Studies in Fuzziness and Soft Computing

Chi-Bin Cheng
Hsu-Shih Shih
E. Stanley Lee

Fuzzy and Multi-Level Decision Making: Soft Computing Approaches

Second Edition

Springer

Studies in Fuzziness and Soft Computing

Volume 368

Series editor

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland
e-mail: kacprzyk@ibspan.waw.pl

The series “Studies in Fuzziness and Soft Computing” contains publications on various topics in the area of soft computing, which include fuzzy sets, rough sets, neural networks, evolutionary computation, probabilistic and evidential reasoning, multi-valued logic, and related fields. The publications within “Studies in Fuzziness and Soft Computing” are primarily monographs and edited volumes. They cover significant recent developments in the field, both of a foundational and applicable character. An important feature of the series is its short publication time and world-wide distribution. This permits a rapid and broad dissemination of research results. Contact the series editor by e-mail: kacprzyk@ibspan.waw.pl

More information about this series at http://www.springer.com/series/2941

Chi-Bin Cheng · Hsu-Shih Shih
E. Stanley Lee

Fuzzy and Multi-Level Decision Making: Soft Computing Approaches

Second Edition

Springer

Chi-Bin Cheng
Department of Information Management
Tamkang University
New Taipei City, Taiwan

Hsu-Shih Shih
Department of Management Sciences
Tamkang University
New Taipei City, Taiwan

E. Stanley Lee
Department of Industrial
and Manufacturing Systems
Engineering
Kansas State University
Manhattan, KS, USA

ISSN 1434-9922 ISSN 1860-0808 (electronic)
Studies in Fuzziness and Soft Computing
ISBN 978-3-319-92524-0 ISBN 978-3-319-92525-7 (eBook)
https://doi.org/10.1007/978-3-319-92525-7

Library of Congress Control Number: 2018954857

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

Decision making under a hierarchical structure is commonly found in the real world, yet the optimization of such decisions is typically difficult due to the highly inter-dependent nature among the decisions by the decision-makers at different levels of the hierarchical organization. Problems in such a domain are referred to as multi-level decision making. The basic concept of multi-level decision making is that an upper-level decision-maker sets his or her goal and/or decision and then asks each subordinate level of the organization for their decisions. The decisions of the lower levels are then submitted and modified by the upper level with consideration of the overall benefits of the organization. This mutually interactive process is continued until reaching a solution, which is satisfactory to all the decision-makers. Apparently, the degree of interaction and the degree of satisfaction depend on the management style of the upper level. This decision-making process is extremely useful to the hierarchy decentralized organizations that are pervasive in various industries.

To solve the multi-level decision making, the problems are typically modeled by multi-level programming, which contains a set of nested optimization problems over a single feasible region and the control of the decision variables is partitioned among the levels where one decision variable may impact the objective of several levels. There have been many traditional approaches proposed for solving the multi-level programming problems, such as the decomposition principle, goal programming, multi-objective programming, and game theory. However, almost all of these traditional approaches cannot meet the common features, the interactive nature in particular, of the decision process of a multi-level decentralized organization. By contrast, soft computing approaches use a collection of algorithms to find inexact solutions to computationally difficult tasks for which conventional methods do not yet provide low cost and time-feasible solutions. Approaches in this category such as fuzzy logic can facilitate the implementation of interactive decision making among levels.

The principal constituents of soft computing are fuzzy logic, artificial neural networks, meta-heuristics, etc. In order to increase the computational efficiency of the basic multi-level programming algorithms, fuzzy set theory that suggests a completely different philosophy of exploring the typical fuzziness, vagueness, or the not-well-defined nature of a large decentralized hierarchy organization has been applied to solve the problems. The resulting fuzzy interactive sequential approach appears to be a useful and efficient one. The advantages of the fuzzy approaches are that not only the computational requirements are reduced tremendously, but the representation of the system is also more realistic.

A particular type of artificial neural networks, recurrent neural networks such as Hopfield network, with their dynamic learning capability appears to be a suitable tool to cope with the dynamic nature of multi-level programming problems. Hybrid neural network approaches that combine neural networks with meta-heuristics have also be proposed to solve the bi-level programming problems. Meta-heuristics are a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms. Typical meta-heuristics, such as genetic algorithms, simulated annealing, and swarm particle optimization, are widely used to solve discrete or nonlinear optimization problems. Meta-heuristic algorithms do not use gradient information and thus are less likely to be trapped in a local optimum. Their structures also enable the implementation of parallel search of the solution space and thus are able to improve the efficiency of the solution procedure.

This book can be divided approximately into five subjects. The first subject, including Chaps. 1 and 2, summarizes the solution approaches of multi-level programming and introduces the classical approaches for solving the problems. The emphasis is on the numerical solution aspects, and no theoretical treatment is included.

The second subject, which includes Chaps. 3–6, presents knowledge representation and fuzzy decision making. For the frequent use of linguistic expressions in the interactions between the various levels of management in a hierarchy organization, emphasis is placed on linguistic representation by the use of fuzzy concepts. The major content of this part is reproduced from our previous book, entitled *Fuzzy and Multi-level Decision Making: An Interactive Computational Approach*, published by Springer.

The third subject (Chap. 7) presents the use of auction mechanism in solving the bi-level programming problems, where the managers make resources allocation decisions in a heterogeneous and distributed fashion. The fourth subject (Chap. 8) summarizes the three popular meta-heuristics, genetic algorithm, particle swarm optimization, and tabu search, and their applications to solving the multi-level programming problems. The final subject (Chap. 9) introduces the use of Hopfield networks in solving optimization problems and their application to bi-level programming problems.

Several examples and algorithms are adopted from the original publications as acknowledged in the text. In particular, we are grateful for the following permissions from the copyright owners: Elsevier, Fig. 6.1, Figs. 7.1–7.4, Tables 7.1–7.5,

Example 7.1, Figs. 9.1–9.5, Table 9.1, Examples 9.1 and 9.2; and Inderscience, Sect. 8.2.

We wish to express our appreciation to the co-workers of our previous studies, Dr. Ue-Pyng Wen, Dr. Young-Jou Lai, Dr. Kuen-Ming Lan, Mr. Han-Chyi Hsiao, and Mr. Kun Chan, for their permissions of using the materials in this book.

New Taipei City, Taiwan Chi-Bin Cheng
New Taipei City, Taiwan Hsu-Shih Shih
Manhattan, USA E. Stanley Lee

Contents

Chapter 1
Introduction

The multiple level optimization model (Bard 1983c; Bard and Moore 1990; Ben-Ayed and Blair 1990; Bialas and Karwan 1982; Bialas and Karwan 1984; Hansen et al. 1992; Judice and Faustino 1992) is a practical and useful tool for solving decision problems in a hierarchical system. The model solves problems where the decision is executed in a top-to-lower level sequential manner and where the lower-level decision-makers have freedom to make decisions within a broad range that is set by senior managers or decision-makers. The use of the approach is essentially restricted to dual- or multi-ploy type decision processes, such as those that are encountered in economic systems.

In the multi-level model, the system explicitly assigns each agent a unique objective and a set of decision variables, as well as a set of common constraints that affect all agents (Anandalingam and Friesz 1992). There are four common features of this multi-level organization: (a) interactive decision-making units exist within a predominantly hierarchical structure; (b) execution of decisions is sequential, from top level to bottom level; (c) each unit independently maximizes its own net benefits, but is affected by the actions of other units through externalities and (d) the external effect on a decision-maker's problem is reflected in both the objective function and the set of feasible decision spaces. The principle of multi-level programming technique relies on an upper-level decision maker (DM) setting a goal and/or decisions and then each subordinate level of the organization gives the optima, which are calculated in isolation. Lower-level decision makers decisions are then submitted and modified by the upper-level DM after consideration of the overall benefits for the organization. The process is continued until a satisfactory solution is reached (Bard 1983a; Burton 1977). This decision-making process is extremely useful in de-centralized systems, such as agriculture, government policy-making, economic systems, finance, warfare, transportation and network design, and is suitable for conflict resolution (Anandalingam 1988; Anandalingam and White 1990; Ben-Ayed 1993; Kim and Suh 1988; Migdalas 1995; Shimizu et al. 1997; Vicente and Calamai 1994; Wen and Hsu 1991).

C.-B. Cheng et al., *Fuzzy and Multi-Level Decision Making: Soft Computing Approaches*, Studies in Fuzziness and Soft Computing 368,
https://doi.org/10.1007/978-3-319-92525-7_1

Many approaches have been proposed for solving multi-level programming problems (MLPP's), and bi-level problems in particular. However, traditional programming techniques, such as the decomposition principle (in linear programming), goal programming, multi-objective programming, or game theory, do not fully incorporate the common features of multi-level systems (Wen and Hsu 1991). Therefore, many heuristic methodologies have been proposed to solve MLPP's over the last three decades. Most methods for linear cases use an extreme point search and transformation methods. The former determines a compromise vertex using a simplex algorithm that adjusts the control variables at the higher level. This type of strategy is inefficient, especially for large problems (Wen 1981). Although Bard (1983b) devised a short-cut to overcome the issued with large problems, it is an ad hoc solution so it lacks generality (Ben-Ayed and Blair 1990; Haurie et al. 1990).

Transformation methods involve transforming lower-level programming problems as the constraints for the higher level using Karush-Kuhn-Tucker (KKT) conditions or other functions (Anandalingam and White 1990; Wen and Hsu 1991). Because the use of non-linear terms in constraints, auxiliary problems often become complex and sometimes unmanageable. Gradient-based methods or other searching heuristics for non-linear, multi-level programming address the problem of the nonlinearity that results from the transformation (Kolstad 1985; Kolstad and Lasdon 1990; Marcotte 1986). However, due to the non-convex property, optimality cannot be guaranteed, even when the search seems exhausted.

Presently, in terms of multi-level programming, only bi-level programming problems (BLPPs), bi-level decentralized programming problems (BLDPPs), three-level programming problems (TLPPs) and three-level de-centralized programming problems (TLDPPs) are soluble. However, even for the simplest case (BLPPs with linear coefficients), they are still categorized as non-convex programming and were proven to be NP-hard by Ben-Ayed and Blair (1990). In order to solve multi-level programming problems efficiently, some approaches using soft computing have been proposed in recent years. Soft computing became a formal area of study in information, computing and decision science in the early 1990s (Zadeh 1994). According to L.A. Zadeh, the guiding principle of soft computing is:

> Exploit the tolerance for imprecision, uncertainty, partial truth, and approximation to achieve tractability, robustness and low solution cost.

Soft computing uses a collection of algorithms to find inexact solutions to computationally difficult tasks, such as the solution of NP-complete problems, for which conventional methods do not yet provide low cost and time-feasible solutions. The principal constituents of SC areFuzzy Logic (FL), Neural Networks (NN), Evolutionary Computation (EC), Meta-heuristics, Machine Learning (ML) and Probabilistic Reasoning (PR).

