cost will not exceed a given level C^0, then we have a probability maximization model (Zhou and Liu [340]),

$$\begin{cases} \max\limits_{\boldsymbol{x},\boldsymbol{y}} \Pr\left\{\omega \in \Omega \;\middle|\; C(\boldsymbol{x}, \boldsymbol{y}|\omega) \leq C^0\right\} \\ \text{subject to:} \\ \quad g_j(\boldsymbol{x}, \boldsymbol{y}) \leq 0, \;\; j = 1, 2, \cdots, p \end{cases} \tag{10.11}$$

where $C(\boldsymbol{x}, \boldsymbol{y}|\omega)$ is defined by equation (10.4).

Example 10.3: Now we want to maximize the probability that the transportation cost does not exceed 1300. A run of the hybrid intelligent algorithm (5000 cycles in simulation, and 300 generations in GA) shows that the optimal locations of the 3 facilities are

$$\begin{pmatrix} x_1^*, y_1^* \\ x_2^*, y_2^* \\ x_3^*, y_3^* \end{pmatrix} = \begin{pmatrix} 30.21, 42.75 \\ 70.47, 17.94 \\ 72.03, 97.98 \end{pmatrix}$$

whose probability is 0.86.

10.3 Fuzzy Models

We have discussed facility location problem with stochastic demands. However, in many cases, the probability distributions are not easy to obtain due to the lack of data. Instead, expert knowledge is used to estimate the demands. Thus we have facility location problem with fuzzy demands.

Fuzzy Expected Cost Minimization Model

In this section, we suppose that ξ_j are fuzzy demands of customers j defined on the credibility space $(\Theta, \mathcal{P}(\Theta), \mathrm{Cr})$, $j = 1, 2, \cdots, m$, respectively, and denote $\boldsymbol{\xi} = (\xi_1, \xi_2, \cdots, \xi_n)$.

For each $\theta \in \Theta$, $\boldsymbol{\xi}(\theta)$ is a realization of fuzzy vector $\boldsymbol{\xi}$. We denote the feasible allocation set by

$$Z(\theta) = \left\{ \boldsymbol{z} \;\middle|\; \begin{array}{l} z_{ij} \geq 0,\ i = 1, 2, \cdots, n,\ j = 1, 2, \cdots, m \\ \sum\limits_{i=1}^{n} z_{ij} = \xi_j(\theta),\ j = 1, 2, \cdots, m \\ \sum\limits_{j=1}^{m} z_{ij} \leq s_i,\ i = 1, 2, \cdots, n \end{array} \right\}. \tag{10.12}$$

Note that $Z(\theta)$ may be an empty set for some θ.

For each $\theta \in \Theta$, the minimal conveying cost from facilities to customers is

$$C(\boldsymbol{x}, \boldsymbol{y}|\theta) = \min_{\boldsymbol{z} \in Z(\theta)} \sum_{i=1}^{n} \sum_{j=1}^{m} z_{ij} \sqrt{(x_i - a_j)^2 + (y_i - b_j)^2} \tag{10.13}$$

whose optimal solution $\boldsymbol{z}^*$ is called the optimal allocation. If $Z(\theta) = \emptyset$, then the demands of some customers are impossible to be met. As a penalty, we define

$$C(\boldsymbol{x}, \boldsymbol{y}|\theta) = \sum_{j=1}^{m} \max_{1 \leq i \leq n} \xi_j(\theta) \sqrt{(x_i - a_j)^2 + (y_i - b_j)^2}. \tag{10.14}$$

Note that the conveying cost $C(\boldsymbol{x}, \boldsymbol{y}|\theta)$ is a fuzzy variable. In order to evaluate the location design, we use its expected cost

$$E[C(\boldsymbol{x}, \boldsymbol{y}|\theta)] = \int_0^{\infty} \mathrm{Cr}\left\{\theta \in \boldsymbol{\Theta} \;\middle|\; C(\boldsymbol{x}, \boldsymbol{y}|\theta) \geq r\right\} \mathrm{d}r. \tag{10.15}$$

In order to minimize the expected cost, Zhou and Liu [342] presented the following expected cost minimization model for fuzzy capacitated facility location problem,

$$\left\{ \begin{array}{l} \min\limits_{\boldsymbol{x}, \boldsymbol{y}} \displaystyle\int_0^{\infty} \mathrm{Cr}\left\{\theta \in \Theta | C(\boldsymbol{x}, \boldsymbol{y}|\theta) \geq r\right\} \mathrm{d}r \\ \text{subject to:} \\ \quad g_j(\boldsymbol{x}, \boldsymbol{y}) \leq 0,\ j = 1, 2, \cdots, p \end{array} \right. \tag{10.16}$$

where $g_j(\boldsymbol{x}, \boldsymbol{y}) \leq 0$, $j = 1, 2, \cdots, p$ represent the potential region of locations of new facilities and $C(\boldsymbol{x}, \boldsymbol{y}|\theta)$ is defined by equation (10.13).

Example 10.4: Now we assume that there are 8 customers whose locations and trapezoidal fuzzy demands are given in Table 10.2, and 3 facilities with capacities 70, 80 and 90.

Table 10.2: Locations and Fuzzy Demands of Customers

j	$(a_j,\ b_j)$	ξ_j	j	$(a_j,\ b_j)$	ξ_j
1	(28, 42)	(14,15,16,17)	5	(70, 18)	(21,23,24,26)
2	(18, 50)	(13,14,16,18)	6	(72, 98)	(24,25,26,28)
3	(74, 34)	(12,14,15,16)	7	(60, 50)	(13,14,15,16)
4	(74, 6)	(17,18,19,20)	8	(36, 40)	(12,14,16,17)

A run of the hybrid intelligent algorithm (10000 cycles in fuzzy simulation, 1000 generations in GA) shows that the optimal locations of the 3 facilities are

$$\begin{pmatrix} x_1^*, y_1^* \\ x_2^*, y_2^* \\ x_3^*, y_3^* \end{pmatrix} = \begin{pmatrix} 30.34, 42.50 \\ 70.14, 17.97 \\ 71.99, 98.04 \end{pmatrix}$$

whose expected conveying cost is 1255.

Fuzzy α-Cost Minimization Model

Now we define the α-cost as the minimum number $\overline{f}$ such that

$$\mathrm{Cr}\{C(\boldsymbol{x}, \boldsymbol{y}|\theta) \leq \overline{f}\} \geq \alpha.$$

If we want to minimize the α-cost, then we have a fuzzy α-cost minimization model (Zhou and Liu [342]),

$$\begin{cases} \min\limits_{\boldsymbol{x}, \boldsymbol{y}} \overline{f} \\ \text{subject to:} \\ \quad \mathrm{Cr}\left\{\theta \in \Theta \mid C(\boldsymbol{x}, \boldsymbol{y}|\theta) \leq \overline{f}\right\} \geq \alpha \\ \quad g_j(\boldsymbol{x}, \boldsymbol{y}) \leq 0,\ j = 1, 2, \cdots, p \end{cases} \tag{10.17}$$

where $\overline{f}$ is the α-cost and $C(\boldsymbol{x}, \boldsymbol{y}|\theta)$ is defined by equation (10.13).

Example 10.5: Let us minimize the 0.9-cost. A run of the hybrid intelligent algorithm (10000 cycles in fuzzy simulation, 1000 generations in GA) shows that the optimal locations of the 3 facilities are

$$\begin{pmatrix} x_1^*, y_1^* \\ x_2^*, y_2^* \\ x_3^*, y_3^* \end{pmatrix} = \begin{pmatrix} 30.39, 43.49 \\ 70.10, 17.64 \\ 71.98, 98.03 \end{pmatrix}$$

whose 0.9-cost is 1354.

Credibility Maximization Model

If we hope to maximize the credibility that the conveying cost does not exceed a given level C^0, then we have a credibility maximization model (Zhou and Liu [342]),

$$\begin{cases} \max\limits_{\boldsymbol{x},\boldsymbol{y}} \ \mathrm{Cr}\left\{\theta \in \Theta \ \middle| \ C(\boldsymbol{x},\boldsymbol{y}|\theta) \le C^0\right\} \\ \text{subject to:} \\ \quad g_j(\boldsymbol{x},\boldsymbol{y}) \le 0, \ \ j=1,2,\cdots,p \end{cases} \tag{10.18}$$

where $C(\boldsymbol{x},\boldsymbol{y}|\theta)$ is defined by equation (10.13).

Example 10.6: Assume $C^0 = 1350$. A run of the hybrid intelligent algorithm (10000 cycles in fuzzy simulation, 1000 generations in GA) shows that the optimal locations of the 3 facilities are

$$\begin{pmatrix} x_1^*, y_1^* \\ x_2^*, y_2^* \\ x_3^*, y_3^* \end{pmatrix} = \begin{pmatrix} 32.18, 42.04 \\ 70.15, 17.81 \\ 72.04, 98.09 \end{pmatrix}$$

whose credibility is 0.87.

10.4 Hybrid Models

In this section, we suppose that the demands of customers are hybrid variables defined on $(\Theta, \mathcal{P}(\Theta), \mathrm{Cr}) \times (\Omega, \mathcal{A}, \Pr)$.

For each $(\theta,\omega) \in \Theta \times \Omega$, the value $\boldsymbol{\xi}(\theta,\omega)$ is a realization of hybrid vector $\boldsymbol{\xi}$. We denote the feasible allocation set by

$$Z(\theta,\omega) = \left\{ \boldsymbol{z} \ \middle| \ \begin{array}{l} z_{ij} \ge 0, \ i=1,2,\cdots,n, \ j=1,2,\cdots,m \\ \sum\limits_{i=1}^{n} z_{ij} = \xi_j(\theta,\omega), \ j=1,2,\cdots,m \\ \sum\limits_{j=1}^{m} z_{ij} \le s_i, \ i=1,2,\cdots,n \end{array} \right\}. \tag{10.19}$$

Note that $Z(\theta,\omega)$ may be an empty set for some (θ,ω).

For each $(\theta,\omega) \in \Theta \times \Omega$, the minimal conveying cost from facilities to customers is

$$C(\boldsymbol{x},\boldsymbol{y}|\theta,\omega) = \min_{\boldsymbol{z} \in Z(\theta,\omega)} \sum_{i=1}^{n}\sum_{j=1}^{m} z_{ij}\sqrt{(x_i-a_j)^2+(y_i-b_j)^2} \tag{10.20}$$

whose optimal solution $\boldsymbol{z}^*$ is called the optimal allocation. If $Z(\theta,\omega) = \emptyset$, then the demands of some customers are impossible to be met. As a penalty, we define

$$C(\boldsymbol{x},\boldsymbol{y}|\theta,\omega) = \sum_{j=1}^{m} \max_{1\le i\le n} \xi_j(\theta,\omega)\sqrt{(x_i-a_j)^2+(y_i-b_j)^2}. \tag{10.21}$$

Note that the conveying cost $C(\boldsymbol{x}, \boldsymbol{y}|\theta, \omega)$ is a hybrid variable. In order to evaluate the location design, we use its expected cost

$$E[C(\boldsymbol{x}, \boldsymbol{y}|\theta, \omega)] = \int_0^\infty \mathrm{Ch}\left\{(\theta, \omega) \in \Theta \times \Omega \in \boldsymbol{\Theta} \;\middle|\; E[C(\boldsymbol{x}, \boldsymbol{y}|\theta, \omega)] \geq r\right\} \mathrm{d}r.$$

Hybrid Expected Cost Minimization Model

In order to minimize the expected cost, we have the following expected cost minimization model for hybrid capacitated facility location problem,

$$\begin{cases} \min\limits_{\boldsymbol{x}, \boldsymbol{y}} \displaystyle\int_0^\infty \mathrm{Ch}\left\{(\theta, \omega) \in \Theta \times \Omega \in \Theta | E[C(\boldsymbol{x}, \boldsymbol{y}|\theta, \omega)] \geq r\right\} \mathrm{d}r \\ \text{subject to:} \\ \quad g_j(\boldsymbol{x}, \boldsymbol{y}) \leq 0, \;\; j = 1, 2, \cdots, p \end{cases} \tag{10.22}$$

where $g_j(\boldsymbol{x}, \boldsymbol{y}) \leq 0$, $j = 1, 2, \cdots, p$ represent the potential region of locations of new facilities and $C(\boldsymbol{x}, \boldsymbol{y}|\theta, \omega)$ is defined by equation (10.20).

Hybrid α-Cost Minimization Model

Now we define the α-cost as the minimum number $\overline{f}$ such that

$$\mathrm{Ch}\{C(\boldsymbol{x}, \boldsymbol{y}|\theta, \omega) \leq \overline{f}\} \geq \alpha.$$

If we want to minimize the α-cost, then we have a hybrid α-cost minimization model,

$$\begin{cases} \min\limits_{\boldsymbol{x}, \boldsymbol{y}} \overline{f} \\ \text{subject to:} \\ \quad \mathrm{Ch}\left\{(\theta, \omega) \in \Theta \times \Omega \;\middle|\; C(\boldsymbol{x}, \boldsymbol{y}|\theta, \omega) \leq \overline{f}\right\} \geq \alpha \\ \quad g_j(\boldsymbol{x}, \boldsymbol{y}) \leq 0, \;\; j = 1, 2, \cdots, p \end{cases} \tag{10.23}$$

where $\overline{f}$ is the α-cost and $C(\boldsymbol{x}, \boldsymbol{y}|\theta, \omega)$ is defined by equation (10.20).

Chance Maximization Model

If we hope to maximize the chance that the conveying cost does not exceed a given level C^0, then we have a chance maximization model,

$$\begin{cases} \max\limits_{\boldsymbol{x}, \boldsymbol{y}} \mathrm{Ch}\left\{(\theta, \omega) \in \Theta \times \Omega \;\middle|\; C(\boldsymbol{x}, \boldsymbol{y}|\theta, \omega) \leq C^0\right\} \\ \text{subject to:} \\ \quad g_j(\boldsymbol{x}, \boldsymbol{y}) \leq 0, \;\; j = 1, 2, \cdots, p \end{cases} \tag{10.24}$$

where $C(\boldsymbol{x}, \boldsymbol{y}|\theta, \omega)$ is defined by equation (10.20).

Problem 10.1. Design a hybrid intelligent algorithm to solve hybrid models for facility location problem.

Chapter 11

Machine Scheduling Problem

Machine scheduling problem is concerned with finding an efficient schedule during an uninterrupted period of time for a set of machines to process a set of jobs. Much of research work has been done on this type of problem during the past five decades.

11.1 Problem Description

In a machine scheduling problem, we assume that (a) each job can be processed on any machine without interruption; and (b) each machine can process only one job at a time.

Let us first introduce the following indices and parameters.

$i = 1, 2, \cdots, n$: jobs;

$k = 1, 2, \cdots, m$: machines;

ξ_{ik}: uncertain processing time of job i on machine k;

D_i: the due date of job i, $i = 1, 2, \cdots, n$.

The schedule is represented by Liu's formulation [182] via two decision vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, where

$\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$: integer decision vector representing n jobs with $1 \leq x_i \leq n$ and $x_i \neq x_j$ for all $i \neq j$, $i, j = 1, 2, \cdots, n$. That is, the sequence $\{x_1, x_2, \cdots, x_n\}$ is a rearrangement of $\{1, 2, \cdots, n\}$;

$\boldsymbol{y} = (y_1, y_2, \cdots, y_{m-1})$: integer decision vector with $y_0 \equiv 0 \leq y_1 \leq y_2 \leq \cdots \leq y_{m-1} \leq n \equiv y_m$.

We note that the schedule is fully determined by the decision vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ in the following way. For each k ($1 \leq k \leq m$), if $y_k = y_{k-1}$, then machine k is not used; if $y_k > y_{k-1}$, then machine k is used and processes jobs $x_{y_{k-1}+1}, x_{y_{k-1}+2}, \cdots, x_{y_k}$ in turn. Thus the schedule of all machines is

as follows:

$$\begin{aligned} &\text{Machine 1: } x_{y_0+1} \to x_{y_0+2} \to \cdots \to x_{y_1}; \\ &\text{Machine 2: } x_{y_1+1} \to x_{y_1+2} \to \cdots \to x_{y_2}; \\ &\quad \cdots \\ &\text{Machine } m\text{: } x_{y_{m-1}+1} \to x_{y_{m-1}+2} \to \cdots \to x_{y_m}. \end{aligned} \tag{11.1}$$

Let $C_i(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})$ be the completion times of jobs i, $i = 1, 2, \cdots, n$, respectively. They can be calculated by the following equations,

$$C_{x_{y_{k-1}+1}}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) = \xi_{x_{y_{k-1}+1}k} \tag{11.2}$$

and

$$C_{x_{y_{k-1}+j}}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) = C_{x_{y_{k-1}+j-1}}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) + \xi_{x_{y_{k-1}+j}k} \tag{11.3}$$

for $2 \le j \le y_k - y_{k-1}$ and $k = 1, 2, \cdots, m$.

We denote the tardiness and makespan of the schedule $(\boldsymbol{x}, \boldsymbol{y})$ by $f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})$ and $f_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})$, respectively. Then we have

$$f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) = \max_{1 \le i \le n} \{C_i(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) - D_i\} \vee 0, \tag{11.4}$$

$$f_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) = \max_{1 \le k \le m} C_{x_{y_k}}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}). \tag{11.5}$$

11.2 Stochastic Models

In this section, we introduce an expected time goal programming model for parallel machine scheduling problems proposed by Peng and Liu [253].

Stochastic Expected Time Goal Programming

In order to balance the above conflicting objectives, we may have the following target levels and priority structure.

At the first priority level, the expected tardiness $E[f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})]$ should not exceed the target value b_1. Thus we have a goal constraint

$$E[f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})] - b_1 = d_1^+$$

in which $d_1^+ \vee 0$ will be minimized.

