STUDIES IN *FUZZINESS*
AND *SOFT COMPUTING*

Studies in Fuzziness and Soft Computing

Baoding Liu

Theory and Practice of Uncertain Programming

2nd Edition

Springer

Baoding Liu

Theory and Practice of Uncertain Programming

Studies in Fuzziness and Soft Computing, Volume 239

Editor-in-Chief

Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

Baoding Liu

Theory and Practice of Uncertain Programming

2nd Edition

Springer

Author

Baoding Liu
Uncertainty Theory Laboratory
Department of Mathematical Sciences
Tsinghua University
Beijing 100084
China
E-Mail: liu@tsinghua.edu.cn

ISBN 978-3-540-89483-4 e-ISBN 978-3-540-89484-1

DOI 10.1007/978-3-540-89484-1

Studies in Fuzziness and Soft Computing ISSN 1434-9922

Library of Congress Control Number: 2008941016

The first edition appeared as volume 102 of Studies in Fuzziness and Soft Computing ISBN 3-7908-1490-3 Physica-Verlag Heidelberg, 2002.

Typeset & Cover Design: Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed in acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

To My Wife Jinlan

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.

For this new edition the entire text has been totally rewritten. More importantly, hybrid variable and hybrid programming are completely new.

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.

Acknowledgment

I am indebted to a series of grants from National Natural Science Foundation, Ministry of Education, and Ministry of Science and Technology of China. I would also like to thank Professor Janusz Kacprzyk for the invitation to publish this book in his series, and Dr. Thomas Ditzinger of Springer for his wonderful cooperation and helpful comments.

September 2008

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

Contents

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,

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

which maximizes a real-valued function f of $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ subject to a set of constraints $g_j(\boldsymbol{x}) \le 0$, $j = 1, 2, \cdots, p$.

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*

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

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*

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

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 $g_j(\boldsymbol{x}) \le 0$ is said to be active at a point $\boldsymbol{x}^*$ if $g_j(\boldsymbol{x}^*) = 0$. A point $\boldsymbol{x}^*$ satisfying $g_j(\boldsymbol{x}^*) \le 0$ 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,

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

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 [52] 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.,

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

where $f_i(\boldsymbol{x})$ are objective functions, $i = 1, 2, \cdots, m$, and $g_j(\boldsymbol{x}) \le 0$ 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*

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

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.,

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

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.,

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

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 [38] and subsequently studied by many researchers. GP can be regarded as a special