This study uses soft computing approaches to solve multi-level programming problems. In particular, approaches involving fuzzy interactive decision-making process, Hopfield neural networks, meta-heuristic algorithms and an interactive auction mechanism that incorporates the former three approaches are discussed in detail. Fuzzy interactive decision-making is generally applicable in large hierarchy organi-

zations, which are characterized by mutual interactions in a top-to-bottom sequence and by the vague and ill-defined nature of an organization with a large hierarchy. Recurrent neural networks, such as Hopfield neural networks (HNNs) (Hopfield 1982), self-organizing feature maps (Kohonen 1982) and Boltzmann machines (Ackley et al. 1985), are used to solve the optimization problem. HNNs and their extensions have been developed for almost all classes of optimization problems. Shih et al. (2004) used the network of Hopfield and Tank (1985) to solve a multi-level programming problem, whereby the original problem is converted to a system of nonlinear differential equations using an energy function and Lagrange multipliers. A meta-heuristic is a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms. Typical meta-heuristics, such as genetic algorithms (GA), simulated annealing (SA) and particle swarm optimization (PSO), are widely used to solve discrete or nonlinear optimization problems. They are especially suited to NP-hard problems, such as multi-ploy problems. A Tabu search (Glover 1986) is used to solve both a linear BLPP (Gendreau et al. 1996) and a mixed-integer linear BLPP (Wen and Huang 1996). Mathieu et al. (1994) proposed the use of simulated annealing to solve this NP-hard problem. Liu (1998) illustrated the use of a genetic algorithm to solve the Stackelberg-Nash equilibrium for multi-level programming problems. In recent years, auction mechanisms have been applied to solve decision-making problems in a heterogeneous and distributed decision environment. Auction formats, particularly a reverse auction and a combinatorial auction, are used to solve multi-level programming problems (Cheng 2011; Cheng and Lo 2017).

The remainder of this chapter gives brief reviews of hierarchical decision-making, bi-level and multi-level programming, as well as the use of fuzzy interactive decision-making, neural networks, meta-heuristics and auction mechanism in solving multi-level programming problems. Chapter 2 of this book introduces classical approaches for solving different types of multi-level programming problems. Chapter 3 discusses the concepts and theorems of possibility measure and fuzzy sets theory, which form the theoretical basis of fuzzy interactive decision-making. Chapters 4 and 5 present the use of fuzzy interactive decision making in solving various types of multi-level programming problems, while Chap. 6 solves a bi-level programming problem by fuzzy dynamic programming. Chapter 7 introduces the auction theory and demonstrates its application to bi-level decentralized problems. Chapter 8 introduces the use of neural networks in solving mathematical programming problems and their applications to multi-level programming problems. Chapter 9 introduces a few popular meta-heuristic algorithms and presents the use of these algorithms to solve bi-level programming problems.

1.1 Decision Making in Hierarchical Systems: Multi-ploy Versus Interactive Decisions

For simplicity, the discussion is initially restricted to a two-level hierarchical system. The decision-maker (DM) at the top or first level is designated as the leader and the DM at the second or lower level, the follower. Depending on the degree of interaction or cooperation between the two levels, various decision-making processes can be formulated. At one extreme, the objective of the follower is in direct opposition to the objective of the leader so the problem is reduced to the classical max-min problem. At the other extreme, the follower is in complete agreement with the leader. However, most problems that are encountered in practice fall between these two extremes. This study is concerned with these practical problems that lie between the extremes, for which the two most important processes are: (1) a decision process that is similar to the static two person Stackelberg game and (2) the interactive decision-making process. For convenience of discussion, the first type of decision process is referred to as a duo-ploy decision-making process. The corresponding decision making process for a multi-level hierarchical system is referred to as multi-ploy decision making. As well as the practical reasons mentioned above, the multi-ploy process is also separated because of historical developments. Many algorithms that are known as bi-level programming problem (BLPP) algorithms have been developed to solve these duo-ploy processes.

A process with two levels, or the duo-ploy system is used to illustrate the characteristics of the multi-ploy process. The decision for a typical duo-ploy process is carried out in the following manner. The upper-level DM, the leader, initially makes a decision with full information about both levels and then the second level DM, the follower, makes a decision in isolation, based on the first level's decision. In most practical cases, the second-level decision is the final decision and there is no time or energy for the upper level to make another decision afterward. Notice that although the second level cannot control the decision at the first level, the final decision at the lower level does eventually influence the upper level and the overall result. There are many practical examples of this game type of interaction, such as the traffic planning problem (Migdalas 1995), where the planner or the decision-maker, the leader, seeks to improve the performance of the traffic network and at another level, the network users, the followers, make choices with regard to the details of their travel based on the network that is formed by the planner. Another example is the taxation problem, where the government, the leader, decides the taxation rules and rates and the taxpayers, the followers, try to minimize their payments based on these rules or rates. A third example is the pricing and purchasing of fertilizers within an agricultural region (Fortuny-Amat and McCarl 1981). Even the issue of setting penalties for illegal drug importation can be formulated in terms of the duo-ploy problem (Candler and Townsley 1982). Triple- or multi-ploy problems, although much more difficult to solve, also occur frequently in practice. One example is the allocation of tax money by the federal government to the various states, which in turn allocate to different cities and these allocate funds different project managers. Notice that, in all of these

examples, the leader must consider both the immediate gains and the influences of the decision on the followers, whose final decision ultimately influences the gains for the leader and for the overall system. In this situation, the follower's decisions are usually final. In actual practice, the planner or the policy maker cannot change the decision after examining the followers' decision. It is true that the network or the policy could be changed, but in practice, these changes are seldom immediately enacted.

As well as these examples, multi-ploy type structures are found in areas as diverse as economic systems, ecology and environmental studies, biology and chemical engineering, network design, transportation, game theory, databases and the theory of classification. A multi-ploy strategy for decision-making can seldom be completely adapted without change by a hierarchical organization, such as a manufacturing or a service company. In general in a multi-hierarchical company, when the followers or the divisions of the company make their decisions based on the leader's decision, the divisions must submit their decisions and these are then modified by the upper-level DM to maximize the overall benefits for the organization. Ideally, this mutually interactive decision process is continued until a satisfactory solution or compromise is reached. This continuously interactive process is referred to as an interactive decision-making process. Depending on the management style, the degree of cooperation is different for different companies. This typed of decision-making process is eminently practical and is applicable in various de-centralized large hierarchical companies. Unfortunately, this practical decision making approach cannot be solved easily, even for a bi-level organization. This study uses recently developed soft computing methods to solve this multi-level interactive decision problem. Although the decision making process in a hierarchical organization, such as an organization in the service or manufacturing industries, is usually not exactly like a multi-ploy decision process, it is also seldom completely controlled by senior management. In general, senior management is more concerned with the overall strategy, the market share and compatibility, and not with the detail of day-to-day activities, so it rarely exercises direct control over its divisions, although its general policy influences and partially controls the decision processes for the divisions. The divisional managers can make independent decisions, as long as these are not in conflict with those of senior management. Because there is no central decision-maker, different degrees of interaction between the various hierarchical levels are possible, so that overall organization profit is maximized.

1.2 Bi-level and Multi-level Programming

As mentioned previously, the traditional method of solving duo-ploy and multi-ploy decision problems is to respectively define them as bi-level and multi-level programming problems. Bracken and McGill (1973) defined this hierarchical decision problem as a generalized mathematical programming problem and various studies have since used this multiple level programming problem. In order to summarize the

results, it is convenient to classify the problem in terms of both the structural complexity and the method for the solution. Structurally, this programming problem can be classified as a bi-level programming problem (BLPP), a bi-level de-centralized programming problem (BLDPP), a multi-level programming problem (MLPP) and a multi-level de-centralized programming problem (MLDPP). A de-centralized problem has multiple divisions in each level, except the highest level, and a BLPP or a MLPP has only one division in each level. A MLPP can be further subdivided into a three-level programming problem (TLPP), a four-level programming problem or other levels. Classical solution methods to solve the various types of multi-level programming problems are classified into three categories, as shown in Table 1.1. The methods in Table 1.1 are relatively mature and use traditional optimization or decision concepts. The first two categories—extreme-point search and transformation approaches—are solely used to solve linear or linear-quadratic bi-level programming problems using traditional optimization concepts. Some of these can also be extended to linear BLDPP and TLPP. The principle of an extreme-point search is to determine a compromise vertex using a simplex algorithm by adjusting the control variables. The transformation approach transforms the lower-level problems into constraints for the higher level using techniques such as the Karush-Kuhn-Tucker (KKT) conditions or penalty functions. This transformed problem becomes a single-level mathematical programming problem. Because there are non-linear terms, the reformulated problem in the transformation approach becomes complex, non-linear and difficult to solve numerically. However, several relatively efficient numerical iterative techniques have been proposed. Although Categories I and II can be extended or modified for non-linear problems, Category III is solely for discrete or non-linear BLPPs and uses existing search or heuristic decision-making approaches, such as a gradient technique, a cutting-plane algorithm, or branch-and-bound heuristics. In general, the problem is non-convex, so optimality cannot be guaranteed, even when the search procedure is exhausted. Some typical examples of each category are listed in Table 1.1. This list is certainly not complete. Most of the computational algorithms are used for the simplest linear BLPPs, but even for this simplest problem, there exist no simple solutions. It has been proven that even for the simplest two-level linear case, the problem is strongly NP-hard. The geometric properties of the bi-level linear problem are also much more complicated than the usual mathematical programming problem so the set of feasible solutions is non-convex and non-unique. Another difference is that no general hypothesis or cost function that guarantees a Pareto optimal can be obtained, even for linear BLPPs, unless both the upper and lower objectives coincide. In that case, both DMs completely cooperate and the result is an optimal solution for the BLPP that is a Pareto optimal. Therefore, the solutions of a BLPP or a MLPP are, in general, not Pareto optima. It is difficult to obtain an effective solution for multi-ploy problems for two reasons: the complexity due to interactions between the various DMs at the various levels and the implicit nature of these interactions. Top-level problems are embedded into the lower levels so most optimization or decision-making techniques are too restrictive to be sufficiently flexible. Because lower level decision-makers have reduced freedom, there exists no simple solution to the overall problem.

Table 1.1 Classical methods for bi-level and multi-level programming

Category	Method	References
I. Extreme point search	*k*th-best algorithm	Bialas and Karwan (1982, 1984) Wen and Bialas (1986)
	Grid-search algorithm	Bard (1982, 1983b)
II. Transformation method	Complementary pivot	Judice and Faustino (1988. 1992) Bialas and Karwan (1984)
	Branch-and-bound	Bard and Falk (1982) Bard and Moore (1990) Hansen et al. (1992)
	Penalty function	Anandalingam and White (1990) Aiyoshi and Shimizus (1981a, b, 1984) Anandalingam and Apprey (1991)
III. Descent and heuristic methods	Descent method	Savard and Gauvin (1994)
	Branch-and-bound	Bard and Moore (1990)
	Cutting plane	Bard (1984b)

1.3 Characteristics of Duo-ploy Systems

To determine the basic characteristics of a duo-ploy game type or a leader-follower decision process, consider the following bi-level problem:

$$\max_{x_1} f_1(x_1, x_2) = c_{11}^T x_1 + c_{12}^T x_2 \quad \text{(Upper level)} \tag{1.1}$$

where x_2 solves:

$$\max_{x_2} f_2(x_1, x_2) = c_{21}^T x_1 + c_{22}^T x_2 \text{ (Lower level)}$$

subject to:

$$\begin{aligned} A_1 x_1 + A_2 x_2 &\le b \\ x_1 &\ge 0 \\ x_2 &\ge 0 \end{aligned} \tag{1.2}$$

wherein A_1 and A_2 are $m \times n_1$- and $m \times n_2$-dimensional matrices, respectively, c_{11}, c_{21} and x_1 are n_1-dimensional vectors, c_{12}, c_{22} and x_2 are n_2-dimensional vectors and b is an m-dimensional vector, f_1 and f_2 are objective functions that are respectively optimized by the leader and the follower and x_1 and x_2 are decision variables that are respectively controlled by the leader and the follower. The decision-making process is enacted in a leader to follower sequence. The leader initially makes the decision

and then the follower must make a decision based on the leader's decision, or based on x_1, which has already decided by the leader. The game is static in the sense that the follower's decision is final and the leader cannot make another decision after the follower decides.