At the second priority level, the expected makespan $E[f_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})]$ should not exceed the target value b_2. That is, we have a goal constraint

$$E[f_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})] - b_2 = d_2^+$$

in which $d_2^+ \vee 0$ will be minimized.

Then according to the priority structure, we have the following stochastic expected time goal programming model:

$$\begin{cases} \text{lexmin}\left\{d_1^+ \vee 0, d_2^+ \vee 0\right\} \\ \text{subject to:} \\ \quad E[f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})] - b_1 = d_1^+ \\ \quad E[f_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})] - b_2 = d_2^+ \\ \quad 1 \leq x_i \leq n, \quad i = 1, 2, \cdots, n \\ \quad x_i \neq x_j, \quad i \neq j, \ i, j = 1, 2, \cdots, n \\ \quad 0 \leq y_1 \leq y_2 \cdots \leq y_{m-1} \leq n \\ \quad x_i, y_j, \quad i = 1, 2, \cdots, n, \quad j = 1, 2, \cdots, m-1, \quad \text{integers.} \end{cases} \tag{11.6}$$

Hybrid Intelligent Algorithm

The hybrid intelligent algorithm documented in Chapter 4 may solve this model provided that the initialization, crossover and mutation operations are revised as follows.

We encode a schedule into a chromosome $V = (\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x}, \boldsymbol{y}$ are the same as the decision vectors. For the gene section $\boldsymbol{x}$, we define a sequence $\{x_1, x_2, \cdots, x_n\}$ with $x_i = i$, $i = 1, 2, \cdots, n$. In order to get a random rearrangement of $\{1, 2, \cdots, n\}$, we repeat the following process from $j = 1$ to n: generating a random position n' between j and n, and exchanging the values of x_j and $x_{n'}$. For each i with $1 \leq i \leq m-1$, we set y_i as a random integer between 0 and n. Then we rearrange the sequence $\{y_1, y_2, \cdots, y_{m-1}\}$ from small to large and thus obtain a gene section $\boldsymbol{y} = (y_1, y_2, \cdots, y_{m-1})$. We can ensure that the produced chromosome $V = (\boldsymbol{x}, \boldsymbol{y})$ is always feasible.

Let us illustrate the crossover operator on the pair V_1 and V_2. We denote $V_1 = (\boldsymbol{x}_1, \boldsymbol{y}_1)$ and $V_2 = (\boldsymbol{x}_2, \boldsymbol{y}_2)$. Two children V_1' and V_2' are produced by the crossover operation as follows: $V_1' = (\boldsymbol{x}_1, \boldsymbol{y}_2)$ and $V_2' = (\boldsymbol{x}_2, \boldsymbol{y}_1)$. Note that the obtained chromosomes $V_1' = (\boldsymbol{x}_1, \boldsymbol{y}_2)$ and $V_2' = (\boldsymbol{x}_2, \boldsymbol{y}_1)$ in this way are always feasible.

We mutate the parent $V = (\boldsymbol{x}, \boldsymbol{y})$ in the following way. For the gene $\boldsymbol{x}$, we randomly generate two mutation positions n_1 and n_2 between 1 and n, and rearrange the sequence $\{x_{n_1}, x_{n_1+1}, \cdots, x_{n_2}\}$ at random to form a new sequence $\{x'_{n_1}, x'_{n_1+1}, \cdots, x'_{n_2}\}$, thus we obtain a new gene section

$$\boldsymbol{x}' = (x_1, \cdots, x_{n_1-1}, x'_{n_1}, x'_{n_1+1}, \cdots, x'_{n_2}, x_{n_2+1}, \cdots, x_n).$$

Similarly, for the gene $\boldsymbol{y}$, we generate two random mutation positions n_1 and n_2 between 1 and $m-1$, and set y_i as a random integer number y'_i between 0 and n for $i = n_1, n_1+1, \cdots, n_2$. We then rearrange the sequence $y_1, \cdots, y_{n_1-1}, y'_{n_1}, y'_{n_1+1}, \cdots, y'_{n_2}, y_{n_2+1}, \cdots, y_{m-1}$ from small to large and obtain a new gene section $\boldsymbol{y}'$. Finally, we replace the parent V with the offspring $V' = (\boldsymbol{x}', \boldsymbol{y}')$.

Example 11.1: Assume that there are 8 jobs and 3 machines. The processing times of jobs on different machines are all uniformly distributed variables. The random processing times and due dates are listed in Table 11.1.

Table 11.1: Random Processing Times and Due Dates

Jobs	Random Processing Times			Due Dates
	Machine 1	Machine 2	Machine 3	
1	$(10, 16)$	$(11, 15)$	$(12, 17)$	30
2	$(10, 18)$	$(10, 16)$	$(12, 18)$	150
3	$(12, 16)$	$(11, 16)$	$(12, 18)$	105
4	$(18, 24)$	$(20, 23)$	$(20, 25)$	130
5	$(10, 15)$	$(12, 17)$	$(10, 15)$	60
6	$(10, 18)$	$(12, 17)$	$(12, 20)$	30
7	$(10, 16)$	$(10, 14)$	$(10, 13)$	75
8	$(15, 20)$	$(14, 20)$	$(12, 16)$	50

We suppose that the target levels of expected tardiness and expected makespan are $b_1 = 0$ and $b_2 = 43$. A run of the hybrid intelligent algorithm (10000 cycles in simulation, 1000 generations in GA) shows that the optimal schedule is

$$\begin{array}{ll} \text{Machine 1:} & 1 \to 5 \to 3; \\ \text{Machine 2:} & 6 \to 7 \to 2; \\ \text{Machine 3:} & 8 \to 4 \end{array}$$

which can satisfy the two goals.

Stochastic Chance-Constrained Goal Programming

We assume the following priority structure. At the first priority level, the tardiness $f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})$ should not exceed the target value b_1 with a confidence level α_1. Thus we have a goal constraint

$$\Pr\left\{f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) - b_1 \leq d_1^+\right\} \geq \alpha_1$$

in which $d_1^+ \vee 0$ will be minimized.

At the second priority level, the makespan $f_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})$ should not exceed the target value b_2 with a confidence level α_2. Thus we have a goal constraint

$$\Pr\left\{f_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) - b_2 \leq d_2^+\right\} \geq \alpha_2$$

in which $d_2^+ \vee 0$ will be minimized.

Then we have the following CCGP model for parallel machine scheduling

problem (Peng and Liu [253]),

$$\begin{cases} \text{lexmin}\{d_1^+ \vee 0, d_2^+ \vee 0\} \\ \text{subject to:} \\ \quad \Pr\left\{f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) - b_1 \leq d_1^+\right\} \geq \alpha_1 \\ \quad \Pr\left\{f_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) - b_2 \leq d_2^+\right\} \geq \alpha_2 \\ \quad 1 \leq x_i \leq n, \quad i = 1, 2, \cdots, n \\ \quad x_i \neq x_j, \quad i \neq j, \ i, j = 1, 2, \cdots, n \\ \quad 0 \leq y_1 \leq y_2 \cdots \leq y_{m-1} \leq n \\ \quad x_i, \, y_j, \quad i = 1, 2, \cdots, n, \quad j = 1, 2, \cdots, m-1, \quad \text{integers.} \end{cases} \tag{11.7}$$

Example 11.2: Suppose that the target levels of tardiness and makespan are $b_1 = 0$ and $b_2 = 43$ with confidence levels $\alpha_1 = 0.95$ and $\alpha_2 = 0.90$. A run of the hybrid intelligent algorithm (10000 cycles in simulation, 1000 generations in GA) shows that the optimal schedule is

$$\begin{array}{ll} \text{Machine 1:} & 5 \to 1 \to 3; \\ \text{Machine 2:} & 7 \to 6 \to 2; \\ \text{Machine 3:} & 8 \to 4 \end{array}$$

which can satisfy the first goal, but the second objective is 0.52.

Stochastic Dependent-Chance Goal Programming

Suppose that the management goals have the following priority structure.

At the first priority level, the probability that the tardiness $f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})$ does not exceed the given value b_1 should achieve a confidence level α_1. Thus we have a goal constraint

$$\alpha_1 - \Pr\left\{f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) \leq b_1\right\} = d_1^-$$

in which $d_1^- \vee 0$ will be minimized.

At the second priority level, the probability that the makespan $f_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})$ does not exceed the given value b_2 should achieve a confidence level α_2. Thus we have a goal constraint

$$\alpha_2 - \Pr\left\{f_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) \leq b_2\right\} = d_2^-$$

in which $d_2^- \vee 0$ will be minimized.

Then we have the following DCGP model formulated by Peng and Liu

[253],

$$\begin{cases} \text{lexmin}\{d_1^- \vee 0, d_2^- \vee 0\} \\ \text{subject to:} \\ \quad \alpha_1 - \Pr\{f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) \leq b_1\} = d_1^- \\ \quad \alpha_2 - \Pr\{f_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) \leq b_2\} = d_2^- \\ \quad 1 \leq x_i \leq n, \quad i = 1, 2, \cdots, n \\ \quad x_i \neq x_j, \quad i \neq j, \ i, j = 1, 2, \cdots, n \\ \quad 0 \leq y_1 \leq y_2 \cdots \leq y_{m-1} \leq n \\ \quad x_i, y_j, \quad i = 1, 2, \cdots, n, \quad j = 1, 2, \cdots, m-1, \quad \text{integers.} \end{cases} \tag{11.8}$$

Example 11.3: Suppose that the upper bounds of tardiness and makespan are $b_1 = 0$ and $b_2 = 43$, and the target levels are $\alpha_1 = 0.95$ and $\alpha_2 = 0.90$. A run of the hybrid intelligent algorithm (10000 cycles in simulation, 1000 generations in GA) shows that the optimal schedule is

$$\begin{array}{ll} \text{Machine 1:} & 1 \to 5 \to 3; \\ \text{Machine 2:} & 6 \to 2 \to 7; \\ \text{Machine 3:} & 8 \to 4 \end{array}$$

which can satisfy the first goal, but the second objective is 0.05.

11.3 Fuzzy Models

Sometimes, the probability distributions are not easy to obtain due to the lack of data. Instead, expert knowledge is used to estimate the processing times.

Fuzzy Expected Time Goal Programming

In order to balance the conflicting objectives, we may have the following target levels and priority structure:

At the first priority level, the expected tardiness $E[f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})]$ should not exceed the target value b_1. Thus we have a goal constraint

$$E[f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})] - b_1 = d_1^+$$

in which $d_1^+ \vee 0$ will be minimized.

At the second priority level, the expected makespan $E[f_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})]$ should not exceed the target value b_2. That is, we have a goal constraint

$$E[f_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})] - b_2 = d_2^+$$

in which $d_2^+ \vee 0$ will be minimized.

Then we have the following fuzzy expected time goal programming model for the parallel machine scheduling problem (Peng and Liu [254]),

$$\begin{cases} \text{lexmin } \left\{d_1^+ \vee 0, d_2^+ \vee 0\right\} \\ \text{subject to:} \\ \quad E[f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})] - b_1 = d_1^+ \\ \quad E[f_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})] - b_2 = d_2^+ \\ \quad 1 \le x_i \le n, \quad i = 1, 2, \cdots, n \\ \quad x_i \ne x_j, \quad i \ne j, \ i, j = 1, 2, \cdots, n \\ \quad 0 \le y_1 \le y_2 \cdots \le y_{m-1} \le n \\ \quad x_i, y_j, \quad i = 1, 2, \cdots, n, \quad j = 1, 2, \cdots, m-1, \quad \text{integers.} \end{cases} \tag{11.9}$$

Example 11.4: Assume that there are 8 jobs and 3 machines. The trapezoidal fuzzy processing times and the due dates are listed in Table 11.2.

Table 11.2: Fuzzy Processing Times and Due Dates

Jobs	Fuzzy Processing Times			Due Dates
	Machine 1	Machine 2	Machine 3	
1	(10, 11, 14, 16)	(11, 12, 14, 15)	(12, 13, 15, 17)	30
2	(10, 13, 16, 18)	(10, 11, 15, 16)	(12, 13, 14, 18)	150
3	(12, 14, 15, 16)	(11, 13, 14, 16)	(12, 15, 16, 18)	105
4	(18, 21, 22, 24)	(20, 21, 22, 23)	(20, 23, 24, 25)	130
5	(10, 12, 13, 15)	(12, 14, 15, 17)	(10, 12, 13, 15)	60
6	(10, 14, 15, 18)	(12, 13, 16, 17)	(12, 16, 17, 20)	30
7	(10, 11, 12, 16)	(10, 11, 12, 14)	(10, 11, 12, 13)	75
8	(15, 17, 18, 20)	(14, 15, 16, 20)	(12, 14, 15, 16)	50

Suppose that $b_1 = 0$ and $b_2 = 45$. A run of the hybrid intelligent algorithm (10000 cycles in fuzzy simulation, 500 generations in GA) shows that the optimal schedule is

Machine 1: $1 \to 3 \to 5$
Machine 2: $6 \to 7 \to 2$
Machine 3: $4 \to 8$

which can satisfy the two goals.

Fuzzy Chance-Constrained Goal Programming

We assume the following priority structure. At the first priority level, the tardiness $f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})$ should not exceed the target value b_1 with a confidence

level α_1. Thus we have a goal constraint

$$\mathrm{Cr}\left\{f_1(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\xi})-b_1\le d_1^+\right\}\ge\alpha_1$$

in which $d_1^+\vee 0$ will be minimized.

At the second priority level, the makespan $f_2(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\xi})$ should not exceed the target value b_2 with a confidence level α_2. Thus we have a goal constraint

$$\mathrm{Cr}\left\{f_2(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\xi})-b_2\le d_2^+\right\}\ge\alpha_2$$

in which $d_2^+\vee 0$ will be minimized.

Then we have the following fuzzy CCGP model for the parallel machine scheduling problem (Peng and Liu [254]),

$$\begin{cases}\text{lexmin}\{d_1^+\vee 0, d_2^+\vee 0\}\\ \text{subject to:}\\ \quad \mathrm{Cr}\left\{f_1(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\xi})-b_1\le d_1^+\right\}\ge\alpha_1\\ \quad \mathrm{Cr}\left\{f_2(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\xi})-b_2\le d_2^+\right\}\ge\alpha_2\\ \quad 1\le x_i\le n,\quad i=1,2,\cdots,n\\ \quad x_i\ne x_j,\quad i\ne j,\ i,j=1,2,\cdots,n\\ \quad 0\le y_1\le y_2\cdots\le y_{m-1}\le n\\ \quad x_i,\ y_j,\quad i=1,2,\cdots,n,\quad j=1,2,\cdots,m-1,\quad \text{integers.}\end{cases}\tag{11.10}$$

Example 11.5: Suppose that $b_1=0$, $b_2=45$, $\alpha_1=0.95$ and $\alpha_2=0.90$. A run of the hybrid intelligent algorithm (10000 cycles in fuzzy simulation, 1000 generations in GA) shows that the optimal schedule is

Machine 1: $6\to 4$
Machine 2: $1\to 2\to 3$
Machine 3: $5\to 8\to 7$

which can satisfy the first goal, but the second objective is 0.12.

Fuzzy Dependent-Chance Goal Programming

Suppose that the management goals have the following priority structure.

At the first priority level, the credibility that the tardiness $f_1(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\xi})$ does not exceed the given value b_1 should achieve a confidence level α_1. Thus we have a goal constraint

$$\alpha_1-\mathrm{Cr}\left\{f_1(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\xi})\le b_1\right\}=d_1^-$$

in which $d_1^-\vee 0$ will be minimized.

At the second priority level, the credibility that the makespan $f_2(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\xi})$ does not exceed the given value b_2 should achieve a confidence level α_2. Thus we have a goal constraint

$$\alpha_2-\mathrm{Cr}\left\{f_2(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\xi})\le b_2\right\}=d_2^-$$

in which $d_2^- \vee 0$ will be minimized.

Then Peng and Liu [254] proposed the following fuzzy DCGP model for parallel machine scheduling problem,

$$\begin{cases} \text{lexmin}\{d_1^- \vee 0, d_2^- \vee 0\} \\ \text{subject to:} \\ \quad \alpha_1 - \text{Cr}\{f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) \leq b_1\} = d_1^- \\ \quad \alpha_2 - \text{Cr}\{f_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) \leq b_2\} = d_2^- \\ \quad 1 \leq x_i \leq n, \quad i = 1, 2, \cdots, n \\ \quad x_i \neq x_j, \quad i \neq j, \ i, j = 1, 2, \cdots, n \\ \quad 0 \leq y_1 \leq y_2 \cdots \leq y_{m-1} \leq n \\ \quad x_i, y_j, \quad i = 1, 2, \cdots, n, \quad j = 1, 2, \cdots, m-1, \quad \text{integers}. \end{cases} \tag{11.11}$$

Example 11.6: Suppose that $b_1 = 0$, $b_2 = 45$, $\alpha_1 = 0.95$ and $\alpha_2 = 0.90$. A run of the hybrid intelligent algorithm (10000 cycles in fuzzy simulation, 1000 generations in GA) shows that the optimal schedule is

Machine 1: $1 \to 5 \to 3$
Machine 2: $6 \to 4 \to 2$
Machine 3: $8 \to 7$

which can satisfy the first goal, but the second objective is 0.05.

11.4 Hybrid Models

We assume that the processing times are hybrid variables, and introduce some hybrid models for machine scheduling problem.

Hybrid Expected Time Goal Programming

In order to balance the conflicting objectives, we may have the following target levels and priority structure:

At the first priority level, the expected tardiness $E[f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})]$ should not exceed the target value b_1. Thus we have a goal constraint

$$E[f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})] - b_1 = d_1^+$$

in which $d_1^+ \vee 0$ will be minimized.