The problem can also be interpreted in another way. Since x_2 is a function of x_1, the problem can be formulated as a parametric programming problem, in which the follower must solve a problem that is parameterized by the leader. To determine the basic characteristics of the problem, the following generally used definitions are necessary:

The follower's feasible region with fixed $x_1 \in X_1$:

$$\Omega(x_1) = \{x_2 | x_2 \rangle 0,\ A_1 x_1 + A_2 x_2 \leq b\} \tag{1.3}$$

The follower's rational reaction set:

$$\Psi(x_1) = \left\{x_2 | x_2 \text{ solves: } \max_{x_2}[f_2(x_1, x_2) \text{ with } x_2 \in \Omega(x_1)]\right\} \tag{1.4}$$

The inducible region:

$$\text{IR} = \{(x_1, x_2) | x_1 \in \text{X}_1, x_2 \in \Psi(x_1)\} \tag{1.5}$$

The follower's rational reaction set is a response to a fixed x_1 and is the best decision set for the follower only. The inducible region is different to the feasible region of the BLPP in that the IR must be contained within the optimal set for the follower. The IR is a subset of the feasible set of the problem. The IR also represents the set over which the leader can optimize if the leader has the control over all of the variables. Because there is a leader-follower decision sequence, the problem is much more complicated. For example, even if it is assumed that: (i) the feasible region of the BLPP is non-empty and bounded and (ii) $\Omega(x_1) \neq 0$, so for each decision by the leader, the follower has a feasible response, the problem can still be difficult. This is principally because the leader cannot respond after the follower decides. If the follower's rational reaction set is not a singleton, then different values of x_2 can produce different values for the leader's objective function. It is most probable that not all of the different values of x_2 maximize f_1. Most algorithms that use the Karush-Kuhn-Tucker condition to imbed the lower problem into the leader's problem exploit the set of the inducible region. However, if the algorithm strictly follows the decision sequence or the algorithm cannot account for the case whereby the follower's rational reaction set is not a singleton, this problem of multiple solutions for the lower decision maker is crucial.

1.4 Characteristics of Duo-ploy Systems with Multiple Followers

For a duo-ploy system with multiple followers, which is also known as a BLDPP, the problem is even more complicated. For a non-linear BLDPP with p followers:

$$\begin{aligned} &\max_x F(x, y_1, y_2, \ldots, y_p) \\ &\text{subject to:} \\ &\quad G(x) \leq 0 \\ &\text{where each } y_i,\ i = 1, 2, \ldots, p, \text{ solves:} \end{aligned} \tag{1.6}$$

$$\begin{aligned} &\max_{y_i} f_i(x, y_1, y_2, \ldots, y_p) \\ &\text{subject to:} \\ &\quad g(x, y_1, y_2, \ldots, y_p) \leq 0 \end{aligned} \tag{1.7}$$

wherein x is the control variable for the leader, y_1, y_2,..., y_P are the controls for the p followers and f_i is the objective function for the i-th follower. It is assumed that the followers know the leader's strategy and also that the followers simultaneously reveal their strategies to each other. When the leader and other followers reveal their strategies, the reaction that is the i-th follower's optimal solution, y_i^*, is obtained by solving:

$$\begin{aligned} &\max_{y_i} f_i(x, y_1, y_2, \ldots, y_p), i = 1, 2, \ldots, p \\ &\text{subject to} \\ &\quad g(x, y_1, y_2, \ldots, y_p) \leq 0 \end{aligned} \tag{1.8}$$

Equation (1.8) can have multiple solutions. Using the definition from the previous section, these solutions are known as the rational reaction set for the i-th follower. Since all the followers have equal status and each follower optimizes his or her own objective individually, the Nash equilibrium solution, which obeys the following optimality condition, is obtained:

$$\begin{aligned} &f_i(x, y_1^*, \ldots, y_{i-1}^*, y_i, y_{i+1}^*, \ldots, y_p^*) \leq f_i(x, y_1^*, \ldots, y_{i-1}^*, y_i^*, y_{i+1}^*, \ldots, y_p^*) \\ &\quad \text{for } i = 1, 2, \ldots, p. \end{aligned} \tag{1.9}$$

The array $(y_1^*, \ldots, y_{i-1}^*, y_i^*, y_{i+1}^*, \ldots, y_p^*)$ is known as the Nash equilibrium with respect to x. In general, a Nash equilibrium is not unique and is also not efficient in the same sense as a Pareto optimum, so any one of the followers can move to a better position by deviating from the Nash equilibrium. For a Nash equilibrium, only the followers obtain their best solutions with respect to any given $x \in X$. The Stackelberg-Nash equilibrium is the array $(x^*, y_1^*, \ldots, y_{i-1}^*, y_i^*, y_{i+1}^*, \ldots, y_p^*)$, which also contains the

leader's optimum. In other words, the Stackelberg-Nash equilibrium is essentially the optimal solution for a BLDPP problem.

1.5 Soft Computing for Multiple Level Programming

In order to simplify the solution procedure and to provide efficient algorithms for multiple level programming problems, soft computing approaches such as fuzzy decision making, neural networks, meta-heuristics and auction mechanisms have been used to solve the problems. This section highlights some of the applications of these approaches to multiple level programming problems.

1.5.1 Fuzzy Interactive Decision-Making

As detailed in an earlier section, most of the classical algorithms can solve the simplest bi-level problems of the duo-ploy type and none are computationally efficient for large hierarchical organizations. They are not even very efficient for bi-level problems. A completely different approach that explores the typical fuzziness, vagueness, or the ill-defined nature of a large hierarchical organization using fuzzy set theory was proposed by Lee and coworkers (Shih et al. 1996; Shih and Lee 1999). The resulting fuzzy interactive sequential approach has been proven to be powerful and can be used by decision-makers to solve practical problems that are encountered in large de-centralized companies. The advantages of the proposed approach are two-fold: the problem becomes much more simplified so it can be solved reasonably easily for fairly large practical problems and the representation of the original problem is not only simplified, but also much more realistic. In other words, since real-world problems for large organizations are generally fuzzy or not well defined, the current classic algorithms are trying to solve a problem that does not exist by assuming unrealistically accurate models and by ignoring the inherent fuzziness of large organizations. The fuzzy interactive approach combines the concepts of fuzzy tolerance membership functions and multiple objective decision-making. A system is modeled approximately and there then follows an iterative process of improvement or a supervised search to obtain a better model. The higher-level DM initially defines the objectives and decisions and tolerances are represented by fuzzy membership functions. This tolerance information for the upper-level constrains the lower-level DM's feasible space. The lower-level DMs also form individual fuzzy membership functions, based on a specific tolerance. A search is then used to sequentially improve the membership functions, which represent the degrees of satisfaction for both the upper- and lower-level decision-makers. In order to satisfy all of the DMs, multiple-objective programming or other interactive approaches are used to obtain satisfactory membership functions. Unlike the extreme-point search procedure, the fuzzy approach does not presume that an optimal solution is at a corner point. Since

a true optimum cannot be defined easily because there are interactions, there is no reason to assume that the optimum is at a corner point. The optimum solution for a multiple-objective problem is a non-dominant solution, so a compromise solution that is acceptable to all of the DMs is obtained. The fuzzy approach is very efficient and does not increase the complexity or the size of the original problem. Since its solution process is very robust, it can be extended easily to multi-level problems with non-linear parameters that are convex.

In addition to the interactions between DMs in a large, de-centralized and hierarchical organization, vagueness or poor definition results from the frequent use of linguistic terms. It is common for the DM to describe current earnings or the demand for a certain product as "*extremely good*, *good*, *average*, *bad*, or *extremely bad*;" or, to give directions such as "*put more emphasis on the environment and worker's safety*". Fuzzy set theory is ideally suited to represent and manipulate these linguistic modifiers.

1.5.2 The Use of Neural Networks to Solve Multiple Level Programming Problems

HNNs and their extensions have been developed for almost all classes of optimization problems. Kennedy and Chua (1988) extended Tank and Hopfield's work to solve a non-linear programming (NLP) problem. Rodríguez-Vázquez et al. (1988) used switched-capacitor NNs to solve LP and NLP problems. Zhang and Constantinides (1992) used Lagrange NNs to solve general NLP problems and Cichocki and Unbehauen (1993) proposed a neural network that uses a Lagrange multiplier to solve a NLP problem. Xia et al. (2005) proposed a primal and dual method for solving linear and quadric programming problems.

HNNs consist of a set of neurons and a corresponding set of unit delays, which form a multiple loop feedback system. The output of each neuron is fed back, via a unit delay element, to each of the other neurons in the system. Each neuron is fully connected to the remaining neurons by weights, which are constantly updated until an energy function is minimized. The energy function characterizes the behavior of the network and directs the search to solutions that allow real-time optimization. The function is the key to bridging the HNNs and the optimization problems.

Shih et al. (2004) used the approach that was proposed by Hopfield and Tank (1985) to solve multiple level programming problems. The optimization problem is converted to a system of nonlinear differential equations using an energy function and Lagrange multipliers. To solve the resulting differential equations, a steepest descent search technique is used. Lan et al. (2007) also proposed a hybrid algorithm that combines the network of Rodríguez-Vázquez (1988) with a tabu search to solve bi-level programming problems. The bi-level programming is first transformed using KKT conditions and the complementary slackness conditions are then reformulated

using the 0–1 vector that was proposed by Fortuny and McCarl (1981). The resulting mixed 0–1 integer programming problem is finally solved using the hybrid algorithm.

1.5.3 Meta-heuristic Algorithms for Multiple Level Programming

Meta-heuristic algorithms are generally to search for solutions to multi-level programming problems. The algorithm is either used to search for a solution in the transformed single level problem or to search individual level problems separately. Depending on the strategies that are used, this category of methods can be classified as a nested sequential approach (Li et al. 2006), a single-level transformation (Bard and Falk 1982; Aiyoshi and Shimizu 1984), a multiple objective approach (Fliege and Vicente 2006), or a co-evolutionary approach (Oduguwa and Roy 2002; Legillon et al. 2012). The first and the last strategies solve the upper and the lower problems separately using metaheuristics and the second and the third strategies involve transforming the original problem to a single-level problem or to a multiple objective problem.

1.5.4 The Use of Auction Mechanisms for Multiple Level Programming

In recent years, auction mechanisms have been used to solve bi-level (decentralized) programming problems. Cheng (2011) modeled business procurement as a bi-level decentralized programming problem and used a reverse auction mechanism to solve the problem. Reverse auctions are simply traditional auctions in reverse (Smart and Harrison 2002). In traditional (i.e. forward) auctions, a seller offers a product or service for sale to the highest bidder. In a reverse auction, a buyer invites suppliers to tender to supply a specific quantity of goods or services on a particular date. In this BLDDP, the buyer is the upper level decision-maker and the multiple suppliers are at the lower level, making decisions independently of each other. The negotiation process between the buyer and the suppliers involves the iterative exchange of decision information between the upper and lower levels of the model. A fuzzy-based algorithm establishes the outcome that allows a satisfactory compromise between the objectives of the decision-maker at the upper-level and the objectives of the decision-makers at the lower level. Cheng et al. (2011) solved the same problem using the same BLDDP formulation, but embedded a genetic algorithm that establishes the optimum decision at the upper level.

Cheng and Lo (2017) modeled a resource allocation and scheduling problem for multiple projects as a bi-level de-centralized programming problem, wherein a higher-level manager determines the distribution of resources to multiple projects and

individual project managers must optimize the output of the project. For this problem, activities in a project can be executed in alternative modes that involve the use of pre-determined combinations of resources and corresponding durations for specific activities. Each execution mode is a combination of the resource requirements for the activity. This characteristic of multiple mode project scheduling embodies the concept of a combinatorial auction (CA), whereby bidders are allowed to submit bids for combinations of items.