At the second priority level, the expected makespan $E[f_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})]$ should not exceed the target value b_2. That is, we have a goal constraint

$$E[f_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})] - b_2 = d_2^+$$

in which $d_2^+ \vee 0$ will be minimized.

Then we have the following hybrid expected time goal programming model for the parallel machine scheduling problem,

$$\begin{cases} \text{lexmin} \left\{ d_1^+ \vee 0, d_2^+ \vee 0 \right\} \\ \text{subject to:} \\ \quad E[f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})] - b_1 = d_1^+ \\ \quad E[f_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})] - b_2 = d_2^+ \\ \quad 1 \leq x_i \leq n, \quad i = 1, 2, \cdots, n \\ \quad x_i \neq x_j, \quad i \neq j, \ i, j = 1, 2, \cdots, n \\ \quad 0 \leq y_1 \leq y_2 \cdots \leq y_{m-1} \leq n \\ \quad x_i, y_j, \quad i = 1, 2, \cdots, n, \quad j = 1, 2, \cdots, m-1, \quad \text{integers.} \end{cases} \tag{11.12}$$

Hybrid Chance-Constrained Goal Programming

We assume the following priority structure. At the first priority level, the tardiness $f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})$ should not exceed the target value b_1 with confidence level α_1. Thus we have a goal constraint

$$\text{Ch} \left\{ f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) - b_1 \leq d_1^+ \right\} \geq \alpha_1$$

in which $d_1^+ \vee 0$ will be minimized.

At the second priority level, the makespan $f_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})$ should not exceed the target value b_2 with confidence level α_2. Thus we have a goal constraint

$$\text{Ch} \left\{ f_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) - b_2 \leq d_2^+ \right\} \geq \alpha_2$$

in which $d_2^+ \vee 0$ will be minimized.

Then we have the following hybrid CCGP model for the parallel machine scheduling problem,

$$\begin{cases} \text{lexmin}\{d_1^+ \vee 0, d_2^+ \vee 0\} \\ \text{subject to:} \\ \quad \text{Ch} \left\{ f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) - b_1 \leq d_1^+ \right\} \geq \alpha_1 \\ \quad \text{Ch} \left\{ f_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) - b_2 \leq d_2^+ \right\} \geq \alpha_2 \\ \quad 1 \leq x_i \leq n, \quad i = 1, 2, \cdots, n \\ \quad x_i \neq x_j, \quad i \neq j, \ i, j = 1, 2, \cdots, n \\ \quad 0 \leq y_1 \leq y_2 \cdots \leq y_{m-1} \leq n \\ \quad x_i, \, y_j, \quad i = 1, 2, \cdots, n, \quad j = 1, 2, \cdots, m-1, \quad \text{integers.} \end{cases} \tag{11.13}$$

Hybrid Dependent-Chance Goal Programming

Suppose that the management goals have the following priority structure.

At the first priority level, the chance that the tardiness $f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})$ does not exceed the given value b_1 should achieve a confidence level β_1. Thus we have a goal constraint

$$\beta_1 - \mathrm{Ch}\{f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) \leq b_1\} = d_1^-$$

in which $d_1^- \vee 0$ will be minimized.

At the second priority level, the chance that the makespan $f_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})$ does not exceed the given value b_2 should achieve a confidence level β_2. Thus we have a goal constraint

$$\beta_2 - \mathrm{Ch}\{f_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) \leq b_2\} = d_2^-$$

in which $d_2^- \vee 0$ will be minimized.

Then we have the following hybrid DCGP model for parallel machine scheduling problem,

$$\begin{cases} \mathrm{lexmin}\{d_1^- \vee 0, d_2^- \vee 0\} \\ \text{subject to:} \\ \quad \beta_1 - \mathrm{Ch}\{f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) \leq b_1\} = d_1^- \\ \quad \beta_2 - \mathrm{Ch}\{f_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) \leq b_2\} = d_2^- \\ \quad 1 \leq x_i \leq n, \quad i = 1, 2, \cdots, n \\ \quad x_i \neq x_j, \quad i \neq j, \ i, j = 1, 2, \cdots, n \\ \quad 0 \leq y_1 \leq y_2 \cdots \leq y_{m-1} \leq n \\ \quad x_i, y_j, \quad i = 1, 2, \cdots, n, \quad j = 1, 2, \cdots, m-1, \quad \text{integers}. \end{cases} \tag{11.14}$$

Problem 11.1. Design a hybrid intelligent algorithm to solve hybrid models for machine scheduling problem.

Chapter 12

Uncertain Programming

Uncertain programming was defined by Liu [175] as the optimization theory in generally uncertain environments, and provided the commonness of stochastic programming, fuzzy programming and hybrid programming.

12.1 Uncertain Variables

Let Γ be a nonempty set, and $\mathcal{L}$ a σ-algebra over Γ. Each element $\Lambda \in \mathcal{L}$ is called an *event*. In order to present an axiomatic definition of uncertain measure, it is necessary to assign to each event Λ a number $\mathcal{M}\{\Lambda\}$ which indicates the level that Λ will occur. In order to ensure that the number $\mathcal{M}\{\Lambda\}$ has certain mathematical properties, Liu [190] presented the following four axioms:

Axiom 1. *(Normality)* $\mathcal{M}\{\Gamma\} = 1$.

Axiom 2. *(Monotonicity)* $\mathcal{M}\{\Lambda_1\} \leq \mathcal{M}\{\Lambda_2\}$ *whenever* $\Lambda_1 \subset \Lambda_2$.

Axiom 3. *(Self-Duality)* $\mathcal{M}\{\Lambda\} + \mathcal{M}\{\Lambda^c\} = 1$ *for any event* Λ.

Axiom 4. *(Countable Subadditivity) For every countable sequence of events* $\{\Lambda_i\}$*, we have*

$$\mathcal{M}\left\{\bigcup_{i=1}^{\infty} \Lambda_i\right\} \leq \sum_{i=1}^{\infty} \mathcal{M}\{\Lambda_i\}. \tag{12.1}$$

.

Definition 12.1 *(Liu [190]) The set function* $\mathcal{M}$ *is called an uncertain measure if it satisfies the normality, monotonicity, self-duality, and countable subadditivity axioms.*

Example 12.1: Probability measure, credibility measure and chance measure are instances of uncertain measure.

Example 12.2: Let $\Gamma = \{\gamma_1, \gamma_2, \gamma_3\}$. For this case, there are only 8 events. Define

$$\mathcal{M}\{\gamma_1\} = 0.6, \quad \mathcal{M}\{\gamma_2\} = 0.3, \quad \mathcal{M}\{\gamma_3\} = 0.2,$$

$$\mathcal{M}\{\gamma_1, \gamma_2\} = 0.8, \quad \mathcal{M}\{\gamma_1, \gamma_3\} = 0.7, \quad \mathcal{M}\{\gamma_2, \gamma_3\} = 0.4,$$

$$\mathcal{M}\{\emptyset\} = 0, \quad \mathcal{M}\{\Gamma\} = 1.$$

It is clear that the set function $\mathcal{M}$ is neither probability measure nor credibility measure. However, $\mathcal{M}$ is an uncertain measure because it satisfies the four axioms.

Theorem 12.1 *Suppose that $\mathcal{M}$ is an uncertain measure. Then $\mathcal{M}\{\emptyset\} = 0$ and $0 \leq \mathcal{M}\{\Lambda\} \leq 1$ for any event Λ.*

Proof: It follows from Axioms 1 and 3 that $\mathcal{M}\{\emptyset\} = 1 - \mathcal{M}\{\Gamma\} = 1 - 1 = 0$. It follows from Axiom 2 that $0 \leq \mathcal{M}\{\Lambda\} \leq 1$ because of $\emptyset \subset \Lambda \subset \Gamma$.

Definition 12.2 *(Liu [190]) Let Γ be a nonempty set, $\mathcal{L}$ a σ-algebra over Γ, and $\mathcal{M}$ an uncertain measure. Then the triplet $(\Gamma, \mathcal{L}, \mathcal{M})$ is called an uncertainty space.*

Definition 12.3 *(Liu [190]) An uncertain variable is a measurable function from an uncertainty space $(\Gamma, \mathcal{L}, \mathcal{M})$ to the set of real numbers, i.e., for any Borel set B of real numbers, we have*

$$\{\xi \in B\} = \{\gamma \in \Gamma \mid \xi(\gamma) \in B\} \in \mathcal{L}. \tag{12.2}$$

Example 12.3: Random variable, fuzzy variable and hybrid variable are instances of uncertain variable.

An n-dimensional uncertain vector is a measurable function from an uncertainty space $(\Gamma, \mathcal{L}, \mathcal{M})$ to the set of n-dimensional real vectors, i.e., for any Borel set B of $\Re^n$, the set

$$\{\boldsymbol{\xi} \in B\} = \{\gamma \in \Gamma \mid \boldsymbol{\xi}(\gamma) \in B\} \tag{12.3}$$

is an event. It has been proved that $(\xi_1, \xi_2, \cdots, \xi_n)$ is an uncertain vector if and only if $\xi_1, \xi_2, \cdots, \xi_n$ are uncertain variables.

Definition 12.4 *Suppose that $f : \Re^n \to \Re$ is a measurable function, and $\xi_1, \xi_2, \cdots, \xi_n$ uncertain variables on the uncertainty space $(\Gamma, \mathcal{L}, \mathcal{M})$. Then $\xi = f(\xi_1, \xi_2, \cdots, \xi_n)$ is an uncertain variable defined as*

$$\xi(\gamma) = f(\xi_1(\gamma), \xi_2(\gamma), \cdots, \xi_n(\gamma)), \quad \forall \gamma \in \Gamma. \tag{12.4}$$

Definition 12.5 *(Liu [190]) Let ξ be an uncertain variable. Then the expected value of ξ is defined by*

$$E[\xi] = \int_0^{+\infty} \mathcal{M}\{\xi \geq r\}\mathrm{d}r - \int_{-\infty}^{0} \mathcal{M}\{\xi \leq r\}\mathrm{d}r \tag{12.5}$$

provided that at least one of the two integrals is finite.

Definition 12.6 *(Liu [190]) Let ξ be an uncertain variable, and $\alpha \in (0, 1]$. Then*

$$\xi_{\sup}(\alpha) = \sup\left\{r \mid \mathcal{M}\left\{\xi \geq r\right\} \geq \alpha\right\} \tag{12.6}$$

is called the α-optimistic value to ξ, and

$$\xi_{\inf}(\alpha) = \inf\left\{r \mid \mathcal{M}\left\{\xi \leq r\right\} \geq \alpha\right\} \tag{12.7}$$

is called the α-pessimistic value to ξ.

Ranking Criteria

Let ξ and η be two uncertain variables. Different from the situation of real numbers, there does not exist a natural ordership in an uncertain world. Thus an important problem appearing in this area is how to rank uncertain variables. Here we give four ranking criteria.

Expected Value Criterion: We say $\xi > \eta$ if and only if $E[\xi] > E[\eta]$.

Optimistic Value Criterion: We say $\xi > \eta$ if and only if, for some predetermined confidence level $\alpha \in (0, 1]$, we have $\xi_{\sup}(\alpha) > \eta_{\sup}(\alpha)$, where $\xi_{\sup}(\alpha)$ and $\eta_{\sup}(\alpha)$ are the α-optimistic values of ξ and η, respectively.

Pessimistic Value Criterion: We say $\xi > \eta$ if and only if, for some predetermined confidence level $\alpha \in (0, 1]$, we have $\xi_{\inf}(\alpha) > \eta_{\inf}(\alpha)$, where $\xi_{\inf}(\alpha)$ and $\eta_{\inf}(\alpha)$ are the α-pessimistic values of ξ and η, respectively.

Measure Criterion: We say $\xi > \eta$ if and only if, for some predetermined levels $\overline{r}$, we have $\mathcal{M}\left\{\xi \geq \overline{r}\right\} > \mathcal{M}\left\{\eta \geq \overline{r}\right\}$.

12.2 Expected Value Model

In order to obtain the decision with maximum expected return subject to expected constraints, we have the following uncertain EVM,

$$\begin{cases} \max E[f(\boldsymbol{x}, \boldsymbol{\xi})] \\ \text{subject to:} \\ \quad E[g_j(\boldsymbol{x}, \boldsymbol{\xi})] \leq 0, \ j = 1, 2, \cdots, p \end{cases} \tag{12.8}$$

where $\boldsymbol{x}$ is a decision vector, $\boldsymbol{\xi}$ is a uncertain vector, f is the objective function, and g_j are the constraint functions for $j = 1, 2, \cdots, p$.

In practice, a decision maker may want to optimize multiple objectives. Thus we have the following uncertain expected value multiobjective programming (EVMOP),

$$\begin{cases} \max\left[E[f_1(\boldsymbol{x},\boldsymbol{\xi})], E[f_2(\boldsymbol{x},\boldsymbol{\xi})], \cdots, E[f_m(\boldsymbol{x},\boldsymbol{\xi})]\right] \\ \text{subject to:} \\ \quad E[g_j(\boldsymbol{x},\boldsymbol{\xi})] \leq 0, \ j = 1,2,\cdots,p \end{cases} \tag{12.9}$$

where $f_i(\boldsymbol{x},\boldsymbol{\xi})$ are objective functions for $i = 1,2,\cdots,m$, and $g_j(\boldsymbol{x},\boldsymbol{\xi})$ are constraint functions for $j = 1,2,\cdots,p$.

In order to balance the multiple conflicting objectives, a decision-maker may establish a hierarchy of importance among these incompatible goals so as to satisfy as many goals as possible in the order specified. Thus we have an uncertain expected value goal programming (EVGP),

$$\begin{cases} \min \displaystyle\sum_{j=1}^{l} P_j \sum_{i=1}^{m} (u_{ij} d_i^+ \vee 0 + v_{ij} d_i^- \vee 0) \\ \text{subject to:} \\ \quad E[f_i(\boldsymbol{x},\boldsymbol{\xi})] - b_i = d_i^+, \quad i = 1,2,\cdots,m \\ \quad b_i - E[f_i(\boldsymbol{x},\boldsymbol{\xi})] = d_i^-, \quad i = 1,2,\cdots,m \\ \quad E[g_j(\boldsymbol{x},\boldsymbol{\xi})] \leq 0, \qquad\quad j = 1,2,\cdots,p \end{cases} \tag{12.10}$$

where P_j is the preemptive priority factor which expresses the relative importance of various goals, $P_j \gg P_{j+1}$, 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, $d_i^+ \vee 0$ is the positive deviation from the target of goal i, $d_i^- \vee 0$ is the negative deviation from the target of goal i, 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.

12.3 Chance-Constrained Programming

Assume that $\boldsymbol{x}$ is a decision vector, $\boldsymbol{\xi}$ is an uncertain vector, $f(\boldsymbol{x},\boldsymbol{\xi})$ is a return function, and $g_j(\boldsymbol{x},\boldsymbol{\xi})$ are constraint functions, $j = 1,2,\cdots,p$. Since the uncertain constraints $g_j(\boldsymbol{x},\boldsymbol{\xi}) \leq 0, j = 1,2,\cdots,p$ do not define a deterministic feasible set, it is naturally desired that the uncertain constraints hold with chance α, where α is a specified confidence level. Then we have a chance constraint as follows,

$$\mathcal{M}\left\{g_j(\boldsymbol{x},\boldsymbol{\xi}) \leq 0, j = 1,2,\cdots,p\right\} \geq \alpha. \tag{12.11}$$

Maximax Chance-Constrained Programming

If we want to maximize the optimistic value to the uncertain return subject to some chance constraints, we have the following uncertain maximax CCP,

$$\begin{cases} \max\limits_{\boldsymbol{x}} \max\limits_{\overline{f}} \overline{f} \\ \text{subject to:} \\ \quad \mathcal{M}\left\{f(\boldsymbol{x},\boldsymbol{\xi}) \geq \overline{f}\right\} \geq \beta \\ \quad \mathcal{M}\left\{g_j(\boldsymbol{x},\boldsymbol{\xi}) \leq 0\right\} \geq \alpha_j, \quad j=1,2,\cdots,p \end{cases} \tag{12.12}$$

where α_j and β are specified confidence levels for $j=1,2,\cdots,p$, and $\max \overline{f}$ is the β-optimistic return.

In practice, we may have multiple objectives. We thus have the following uncertain maximax chance-constrained multiobjective programming (CCMOP),

$$\begin{cases} \max\limits_{\boldsymbol{x}} \left[\max\limits_{\overline{f}_1} \overline{f}_1, \max\limits_{\overline{f}_2} \overline{f}_2, \cdots, \max\limits_{\overline{f}_m} \overline{f}_m\right] \\ \text{subject to:} \\ \quad \mathcal{M}\left\{f_i(\boldsymbol{x},\boldsymbol{\xi}) \geq \overline{f}_i\right\} \geq \beta_i, \quad i=1,2,\cdots,m \\ \quad \mathcal{M}\left\{g_j(\boldsymbol{x},\boldsymbol{\xi}) \leq 0\right\} \geq \alpha_j, \quad j=1,2,\cdots,p \end{cases} \tag{12.13}$$

where β_i are predetermined confidence levels for $i=1,2,\cdots,m$, and $\max \overline{f}_i$ are the β-optimistic values to the return functions $f_i(\boldsymbol{x},\boldsymbol{\xi})$, $i=1,2,\cdots,m$, respectively.