A solution procedure that uses combinatorial auction mechanisms is used to determine the allocation of resources to projects, with the higher-level manager serving as the auctioneer and the project managers at the lower level bidding for resources.

Chapter 2
Linear Bi-level Programming

As has been pointed out in Chap. 1, most of the developments on multiple-level programming problem (MLPP) start with this structurally simplest bi-level programming problem (BLPP). Another reason is that many decision problems encountered in practice can be formulated as the duo-ploy type decision problem. Thus, various algorithms have been developed for solving these practical problems. Obviously, it is impossible to cover all the algorithms. Only several typical algorithms to illustrate the approaches will be discussed in this chapter. We shall restrict our discussions to the various numerical solution algorithms and avoid any theoretical treatments. Only linear and non-fuzzy approaches or Categories I and II in Table 1.1 will be discussed in this chapter and fuzzy interactive approaches, which form the main aim of this book, will be discussed in detail in later chapters. Non-linear and other MLPPs will be discussed in Chap. 3.

2.1 Linear Bi-level Programming Problems

BLPP is usually viewed as a problem with two DMs at two different hierarchical levels. The upper-level decision maker, the leader, selects his or her decision vector first and the lower decision maker, the follower, select his or her afterward based on the decisions of the upper level. The leader knows the functions of the follower, who may or may not know the functions of the leader. This duo-ploy type bi-level problem has been studied extensively in the field of economics, where social objectives versus the objectives of the individual economic agents form a typical Stackelberg's leader-follower duo-ploy model. Thus, the bi-level linear programming problem is often viewed as a special case of two-person, non-zero sum non-cooperative game in which one DM, the leader, has the ability to enforce his/her strategy on the follower. The problem forms a nested optimization model involving two problems, an upper one and a lower one. In a way, the upper-level decision is imbedded in the lower-level decision. The main problem is the aggregation of these imbedded two problems. To

C.-B. Cheng et al., *Fuzzy and Multi-Level Decision Making: Soft Computing Approaches*, Studies in Fuzziness and Soft Computing 368,
https://doi.org/10.1007/978-3-319-92525-7_2

formulate the BLPP, let us introduce the decision vectors, x_1 and x_2, where the top-level DM has control over the vector x_1 and the lower-level DM has control over the vector x_2. We shall assume that the performance functions, f_1 and f_2, for the two DMs are linear and bounded. The BLPP can be stated as:

$$\begin{aligned} &\max_{x_1} f_1(x_1, x_2) = c_{11}^{\mathrm{T}} x_1 + c_{12}^{\mathrm{T}} x_2 \quad \text{(Upper level)} \\ &\text{where } x_2 \text{ solves:} \\ &\max_{x_2} f_2(x_1, x_2) = c_{21}^{\mathrm{T}} x_1 + c_{22}^{\mathrm{T}} x_2 \quad \text{(Lower level)} \\ &\text{subject to:} \end{aligned} \tag{2.1}$$

$$(x_1, x_2) \in X = \{(x_1, x_2) | A_1 x_1 + A_2 x_2 \leq b, \text{ and } x_1, x_2 \geq 0\}, \tag{2.2}$$

where A_1 and A_2 are $m \times n_1$- and $m \times n_2$-dimensional matrices, respectively; c_{11}, c_{21} and x_1 are n_1-dimensional vectors; c_{12}, c_{22} and x_2 are n_2-dimensional vectors; and b is an m-dimensional vector.

It should be noted that, in general, x_2 is a function of x_1. This is because of the fact that x_2 is chosen after those of x_1. Various degrees of cooperation between the leader and the follower can be assumed. From the point of view of the leader, the best case is when the follower accepts the leader's preferences, the cooperative case, and the worst case is the follower adopts the opposite of the leader's preferences. Notice also that the term $c_{21}^{\mathrm{T}} x_1$ in the objective function of the lower OM can be omitted without affecting the optimum. This is because the decision of the lower DM must be based on the already decided x_1. Because of the various applications in different fields, different names have been used in the literature for the leader. Some of them are "upper", "outer", "level one", or "policy". Similarly, "lower" "inner", "level two", or "behavioral" are used instead of "follower". Notice that once x_1 is decided, the lower DM's problem becomes a regular mathematical programming problem. Thus, for a given x_1, let $G(x)$ denote the set of optimal solutions of the following lower-level problem:

$$\max_{x_2 \in f_1(x_1)} f_2(x_2) = c_{22}^{\mathrm{T}} x_2 \tag{2.3}$$

where $f_1(x_1) = \{x_2 | A_2 x_2 \leq b - A_1 x_1,\ x_2 \geq 0\}$ represents the lower-level DM's feasible decision space.

The set of rational reactions of the lower DM or f_2 over the decision x_1 can also be defined as:

$$\Psi(x_1) = \{x_2 | x_2 \in G(x_1)\} \tag{2.4}$$

Multiple local optima may exist for this bi-level problem and on the other hand, there is the possibility of no solution, which is especially true if both upper and lower problems have constraints. Computationally, the basic problem is the maximization of a piece-wise linear function over a polyhedron, which is formed by the constraints. However, due to the interactions of the two levels, this linear optimization problem cannot be solved easily. Thus, various transformation approaches, which transform

the two-level into a single-level problem, have been proposed. Recently, the various evolutionary or local search approaches have also been proposed to solve this very NP-hard problem. Thus, the various algorithms developed for solving the linear BLPPs can be classified into the following four categories:

- extreme-point search method, which searches over the polyhedron formed by the constraints;
- transformation method, which transforms the problem into a one-level optimization problem by the use of the Karush-Kuhn-Tucker (KKT) theory or other methods such as penalty function;
- the evolutionary methods;
- the fuzzy interactive approach.

Combinations of these various approaches have also been proposed. Table 1.1 listed some of the references for the various classifications. The first two approaches will be discussed in this chapter. The extreme-point-search approach seeks a compromise vertex along the polyhedron based on the simplex algorithm. It is very inefficient, especially for large problems. To improve the algorithm, short-cut (Bard 1983b) has been proposed. However, with these short cuts, the generalities of the original algorithm appear to be lost (Ben-Ayed and Blair 1990; Haurie et al. 1990).

The transformation approach transforms the original two-level problems into single-level by treating the lower-level problem as constraints of the higher level and by using the KKT optimality conditions or other transformation functions (Anandalingam and White 1990; Wen and Hsu 1991). However, the resulting problem becomes non-linear and complex. Several fairly effective algorithms have been proposed to solve this problem.

The first step in the transformation approach is to use the KKT conditions to transform the original problem to a first-level auxiliary problem. The resulting problem is equivalent to the elimination of Eq. (2.2). Thus, the BLPP can be re-formulated as the following first-level auxiliary problem:

$$\begin{aligned}
&\max_{x_1,x_2} f_1(x_1, x_2) = c_{11}^{\mathrm{T}} x_1 + c_{12}^{\mathrm{T}} x_2 \\
&\text{subject to:} \\
&\quad A_1 x_1 + A_2 x_2 \le b \\
&\quad w^{\mathrm{T}}(A_1 x_1 + A_2 x_2 - b) = 0 \\
&\quad w^{\mathrm{T}} A_2 = c_{22} \\
&\quad x_1, x_2, w \ge 0
\end{aligned} \tag{2.5}$$

which was obtained by applying the KKT optimality conditions with w being the dual row vector or called the Lagrange multiplier vector.

For simplicity, we shall assume that the linear BLPP has a solution, which means that the feasible region of the problem with the given constraint is nonempty and, for each decision taken by the upper level, the second level has space to respond within his or her feasible region. However, as has been discussed in the previous chapter, even with these assumptions, the BLPP of the duo-ploy type may still have

complications. One example is when the lower-level rational reaction set is not a singleton, in which case, depending on the reaction of the lower DM, the upper-level DM may not obtain the best value for his or her objective function.

Due to the complementary slackness condition in the constraints, the transformed problem forms a non-convex non-linear programming problem. The various algorithms proposed to solve this non-linear programming problem can be roughly divided into the following groups:

- complementary-pivot algorithm. First proposed-by Bialas et al. (1980) and Bialas and Karwan (1984). Further developments by Judice and Faustino (1988, 1992), Onal (1993), Candler and Townsley (1982), and others;
- branch-and-bound algorithm. Investigators who use this approach are, Bard and Falk (1982), Bard and Moore (1990), Fortuny-Amat and McCarl (1981), Hansen et al. (1992), and others;
- penalty-function approach. Various schemes have been proposed to define the penalty function. For example, the penalty function can be incorporated with the lower-level objective function only. Or, both objective functions can be penalized. Another example is due to White and Anandalingam (1993). These authors used a duality-gap-penalty function format to solve the bilinear version of the linear BLPP.

The complementary-pivot algorithm solves the system iteratively. The basic procedure can be viewed as an implicit enumeration of the lower level, and thus, the solution procedure can be extended easily to the three-level programming problem (TLPP), which will be discussed in the next chapter. However, Ben-Ayed and Blair (1990) showed that the early complementary-pivot algorithm such as that of Bialas and Karwan (1984) does not guarantee the convergence to the optimum. To overcome this deficiency, Judice and Faustino (1992) reformulated this algorithm by first solving a sequence of linear complementary problems, which is formulated by treating the objective function as another constraint. Fortuny-Amat and McCarl (1981), again based on the KKT condition, formulated a much larger mixed-integer problem. The branch-and-bound algorithm such as that of Bard and Falk (1982) is based on a series of transformations through the complimentary slackness conditions to transform the product terms into a series of equalities without altering the solution. The branch-and-bound technique is then applied to partition the feasible region and thus obtain a global optimal solution. Later, Bard and Moore (1990) modified the algorithm with 0–1 variables and extend it to solve quadratic BLPPs, which will be discussed in a later chapter for non-linear multiple-level problems. They claim that the performance and the robustness of the resulting algorithm are superior to all other contenders. However, Ben-Ayed (1993) showed that, for large practical problems, the efficiency of the algorithm is still constrained by the exponential growth of the branch-and-bound tree. Hansen et al. (1992) formed a new branch-and-bound algorithm based on the tightness of the follower's constraint and used a mixed-integer approach. To date, the most efficient traditional approaches appear to be algorithms based on branch-and-bound and the complementary-pivot concept such as those due to Bard and Moore (1990), Judice and Faustino (1992), and Hansen et al. (1992).

2.2 Extreme Point Search

As has already mentioned above, the extreme-point-search approach is not very efficient numerically. However, it does give a very good picture of the problems involved. Thus, we shall give a fairly detailed discussion with the help of an example and graphical illustrations.

Extreme-point search includes the kth-best algorithm and the grid-point-search method. Although some reviews (Kolstad 1985; Anandalingam and Friesz 1992) classify the latter as KKT methods, it seems more suitable to be classified as vertex enumeration (Wen and Hsu 1991).

2.2.1 k*th-Best Algorithm*

Assuming the feasible region is closed and bounded, Bialas and Karwan (1982, 1984) proposed the kth-best algorithm with the following procedure:

Step 1. Start with $i = 1$ and solve Eq. (2.1) with the given decision space X by the simplex algorithm. Let this solution for the upper-level DM be represented by $x^*_{[i]}$, and the possible solution set $S = \{x^*_{[i]}\}$. Go to Step 2.

Step 2. Solve Eq. (2.3) with $x_1 = x^*_{[i]}$. Let x^+ be the optimal solution of the lower level DM. If $x^+ = x^*_{[i]}$, then stop and $x^*_{[i]}$ is the global optimum with iteration index $k = i$. Otherwise, go to Step 3.