If the priority structure and target levels are set by the decision-maker, then we have a minimin chance-constrained goal programming (CCGP),

$$\begin{cases} \min\limits_{\boldsymbol{x}} \sum\limits_{j=1}^{l} P_j \sum\limits_{i=1}^{m} \left(u_{ij}\left(\min\limits_{d_i^+} d_i^+ \vee 0\right) + v_{ij}\left(\min\limits_{d_i^-} d_i^- \vee 0\right)\right) \\ \text{subject to:} \\ \quad \mathcal{M}\left\{f_i(\boldsymbol{x},\boldsymbol{\xi}) - b_i \leq d_i^+\right\} \geq \beta_i^+, \quad i=1,2,\cdots,m \\ \quad \mathcal{M}\left\{b_i - f_i(\boldsymbol{x},\boldsymbol{\xi}) \leq d_i^-\right\} \geq \beta_i^-, \quad i=1,2,\cdots,m \\ \quad \mathcal{M}\left\{g_j(\boldsymbol{x},\boldsymbol{\xi}) \leq 0\right\} \geq \alpha_j, \qquad j=1,2,\cdots,p \end{cases} \tag{12.14}$$

where P_j is the preemptive priority factor which expresses the relative importance of various goals, $P_j \gg P_{j+1}$, 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, $\min d_i^+ \vee 0$ is the β_i^+-optimistic positive deviation from the target of goal i, $\min d_i^- \vee 0$ is the β_i^--optimistic negative deviation from the target of goal i, b_i is the target value according to goal i, and l is the number of priorities.

Minimax Chance-Constrained Programming

If we want to maximize the pessimistic value subject to some chance constraints, we have the following uncertain minimax CCP,

$$\begin{cases} \max\limits_{\boldsymbol{x}} \min\limits_{\overline{f}} \overline{f} \\ \text{subject to:} \\ \quad \mathcal{M}\left\{f(\boldsymbol{x},\boldsymbol{\xi}) \le \overline{f}\right\} \ge \beta \\ \quad \mathcal{M}\left\{g_j(\boldsymbol{x},\boldsymbol{\xi}) \le 0\right\} \ge \alpha_j, \quad j=1,2,\cdots,p \end{cases} \tag{12.15}$$

where α_j and β are specified confidence levels for $j=1,2,\cdots,p$, and $\min \overline{f}$ is the β-pessimistic return.

If there are multiple objectives, then we have the following uncertain minimax CCMOP,

$$\begin{cases} \max\limits_{\boldsymbol{x}} \left[\min\limits_{\overline{f}_1} \overline{f}_1,\ \min\limits_{\overline{f}_2} \overline{f}_2,\cdots,\ \min\limits_{\overline{f}_m} \overline{f}_m\right] \\ \text{subject to:} \\ \quad \mathcal{M}\left\{f_i(\boldsymbol{x},\boldsymbol{\xi}) \le \overline{f}_i\right\} \ge \beta_i, \quad i=1,2,\cdots,m \\ \quad \mathcal{M}\left\{g_j(\boldsymbol{x},\boldsymbol{\xi}) \le 0\right\} \ge \alpha_j, \quad j=1,2,\cdots,p \end{cases} \tag{12.16}$$

where $\min \overline{f}_i$ are the β_i-pessimistic values to the return functions $f_i(\boldsymbol{x},\boldsymbol{\xi})$, $i=1,2,\cdots,m$, respectively.

We can also formulate a uncertain decision system as an uncertain minimax CCGP according to the priority structure and target levels set by the decision-maker:

$$\begin{cases} \min\limits_{\boldsymbol{x}} \sum\limits_{j=1}^{l} P_j \sum\limits_{i=1}^{m} \left[u_{ij}\left(\max\limits_{d_i^+} d_i^+ \vee 0\right) + v_{ij}\left(\max\limits_{d_i^-} d_i^- \vee 0\right)\right] \\ \text{subject to:} \\ \quad \mathcal{M}\left\{f_i(\boldsymbol{x},\boldsymbol{\xi}) - b_i \ge d_i^+\right\} \ge \beta_i^+, \quad i=1,2,\cdots,m \\ \quad \mathcal{M}\left\{b_i - f_i(\boldsymbol{x},\boldsymbol{\xi}) \ge d_i^-\right\} \ge \beta_i^-, \quad i=1,2,\cdots,m \\ \quad \mathcal{M}\left\{g_j(\boldsymbol{x},\boldsymbol{\xi}) \le 0\right\} \ge \alpha_j, \qquad j=1,2,\cdots,p \end{cases} \tag{12.17}$$

where P_j is the preemptive priority factor which expresses the relative importance of various goals, $P_j \gg P_{j+1}$, 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, $\max d_i^+ \vee 0$ is the β_i^+-pessimistic positive deviation from the target of goal i, $\max d_i^- \vee 0$ is the β_i^--pessimistic negative deviation from the target of goal i, b_i is the target value according to goal i, and l is the number of priorities.

12.4 Dependent-Chance Programming

This section provides uncertain DCP in which the underlying philosophy is based on selecting the decision with maximum chance to meet the event. A generally uncertain DCP has the following form,

$$\begin{cases} \max \mathcal{M}\{h_k(\boldsymbol{x},\boldsymbol{\xi}) \le 0, k=1,2,\cdots,q\} \\ \text{subject to:} \\ \quad g_j(\boldsymbol{x},\boldsymbol{\xi}) \le 0, \quad j=1,2,\cdots,p \end{cases} \tag{12.18}$$

where $\boldsymbol{x}$ is an n-dimensional decision vector, $\boldsymbol{\xi}$ is a uncertain vector, the event $\mathcal{E}$ is characterized by $h_k(\boldsymbol{x},\boldsymbol{\xi}) \le 0, k=1,2,\cdots,q$, and the uncertain environment is described by the uncertain constraints $g_j(\boldsymbol{x},\boldsymbol{\xi}) \le 0, j=1,2,\cdots,p$.

If there are multiple events in the uncertain environment, then we have the following uncertain dependent-chance multiobjective programming (DCMOP),

$$\begin{cases} \max \begin{bmatrix} \mathcal{M}\{h_{1k}(\boldsymbol{x},\boldsymbol{\xi}) \le 0, k=1,2,\cdots,q_1\} \\ \mathcal{M}\{h_{2k}(\boldsymbol{x},\boldsymbol{\xi}) \le 0, k=1,2,\cdots,q_2\} \\ \cdots \\ \mathcal{M}\{h_{mk}(\boldsymbol{x},\boldsymbol{\xi}) \le 0, k=1,2,\cdots,q_m\} \end{bmatrix} \\ \text{subject to:} \\ \quad g_j(\boldsymbol{x},\boldsymbol{\xi}) \le 0, \quad j=1,2,\cdots,p \end{cases} \tag{12.19}$$

where the events $\mathcal{E}_i$ are characterized by $h_{ik}(\boldsymbol{x},\boldsymbol{\xi}) \le 0, k=1,2,\cdots,q_i$, $i=1,2,\cdots,m$, respectively.

Uncertain dependent-chance goal programming (DCGP) is employed to formulate uncertain decision systems according to the priority structure and target levels set by the decision-maker,

$$\begin{cases} \min \sum_{j=1}^{l} P_j \sum_{i=1}^{m} (u_{ij} d_i^+ \vee 0 + v_{ij} d_i^- \vee 0) \\ \text{subject to:} \\ \quad \mathcal{M}\{h_{ik}(\boldsymbol{x},\boldsymbol{\xi}) \le 0, k=1,2,\cdots,q_i\} - b_i = d_i^+, \quad i=1,2,\cdots,m \\ \quad b_i - \mathcal{M}\{h_{ik}(\boldsymbol{x},\boldsymbol{\xi}) \le 0, k=1,2,\cdots,q_i\} = d_i^-, \quad i=1,2,\cdots,m \\ \quad g_j(\boldsymbol{x},\boldsymbol{\xi}) \le 0, \qquad\qquad\qquad\qquad\qquad\qquad j=1,2,\cdots,p \end{cases}$$

where P_j is the preemptive priority factor which expresses the relative importance of various goals, $P_j \gg P_{j+1}$, 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, $d_i^+ \vee 0$ is the positive deviation from the target of goal i, $d_i^- \vee 0$ is the negative deviation from the target of goal i, 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.

12.5 Uncertain Dynamic Programming

Stochastic dynamic programming has been studied widely in the literature. In human decision processes such as diagnosis, psychotherapy, and even design, some parameters may be regarded as fuzzy, or hybrid variables. In order to model generally uncertain decision processes, Liu [182] proposed a general principle of uncertain dynamic programming, including expected value dynamic programming, chance-constrained dynamic programming and dependent-chance dynamic programming.

Expected Value Dynamic Programming

Consider an N-stage decision system in which $(\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_N)$ represents the state vector, $(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N)$ the decision vector, $(\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \cdots, \boldsymbol{\xi}_N)$ the uncertain vector. We also assume that the state transition function is

$$\boldsymbol{a}_{n+1} = T(\boldsymbol{a}_n, \boldsymbol{x}_n, \boldsymbol{\xi}_n), \quad n = 1, 2, \cdots, N-1. \tag{12.20}$$

In order to maximize the expected return over the horizon, we may use the following expected value dynamic programming,

$$\begin{cases} f_N(\boldsymbol{a}) = \max\limits_{E[g_N(\boldsymbol{a},\boldsymbol{x},\boldsymbol{\xi}_N)]\leq 0} E[r_N(\boldsymbol{a},\boldsymbol{x},\boldsymbol{\xi}_N)] \\ f_n(\boldsymbol{a}) = \max\limits_{E[g_n(\boldsymbol{a},\boldsymbol{x},\boldsymbol{\xi}_n)]\leq 0} E[r_n(\boldsymbol{a},\boldsymbol{x},\boldsymbol{\xi}_n) + f_{n+1}(T(\boldsymbol{a},\boldsymbol{x},\boldsymbol{\xi}_n)] \\ n \leq N-1 \end{cases} \tag{12.21}$$

where r_n are the return functions at the nth stages, $n = 1, 2, \cdots, N$, respectively, and E denotes the expected value operator. This type of uncertain (especially stochastic) dynamic programming has been applied to a wide variety of problems, for example, inventory systems.

Chance-Constrained Dynamic Programming

In order to maximize the optimistic return over the horizon, we may use the following chance-constrained dynamic programming (Liu [182]),

$$\begin{cases} f_N(\boldsymbol{a}) = \max\limits_{\mathcal{M}\{g_N(\boldsymbol{a},\boldsymbol{x},\boldsymbol{\xi}_N)\leq 0\}\geq\alpha} \overline{r}_N(\boldsymbol{a},\boldsymbol{x},\boldsymbol{\xi}_N) \\ f_n(\boldsymbol{a}) = \max\limits_{\mathcal{M}\{g_n(\boldsymbol{a},\boldsymbol{x},\boldsymbol{\xi}_n)\leq 0\}\geq\alpha} \{\overline{r}_n(\boldsymbol{a},\boldsymbol{x},\boldsymbol{\xi}_n) + f_{n+1}(T(\boldsymbol{a},\boldsymbol{x},\boldsymbol{\xi}_n)\} \\ n \leq N-1 \end{cases} \tag{12.22}$$

where the functions $\overline{r}_n$ are defined by

$$\overline{r}_n(\boldsymbol{a}, \boldsymbol{x}, \boldsymbol{\xi}_n) = \sup\left\{\overline{r} \,\middle|\, \mathcal{M}\{r_n(\boldsymbol{a}, \boldsymbol{x}, \boldsymbol{\xi}_n) \geq \overline{r}\} \geq \beta\right\} \tag{12.23}$$

for $n = 1, 2, \cdots, N$. If we want to maximize the pessimistic return over the horizon, then we must define the functions $\overline{r}_n$ as

$$\overline{r}_n(\boldsymbol{a}, \boldsymbol{x}, \boldsymbol{\xi}_n) = \inf\left\{\overline{r} \,\middle|\, \mathcal{M}\{r_n(\boldsymbol{a}, \boldsymbol{x}, \boldsymbol{\xi}_n) \leq \overline{r}\} \geq \beta\right\} \tag{12.24}$$

for $n = 1, 2, \cdots, N$.

Dependent-Chance Dynamic Programming

In order to maximize the chance over the horizon, we may employ the following dependent-chance dynamic programming (Liu [182]),

$$\begin{cases} f_N(\boldsymbol{a}) = \max\limits_{g_N(\boldsymbol{a},\boldsymbol{x},\boldsymbol{\xi}_N)\leq 0} \mathcal{M}\{h_N(\boldsymbol{a}, \boldsymbol{x}, \boldsymbol{\xi}_N) \leq 0\} \\ f_n(\boldsymbol{a}) = \max\limits_{g_n(\boldsymbol{a},\boldsymbol{x},\boldsymbol{\xi}_n)\leq 0} \{\mathcal{M}\{h_n(\boldsymbol{a}, \boldsymbol{x}, \boldsymbol{\xi}_n) \leq 0\} + f_{n+1}(T(\boldsymbol{a}, \boldsymbol{x}, \boldsymbol{\xi}_n)\} \\ n \leq N-1 \end{cases}$$

where $h_n(\boldsymbol{a}, \boldsymbol{x}, \boldsymbol{\xi}_n) \leq 0$ are the events, and $g_n(\boldsymbol{a}, \boldsymbol{x}, \boldsymbol{\xi}_n) \leq 0$ are the uncertain environments at the nth stages, $n = 1, 2, \cdots, N$, respectively.

12.6 Uncertain Multilevel Programming

In order to model generally uncertain decentralized decision systems, Liu [182] proposed three types of uncertain multilevel programming, including expected value multilevel programming, chance-constrained multilevel programming and dependent-chance multilevel programming, and gave the concept of Stackelberg-Nash equilibrium to uncertain multilevel programming. After that, Gao, Liu and Gen [79] and Gao and Liu [80] investigated stochastic and fuzzy multilevel programming, and designed a series of hybrid intelligent algorithm for finding the Stackelberg-Nash equilibrium.

Expected Value Multilevel Programming

Assume that in a decentralized two-level decision system there is one leader and m followers. Let $\boldsymbol{x}$ and $\boldsymbol{y}_i$ be the control vectors of the leader and the ith followers, $i = 1, 2, \cdots, m$, respectively. We also assume that the objective functions of the leader and ith followers are $F(\boldsymbol{x}, \boldsymbol{y}_1, \cdots, \boldsymbol{y}_m, \boldsymbol{\xi})$ and $f_i(\boldsymbol{x}, \boldsymbol{y}_1, \cdots, \boldsymbol{y}_m, \boldsymbol{\xi})$, $i = 1, 2, \cdots, m$, respectively, where $\boldsymbol{\xi}$ is an uncertain (random, fuzzy, hybrid) vector.

Let the feasible set of control vector $\boldsymbol{x}$ of the leader be defined by the expected constraint

$$E[G(\boldsymbol{x}, \boldsymbol{\xi})] \leq 0 \tag{12.25}$$

where G is a vector-valued function and 0 is a zero vector. Then for each decision $\boldsymbol{x}$ chosen by the leader, the feasibility of control vectors $\boldsymbol{y}_i$ of the ith followers should be dependent on not only $\boldsymbol{x}$ but also $\boldsymbol{y}_1, \cdots, \boldsymbol{y}_{i-1}, \boldsymbol{y}_{i+1}, \cdots, \boldsymbol{y}_m$, and generally represented by the expected constraints,

$$E[g_i(\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m, \boldsymbol{\xi})] \leq 0 \tag{12.26}$$

where g_i are vector-valued functions, $i = 1, 2, \cdots, m$, respectively.

Assume that the leader first chooses his control vector $\boldsymbol{x}$, and the followers determine their control array $(\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m)$ after that. In order to maximize the expected objective of the leader, we have the following expected value bilevel programming,

$$\left\{\begin{array}{l} \max\limits_{\boldsymbol{x}} E[F(\boldsymbol{x}, \boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*, \boldsymbol{\xi})] \\ \text{subject to:} \\ \quad E[G(\boldsymbol{x}, \boldsymbol{\xi})] \leq 0 \\ \quad (\boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*) \text{ solves problems } (i = 1, 2, \cdots, m) \\ \quad \left\{\begin{array}{l} \max\limits_{\boldsymbol{y_i}} E[f_i(\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m, \boldsymbol{\xi})] \\ \text{subject to:} \\ \quad E[g_i(\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m, \boldsymbol{\xi})] \leq 0. \end{array}\right. \end{array}\right. \tag{12.27}$$

Definition 12.7 *Let $\boldsymbol{x}$ be a feasible control vector of the leader. A Nash equilibrium of followers is the feasible array $(\boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*)$ with respect to $\boldsymbol{x}$ if*

$$\begin{aligned} &E[f_i(\boldsymbol{x}, \boldsymbol{y}_1^*, \cdots, \boldsymbol{y}_{i-1}^*, \boldsymbol{y}_i, \boldsymbol{y}_{i+1}^*, \cdots, \boldsymbol{y}_m^*, \boldsymbol{\xi})] \\ &\quad \leq E[f_i(\boldsymbol{x}, \boldsymbol{y}_1^*, \cdots, \boldsymbol{y}_{i-1}^*, \boldsymbol{y}_i^*, \boldsymbol{y}_{i+1}^*, \cdots, \boldsymbol{y}_m^*, \boldsymbol{\xi})] \end{aligned} \tag{12.28}$$

for any feasible array $(\boldsymbol{y}_1^*, \cdots, \boldsymbol{y}_{i-1}^*, \boldsymbol{y}_i, \boldsymbol{y}_{i+1}^*, \cdots, \boldsymbol{y}_m^*)$ and $i = 1, 2, \cdots, m$.

Definition 12.8 *Suppose that $\boldsymbol{x}^*$ is a feasible control vector of the leader and $(\boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*)$ is a Nash equilibrium of followers with respect to $\boldsymbol{x}^*$. We call the array $(\boldsymbol{x}^*, \boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*)$ a Stackelberg-Nash equilibrium to the expected value bilevel programming (12.27) if and only if*

$$E[F(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}_1, \overline{\boldsymbol{y}}_2, \cdots, \overline{\boldsymbol{y}}_m, \boldsymbol{\xi})] \leq E[F(\boldsymbol{x}^*, \boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*, \boldsymbol{\xi})] \tag{12.29}$$

for any feasible control vector $\overline{\boldsymbol{x}}$ and the Nash equilibrium $(\overline{\boldsymbol{y}}_1, \overline{\boldsymbol{y}}_2, \cdots, \overline{\boldsymbol{y}}_m)$ with respect to $\overline{\boldsymbol{x}}$.

Chance-Constrained Multilevel Programming

In order to maximize the optimistic return subject to the chance constraint, we may use the following chance-constrained bilevel programming,

$$\begin{cases} \max\limits_{\boldsymbol{x}} \max\limits_{\overline{F}} \overline{F} \\ \text{subject to:} \\ \quad \mathcal{M}\{F(\boldsymbol{x}, \boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*, \boldsymbol{\xi}) \geq \overline{F}\} \geq \beta \\ \quad \mathcal{M}\{G(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0\} \geq \alpha \\ \quad (\boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*) \text{ solves problems } (i = 1, 2, \cdots, m) \\ \quad \begin{cases} \max\limits_{\boldsymbol{y_i}} \max\limits_{\overline{f}_i} \overline{f}_i \\ \text{subject to:} \\ \quad \mathcal{M}\{f_i(\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m, \boldsymbol{\xi}) \geq \overline{f}_i\} \geq \beta_i \\ \quad \mathcal{M}\{g_i(\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m, \boldsymbol{\xi}) \leq 0\} \geq \alpha_i \end{cases} \end{cases} \tag{12.30}$$

where $\alpha, \beta, \alpha_i, \beta_i$, $i = 1, 2, \cdots, m$ are predetermined confidence levels.

Definition 12.9 *Let $\boldsymbol{x}$ be a feasible control vector of the leader. A Nash equilibrium of followers is the feasible array $(\boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*)$ with respect to $\boldsymbol{x}$ if*

$$\begin{aligned} &\overline{f}_i(\boldsymbol{x}, \boldsymbol{y}_1^*, \cdots, \boldsymbol{y}_{i-1}^*, \boldsymbol{y}_i, \boldsymbol{y}_{i+1}^*, \cdots, \boldsymbol{y}_m^*) \\ &\quad \leq \overline{f}_i(\boldsymbol{x}, \boldsymbol{y}_1^*, \cdots, \boldsymbol{y}_{i-1}^*, \boldsymbol{y}_i^*, \boldsymbol{y}_{i+1}^*, \cdots, \boldsymbol{y}_m^*) \end{aligned} \tag{12.31}$$

for any feasible array $(\boldsymbol{y}_1^*, \cdots, \boldsymbol{y}_{i-1}^*, \boldsymbol{y}_i, \boldsymbol{y}_{i+1}^*, \cdots, \boldsymbol{y}_m^*)$ and $i = 1, 2, \cdots, m$.

Definition 12.10 *Suppose that $\boldsymbol{x}^*$ is a feasible control vector of the leader and $(\boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*)$ is a Nash equilibrium of followers with respect to $\boldsymbol{x}^*$. The array $(\boldsymbol{x}^*, \boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*)$ is called a Stackelberg-Nash equilibrium to the chance-constrained bilevel programming (12.30) if and only if,*

$$\overline{F}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}_1, \overline{\boldsymbol{y}}_2, \cdots, \overline{\boldsymbol{y}}_m) \leq \overline{F}(\boldsymbol{x}^*, \boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*) \tag{12.32}$$

for any feasible control vector $\overline{\boldsymbol{x}}$ and the Nash equilibrium $(\overline{\boldsymbol{y}}_1, \overline{\boldsymbol{y}}_2, \cdots, \overline{\boldsymbol{y}}_m)$ with respect to $\overline{\boldsymbol{x}}$.

In order to maximize the pessimistic return, we have the following minimax chance-constrained bilevel programming,

$$\begin{cases} \max\limits_{\boldsymbol{x}} \min\limits_{\overline{F}} \overline{F} \\ \text{subject to:} \\ \quad \mathcal{M}\{F(\boldsymbol{x}, \boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*, \boldsymbol{\xi}) \le \overline{F}\} \ge \beta \\ \quad \mathcal{M}\{G(\boldsymbol{x}, \boldsymbol{\xi}) \le 0\} \ge \alpha \\ \quad (\boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*) \text{ solves problems } (i = 1, 2, \cdots, m) \\ \quad \begin{cases} \max\limits_{\boldsymbol{y_i}} \min\limits_{\overline{f}_i} \overline{f}_i \\ \text{subject to:} \\ \quad \mathcal{M}\{f_i(\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m, \boldsymbol{\xi}) \le \overline{f}_i\} \ge \beta_i \\ \quad \mathcal{M}\{g_i(\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m, \boldsymbol{\xi}) \le 0\} \ge \alpha_i. \end{cases} \end{cases} \tag{12.33}$$

Dependent-Chance Multilevel Programming

Let $H(\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m, \boldsymbol{\xi}) \le 0$ and $h_i(\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m, \boldsymbol{\xi}) \le 0$ be the tasks of the leader and ith followers, $i = 1, 2, \cdots, m$, respectively. In order to maximize the chance functions of the leader and followers, we have the following dependent-chance bilevel programming,

$$\begin{cases} \max\limits_{\boldsymbol{x}} \mathcal{M}\{H(\boldsymbol{x}, \boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*, \boldsymbol{\xi}) \le 0\} \\ \text{subject to:} \\ \quad G(\boldsymbol{x}, \boldsymbol{\xi}) \le 0 \\ \quad (\boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*) \text{ solves problems } (i = 1, 2, \cdots, m) \\ \quad \begin{cases} \max\limits_{\boldsymbol{y_i}} \mathcal{M}\{h_i(\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m, \boldsymbol{\xi}) \le 0\} \\ \text{subject to:} \\ \quad g_i(\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m, \boldsymbol{\xi}) \le 0. \end{cases} \end{cases} \tag{12.34}$$

Definition 12.11 *Let $\boldsymbol{x}$ be a control vector of the leader. We call the array $(\boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*)$ a Nash equilibrium of followers with respect to $\boldsymbol{x}$ if*

$$\begin{aligned} &\mathcal{M}\{h_i(\boldsymbol{x}, \boldsymbol{y}_1^*, \cdots, \boldsymbol{y}_{i-1}^*, \boldsymbol{y}_i, \boldsymbol{y}_{i+1}^*, \cdots, \boldsymbol{y}_m^*, \boldsymbol{\xi}) \le 0\} \\ &\quad \le \mathcal{M}\{h_i(\boldsymbol{x}, \boldsymbol{y}_1^*, \cdots, \boldsymbol{y}_{i-1}^*, \boldsymbol{y}_i^*, \boldsymbol{y}_{i+1}^*, \cdots, \boldsymbol{y}_m^*, \boldsymbol{\xi}) \le 0\} \end{aligned} \tag{12.35}$$

subject to the uncertain environment $g_i(\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m, \boldsymbol{\xi}) \le 0, i = 1, 2, \cdots, m$ for any array $(\boldsymbol{y}_1^*, \cdots, \boldsymbol{y}_{i-1}^*, \boldsymbol{y}_i, \boldsymbol{y}_{i+1}^*, \cdots, \boldsymbol{y}_m^*)$ and $i = 1, 2, \cdots, m$.

Definition 12.12 *Suppose that $\boldsymbol{x}^*$ is a control vector of the leader, and $(\boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*)$ a Nash equilibrium of followers with respect to $\boldsymbol{x}^*$. Then $(\boldsymbol{x}^*, \boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*)$ is called a Stackelberg-Nash equilibrium to the dependent-chance bilevel programming (12.34) if and only if,*

$$\mathcal{M}\{H(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}_1, \overline{\boldsymbol{y}}_2, \cdots, \overline{\boldsymbol{y}}_m, \boldsymbol{\xi}) \le 0\} \le \mathcal{M}\{H(\boldsymbol{x}^*, \boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \cdots, \boldsymbol{y}_m^*, \boldsymbol{\xi}) \le 0\}$$

subject to the uncertain environment $G(\boldsymbol{x}, \boldsymbol{\xi}) \leq 0$ *for any control vector* $\overline{\boldsymbol{x}}$ *and the Nash equilibrium* $(\overline{\boldsymbol{y}}_1, \overline{\boldsymbol{y}}_2, \cdots, \overline{\boldsymbol{y}}_m)$ *with respect to* $\overline{\boldsymbol{x}}$.

12.7 Ψ Graph of Uncertain Programming

There are many possible ways that we can classify uncertain programming models. For example, we may classify them according to the state of knowledge about information, modeling structure, and uncertainty-handling philosophy. Let us now briefly outline some important aspects to be considered when discussing each of these.

1. State of knowledge about information
   a. Stochastic variable
   b. Fuzzy variable
   c. Hybrid variable
   d. Uncertain variable

2. Modeling structure
   a. Single-objective programming
   b. Multiobjective programming
   c. Goal programming
   d. Dynamic programming
   e. Multilevel programming

3. Uncertainty-handling philosophy
   a. Expected value model
   b. Chance-constrained programming
   c. Dependent-chance programming

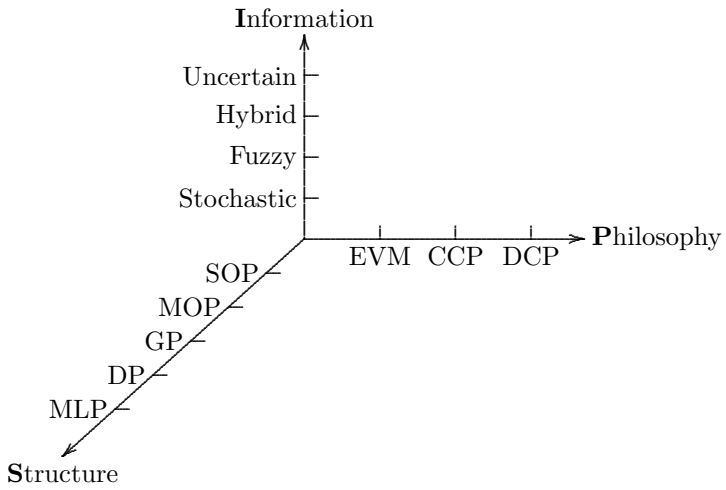

Figure 12.1: Ψ Graph for Uncertain Programming Classifications. Any type of uncertain programming may be represented by the Ψ graph which is essentially a coordinate system (**P**hilosophy, **S**tructure, **I**nformation). For example, the plane "**P**=CCP" represents chance-constrained programming; the plane "**I**=Stochastic" represents stochastic programming; the point "(**P**,**S**,**I**)=(DCP, GP, Fuzzy)" represents fuzzy dependent-chance goal programming.

Bibliography

[1] Abe S, *Neural Networks and Fuzzy Systems: Theory and Applications*, Kluwer Academic Publishers, Boston, 1997.

[2] Aggarwal KK, Chopra YC, Bajwa JS, Topological layout of links for optimizing the s-t reliability in a computer communication system, *Microelectronics & Reliability*, Vol.22, No.3, 341-345, 1982.

[3] Aggarwal KK, Chopra YC, Bajwa JS, Topological layout of links for optimizing the overall reliability in a computer communication system, *Microelectronics & Reliability*, Vol.22, No.3, 347-351, 1982.

[4] Angelov P, A generalized approach to fuzzy optimization, *International Journal of Intelligent Systems*, Vol.9, 261-268, 1994.

[5] Armentano VA, and Ronconi DP, Tabu search for total tardiness minimization in flowshop scheduling problems, *Computers and Operations Research*, Vol.26, No.3, 219-235, 1999.

[6] Ballestero E, Using stochastic goal programming: Some applications to management and a case of industrial production, *INFOR*, Vol.43, No.2, 63-77, 2005.

[7] Bandemer H, and Nather W, *Fuzzy Data Analysis*, Kluwer Academic Publishers, Dordrecht, 1992.

[8] Bard JF, An algorithm for solving the general bilevel programming problem, *Mathematics of Operations Research*, Vol.8, 260-272, 1983.

[9] Bard JF, and Moore JT, A branch and bound algorithm for the bilevel programming problem, *SIAM J. Sci. Statist. Comput.*, Vol.11, 281-292, 1990.

[10] Bard JF, Optimality conditions for the bilevel programming problem, *Naval Research Logistics Quarterly*, Vol.31, 13-26, 1984.

[11] Bastian C, and Rinnooy Kan AHG, The stochastic vehicle routing problem revisited, *European Journal of Operational Research*, Vol.56, 407-412, 1992.

[12] Bellman RE, *Dynamic Programming*, Princeton University Press, New Jersey, 1957.

[13] Bellman RE, and Zadeh LA, Decision making in a fuzzy environment, *Management Science*, Vol.17, 141-164, 1970.

[14] Ben-Ayed O, and Blair CE, Computational difficulties of bilevel linear programming, *Operations Research*, Vol.38, 556-560, 1990.

[15] Ben Abdelaziz F, Enneifar L, and Martel JM, A multiobjective fuzzy stochastic program for water resources optimization: The case of lake management, *INFOR*, Vol.42, No.3, 201-214, 2004.

[16] Bertsekas DP, and Tsitsiklis JN, *Parallel and Distributed Computation: Numerical Methods*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[17] Bertsimas DJ, and Simchi-Levi D, A new generation of vehicle routing research: Robust algorithms, addressing uncertainty, *Operations Research*, Vol.44, No.2, 286-304, 1996.

[18] Bialas WF, and Karwan MH, Two-level linear programming, *Management Science*, Vol.30, 1004-1020, 1984.

[19] Biswal MP, Fuzzy programming technique to solve multi-objective geometric programming problems, *Fuzzy Sets and Systems*, Vol.51, 61-71, 1992.

[20] Bit AK, Biswal MP, and Alam SS, Fuzzy programming approach to multicriteria decision making transportation problem, *Fuzzy Sets and Systems*, Vol.50, 135-141, 1992.

[21] Bitran GR, Linear multiple objective problems with interval coefficients, *Management Science*, Vol.26, 694-706, 1980.

[22] Bouchon-Meunier B, Kreinovich V, Lokshin A, and Nguyen HT, On the formulation of optimization under elastic constraints (with control in mind), *Fuzzy Sets and Systems*, Vol.81, 5-29, 1996.

[23] Bratley P, Fox BL, and Schrage LE, *A Guide to Simulation*, Springer-Verlag, New York, 1987.

[24] Brucker P, *Scheduling Algorithms*, 2nd ed., Springer, Berlin, New York, 1998.

[25] Buckley JJ, Possibility and necessity in optimization, *Fuzzy Sets and Systems*, Vol.25, 1-13, 1988.

[26] Buckley JJ, Stochastic versus possibilistic programming, *Fuzzy Sets and Systems*, Vol.34, 173-177, 1990.

[27] Buckley JJ, Multiobjective possibilistic linear programming, *Fuzzy Sets and Systems*, Vol.35, 23-28, 1990.

[28] Buckley JJ, and Hayashi Y, Fuzzy genetic algorithm and applications, *Fuzzy Sets and Systems*, Vol.61, 129-136, 1994.

[29] Buckley JJ, and Feuring T, Evolutionary algorithm solution to fuzzy problems: Fuzzy linear programming, *Fuzzy Sets and Systems*, Vol.109, No.1, 35-53, 2000.

[30] Cadenas JM, and Verdegay JL, Using fuzzy numbers in linear programming, *IEEE Transactions on Systems, Man and Cybernetics–Part B*, Vol.27, No.6, 1016-1022, 1997.

[31] Campos L, and González A, A subjective approach for ranking fuzzy numbers, *Fuzzy Sets and Systems*, Vol.29, 145-153, 1989.

[32] Campos L, and Verdegay JL, Linear programming problems and ranking of fuzzy numbers, *Fuzzy Sets and Systems*, Vol.32, 1-11, 1989.

[33] Campos FA, Villar J, and Jimenez M, Robust solutions using fuzzy chance constraints, *Engineering Optimization*, Vol.38, No.6, 627-645, 2006.

[34] Candler W, and Townsley R, A linear two-level programming problem, *Computers and Operations Research*, Vol.9, 59-76, 1982.

[35] Carlsson C, Fuller R, and Majlender P, A possibilistic approach to selecting portfolios with highest utility score, *Fuzzy Sets and Systems*, Vol.131, No.1, 13-21, 2002.

[36] Changchit C, and Terrell MP, CCGP model for multiobjective reservoir systems, *Journal of Water Resources Planning and Management*, Vol.115, No.5, 658-670, 1989.

[37] Chankong V, and Haims YY, *Multiobjective Decision Making: Theory and Methodology*, North-Holland, Amsterdam, 1983.

[38] Charnes A, and Cooper WW, Chance-constrained programming, *Management Science*, Vol.6, No.1, 73-79, 1959.

[39] Charnes A, and Cooper WW, *Management Models and Industrial Applications of Linear Programming,* Wiley, New York, 1961.

[40] Charnes A, and Cooper WW, Chance constraints and normal deviates, *Journal of the American Statistical Association*, Vol.57, 134-148, 1962.

[41] Charnes A, and Cooper WW, Deterministic equivalents for optimizing and satisficing under chance-constraints, *Operations Research*, Vol.11, No.1, 18-39, 1963.

[42] Charnes A, and Cooper WW, Goal programming and multiple objective optimizations: Part I, *European Journal of Operational Research*, Vol.1, No.1, 39-54, 1977.

[43] Chanas S, Fuzzy programming in multiobjective linear programming—a parametric approach, *Fuzzy Sets and Systems*, Vol.29, 303-313, 1989.