Step 3. Let $S_{[i]}$ denote the set of extreme points x which are adjacent to x^+ and such that $c_1^{\mathrm{T}} x \le c_1^{\mathrm{T}} x^*_{[i]}$. Let $I = I \cup \{x^*_{[i]}\}$ and $S = (S \cup S_{[i]}) \cap I^{\mathrm{C}}$. Go to Step 4.

Step 4. Set $i = i + 1$ and choose $x^*_{[i]}$ so that $\boldsymbol{c}_1 x^*_{[i]} = \max_{x \in S}\{\boldsymbol{c}_1 x\}$. Go to Step 2.

The search starts at the optimum of the upper-level problem. An overall optimal solution is reached if the upper-level optimum matches the lower-level one. Otherwise, search for the neighboring comers or extreme points of the previous point until the upper-level DM's proposed decision match the lower-level DM's optimum. From this iterative algorithm, we can see that the upper-level DM widens her tolerance or decision space continuously so that an overall optimum can be obtained. Thus, an implicitly compromise process is continuous carried out with each iteration.

To examine in more detail about this comprising process, let us consider the situation where, at the ith iteration, the solutions for the two problems are located at two neighboring vertices. The kth-best algorithm will force its solution to be either one of the two, depending on who moves first. The DM who moves first dominates the solution. Indeed this situation happens in all the extreme-point search procedures. The fundamental assumption of the kth-best algorithm is that the optimum should exist at the comer points. Since it is essentially impossible to define true optimality for multiple-objective problems, compromise or cooperation is usually needed to reach a solution. Instead of moving to one of the adjacent comer points, a more averaging compromise between these two extreme points would be more reasonable. In other

words, instead of only one DM sacrifices, the sacrifice should be shared by both DMs. We shall see later that the proposed fuzzy approach compromises between both DMs and thus the results are more reasonable and, furthermore, the solution is not necessarily at the comer point.

The following example demonstrates the computational procedure of the kth-best algorithm.

Example 2.1 A bi-level programming problem for balancing between trade surplus and profit.

$$\begin{aligned}
&\max_{x_1} f_1 = 2x_1 - x_2 \quad \text{(trade surplus, upper level)}\\
&\text{where } x_2 \text{ solves}\\
&\max_{x_2} f_2 = x_1 + 2x_2 \quad \text{(profit, lower level)}\\
&\quad \text{subject to:}\\
&\qquad 3x_1 - 5x_2 \le 15 \quad \text{(capacity)}\\
&\qquad 3x_1 - x_2 \le 21 \quad \text{(management)}\\
&\qquad 3x_1 + x_2 \le 27 \quad \text{(space)}\\
&\qquad 3x_1 + 4x_2 \le 45 \quad \text{(material)}\\
&\qquad x_1 + 3x_2 \le 30 \quad \text{(labor hours)}\\
&\qquad x_1, x_2 \ge 0
\end{aligned}$$

For simplicity, X will be used to represent the above constraint set in the following discussions. The actual solution procedure, based on the kth-best algorithm, can be summarized as:

Step 1. Solve the upper level problem,

$$\begin{aligned}
&\max f_1 = 2x_1 - x_2\\
&\text{subject to: } x \in X.
\end{aligned}$$

The solution is $x^*_{[1]} = (7.5, 1.5)$ at vertex B (see Fig. 2.1). Let the possible solution set $S = \{(7.5, 1.5)\}$ and index set $I = \varnothing$. Go to Step 2.

Step 2. Solve the following lower-level problem with the results just obtained for the upper level:

$$\begin{aligned}
&\max f_2 = x_1 + 2x_2\\
&\text{subject to: } x \in X
\end{aligned}$$

with $x_1 = 7.5$. The solution of the problem is $x^+ = (7.5, 4.5)$. Since $x^+ \neq x^*_{[1]}$, go to Step 3.

Step 3. Form the neighboring vertex set $S_{[1]} = \{(8, 3), (5, 0)\}$, which includes neighboring vertices A and C (see Fig. 2.1). Let $I = I \cup \{x^*_{[1]}\} = \{(7.5, 1.5)\}$.

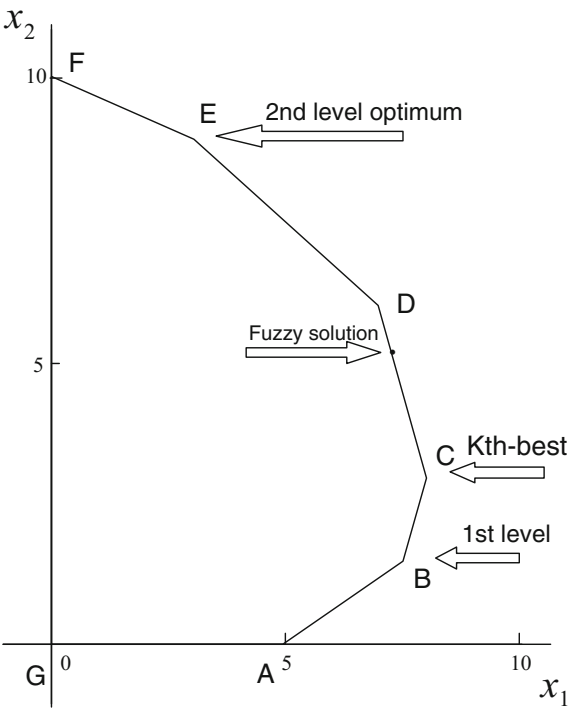

Fig. 2.1 Decision variable space for Example 2.1

Table 2.1 Numerical values at the various vertices, Example 2.1

Vertex	A	B	C	D	E	F	G
(x_1, x_2)	(5,0)	(7.5, 1.5)	(8, 3)	(7, 6)	(3, 9)	(0, 10)	(0, 0)
(f_1, f_2)	(10, 0)	(13.5, 10.5)	(13, 14)	(8, 19)	$(-3, 21)$	$(-10, 20)$	(0, 0)

The possible solution set is $S = (S \cup S_{[1]}) \cap I^C = \{(8, 3), (5, 0)\}$. Go to Step 4.

Step 4. Set $i = i + 1 = 2$, and choose $x^*_{[2]} = \max_{x \in S}\{c_1 x\} = (8, 3)$. Thus the current solution is at vertex C. Go back to Step 2.

Step 5. Solve the lower-level problem again with $x_1 = 8$. The solution is $x^+ = (8, 3)$. Since $x^+ = x^*_{[2]}$, the procedure terminates and $x^*_{[2]}$ is the global optimum of the bi-level system with index $k = i = 2$.

Thus, the optimum obtained is $x = (8, 3)$, $f = (13, 14)$, which is at vertex C. The decision and objective spaces are shown in Figs. 2.1 and 2.2, respectively. The values of the decision variables and the objective functions at the various vertexes are summarized in Table 2.1. ■

The example can also be treated as a Stackelberg game. The pay-off table or the bi-matrix form of the Stackelberg game is shown in Table 2.2. In addition to the

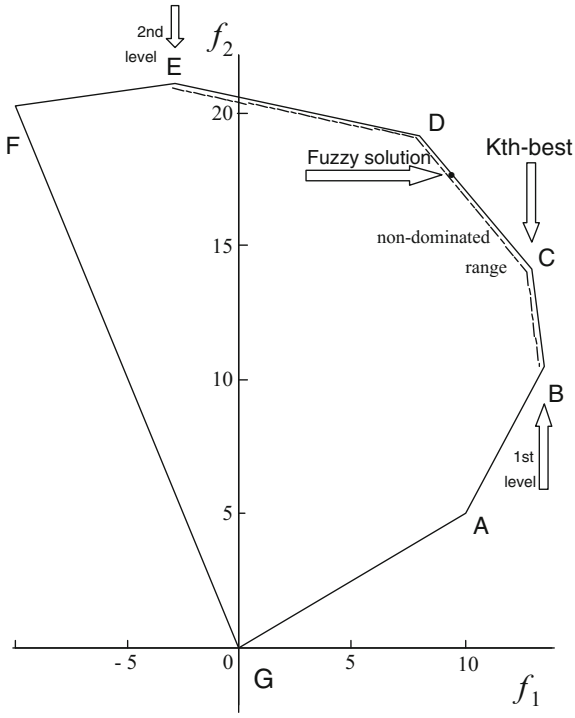

Fig. 2.2 Objective function space of Example 2.1

vertexes and their values, the iteration sequences are also shown in parentheses. The moves in the game are searching sequences for the desired solution. The leader or upper-level DM as well as the follower or lower-level DM interacts with each other throughout the decision process as in Stackelberg game. This interaction terminates when the leader can no longer maximize his objective function. In such a situation, the Stackelberg equilibrium is reached. Since our problem has very few comer points, the matrix is very sparse as shown in Table 2.2.

2.2.2 Grid Search Algorithm

Bard (1982, 1983b) proposed an algorithm based on sensitivity analysis to solve the linear two-stage optimization problem. The underlying theory rests on a set of first order optimality conditions that parallel to the KKT conditions associated with a one-level parameterized linear program. By systematically varying the parameter over the unit interval and solving the corresponding linear program, the solution to the original problem is eventually obtained. The transformed parameterized expression can be expressed as:

Table 2.2 Bi-matrix form, Example 2.1

X_2 / X_1	0	1.5	3	6	9	10
0	G(0, 0)	–	–	–	–	F(−10, 20)
3	–	–	–	–	E(−3, 21)	–
5	A(10, 5)	–	–	–	–	–
7	–	–	–	D(8, 19)	–	–
7.5	–	B, (1) (13.5, 10.5)	–	–	–	–
8	–	–	C, (2) (13, 14)	–	–	–

$$\begin{aligned}
&\max P(\lambda', x_1, x_2) = \lambda'(c_{11}^{\mathrm{T}} x_1 + c_{12}^{\mathrm{T}} x_2) + (1 - \lambda') c_{22}^{\mathrm{T}} x_2 \\
&\text{subject to:} \\
&\quad A_1 x_1 + A_2 x_2 \le b, \\
&\quad x_1, x_2 \ge 0, \\
&\quad \lambda' \in [0,\ 1],
\end{aligned} \tag{2.6}$$

Again, in the following discussions, X represents the constraint set.

Once x_1 is obtained, the next step is to check the feasibility by solving the following inside problem:

$$\begin{aligned}
&\max \ \ f_2(x_2) = c_{22}^{\mathrm{T}} x_2 \\
&\text{subject to:} \\
&\quad A_2 x_2 \le b - A_1 \underline{x}_1, \\
&\quad x_2 \ge 0\,,
\end{aligned} \tag{2.7}$$

where $\underline{x}_1$ was obtained by solving Eq. (2.6).

Bard (1982, 1983b) argues that there exists an $\lambda'^* \in [0, 1]$ such that the corresponding solution $(\boldsymbol{x}_1^*, \boldsymbol{x}_2^*)$ of Eq. (2.6) is both feasible and optimal to the original bi-level programming problem. Under certain non-degeneracy assumptions, the proposed algorithm is described as:

Step 1. Let $\lambda' = 1$ and solve Eq. (2.6) to obtain $(\boldsymbol{x}_1^1, \boldsymbol{x}_2^1)$.

Step 2. Check $(\boldsymbol{x}_1^1, \boldsymbol{x}_2^1)$ for feasibility by solving the inside problem, i.e. Equation (2.7), to obtain $\underline{x}_2^1$. If $\underline{x}_2^1 = x_2^1$, stop. Otherwise, go to Step 3.

Step 3. Let (x_1^k, x_2^k) be the current solution of Eq. (2.6) with $\lambda' = \lambda'_k$. From sensitivity analysis, determine the minimum value of λ' such that (x_1^k, x_2^k) remains

Table 2.3 Iteration results for Example 2.2

λ'	(x_1, x_2, y_1, y_2)	f_1	Feasibility
$1.0 - 0.98$	(1, 0, 0, 1)	2.0	No
$0.97 - 0.84$	(1, 0, 0, 0.5)	2.0	No
$0.83 - 0.67$	(1, 0, 0.5, 1)	1.75	Yes
$0.66 - 0$	(0, 2, 5.5, 8)	−4.75	Yes

optimal. Call this value $\lambda'_{\min}$ and put $\lambda'_{k+1} = \lambda'_{\min} - \delta$ where $\delta > 0$ is sufficiently small so that no vertices are missed.

Step 4. Solve Eq. (2.6) for $P(\lambda'_{k+1}, x_1, x_2)$ to obtain (x_1^{k+1}, x_2^{k+1}).

Step 5. Check (x_1^{k+1}, x_2^{k+1}) for feasibility by solving the inside problem, i.e. Equation (2.7), to obtain $\underline{x}_2^{k+1}$. If $\underline{x}_2^{k+1} = x_2^{k+1}$, stop; otherwise, let $k \leftarrow k+1$ and go to Step 3.

Notice that there may have multiple optimal solutions in Step 4 if x_2 with a given x_1 is not a singleton. In that case, the algorithm must iterate for each of these vertices. Let us illustrate the solution procedure with the following example.

Example 2.2 Consider the linear problem (Bard and Falk 1982):

$$\begin{aligned}
&\max_x f_1 = 2x_1 - x_2 - 0.5y_1 \\
&\text{where } x \text{ solves} \\
&\max_y f_2 = -x_1 - x_2 + 4y_1 - y_2 \\
&\quad \text{subject to:} \\
&\qquad -2x_1 + y_1 - y_2 \le -2.5, \\
&\qquad x_1 - 3x_2 + y_2 \le 2, \\
&\qquad 2x_1 + x_2 \le 2, \\
&\qquad x, y \ge 0\,,
\end{aligned}$$

The parameterized linear program for the above problem is:

$$\begin{aligned}
&\max P(\lambda', x_1, x_2) = 2\lambda' x_1 - \lambda' \boldsymbol{x}_2 + (4 - 4.5\lambda')y_1 + (1 - \lambda')y_2 \\
&\text{subject to:} \quad (x, y) \in X.
\end{aligned}$$

where X represents the set of the original constraints.

The above problem was solved. The iteration results for the grid search are summarized in Table 2.3. Feasibility is determined by solving the following inside problem:

$$\begin{aligned}
&\text{Max } f_2 = 4y_1 - y_2 \\
&\text{subject to:} \\
&\qquad y_1 - y_2 \le -2.5 + 2x_1^k,
\end{aligned}$$

Table 2.4 Comparison of algorithms (Bard 1983b)

Algorithm	Problem size (n_1, n_2, m)	Average CPU time	Average number of iterations	Range of iterations
Branch-and-bound	(6, 3, 7) (10, 10, 6) (30, 20, 25)	77 136 215	321 786 1142	106–630 340–1010 510–570
Implicit search	(6, 3, 7) (10, 10, 16) (30, 20, 25)	34 80 194	63 107 232	53–88 47–191 126–430
PCP algorithm	(6, 3, 7) (10,10,16) (30,20,25)	2 5 9	12 21 47	7–18 14–26 13–123
kth-best	(6,3,7) (10, 10, 16) (30, 20, 25)	4 18 66	77 229 966	61–109 153–576 423–2011
Grid search	(6, 3, 7) (10, 10, 16) (30, 20, 25)	1 1 1	5 11 23	3–8 4–16 11–38

$$y_2 \leq 2 - x_1^k + 3x_2^k,$$
$$y \geq 0\,,$$

with the values for (x_1^k, x_2^k) are obtained from the above parameterized linear program. As shown in Table 2.3, the objective of the upper level decreases as λ' decreases. The global optimum is obtained at the first feasible point. The optimum is $(x_1, x_2, y_1, y_2) = (1, 0, 0.5, 1)$ and the objectives $f = (1.75, -1)$ for the upper and the lower levels, respectively. For purpose of illustration, the computation is continued until λ' is reduced to zero. ■

The grid-search algorithm is designed to solve Eq. (2.6) parametrically and to terminate when the first feasible point is reached. Bard's (1983b) computational results are reproduced in Table 2.4. The symbols n_1 and n_2 are the number of decision variables of the upper and lower levels, respectively, and m represents the number of constraints. The CPU time listed for comparison purposes only. As can be seen from this table, grid-search algorithm has an overall advantage over the algorithms of the kth-best approach (Bialas and Karwan 1982, 1984), parametric complementary-pivot (PCP) algorithm (Bialas and Karwan 1982), implicit search (Candler and Townsley 1982), and the branch-and-bound algorithm (Bard and Falk 1982). However, Haurie et al. (1990) and Ben-Ayed and Blair (1990) have shown that this algorithm may fail to obtain the optimum.

2.3 Transformation Approach

As discussed before, the transformation approach uses the KKT optimality condition or other similar functions to convert the lower-level problem into the constraint of the upper level. The transformed problem such as Eq. (2.5) is one less level than the original two-level one. Thus, this nested optimization problem is reduced to the traditional one-level non-linear programming problem, which is non-convex. By using the traditional optimization approaches such as branch-and-bound and penalty-function, various algorithms have been developed. Some of them appear to be the most efficient approaches among the traditional optimization techniques for this duo-ploy type problem.

2.3.1 Mixed-Integer Approach

Fortuny-Amat and McCarl (1981) developed a mixed-integer approach by considering the disjunctive nature of the complementary slackness of the nonlinear term in the transformed equation. Due to the introduction of the zero-one variables, the resulting formulation forms a much larger programming problem. However, the basic idea is a very useful one and has been incorporated into several algorithms. Notice that in the transformed problem, Eq. (2.5) the only non-linear equation is the complimentary slackness equation:

$$w^{\mathrm{T}}(A_1x_1 + A_2x_2 - b) = 0$$

If we let:

$$y = A_1x_1 + A_2x_2 - b$$

then, we obtain $w^{\mathrm{T}}y = 0$, which is consisted of the sums of the products of two variables. In order to satisfy this equation, one of the two variables, either w_i or y_i must be equal to zero. This condition can be achieved by introducing the zero-one variable η for each product. Thus, we have:

$$\begin{aligned} &w \le (1 - \eta)M, \text{ and} \\ &y = A_1x_1 + A_2x_2 - b \le M\eta. \end{aligned}$$

where M is a very large positive constant. Using the above two equations, Eq. (2.5) can be restated as:

$$
\begin{aligned}
&\max\ f_1(x_1, x_2) = c_{11}^{\mathrm{T}} x_1 + c_{12}^{\mathrm{T}} x_2 \\
&\text{subject to:} \\
&\qquad A_1 x_1 + A_2 x_2 \le b, \\
&\qquad w \le (1-\eta) M, \\
&\qquad A_1 x_1 + A_2 x_2 - b \le M\eta, \\
&\qquad w^{\mathrm{T}} A_2 = c_{22}, \\
&\qquad \eta \in \{0,\ 1\}, \\
&\qquad x_1, x_2, w \ge 0\,,
\end{aligned}
\tag{2.8}
$$

Compared to Eq. (2.5), the non-linear term disappeared and is replaced by two linear inequalities. Due to the presence of the zero-one integer, the problem represented by Eq. (2.8) is a mixed-integer problem.

To illustrate the algorithm, the numerical example solved in Example 2.1 is again solved by this mixed-integer approach in the following.

Example 2.3 Using Eq. (2.8), the numerical example used in Example 2.1 becomes:

$$
\begin{aligned}
&\max\ f_1 = 2x_1 - x_2 \\
&\text{subject to:} \\
&\qquad 3x_1 - 5x_2 \le 15, \\
&\qquad 3x_1 - x_2 \le 21, \\
&\qquad 3x_1 + x_2 \le 27, \\
&\qquad 3x_1 + 4x_2 \le 45, \\
&\qquad x_1 + 3x_2 \le 30, \\
&\qquad -3x_1 + 5x_2 + 15 \le M\eta_1, \quad w_1 \le (1-\eta_1)M, \\
&\qquad -3x_1 + x_2 + 21 \le M\eta_2, \quad w_2 \le (1-\eta_2)M, \\
&\qquad -3x_1 - x_2 + 27 \le M\eta_3, \quad w_3 \le (1-\eta_3)M, \\
&\qquad -3x_1 - 4x_2 + 45 \le M\eta_4, \quad w_4 \le (1-\eta_4)M, \\
&\qquad -x_1 - 3x_2 + 30 \le M\eta_5, \quad w_5 \le (1-\eta_5)M, \\
&\qquad -5w_1 - w_2 + w_3 + 4w_4 + 3w_5 = 2\,, \\
&\qquad \eta_1, \eta_2, \eta_3, \eta_4, \eta_5 \in \{0,\ 1\}, \\
&\qquad w_1, w_2, w_3, w_4, w_5, x_1, x_2 \ge 0.
\end{aligned}
$$

The LINGO is used to solve this problem. With the large constant $M = 10{,}000$, the solution obtained is $x = (x_1, x_2) = (7.5, 1.5), f_1 = 13.5$, and $f_2 = 10.5$. The optimum is at vertex B (see Figs. 2.1 and 2.2). The problem is also solved with $M = 100{,}000$, 1,000,000, 10,000,000, and 100,000,000, the solution remains the same. The solution is quite stable and is different from that obtained by the kth-best approach, whose solution optimum was at point C. It appears that the upper-level DM dominates the solution for this particular case.

Because of the introduction of the zero-one integer variables η, this mixed-integer programming problem is not easy to solve. This is especially true for practical problems, which are usually large and complex. However, the concept has been used to form a branch-and-bound tree, which forms the basis for fairly effective branch-and-bound algorithms.

Fortuny-Amat and McCarl (1981) formulated the problem as a two-level quadratic programming problem and used the distributor and customer as a two-level example. In particular, these investigators considered the problem of purchasing fertilizer from two different sources, the local neighborhood versus competitors from other places.

2.3.2 Complementary Pivot Algorithm

As has been pointed out previously, the complementary-pivot algorithm, introduced by Bialas and Karwan (1984) may not converge to the optimum. However, because the approach forms the basis of later developments, the original algorithm is discussed in the following.

2.3.2.1 Parametric Complementary-Pivot Algorithm

Bialas and Karwan (1984) proposed the parametric complementary-pivot (PCP) algorithm to solve the BLPP problem. The algorithm uses Wolfe's approach for quadratic programming (1959) and transforms the objective function into the constraint, $c_{11}^{\mathrm{T}} x_1 + c_{22}^{\mathrm{T}} x_2 \geq \alpha'$, where α' is a constant. At the beginning of the iteration, a small value for α' should be used. As the computation progresses, the value of α' should be increased gradually and parametrically to improve the upper-level's objective. At the same time, the non-linear complementary slackness terms are taken care of by the use of restricted entry into the basis. Our discussion will follow the work of Bialas and Karwan (1984).

The transformed Eq. (2.5) can be rewritten in the following complementary slackness form:

$$\max_{x_1, x_2} f(x_1, x_2) = c_{11}^{\mathrm{T}} x_1 + c_{12}^{\mathrm{T}} x_2$$

subject to:

$$A_1 x_1 + A_2 x_2 + u = b$$
$$w^T A_2 - v = c_{22}$$
$$w^T u = 0$$
$$x_2 v = 0$$
$$x_1, x_2, u, v, w \geq 0$$

We have introduced the complimentary variables u and v. Instead of solving the above problem directly, the PCP algorithm considers the following system of equations (Bialas and Karwan 1984):

$$\begin{aligned} &A_1x_1 + A_2x_2 + u = b \\ &-\varepsilon Hx_1 + w^{\mathrm{T}}A_2 - v = c_{22} \\ &wu = x_2v = 0 \\ &x_1, x_2, u, v, w \geq 0 \end{aligned}$$

The maximization expression has been replaced by the third inequality in the above equations. The matrix H may be any negative definite matrix, and ε is a suitably small positive scalar so that the term εH leads only to a small perturbation of Eq. (2.5). Bialas and Karwan suggested a value of $H = -I$ with ε ranging from 10^{-2} to 10^{-8}.