[44] Chanas S, and Kuchta D, Multiobjective programming in optimization of interval objective functions—a generalized approach, *European Journal of Operational Research*, Vol.94, 594-598, 1996.

[45] Chen A, and Ji ZW, Path finding under uncertainty, *Journal of Advance Transportation*, Vol.39, No.1, 19-37, 2005.

[46] Chen SJ, and Hwang CL, *Fuzzy Multiple Attribute Decision Making: Methods and Applications*, Springer-Verlag, Berlin, 1992.

[47] Chen Y, Fung RYK, Yang J, Fuzzy expected value modelling approach for determining target values of engineering characteristics in QFD, *International Journal of Production Research*, Vol.43, No.17, 3583-3604, 2005.

[48] Chen Y, Fung RYK, Tang JF, Rating technical attributes in fuzzy QFD by integrating fuzzy weighted average method and fuzzy expected value operator, *European Journal of Operational Research*, Vol.174, No.3, 1553-1566, 2006.

[49] Chopra YC, Sohi BS, Tiwari RK, Aggarwal KK, Network topology for maximizing the terminal reliability in a computer communication network, *Microelectronics & Reliability*, Vol.24, 911-913, 1984.

[50] Clayton E, Weber W, and Taylor III B, A goal programming approach to the optimization of multiresponse simulation models, *IIE Trans.,* Vol.14, 282-287, 1982.

[51] Cooper L, Location-allocation problems, *Operations Research*, Vol.11, 331-344, 1963.

[52] Cybenko G, Approximations by superpositions of a sigmoidal function, *Mathematics of Control, Signals and Systems*, Vol.2, 183-192, 1989.

[53] Dantzig GB, *Linear Programming and Extensions*, Princeton University Press, NJ, 1963.

[54] Das B, Maity K, Maiti A, A two warehouse supply-chain model under possibility/necessity/credibility measures, *Mathematical and Computer Modelling*, Vol.46, No.3-4, 398-409, 2007.

[55] Delgado M, Verdegay JL, and Vila MA, A general model for fuzzy linear programming, *Fuzzy Sets and Systems*, Vol.29, 21-29, 1989.

[56] Dengiz B, Altiparmak F, and Smith AE, Efficient optimization of all-terminal reliable networks using an evolutionary approach, *IEEE Transactions on Reliability*, Vol.46, No.1, 18-26, 1997.

[57] Dror M, Laporte G, and Trudreau P, Vehicle routing with stochastic demands: Properties and solution frameworks, *Transportation Science*, Vol.23, 166-176, 1989.

[58] Dubois D, and Prade H, *Fuzzy Sets and Systems, Theory and Applications*, Academic Press, New York, 1980.

[59] Dubois D, and Prade H, Fuzzy cardinality and the modeling of imprecise quantification, *Fuzzy Sets and Systems*, Vol.16, 199-230, 1985.

[60] Dubois D, and Prade H, The mean value of a fuzzy number, *Fuzzy Sets and Systems*, Vol.24, 279-300, 1987.

[61] Dubois D, and Prade H, *Possibility Theory: An Approach to Computerized Processing of Uncertainty*, Plenum, New York, 1988.

[62] Dubois D, and Prade H, Fuzzy numbers: An overview, in *Analysis of Fuzzy Information*, Vol.2, 3-39, Bezdek JC (Ed.), CRC Press, Boca Raton, 1988.

[63] Dunyak J, Saad IW, and Wunsch D, A theory of independent fuzzy probability for system reliability, *IEEE Transactions on Fuzzy Systems*, Vol.7, No.3, 286-294, 1999.

[64] Ermoliev Y, and Wets RJB(eds.), *Numerical Techniques for Stochastic Optimization*, Springer-Verlag, Berlin, 1988.

[65] Esogbue AO, and Liu B, Reservoir operations optimization via fuzzy criterion decision processes, *Fuzzy Optimization and Decision Making*, Vol.5, No.3, 289-305, 2006.

[66] Feng X, and Liu YK, Measurability criteria for fuzzy random vectors, *Fuzzy Optimization and Decision Making*, Vol.5, No.3, 245-253, 2006.

[67] Feng Y, Yang LX, A two-objective fuzzy k-cardinality assignment problem, *Journal of Computational and Applied Mathematics*, Vol.197, No.1, 233-244, 2006.

[68] Fine TL, *Feedforward Neural Network Methodology*, Springer, New York, 1999.

[69] Fishman GS, *Monte Carlo: Concepts, Algorithms, and Applications*, Springer-Verlag, New York, 1996.

[70] Fogel DB, An introduction to simulated evolutionary optimization, *IEEE Transactions on Neural Networks,* Vol.5, 3-14, 1994.

[71] Fogel DB, *Evolution Computation: Toward a New Philosophy of Machine Intelligence*, IEEE Press, Piscataway, NJ, 1995.

[72] Fortemps P, Jobshop scheduling with imprecise durations: A fuzzy approach, *IEEE Transactions on Fuzzy Systems*, Vol.5, No.4, 557-569, 1997.

[73] Freeman RJ, A generalized PERT, *Operations Research*, Vol.8, No.2, 281, 1960.

[74] Freeman RJ, A generalized network approach to project activity sequencing, *IRE Transactions on Engineering Management*, Vol.7, No.3, 103-107, 1960.

[75] Fuller R, On stability in fuzzy linear programming problems, *Fuzzy Sets and Systems*, Vol.30, 339-344, 1989.

[76] Fung K, A philosophy for allocating component reliabilities in a network, *IEEE Transactions on Reliability,* Vol.34, No.2, 151-153, 1985.

[77] Fung RYK, Chen YZ, Chen L, A fuzzy expected value-based goal programing model for product planning using quality function deployment, *Engineering Optimization*, Vol.37, No.6, 633-647, 2005.

[78] Gao J, and Liu B, New primitive chance measures of fuzzy random event, *International Journal of Fuzzy Systems*, Vol.3, No.4, 527-531, 2001.

[79] Gao J, Liu B, and Gen M, A hybrid intelligent algorithm for stochastic multilevel programming, *IEEJ Transactions on Electronics, Information and Systems*, Vol.124-C, No.10, 1991-1998, 2004.

[80] Gao J, and Liu B, Fuzzy multilevel programming with a hybrid intelligent algorithm, *Computers & Mathematics with Applications*, Vol.49, 1539-1548, 2005.

[81] Gao J, and Lu M, Fuzzy quadratic minimum spanning tree problem, *Applied Mathematics and Computation*, Vol.164, No.3, 773-788, 2005.

[82] Gao J, and Feng XQ, A hybrid intelligent algorithm for fuzzy dynamic inventory problem, *Journal of Information and Computing Science*, Vol.1, No.4, 235-244, 2006.

[83] Gao J, Credibilistic game with fuzzy information, *Journal of Uncertain Systems*, Vol.1, No.1, 74-80, 2007.

[84] Gen M, and Liu B, Evolution program for production plan problem, *Engineering Design and Automation*, Vol.1, No.3, 199-204, 1995.

[85] Gen M, Liu B, and Ida K, Evolution program for deterministic and stochastic optimizations, *European Journal of Operational Research*, Vol.94, No.3, 618-625, 1996.

[86] Gen M, and Liu B, Evolution program for optimal capacity expansion, *Journal of Operations Research Society of Japan*, Vol.40, No.1, 1-9, 1997.

[87] Gen M, and Cheng R, *Genetic Algorithms & Engineering Design*, Wiley, New York, 1997.

[88] Gen M, and Liu B, A genetic algorithm for nonlinear goal programming, *Evolutionary Optimization*, Vol.1, No.1, 65-76, 1999.

[89] Gen M, and Cheng R, *Genetic Algorithms & Engineering Optimization*, Wiley, New York, 2000.

[90] Gendreau M, Laporte G, and Séguin R, Stochastic vehicle routing, *European Journal of Operational Research*, Vol.88, 3-12, 1999.

[91] Goldberg DE, *Genetic Algorithms in Search, Optimization and Machine Learning,* Addison-Wesley, MA, 1989.

[92] González A, A study of the ranking function approach through mean values, *Fuzzy Sets and Systems*, Vol.35, 29-41, 1990.

[93] Han S, Ishii H, Fujii S, One machine scheduling problem with fuzzy duedates, *European Journal of Operational Research*, Vol.79, No.1, 1-12, 1994.

[94] Hayashi H, Buckley JJ, and Czogala E, Fuzzy neural network with fuzzy signals and weights, *International Journal of Intelligent Systems*, Vol.8, 527-537, 1993.

[95] He Y, and Xu J, A class of random fuzzy programming model and its application to vehicle routing problem, *World Journal of Modelling and Simulation*, Vol.1, No.1, 3-11, 2005.

[96] Heilpern S, The expected value of a fuzzy number, *Fuzzy Sets and Systems*, Vol.47, 81-86, 1992.

[97] Hisdal E, *Logical Structures for Representation of Knowledge and Uncertainty*, Physica-Verlag, Heidelberg, 1998.

[98] Holland JH, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[99] Hornik K, Stinchcombe M, White H, Multilayer feedforward networks are universal approximators, *Neural Networks*, Vol.2, 359-366, 1989.

[100] Hu CF, and Fang SC, Solving fuzzy inequalities with piecewise linear membership functions, *IEEE Transactions on Fuzzy Systems*, Vol.7, 230-235, 1999.

[101] Inuiguchi M, Ichihashi H, and Kume Y, Relationships between modality constrained programming problems and various fuzzy mathematical programming problems, *Fuzzy Sets and Systems* Vol.49, 243-259, 1992.

[102] Inuiguchi M, Ichihashi H, and Kume Y, Modality constrained programming problems: An unified approach to fuzzy mathematical programming problems in the setting of possibility theory, *Information Sciences*, Vol.67, 93-126, 1993.

[103] Inuiguchi M, and Ramík J, Possibilistic linear programming: A brief review of fuzzy mathematical programming and a comparison with stochastic programming in portfolio selection problem, *Fuzzy Sets and Systems*, Vol.111, No.1, 3-28, 2000.

[104] Ishibuchi H, and Tanaka H, Multiobjective programming in optimization of the interval objective function, *European Journal of Operational Research*, Vol.48, 219-225, 1990.

[105] Ishibuchi H, and Tanaka H, An architecture of neural networks with interval weights and its application to fuzzy regression analysis, *Fuzzy Sets and Systems*, Vol.57, 27-39, 1993.

[106] Ishibuchi H, Yamamoto N, Murata T, Tanaka H, Genetic algorithms and neighborhood search algorithms for fuzzy flowshop scheduling problems, *Fuzzy Sets and Systems*, Vol.67, No.1, 81-100, 1994.

[107] Iwamura K, and Liu B, A genetic algorithm for chance constrained programming, *Journal of Information & Optimization Sciences*, Vol.17, No.2, 409-422, 1996.

[108] Iwamura K, and Liu B, Chance constrained integer programming models for capital budgeting in fuzzy environments, *Journal of the Operational Research Society*, Vol.49, No.8, 854-860, 1998.

[109] Iwamura K and Liu B, Stochastic operation models for open inventory networks, *Journal of Information & Optimization Sciences*, Vol20, No.3, 347-363, 1999.

[110] Iwamura K and Liu B, Dependent-chance integer programming applied to capital budgeting, *Journal of the Operations Research Society of Japan*, Vol.42, No.2, 117-127, 1999.

[111] Jan RH, Hwang FJ, Chen ST, Topological optimization of a communication network subject to a reliability constraint, *IEEE Transactions on Reliability*, Vol.42, 63-70, 1993.

[112] Ji XY, Models and algorithm for stochastic shortest path problem, *Applied Mathematics and Computation*, Vol.170, No.1, 503-514, 2005.

[113] Ji XY, and Shao Z, Model and algorithm for bilevel newsboy problem with fuzzy demands and discounts, *Applied Mathematics and Computation*, Vol.172, No.1, 163-174, 2006.

[114] Ji XY, Iwamura K, and Shao Z, New models for shortest path problem with fuzzy arc lengths, *Applied Mathematical Modelling*, Vol.31, No.2, 259-269, 2007.

[115] Jiménez F, and Verdegay JL, Uncertain solid transportation problems, *Fuzzy Sets and Systems*, Vol.100, Nos.1-3, 45-57, 1998.

[116] Kacprzyk J, and Esogbue AO, Fuzzy dynamic programming: Main developments and applications, *Fuzzy Sets and Systems*, Vol.81, 31-45, 1996.

[117] Kacprzyk J, Multisatge control of a stochastic system in a fuzzy environment using a genetic algorithm, *International Journal of Intelligent Systems*, Vol.13, 1011-1023, 1998.

[118] Kacprzyk J, Romero RA, and Gomide FAC, Involving objective and subjective aspects in multistage decision making and control under fuzziness: Dynamic programming and neural networks, *International Journal of Intelligent Systems*, Vol.14, 79-104, 1999.

[119] Kacprzyk J, Pasi G, Vojtas P, Fuzzy querying: Issues and perspectives, *Kybernetika*, Vol.36, 605-616, 2000.

[120] Kacprzyk J, Zadrozny S, Computing with words in intelligent database querying: Standalone and Internet-based applications, *Information Sciences*, Vol.134, 71-109, 2001.

[121] Kall P, and Wallace SW, *Stochastic Programming,* Wiley, Chichester, 1994.

[122] Kaufmann A, *Introduction to the Theory of Fuzzy Subsets*, Vol.I, Academic Press, New York, 1975.

[123] Kaufman A, and Gupta MM, *Introduction to Fuzzy Arithmetic: Theory and Applications*, Van Nostrand Reinhold, New York, 1985.

[124] Kaufman A, and Gupta MM, *Fuzzy Mathematical Models in Engineering and Management Science*, 2nd ed., North-Holland, Amsterdam, 1991.

[125] Ke H, and Liu B, Project scheduling problem with stochastic activity duration times, *Applied Mathematics and Computation*, Vol.168, No.1, 342-353, 2005.

[126] Ke H, and Liu B, Project scheduling problem with mixed uncertainty of randomness and fuzziness, *European Journal of Operational Research*, Vol.183, No.1, 135-147, 2007.

[127] Ke H, and Liu B, Fuzzy project scheduling problem and its hybrid intelligent algorithm, Technical Report, 2005.

[128] Keller JM, and Tahani H, Implementation of conjunctive and disjunctive fuzzy logic rules with neural networks, *International Journal of Approximate Reasoning*, Vol.6, 221-240, 1992.

[129] Keown AJ, A chance-constrained goal programming model for bank liquidity management, *Decision Sciences*, Vol.9, 93-106, 1978.

[130] Keown AJ, and Taylor BW, A chance-constrained integer goal programming model for capital budgeting in the production area, *Journal of the Operational Research Society*, Vol.31, No.7, 579-589, 1980.

[131] Kitano H, Neurogenetic learning: An integrated method of designing and training neural networks using genetic algorithms, *Physica D*, Vol.75, 225-228, 1994.

[132] Klein G, Moskowitz H, and Ravindran A, Interactive multiobjective optimization under uncertainty, *Management Science*, Vol.36, No.1, 58-75, 1990.

[133] Klement EP, Puri ML, and Ralescu DA, Limit theorems for fuzzy random variables, *Proceedings of the Royal Society of London*, Vol.407, 171-182, 1986.

[134] Klir GJ, and Folger TA, *Fuzzy Sets, Uncertainty, and Information*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[135] Klir GJ, and Yuan B, *Fuzzy Sets and Fuzzy Logic: Theory and Applications*, Prentice-Hall, New Jersey, 1995.

[136] Kolonko M, Some new results on simulated annealing applied to the job shop scheduling problem, *European Journal of Operational Research*, Vol.113, No.1, 123-136, 1999.

[137] Konno T, Ishii H, An open shop scheduling problem with fuzzy allowable time and fuzzy resource constraint, *Fuzzy Sets and Systems*, Vol.109, No.1, 141-147, 2000.

[138] Kou W, and Prasad VR, An annotated overview of system-reliability optimization, *IEEE Transactions on Reliability*, Vol.49, 176-197, 2000.

[139] Koza JR, *Genetic Programming*, MIT Press, Cambridge, MA, 1992.

[140] Koza JR, *Genetic Programming, II*, MIT Press, Cambridge, MA, 1994.

[141] Kruse R, and Meyer KD, *Statistics with Vague Data*, D. Reidel Publishing Company, Dordrecht, 1987.

[142] Kumar A, Pathak RM, and Gupta YP, Genetic-algorithm-based reliability optimization for computer network expansion, *IEEE Transactions on Reliability*, Vol.44, 63-72, 1995.

[143] Kumar A, Pathak RM, Gupta YP, and Parsaei HR, A Genetic algorithm for distributed system topology design, *Computers and Industrial Engineering*, Vol.28, 659-670, 1995.

[144] Kwakernaak H, Fuzzy random variables–I: Definitions and theorems, *Information Sciences*, Vol.15, 1-29, 1978.

[145] Kwakernaak H, Fuzzy random variables–II: Algorithms and examples for the discrete case, *Information Sciences*, Vol.17, 253-278, 1979.

[146] Lai YJ, and Hwang CL, A new approach to some possibilistic linear programming problems, *Fuzzy Sets and Systems*, Vol.49, 121-133, 1992.

[147] Lai YJ, and Hwang CL, *Fuzzy Multiple Objective Decision Making: Methods and Applications*, Springer-Verlag, New York, 1994.

[148] Laporte G, Louveaux FV, and Mercure H, The vehicle routing problem with stochastic travel times, *Transportation Science*, Vol.26, 161-170, 1992.

[149] Law AM, and Kelton WD, *Simulation Modelling & Analysis*, 2nd edition, McGraw-Hill, New York, 1991.

[150] Lee ES, and Li RJ, Fuzzy multiple objective programming and compromise programming with Pareto optimum, *Fuzzy Sets and Systems*, Vol.53, 275-288, 1993.