The PCP algorithm can be viewed as an implicit enumeration of the lower-level optimal bases. All of the computations for the PCP algorithm are performed within the framework of the simplex-like tableau whose size is roughly that of the original system. In the absence of degeneracy and multiple level-two optima, a simplified version of the PCP algorithm can be described as (Bialas and Karwan 1984):

Step 0. Let $\alpha = \alpha_{\min}$, where $\alpha_{\min}$ is a lower bound for $c_{11}^{\mathrm{T}}x_1 + c_{12}^{\mathrm{T}}x_2$

Step 1. Obtain a feasible solution for:

$$\begin{aligned} &A_1x_1 + A_2x_2 + u = b \\ &-c_{21}^{\mathrm{T}}x_1 - c_{22}^{\mathrm{T}}x_2 + s = -\alpha' \\ &-\varepsilon Hx_2 + w^{\mathrm{T}}A_2 - v = c_{22} \end{aligned} \tag{2.9}$$

by using the Wolfe's two-phase quadratic programming algorithm. The nonlinear expressions, wu and x_2v, can be satisfied by controlling the variables entering the basis. In other words, use the following conditions for variables entering the basis:

Step 1.1. If x_{2k} is in the basis and is not the leaving variable for the current pivot, do not admit v_k into the basis, and vice versa, and

Step 1.2. If u_k is in the basis and is not the leaving variable for the current pivot, do not admit w_k into the basis, and vice versa

If a feasible solution exists, go to Step 2. Otherwise, go to Step 3.

Step 2. Let $x^* = (x_1^*, x_2^*)$ represents the incumbent solution and B represents the current basis. Let $\beta = (b, -\alpha', c_{22})$, represents the right-hand side vector in Eq. (2.9). Define $\beta = B^{-1}\beta$ and let the column corresponding to s in the simplex tableau be the kth column denoted by y_k. Choose $\Delta\alpha' = \min\{\frac{\beta_i}{y_{ki}} | y_{ki} > 0\}$. Let $\alpha' = (\alpha' + \Delta\alpha')(1 + \gamma)$ where γ is a suitable small positive scalar. Go to Step 1.

Step 3. Stop, if x^* is the solution which is feasible within the constrained space of f_2 and $c_{21}^T x_1^* + c_{22}^T x_2^*$ is within $\gamma \times 100\%$ of the optimal objective function value.

The procedure is illustrated in the example below solved by Bialas and Karwan (1984).

Example 2.4 Bialas and Karwan (1984) solved the following two-level linear resource control problem. We follow their numerical discussions.

$$
\begin{aligned}
&\max_{x_1} f_1 = x_2 \\
&\text{where } x_2 \text{ solves:} \\
&\max_{x_2} f_2 = -x_2 \\
&\quad \text{subject to:} \\
&\qquad -x_1 - 2x_2 \le -10 \\
&\qquad x_1 - 2x_2 \le 6 \\
&\qquad 2x_1 - x_2 \le 21 \\
&\qquad x_1 + 2x_2 \le 38 \\
&\qquad -x_1 + 2x_2 \le 18 \\
&\qquad x_1, x_2 \ge 0
\end{aligned}
$$

Following the above outlined procedure, the PCP algorithm deals with the following transformed expression:

$$
\begin{aligned}
&\max f_1 = x_2 \\
&\text{subject to:} \\
&\qquad -x_1 - 2x_2 + u_1 = -10 \\
&\qquad x_1 - 2x_2 + u_2 = 6 \\
&\qquad 2x_1 - x_2 + u_3 = 21 \\
&\qquad x_1 + 2x_2 + u_4 = 38 \\
&\qquad -x_1 + 2x_2 + u_5 = 18 \\
&\qquad -x_2 + s = -\alpha' \\
&\qquad -\varepsilon x_2 - 2w_1 - 2w_2 - w_3 + 2w_4 + 2w_5 - v_1 = -1 \\
&\qquad u_1 w_1 = u_2 w_2 = u_3 w_3 = u_4 w_4 = u_5 w_5 = 0 \\
&\qquad x_1, x_2, u_1, u_2, u_3, u_4, u_5, s, w_1, w_2, w_3, w_4, w_5, v_1 \ge 0
\end{aligned}
$$

Let $\varepsilon = 10^{-8}$. Since $f_1 = x_2 \ge 0$, choose $\alpha'_{\min} = 0$ to start the iteration. In Step 1, consider the first phase of the two-phase simplex. In other words, with $\alpha' = 0$, minimize the sum of the artificial variables:

$$
\begin{aligned}
&\min\ g_1 = u_1 + s + v_1 \\
&\text{subject to:} \\
&\quad x_1 + 2x_2 - u_1 = 10 \\
&\quad x_1 - 2x_2 + u_2 = 6 \\
&\quad 2x_1 - x_2 + u_3 = 21 \\
&\quad x_1 + 2x_2 + u_4 = 38 \\
&\quad -x_1 + 2x_2 + u_5 = 18 \\
&\quad -x_2 + s = 0 \\
&\quad -10^{-8}x_2 - 2w_1 - 2w_2 - w_3 + 2w_4 + 2w_5 - v_1 = -1 \\
&\quad x_1, x_2, u_1, u_2, u_3, u_4, u_5, s, w_1, w_2, w_3, w_4, w_5, v_1 \geq 0
\end{aligned}
$$

The solution for this problem is:

$$
\begin{aligned}
x &= (x_1, x_2) = (8, 1) \\
u &= (u_1, u_2, u_3, u_4, u_5) = (0,\ 0,\ 6,\ 28,\ 24) \\
w &= (w_1, w_2, w_3, w_4, w_5) = (0.5, 0, 0, 0, 0) \\
s &= 1 \\
v_1 &= 0
\end{aligned}
$$

In Step 2, let $\gamma = 0.01$ and $\Delta\alpha = 1$, the value of α is set to $(\alpha' + \Delta\alpha')(1 + 0.01) = 1.01$. Re-solving the above expression with the new values, obtain the solution as:

$$
\begin{aligned}
x &= (12, 3) \\
u &= (8, 0, 0, 20, 24) \\
w &= (0, 0.5, 0, 0, 0) \\
s &= 1.99 \\
v_1 &= 0
\end{aligned}
$$

Computing $\Delta\alpha' = 1.99$ and resetting α' to (1.01 + 1.99) (1.01) = 3.03, solve the above expression again and the solution is:

$$
\begin{aligned}
x &= (16, 11) \\
u &= (28, 12, 0, 0, 12) \\
w &= (0, 0, 1, 0, 0) \\
s &= 7.97 \\
v_1 &= 0
\end{aligned}
$$

Compute $\Delta\alpha' = 7.97$ and set $\alpha' = 11.11$. Return to Step 1, there is no solution existed with $\alpha' = 11.11$. Hence, the solution $x^* = (16, 11)$ is the solution of the two-level system.

Bialas and Karwan (1984) carried out some computational experiments to test the effectiveness of both the *k*th-best and the PCP algorithms. The number of variables controlled by the second level appears to have more influence on the performances and the problem is easier to solve with fewer level-two variables. Furthermore, it appears that the *k*th-best algorithm can be erratic under certain conditions and fails to obtain a solution within a reasonable computational time. Thus, they concluded that the PCP algorithm is a more reliable approach.

2.3.2.2 Sequential Linear Complementary Algorithm with Branch-and-Bound

The PCP algorithm due to Bialas and Karwan (1984) sometimes does not converge and is unable to solve the BLPP in all cases. To overcome this problem, Judice and Fausstino (1992) modified and reformulated the problem and the modified problem was referred to as the sequential linear complementary problem (SLCP). Instead of using the original algorithm, these investigators, with the help of a branch-and-bound search, solved a sequence of linear complementary problems to converge to the desired optimum of the original problem. These authors also carried out a fairly extensive numerical study, and found out that, compared to the branch-and-bound algorithm by Bard and Moore (1990), the proposed approach was consistently more efficient. Furthermore, the current approach appears to be more advantageous as the dimension of the problem increases. Another advantage is that the algorithm is robust and is fairly efficient in finding a global optimum.

By adding complementary variables, Eq. (2.5) can be rewritten as:

$$\begin{aligned}
&\max_{x_1,x_2} f_1(x_1, x_2) = c_{11}^{\mathrm{T}} x_1 + c_{12}^{\mathrm{T}} x_2 \\
&\text{subject to:} \\
&\qquad A_1 x_1 + A_2 x_2 + u = b \\
&\qquad w^{\mathrm{T}} A_2 - v = c_{22} \\
&\qquad w^{\mathrm{T}} u = 0 \\
&\qquad x_2 v = 0 \\
&\qquad x_1, x_2, v, u, w \geq 0
\end{aligned} \tag{2.10}$$

The last two equations before the non-negativity conditions are complementary slackness conditions. Judice and Faustino (1992) introduced a parameter λ, and replaced the objective function with the constraint:

$$c_{11}^{\mathrm{T}} x_1 + c_{12}^{\mathrm{T}} x_2 \geq \lambda$$

After introducing the slack variable, β, the linear complementary problem becomes:

$$
\begin{aligned}
&A_1 x_1 + A_2 x_2 + u = b \\
&A_2^{\mathrm{T}} w - v = c_{22} \\
&c_{11}^{\mathrm{T}} x_1 + c_{12}^{\mathrm{T}} x_2 + \beta = \lambda \\
&w^{\mathrm{T}} u = x_2 v = 0 \\
&x_1, x_2, v, u, w \geq 0
\end{aligned}
$$

which can be presented in a matrix form:

$$
\begin{bmatrix} -u \\ -v \\ \beta \end{bmatrix} = \begin{bmatrix} -b \\ c_{22} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \lambda + \begin{bmatrix} 0 & A_1 & A_2 \\ -A_2^T & 0 & 0 \\ 0 & -c_{11}^T & -c_{12}^T \end{bmatrix} \begin{bmatrix} w \\ x_1 \\ x_2 \end{bmatrix}
$$

$$
\begin{aligned}
&w^{\mathrm{T}} u = x_2 v = 0 \\
&x_1, x_2, v, u, w \geq 0
\end{aligned}
\tag{2.11}
$$

The efficiency of the proposed algorithm depends essentially on the solution of the linear complementary problem, Eq. 2.11. Judice and Faustino (1992) developed a hybrid enumerative tree-search technique to solve this linear complementary problem (LEP). These investigators further improved the efficiency by exploring various strategies to reduce the search. The hybrid enumerative approach is essentially a branch-and bound enumerative technique, whose efficiency is improved by the incorporation of a modified reduced gradient method due to AI-Khayyal (1987). Equation 2.11 can be represented by the following vector equation:

$$x = q + My + Nz \tag{2.12a}$$

$$x \geq 0, y \geq 0, z \geq 0 \tag{2.12b}$$

$$x_i y_i = 0, i = 1, 2, 3, \ldots, n_2 + m, \tag{2.12c}$$

where x and y are $(n_1 + m + 1)$- and $(n_1 + \mathrm{m})$-dimensional vectors, respectively; z is an m-dimensional vector; and q is an $(n_1 + m + 1)$-dimensional vector. Following the tradition of linear programming, a solution of Eq. (2.12a) satisfying the constraints of Eq. (2.12b) is a feasible solution. A complementary solution (x, y, z) must satisfy Eqs. (2.12a)–(2.12c). The enumerative branch-and-bound method attempts to find a complementary solution by using only basic feasible solutions of Eq. 2.12a. This is accomplished by exploring the branch-and- bound tree shown in Fig. 2.3 (Judice and Faustino 1992), where $i_1, i_2, \ldots$ are integer numbers.