[151] Lee ES, Fuzzy multiple level programming, *Applied Mathematics and Computation*, Vol.120, 79-90, 2001.

[152] Lee KH, *First Course on Fuzzy Theory and Applications*, Springer-Verlag, Berlin, 2005.

[153] Lee SM, *Goal Programming for Decision Analysis*, Auerbach, Philadelphia, 1972.

[154] Lee SM, and Olson DL, A gradient algorithm for chance constrained nonlinear goal programming, *European Journal of Operational Research*, Vol.22, 359-369, 1985.

[155] Lertworasirkul S, Fang SC, Joines JA, and Nuttle HLW, Fuzzy data envelopment analysis (DEA): a possibility approach, *Fuzzy Sets and Systems*, Vol.139, No.2, 379-394, 2003.

[156] Li HL, and Yu CS, A fuzzy multiobjective program with quasiconcave membership functions and fuzzy coefficients, *Fuzzy Sets and Systems*, Vol.109, No.1, 59-81, 2000.

[157] Li J, Xu J, and Gen M, A class of multiobjective linear programming model with fuzzy random coefficients, *Mathematical and Computer Modelling*, Vol.44, Nos.11-12, 1097-1113, 2006.

[158] Li P, and Liu B, Entropy of credibility distributions for fuzzy variables, *IEEE Transactions on Fuzzy Systems*, Vol.16, No.1, 123-129, 2008.

[159] Li SY, Hu W, Yang YP, Receding horizon fuzzy optimization under local information environment with a case study, *International Journal of Information Technology & Decision Making*, Vol.3, No.1, 109-127, 2004.

[160] Li X, and Liu B, The independence of fuzzy variables with applications, *International Journal of Natural Sciences & Technology*, Vol.1, No.1, 95-100, 2006.

[161] Li X, and Liu B, New independence definition of fuzzy random variable and random fuzzy variable, *World Journal of Modelling and Simulation*, Vol.2, No.5, 338-342, 2006.

[162] Li X, and Liu B, A sufficient and necessary condition for credibility measures, *International Journal of Uncertainty, Fuzziness & Knowledge-Based Systems*, Vol.14, No.5, 527-535, 2006.

[163] Li X, and Liu B, Chance measure for hybrid events with fuzziness and randomness, *Soft Computing*, to be published.

[164] Litoiu M, Tadei R, Real-time task scheduling with fuzzy deadlines and processing times, *Fuzzy Sets and Systems*, Vol.117, No.1, 35-45, 2001.

[165] Liu B, Dependent-chance goal programming and its genetic algorithm based approach, *Mathematical and Computer Modelling*, Vol.24, No.7, 43-52, 1996.

[166] Liu B, and Esogbue AO, Fuzzy criterion set and fuzzy criterion dynamic programming, *Journal of Mathematical Analysis and Applications*, Vol.199, No.1, 293-311, 1996.

[167] Liu B, Dependent-chance programming: A class of stochastic optimization, *Computers & Mathematics with Applications*, Vol.34, No.12, 89-104, 1997.

[168] Liu B, and Iwamura K, Modelling stochastic decision systems using dependent-chance programming, *European Journal of Operational Research*, Vol.101, No.1, 193-203, 1997.

[169] Liu B, and Iwamura K, Chance constrained programming with fuzzy parameters, *Fuzzy Sets and Systems*, Vol.94, No.2, 227-237, 1998.

[170] Liu B, and Iwamura K, A note on chance constrained programming with fuzzy coefficients, *Fuzzy Sets and Systems*, Vol.100, Nos.1-3, 229-233, 1998.

[171] Liu B, Stackelberg-Nash equilibrium for multilevel programming with multiple followers using genetic algorithms, *Computers & Mathematics with Applications*, Vol.36, No.7, 79-89, 1998.

[172] Liu B, Minimax chance constrained programming models for fuzzy decision systems, *Information Sciences*, Vol.112, Nos.1-4, 25-38, 1998.

[173] Liu B, Dependent-chance programming with fuzzy decisions, *IEEE Transactions on Fuzzy Systems*, Vol.7, No.3, 354-360, 1999.

[174] Liu B, and Esogbue AO, *Decision Criteria and Optimal Inventory Processes*, Kluwer Academic Publishers, Boston, 1999.

[175] Liu B, *Uncertain Programming*, Wiley, New York, 1999.

[176] Liu B, Dependent-chance programming in fuzzy environments, *Fuzzy Sets and Systems*, Vol.109, No.1, 97-106, 2000.

[177] Liu B, and Iwamura K, Topological optimization models for communication network with multiple reliability goals, *Computers & Mathematics with Applications*, Vol.39, 59-69, 2000.

[178] Liu B, Uncertain programming: A unifying optimization theory in various uncertain environments, *Applied Mathematics and Computation*, Vol.120, Nos.1-3, 227-234, 2001.

[179] Liu B, and Iwamura K, Fuzzy programming with fuzzy decisions and fuzzy simulation-based genetic algorithm, *Fuzzy Sets and Systems*, Vol.122, No.2, 253-262, 2001.

[180] Liu B, Fuzzy random chance-constrained programming, *IEEE Transactions on Fuzzy Systems*, Vol.9, No.5, 713-720, 2001.

[181] Liu B, Fuzzy random dependent-chance programming, *IEEE Transactions on Fuzzy Systems*, Vol.9, No.5, 721-726, 2001.

[182] Liu B, *Theory and Practice of Uncertain Programming*, Physica-Verlag, Heidelberg, 2002.

[183] Liu B, Toward fuzzy optimization without mathematical ambiguity, *Fuzzy Optimization and Decision Making*, Vol.1, No.1, 43-63, 2002.

[184] Liu B, and Lai KK, Stochastic programming models for vehicle routing problems, *Asian Information-Science-Life*, Vol.1, No.1, 13-28, 2002.

[185] Liu B, and Liu YK, Expected value of fuzzy variable and fuzzy expected value models, *IEEE Transactions on Fuzzy Systems*, Vol.10, No.4, 445-450, 2002.

[186] Liu B, Random fuzzy dependent-chance programming and its hybrid intelligent algorithm, *Information Sciences*, Vol.141, Nos.3-4, 259-271, 2002.

[187] Liu B, *Uncertainty Theory*, Springer-Verlag, Berlin, 2004.

[188] Liu B, A survey of credibility theory, *Fuzzy Optimization and Decision Making*, Vol.5, No.4, 387-408, 2006.

[189] Liu B, A survey of entropy of fuzzy variables, *Journal of Uncertain Systems*, Vol.1, No.1, 4-13, 2007.

[190] Liu B, *Uncertainty Theory*, 2nd ed., Springer-Verlag, Berlin, 2007.

[191] Liu B, Fuzzy process, hybrid process and uncertain process, *Journal of Uncertain Systems*, Vol.2, No.1, 3-16, 2008.

[192] Liu LZ, Li YZ, The fuzzy quadratic assignment problem with penalty: New models and genetic algorithm, *Applied Mathematics and Computation*, Vol.174, No.2, 1229-1244, 2006.

[193] Liu XW, Measuring the satisfaction of constraints in fuzzy linear programming *Fuzzy Sets and Systems*, Vol.122, No.2, 263-275, 2001.

[194] Liu YK, and Liu B, Random fuzzy programming with chance measures defined by fuzzy integrals, *Mathematical and Computer Modelling*, Vol.36, Nos.4-5, 509-524, 2002.

[195] Liu YK, and Liu B, Fuzzy random programming problems with multiple criteria, *Asian Information-Science-Life*, Vol.1, No.3, 249-256, 2002.

[196] Liu YK, and Liu B, A class of fuzzy random optimization: Expected value models, *Information Sciences*, Vol.155, Nos.1-2, 89-102, 2003.

[197] Liu YK, and Liu B, Fuzzy random variables: A scalar expected value operator, *Fuzzy Optimization and Decision Making*, Vol.2, No.2, 143-160, 2003.

[198] Liu YK, and Liu B, Expected value operator of random fuzzy variable and random fuzzy expected value models, *International Journal of Uncertainty, Fuzziness & Knowledge-Based Systems*, Vol.11, No.2, 195-215, 2003.

[199] Liu YK, and Liu B, On minimum-risk problems in fuzzy random decision systems, *Computers & Operations Research*, Vol.32, No.2, 257-283, 2005.

[200] Liu YK, and Liu B, Fuzzy random programming with equilibrium chance constraints, *Information Sciences*, Vol.170, 363-395, 2005.

[201] Liu YK, Fuzzy programming with recourse, *International Journal of Uncertainty, Fuzziness & Knowledge-Based Systems*, Vol.13, No.4, 381-413, 2005.

[202] Liu YK, Convergent results about the use of fuzzy simulation in fuzzy optimization problems, *IEEE Transactions on Fuzzy Systems*, Vol.14, No.2, 295-304, 2006.

[203] Liu YK, and Wang SM, On the properties of credibility critical value functions, *Journal of Information and Computing Science*, Vol.1, No.4, 195-206, 2006.

[204] Liu YK, and Gao J, The independence of fuzzy variables in credibility theory and its applications, *International Journal of Uncertainty, Fuzziness & Knowledge-Based Systems*, Vol.15, Supp.2, 1-20, 2007.

[205] Liu YK, The approximation method for two-stage fuzzy random programming with recourse, *IEEE Transactions on Fuzzy Systems*, Vol.15, No.6, 1197-1208, 2007.

[206] Loetamonphong J, and Fang SC, An efficient solution procedure for fuzzy relation equations with max-product composition, *IEEE Transactions on Fuzzy Systems*, Vol.7, 441-445, 1999.

[207] Logendran R, and Terrell MP, Uncapacitated plant location-allocation problems with price sensitive stochastic demands, *Computers and Operations Research*, Vol.15, No.2, 189-198, 1988.

[208] Lu M, On crisp equivalents and solutions of fuzzy programming with different chance measures, *Information: An International Journal*, Vol.6, No.2, 125-133, 2003.

[209] Lucas C, and Araabi BN, Generalization of the Dempster-Shafer Theory: A fuzzy-valued measure, *IEEE Transactions on Fuzzy Systems*, Vol.7, No.3, 255-270, 1999.

[210] Luhandjula MK, Linear programming under randomness and fuzziness, *Fuzzy Sets and Systems*, Vol.10, 45-55, 1983.

[211] Luhandjula MK, On possibilistic linear programming, *Fuzzy Sets and Systems*, Vol.18, 15-30, 1986.

[212] Luhandjula MK, Fuzzy optimization: An appraisal, *Fuzzy Sets and Systems*, Vol.30, 257-282, 1989.

[213] Luhandjula MK, Fuzziness and randomness in an optimization framework, *Fuzzy Sets and Systems*, Vol.77, 291-297, 1996.

[214] Luhandjula MK, and Gupta MM, On fuzzy stochastic optimization, *Fuzzy Sets and Systems*, Vol.81, 47-55, 1996.

[215] Luhandjula MK, Optimisation under hybrid uncertainty, *Fuzzy Sets and Systems*, Vol.146, No.2, 187-203, 2004.

[216] Luhandjula MK, Optimization under hybrid uncertainty, *Fuzzy Sets and Systems*, Vol.146, 187-203, 2004.

[217] Luhandjula MK, Fuzzy stochastic linear programming: Survey and future research directions, *European Journal of Operational Research*, Vol.174, No.3, 1353-1367, 2006.

[218] Mabuchi S, An interpretation of membership functions and the properties of general probabilistic operators as fuzzy set operators, *Fuzzy Sets and Systems*, Vol.92, No.1, 31-50, 1997.

[219] Maiti MK, Maiti MA, Fuzzy inventory model with two warehouses under possibility constraints, *Fuzzy Sets and Systems*, Vol.157, No.1, 52-73, 2006.

[220] Maleki HR, Tata M, and Mashinchi M, Linear programming with fuzzy variables, *Fuzzy Sets and Systems*, Vol.109, No.1, 21-33, 2000.

[221] Maniezzo V, Genetic evolution of the topology and weight distribution of neural networks, *IEEE Transactions on Neural Networks*, Vol.5, No.1, 39-53, 1994.

[222] Mareš M, *Computation Over Fuzzy Quantities*, CRC Press, Boca Raton, 1994.

[223] Mareschal B, Stochastic multicriteria decision making and uncertainty, *European Journal of Operational Research,* Vol.26, No.1, 58-64, 1986.

[224] Martel A, and Price W, Stochastic programming applied to human resource planning, *Journal of the Operational Research Society*, Vol.32, 187-196, 1981.

[225] Masud A, and Hwang C, Interactive sequential goal programming, *Journal of the Operational Research Society,* Vol.32, 391-400, 1981.

[226] Marianov V, Rios M, and Barros FJ, Allocating servers to facilities, when demand is elastic to travel and waiting times, *RAIRO-Operations Research*, Vol.39, No.3, 143-162, 2005.

[227] Medsker LR, *Hybrid Intelligent Systems*, Kluwer Academic Publishers, Boston, 1995.

[228] Mendel JM, *Uncertainty Rule-Based Fuzzy Logic Systems: Introduction and New Directions*, Prentice-Hall, New Jersey, 2001.

[229] Michalewicz Z, *Genetic Algorithms + Data Structures = Evolution Programs,* 3rd ed., Springer-Verlag, Berlin, 1996.

[230] Minsky M, and Papert S, *Perceptrons*, MIT Press, Cambridge, MA, 1969.

[231] Mitchell M, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA, 1996.

[232] Mohamed RH, A chance-constrained fuzzy goal program, *Fuzzy Sets and Systems*, Vol.47, 183-186, 1992.

[233] Mohammed W, Chance constrained fuzzy goal programming with right-hand side uniform random variable coefficients, *Fuzzy Sets and Systems*, Vol.109, No.1, 107-110, 2000.

[234] Mohan C, and Nguyen HT, An interactive satisfying method for solving multiobjective mixed fuzzy-stochastic programming problems, *Fuzzy Sets and Systems*, Vol.117, 95-111, 2001.

[235] Morgan B, *Elements of Simulation*, Chapamn & Hall, London, 1984.

[236] Morgan DR, Eheart JW, and Valocchi AJ, Aquifer remediation design under uncertainty using a new chance constrained programming technique, *Water Resources Research*, Vol.29, No.3, 551-561, 1993.

[237] Mukherjee SP, Mixed strategies in chance-constrained programming, *Journal of the Operational Research Society*, Vol.31, 1045-1047, 1980.

[238] Murat C, and Paschos VTH, The probabilistic longest path problem, *Networks*, Vol.33, 207-219, 1999.

[239] Murray AT, and Church RL, Applying simulated annealing to location-planning models, *Journal of Heuristics*, Vol.2, No.1, 31-53, 1996.

[240] Nahmias S, Fuzzy variables, *Fuzzy Sets and Systems*, Vol.1, 97-110, 1978.

[241] Negoita CV, and Ralescu D, On fuzzy optimization, *Kybernetes*, Vol.6, 193-195, 1977.

[242] Negoita CV, and Ralescu D, *Simulation, Knowledge-based Computing, and Fuzzy Statistics*, Van Nostrand Reinhold, New York, 1987.

[243] Nguyen VH, Fuzzy stochastic goal programming problems, *European Journal of Operational Research*, Vol.176, No.1, 77-86, 2007.

[244] Ohlemuller M, Tabu search for large location-allocation problems, *Journal of the Operational Research Society*, Vol.48, No.7, 745-750, 1997.

[245] Ostasiewicz W, A new approach to fuzzy programming, *Fuzzy Sets and Systems*, Vol.7, 139-152, 1982.

[246] Painton L, and Campbell J, Genetic algorithms in optimization of system reliability, *IEEE Transactions on Reliability*, Vol.44, 172-178, 1995.

[247] Pawlak Z, Rough sets, *International Journal of Information and Computer Sciences*, Vol.11, No.5, 341-356, 1982.

[248] Pawlak Z, Rough sets and fuzzy sets, *Fuzzy Sets and Systems*, Vol.17, 99-102, 1985.

[249] Pawlak Z, *Rough Sets: Theoretical Aspects of Reasoning about Data*, Kluwer Academic Publishers, Dordrecht, 1991.

[250] Pawlak Z, and Slowinski R, Rough set approach to multi-attribute decision analysis, *European Journal of Operational Research*, Vol.72, 443-459, 1994.

[251] Pawlak Z, Rough set approach to knowledge-based decision support, *European Journal of Operational Research*, Vol.99, 48-57, 1997.

[252] Pedrycz W, Optimization schemes for decomposition of fuzzy relations, *Fuzzy Sets and Systems*, Vol.100, 301-325, 1998.

[253] Peng J, and Liu B, Stochastic goal programming models for parallel machine scheduling problems, *Asian Information-Science-Life*, Vol.1, No.3, 257-266, 2002.

[254] Peng J, and Liu B, Parallel machine scheduling models with fuzzy processing times, *Information Sciences*, Vol.166, Nos.1-4, 49-66, 2004.

[255] Peng J, and Liu B, A framework of birandom theory and optimization methods, *Information: An International Journal*, Vol.9, No.4, 629-640, 2006.

[256] Peng J, and Zhao XD, Some theoretical aspects of the critical values of birandom variable, *Journal of Information and Computing Science*, Vol.1, No.4, 225-234, 2006.

[257] Peng J, and Liu B, Birandom variables and birandom programming, *Computers & Industrial Engineering*, Vol.53, No.3, 433-453, 2007.

[258] Puri ML, and Ralescu D, Fuzzy random variables, *Journal of Mathematical Analysis and Applications*, Vol.114, 409-422, 1986.

[259] Qin ZF, and Liu B, On some special hybrid variables, Technical Report, 2007.

[260] Raj PA, and Kumer DN, Ranking alternatives with fuzzy weights using maximizing set and minimizing set, *Fuzzy Sets and Systems*, Vol.105, 365-375, 1999.

[261] Ramer A, Conditional possibility measures, *International Journal of Cybernetics and Systems*, Vol.20, 233-247, 1989.

[262] Ramík J, Extension principle in fuzzy optimization, *Fuzzy Sets and Systems*, Vol.19, 29-35, 1986.

[263] Ramík J, and Rommelfanger H, Fuzzy mathematical programming based on some inequality relations, *Fuzzy Sets and Systems*, Vol.81, 77-88, 1996.

[264] Ravi V, Murty BSN, and Reddy PJ, Nonequilibrium simulated annealing-algorithm applied to reliability optimization of complex systems, *IEEE Transactions on Reliability*, Vol.46, 233-239, 1997.

[265] Rommelfanger H, Hanscheck R, and Wolfe J, Linear programming with fuzzy objectives, *Fuzzy Sets and Systems*, Vol.29, 31-48, 1989.

[266] Roubens M, and Teghem Jr J, Comparison of methodologies for fuzzy and stochastic multi-objective programming, *Fuzzy Sets and Systems*, Vol.42, 119-132, 1991.

[267] Roy B, Main sources of inaccurate determination, uncertainty and imprecision in decision models, *Mathematical and Computer Modelling*, Vol.12, 1245-1254, 1989.

[268] Rubinstein RY, *Simulation and the Monte Carlo Method*, Wiley, New York, 1981.

[269] Saade JJ, Maximization of a function over a fuzzy domain, *Fuzzy Sets and Systems*, Vol.62, 55-70, 1994.

[270] Saber HM, and Ravindran A, Nonlinear goal programming theory and practice: A survey, *Computers and Operations Research*, Vol.20, 275-291, 1993.

[271] Sakawa M, and Yano H, Feasibility and Pareto optimality for multiobjective nonlinear programming problems with fuzzy parameters, *Fuzzy Sets and Systems*, Vol.43, 1-15, 1991.

[272] Sakawa M, Kato K, Katagiri H, An interactive fuzzy satisficing method for multiobjective linear programming problems with random variable coefficients through a probability maximization model, *Fuzzy Sets and Systems*, Vol.146, No.2, 205-220, 2004.

[273] Sakawa M, Nishizaki I, and Uemura Y, Interactive fuzzy programming for multi-level linear programming problems with fuzzy parameters, *Fuzzy Sets and Systems*, Vol.109, No.1, 3-19, 2000.

[274] Sakawa M, Kubota R, Fuzzy programming for multiobjective job shop scheduling with fuzzy processing time and fuzzy duedate through genetic algorithms, *European Journal of Operational Research*, Vol.120, No.2, 393-407, 2000.

[275] Sakawa M, Nishizaki I, Uemura Y, Interactive fuzzy programming for two-level linear fractional programming problems with fuzzy parameters, *Fuzzy Sets and Systems*, Vol.115, 93-103, 2000.

[276] Sakawa M, Nishizaki I, Hitaka M, Interactive fuzzy programming for multi-level 0-1 programming problems with fuzzy parameters through genetic algorithms, *Fuzzy Sets and Systems*, Vol.117, 95-111, 2001.

[277] Savard G, and Gauvin J, The steepest descent direction for nonlinear bilevel programming problem, *Operations Research Letters*, Vol.15, 265-272, 1994.

[278] Schalkoff RJ, *Artificial Neural Networks*, McGraw-Hill, New York, 1997.

[279] Schneider M, and Kandel A, Properties of the fuzzy expected value and the fuzzy expected interval in fuzzy environment, *Fuzzy Sets and Systems*, Vol.26, 373-385, 1988.

[280] Shao Z, and Ji XY, Fuzzy multi-product constraint newsboy problem, *Applied Mathematics and Computation*, Vol.180, No.1, 7-15, 2006.

[281] Sherali HD, and Rizzo TP, Unbalanced, capacitated p-median problems on a chain graph with a continuum of link demands, *Networks*, Vol.21, No.2, 133-163, 1991.

[282] Shih HS, Lai YJ, Lee ES, Fuzzy approach for multilevel programming problems, *Computers and Operations Research,* Vol.23, 73-91, 1996.

[283] Shimizu K, Ishizuka Y, and Bard JF, *Nondifferentiable and Two-level Mathematical Programming*, Kluwer, Boston, 1997.

[284] Shin WS, and Ravindran, A, Interactive multiple objective optimization: Survey I - continuous case, *Computers and Operations Research*, Vol.18, No.1, 97-114, 1991.

[285] Silva EF, and Wood RK, Solving a class of stochastic mixed-integer programs with branch and price, *Mathematical Programming*, Vol.108, Nos.2-3, 395-418, 2007.

[286] Slowinski R, and Teghem Jr J, Fuzzy versus stochastic approaches to multicriteria linear programming under uncertainty, *Naval Research Logistics*, Vol.35, 673-695, 1988.

[287] Slowinski R, and Stefanowski J, Rough classification in incomplete information systems, *Mathematical and Computer Modelling*, Vol.12, 1347-1357, 1989.

[288] Slowinski R, and Vanderpooten D, A generalized definition of rough approximations based on similarity, *IEEE Transactions on Knowledge and Data Engineering*, Vol.12, No.2, 331-336, 2000.

[289] Sommer G, and Pollatschek, MA, A fuzzy programming approach to an air pollution regulation problem, *European Journal of Operational Research*, Vol.10, 303-313, 1978.

[290] Soroush HM, The most critical path in a PERT network, *Journal of the Operational Research Society*, Vol.45, 287-300, 1994.

[291] Steuer RE, Algorithm for linear programming problems with interval objective function coefficients, *Mathematics of Operational Research*, Vol.6, 333-348, 1981.

[292] Steuer RE, *Multiple Criteria Optimization: Theory, Computation and Application*, Wiley, New York, 1986.

[293] Stewart WR Jr, and Golden BL, Stochastic vehicle routing: A comprehensive approach, *European Journal of Operational Research*, Vol.14, 371-385, 1983.

[294] Suykens JAK, Vandewalle JPL, De Moor BLR, *Artificial Neural Networks for Modelling and Control of Non-Linear Systems*, Kluwer Academic Publishers, Boston, 1996.

[295] Szmidt E, Kacprzyk J, Entropy for intuitionistic fuzzy sets, *Fuzzy Sets and Systems*, Vol.118, 467-477, 2001.

[296] Taha HA, *Operations Research: An Introduction*, Macmillan, New York, 1982.

[297] Tanaka H, and Asai K, Fuzzy linear programming problems with fuzzy numbers, *Fuzzy Sets and Systems*, Vol.13, 1-10, 1984.

[298] Tanaka H, and Asai K, Fuzzy solutions in fuzzy linear programming problems, *IEEE Transactions on Systems, Man and Cybernetics*, Vol.14, 325-328, 1984.

[299] Tanaka H, Guo P, *Possibilistic Data Analysis for Operations Research*, Physica-Verlag, Heidelberg, 1999.

[300] Tanaka H, Guo P, and Zimmermann HJ, Possibility distribution of fuzzy decision variables obtained from possibilistic linear programming problems, *Fuzzy Sets and Systems*, Vol.113, 323-332, 2000.

[301] Teghem Jr J, DuFrane D, and Kunsch P, STRANGE: An interactive method for multiobjective linear programming under uncertainty, *European Journal Operational Research*, Vol.26, No.1, 65-82, 1986.

[302] Turunen E, *Mathematics Behind Fuzzy Logic*, Physica-Verlag, Heidelberg, 1999.

[303] Van Rooij AJF, Jain LC, Johnson RP, *Neural Network Training Using Genetic Algorithms*, World Scientific, Singapore, 1996.

[304] Wagner BJ, and Gorelick SM, Optimal ground water quality management under parameter uncertainty, *Water Resources Research*, Vol.23, No.7, 1162-1174, 1987.

[305] Wang D, An inexact approach for linear programming problems with fuzzy objective and resources, *Fuzzy Sets and Systems*, Vol.89, 61-68, 1998.

[306] Wang G, and Qiao Z, Linear programming with fuzzy random variable coefficients, *Fuzzy Sets and Systems*, Vol.57, 295-311, 1993.

[307] Wang YP, Jiao YC, Li H, An evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint-handling scheme, *IEEE Transactions on Systems Man and Cybernetics Part C*, Vol.35, No.2, 221-232, 2005.

[308] Waters CDJ, Vehicle-scheduling problems with uncertainty and omitted customers, *Journal of the Operational Research Society*, Vol.40, 1099-1108, 1989.

[309] Weistroffer H, An interactive goal programming method for nonlinear multiple-criteria decision-making problems, *Computers and Operations Research,* Vol.10, No.4, 311-320, 1983.

[310] Wen M, and Iwamura K, Fuzzy facility location-allocation problem under the Hurwicz criterion, *European Journal of Operational Research*, Vol.184, No.2, 627-635, 2008.

[311] Wen M, and Iwamura K, Facility location-allocation problem in random fuzzy environment: Using (α, β)-cost minimization model under the Hurewicz criterion, *Computers and Mathematics with Applications*, Vol.55, No.4, 704-713, 2008.

[312] Whalen T, Decision making under uncertainty with various assumptions about available information, *IEEE Transactions on Systems, Man and Cybernetics*, Vol.14, 888-900, 1984.

[313] Xu J, and Li J, A class of stochastic optimization problems with one quadratic and several linear objective functions and extended portfolio selection model, *Journal of Computational and Applied Mathematics*, Vol.146, No.1, 99-113, 2002.

[314] Yager RR, Mathematical programming with fuzzy constraints and a preference on the objective, *Kybernetes*, Vol.9, 285-291, 1979.

[315] Yager RR, A procedure for ordering fuzzy subsets of the unit interval, *Information Sciences*, Vol.24, 143-161, 1981.

[316] Yager RR, Generalized probabilities of fuzzy events from fuzzy belief structures, *Information Sciences*, Vol.28, 45-62, 1982.

[317] Yager RR, On ordered weighted averaging aggregation operators in multicriteria decision making, *IEEE Transactions on Systems, Man and Cybernetics*, Vol.18, 183-190, 1988.

[318] Yager RR, On the specificity of a possibility distribution, *Fuzzy Sets and Systems*, Vol.50, 279-292, 1992.

[319] Yager RR, Modeling uncertainty using partial information, *Information sciences*, Vol.121, 271-294, 1999.

[320] Yager RR, Decision making with fuzzy probability assessments, *IEEE Transactions on Fuzzy Systems*, Vol.7, 462-466, 1999.

[321] Yager RR, On the evaluation of uncertain courses of action, *Fuzzy Optimization and Decision Making*, Vol.1, 13-41, 2002.

[322] Yang L, and Liu B, On inequalities and critical values of fuzzy random variable, *International Journal of Uncertainty, Fuzziness & Knowledge-Based Systems*, Vol.13, No.2, 163-175, 2005.

[323] Yang L, and Liu B, A sufficient and necessary condition for chance distribution of birandom variable, *Information: An International Journal*, Vol.9, No.1, 33-36, 2006.

[324] Yang N, Wen FS, A chance constrained programming approach to transmission system expansion planning, *Electric Power Systems Research*, Vol.75, Nos.2-3, 171-177, 2005.

[325] Yao YY, Two views of the theory of rough sets in finite universes, *International Journal of Approximate Reasoning*, Vol.15, 291-317, 1996.

[326] Yazenin AV, Fuzzy and stochastic programming, *Fuzzy Sets and Systems*, Vol.22, 171-180, 1987.

[327] Yazenin AV, On the problem of possibilistic optimization, *Fuzzy Sets and Systems*, Vol.81, 133-140, 1996.

[328] Zadeh LA, Fuzzy sets, *Information and Control*, Vol.8, 338-353, 1965.

[329] Zadeh LA, Outline of a new approach to the analysis of complex systems and decision processes, *IEEE Transactions on Systems, Man and Cybernetics*, Vol.3, 28-44, 1973.

[330] Zadeh LA, The concept of a linguistic variable and its application to approximate reasoning, *Information Sciences*, Vol.8, 199-251, 1975.

[331] Zadeh LA, Fuzzy sets as a basis for a theory of possibility, *Fuzzy Sets and Systems*, Vol.1, 3-28, 1978.

[332] Zadeh LA, A theory of approximate reasoning, In: J Hayes, D Michie and RM Thrall, eds, *Mathematical Frontiers of the Social and Policy Sciences*, Westview Press, Boulder, Cororado, 69-129, 1979.

[333] Zhao R, and Liu B, Stochastic programming models for general redundancy optimization problems, *IEEE Transactions on Reliability*, Vol.52, No.2, 181-191, 2003.

[334] Zhao R, and Liu B, Renewal process with fuzzy interarrival times and rewards, *International Journal of Uncertainty, Fuzziness & Knowledge-Based Systems*, Vol.11, No.5, 573-586, 2003.

[335] Zhao R, and Liu B, Redundancy optimization problems with uncertainty of combining randomness and fuzziness, *European Journal of Operational Research*, Vol.157, No.3, 716-735, 2004.

[336] Zhao R, and Liu B, Standby redundancy optimization problems with fuzzy lifetimes, *Computers & Industrial Engineering*, Vol.49, No.2, 318-338, 2005.

[337] Zhao R, Tang WS, and Yun HL, Random fuzzy renewal process, *European Journal of Operational Research*, Vol.169, No.1, 189-201, 2006.

[338] Zhao R, and Tang WS, Some properties of fuzzy random renewal process, *IEEE Transactions on Fuzzy Systems*, Vol.14, No.2, 173-179, 2006.

[339] Zheng Y, and Liu B, Fuzzy vehicle routing model with credibility measure and its hybrid intelligent algorithm, *Applied Mathematics and Computation*, Vol.176, No.2, 673-683, 2006.

[340] Zhou J, and Liu B, New stochastic models for capacitated location-allocation problem, *Computers & Industrial Engineering*, Vol.45, No.1, 111-125, 2003.

[341] Zhou J, and Liu B, Analysis and algorithms of bifuzzy systems, *International Journal of Uncertainty, Fuzziness & Knowledge-Based Systems*, Vol.12, No.3, 357-376, 2004.

[342] Zhou J, and Liu B, Modeling capacitated location-allocation problem with fuzzy demands, *Computers & Industrial Engineering*, Vol.53, No.3, 454-468, 2007.

[343] Zhu Y, and Liu B, Continuity theorems and chance distribution of random fuzzy variable, *Proceedings of the Royal Society of London Series A*, Vol.460, 2505-2519, 2004.

[344] Zhu Y, and Liu B, Some inequalities of random fuzzy variables with application to moment convergence, *Computers & Mathematics with Applications*, Vol.50, Nos.5-6, 719-727, 2005.

[345] Zimmermann HJ, Description and optimization of fuzzy systems, *International Journal of General Systems*, Vol.2, 209-215, 1976.

[346] Zimmermann HJ, Fuzzy programming and linear programming with several objective functions, *Fuzzy Sets and Systems*, Vol.3, 45-55, 1978.

[347] Zimmermann HJ, Fuzzy mathematical programming, *Computers and Operations Research*, Vol.10, 291-298, 1983.

[348] Zimmermann HJ, Applications of fuzzy set theory to mathematical programming, *Information Sciences*, Vol.36, 29-58, 1985.

[349] Zimmermann HJ, *Fuzzy Set Theory and its Applications*, Kluwer Academic Publishers, Boston, 1985.

List of Acronyms

CCDP	Chance-Constrained Dynamic Programming
CCGP	Chance-Constrained Goal Programming
CCMLP	Chance-Constrained Multilevel Programming
CCMOP	Chance-Constrained Multiobjective Programming
CCP	Chance-Constrained Programming
DCDP	Dependent-Chance Dynamic Programming
DCGP	Dependent-Chance Goal Programming
DCMLP	Dependent-Chance Multilevel Programming
DCMOP	Dependent-Chance Multiobjective Programming
DCP	Dependent-Chance Programming
DP	Dynamic Programming
EVDP	Expected Value Dynamic Programming
EVGP	Expected Value Goal Programming
EVM	Expected Value Model
EVMLP	Expected Value Multilevel Programming
EVMOP	Expected Value Multiobjective Programming
GA	Genetic Algorithm
GP	Goal Programming
MLP	Multilevel Programming
MOP	Multiobjective Programming
NN	Neural Network
SOP	Single-Objective Programming

List of Frequently Used Symbols

x, y, z	decision variables
$\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$	decision vectors
$\tilde{a}$, $\tilde{b}$, $\tilde{c}$	fuzzy variables
$\tilde{\boldsymbol{a}}$, $\tilde{\boldsymbol{b}}$, $\tilde{\boldsymbol{c}}$	fuzzy vectors
ξ, η, τ	random, fuzzy, or hybrid variables
$\boldsymbol{\xi}$, $\boldsymbol{\eta}$, $\boldsymbol{\tau}$	random, fuzzy, or hybrid vectors
μ, ν	membership functions
ϕ, ψ	probability density functions
Φ, Ψ	probability distributions
f, f_i	objective functions
g, g_j	constraint functions
$\emptyset$	empty set
Pr	probability measure
Cr	credibility measure
Ch	chance measure
$\mathcal{M}$	uncertain measure
E	expected value
$(\Omega, \mathcal{A}, \Pr)$	probability space
$(\Theta, \mathcal{P}, \mathrm{Cr})$	credibility space
$(\Theta, \mathcal{P}, \mathrm{Cr}) \times (\Omega, \mathcal{A}, \Pr)$	chance space
$(\Gamma, \mathcal{L}, \mathcal{M})$	uncertainty space
α, β	confidence levels
d^+, d^-	positive and negative deviations
$\Re$	set of real numbers
$\Re^n$	set of n-dimensional real vectors
$\vee$	maximum operator
$\wedge$	minimum operator
lexmin	lexicographical minimization

Index