To begin the process, an initial feasible solution for node 1 is obtained by solving the problem:

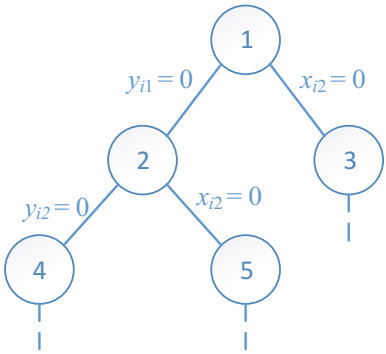

Fig. 2.3 Branch-and-bound tree (Judice and Faustino 1992)

$$\begin{aligned} &\min\ y_0 \\ &\text{subject to:} \\ &\qquad x = q + py_0 + My + Nz \\ &\qquad x \geq 0,\ y \geq 0,\ z \geq 0,\ y_0 \geq 0 \end{aligned} \tag{2.13}$$

where y_0 is an artificial variable and p is a non-negative vector satisfying $p_i > 0$ for all i such that $q_i < 0$, which essentially follows the rule used by Wolfe (1959). This problem is solved by using the simplex algorithm with a modification that whenever possible Eq. (2.12c) is also satisfied. The remaining notes are generated in essentially the same manner, namely, solving a problem similar to Eq. (2.13) except minimizing either y_{ik} or x_{ik} with $k = 2, 3, \ldots$, and with the variables in earlier notes kept to zero. Two situations may occur during the generation of anyone of the notes:

1. if the minimum of zero is obtained, then it is fixed at zero and continues to generate the next generation of notes at the descending path of the tree;
2. if the minimum value is positive, then this branch is pruned and the node is fathomed.

This enumerative branching is continued until either a complementary solution is obtained or the problem has no solution. The algorithm can be summarized as (Judice and Faustino 1992):

Step 1. Obtain an initial feasible solution. If no feasible solution can be obtained, the problem has no solution. Otherwise, go to Step 2.

Step 2. If the solution obtained satisfies Eq. (2.12c), or complementary; then a solution is found and stop. Otherwise, go to Step 3.

Step 3. Choose two sets of complementary positive basic variables x_i and y_i and generate two nodes by solving two linear programming problems, which minimize x_i or y_i subject to the linear constraints with x_i and y_i at earlier nodes set to zero. If the minimum of the variable is positive, then the corresponding node is fathomed.

Step 4. If all the nodes are fathomed, then the problem has no solution and stop. Otherwise, choose an unfathomed node and got to 2.

In order to carry out the above procedure, the following details are needed: the choice of the initial feasible solution in Step 1, the choice of the two sets of the complementary variables x_i and y_i in Step 3, and the choice of the unfathomed node in Step 4. Obviously, these choices influence tremendously the efficiency of the algorithm. This is especially true if there are many nodes to be obtained along the branch-and-bound tree. Judice and Faustino (1992) incorporated a number of procedures to help the above choices and thus increase the search efficiency. One of the heuristics used is the reduced gradient search method due to AI-Khayyal (1987). This gradient method can find a local optimum of the function:

$$f(x, y, z) = \sum_{i=1}^{m+n_2} x_i y_i$$

Fairly large computational experiments were carried out by Judice and Faustino (1992) and compared with other approaches. The maximum number of constraints, the maximum number of variables controlled by the leader, and the maximum number of variables controlled by the follower in these experiments are 150, 300, and 150, respectively. For small-dimensional problems, this improved hybrid enumerative SLCP approach is about par with the branch-and-bound approach due to Bard and Moore (1992). However, for large-dimensional problems, the present approach is significantly more efficient then the branch-and- bound algorithm.

2.3.3 Branch-and-Bound Algorithm

Among the many approaches for solving the BLPP problems, the branch-and-bound algorithm appears to be one of the more effective approaches among the traditional optimization techniques. Fortuny-Amat and McCarl (1981) suggested the use of two equality constraints with zero-one variables to replace the non-linear complementary term. Bard and Moore (1990) used this concept to improve their earlier branch-and-bound algorithm (Bard and Falk 1982) and the resulting formulation can be used to solve linear and quadratic BLPP. In the following discussion, we have restricted our discussions to the linear problem. Bard and Moore formulated their problem in the quadratic form, which will be discussed in the next chapter. Hansen et al. (1992) follows the concept used in global optimization, proposed a new algorithm by exploiting the follower's tight constraint. They did fairly extensive numerical studies and their algorithm appears to be the most efficient one among the proposed branch-and bound approaches.

2.3.3.1 Algorithm of Bard and Moore

To formulate the algorithm, let us rewrite Eq. 2.5 to include the non- negativity constraints of the decision variables for the lower-level decision variable, x_2 explicitly:

$$\begin{aligned}
&\max_{x_1,x_2} f_1(x_1, x_2) = c_{11}^{\mathrm{T}} x_1 + c_{12}^{\mathrm{T}} x_2 \\
&\text{subject to:} \\
&\quad A_1 x_1 + A_2 x_2 \le b \\
&\quad w_1^{\mathrm{T}}(A_1 x_1 + A_2 x_2 - b) + w_2^{\mathrm{T}} x_2 = 0 \\
&\quad w_1^{\mathrm{T}} A_2 + w_2^{\mathrm{T}} = c_{22} \\
&\quad x_1, x_2, w, u \ge 0
\end{aligned} \tag{2.14}$$

where w_1 and w_2 are dual vectors or Lagrange multipliers for the constraints of the original problem and the non-negativity constraints for the lower-level decision variables, respectively. Rewrite the above equations without the complementary non-linear equation, we have:

$$\begin{aligned}
&\max_{x_1,x_2} f_1(x_1, x_2) = c_{11}^{\mathrm{T}} x_1 + c_{12}^{\mathrm{T}} x_2 \\
&\text{subject to:} \\
&\quad A_1 x_1 + A_2 x_2 \le b \\
&\quad w_1^{\mathrm{T}} A_2 + w_2^{\mathrm{T}} = c_{22} \\
&\quad x_1, x_2, w \ge 0
\end{aligned} \tag{2.15}$$

and write the complementary equation separately:

$$w_1^{\mathrm{T}}(A_1 x_1 + A_2 x_2 - b) + w_2^{\mathrm{T}} x_2 = 0 \tag{2.16}$$

which can also be represented as:

$$w_i q_i = 0, \quad i = 1, 2, \ldots, m + n_2 \tag{2.17}$$

where $w^{\mathrm{T}} = [w_1, w_2]$, the first m $q's$ equal to the correspondence elements of $(A_1 x_1 + A_2 x_2 - b)$, and the last $n_2 q's$ equal to the correspondence elements of the vector x_2. Note that Eq. (2.15) is linear and thus can be solved easily. The procedure is to solve Eq. (2.15) iteratively. If the results for any iteration satisfy Eq. (2.16) or (2.17) for all i, then the corresponding point is in the inducible region and thus is a potential solution for the BLPP. If not, branch-and-bound algorithm is used to examine implicitly all combinations of the complementary slackness, Eq. (2.17).

To formulate the algorithm, Bard and Moore (1990) introduced some additional notation. Let $v = \{1, 2, \ldots, m + n_2\}$ be the index set for the terms in Eq. 2.17, and let f_1 be the incumbent lower bound on the upper-level's objective. At the kth level of

the branch-and-bound tree we define a subset of indices $v_k \subseteq v$, and a path vector P_k corresponding to an assignment of $w_i = 0$ or $q_i = 0$ for $i \in v_k$. Then, let:

$$S_k^+ = \{i \,|\, i \in v_k \text{ and } w_i = 0\}$$
$$S_k^- = \{i \,|\, i \in v_k \text{ and } q_i = 0\}$$
$$S_k^0 = \{i \,|\, i \notin v_k\}$$

For $i \in S_k^0$, the variables w_i and q_i are free to assume any non-negative values in the solution of Eq. 2.15 and thus Eq. 2.17 will not necessarily be satisfied. The proposed procedure is summarized in the following (Bard and Moore 1990):

Step 0. (Initialization) Let $k = 0$, $S_k^+ = \varnothing$, $S_k^- = \varnothing$, $S_k^0 = \{1, 2, \ldots, m + n_2\}$, and $f_{-1} = -\infty$.

Step 1. (Iteration k) Set $w_i = 0$ for $i \in S_k^+$ and $q_i = 0$ for $i \in S_k^-$. Solve Eq. 2.15. If the solution is infeasible, go to Step 5. Otherwise, let $k = k + 1$ and label the solution as (x_1^k, x_2^k, w^k). Go to Step 2.

Step 2. (Fathoming) If $f_1(x_1, x_2) \leq f_{-1}$, go to Step 5. Otherwise, go to the next step.

Step 3. (Branching) If $w_i^k q_i(x_1^k, x_2^k) = 0$, $i = 1, \ldots, m + n_2$, go to Step 4. Otherwise, select i for which $w_i^k q_i(x_1^k, x_2^k)$ is the largest and label it as i_1. Let $S_k^+ arrow S_k^+ \cup \{i_1\}$, $S_k^0 arrow S_k^0 \backslash \{i_1\}$, and $S_k^- arrow S_k^-$, append i_1 to P_k, and go to Step 1.

Step 4. (Updating) $f_{-1} = f_1(x_1^k, x_2^k)$. Go to the next step.

Step 5. (Backtracking) If no live node exists, go to Step 6. Otherwise, branch to the newest live vertex and update S_k^+, S_k^-, S_k^0 and P_k. Go to Step 1.

Step6. (Termination) If $f_{-1} = -\infty$, there is no feasible solution to the BLPP. Otherwise, declare the feasible point associated with f_1 the optimal solution.

A live node is one associated with a sub-problem that has not yet been fathomed at either Step 1 due to unfeasibility or at Step 2 due to bounding. To illustrate the algorithm, we follow the example used by Bard and Moore (1990) in the following.

Example 2.5 Consider the linear two-level problem (Candler and Townsley 1982; Bard and Moore 1990):

$$\max_x f_1 = 8x_1 + 4x_2 - 4y_1 + 40y_2 + 4y_3$$
where x solves:
$$\max_y f_2 = -x_1 - 2x_2 - y_1 - y_2 - 2y_3$$
subject to:
$$y_1 - y_2 - y_3 \geq -1$$
$$-2x_1 + y_1 - 2y_2 + 0.5y_3 \geq -1$$
$$-2x_2 - 2y_1 + y_2 + 0.5y_3 \geq -1$$
$$x_1, x_2, y_1, y_2, y_3 \geq 0$$

The transformed problem without the complementary slackness product terms is:

$$\begin{aligned}
&\max f_1 = 8x_1 + 4x_2 - 4y_1 + 40y_2 + 4y_3 \\
&\text{subject to:} \\
&\quad y_1 - y_2 - y_3 \geq -1 \\
&\quad -2x_1 + y_1 - 2y_2 + 0.5y_3 \geq -1 \\
&\quad -2x_2 - 2y_1 + y_2 + 0.5y_3 \geq -1 \\
&\quad -w_1 - w_2 + w_3 - w_4 = -1 \\
&\quad w_1 + 2w_2 - w_3 - w_5 = -1 \\
&\quad w_1 - 0.5w_2 - 0.5w_3 - w_6 = -2 \\
&\quad x_1, x_2, y_1, y_2, y_3, w_1, w_2, w_3, w_4, w_5, w_6 \geq 0
\end{aligned}$$

and the complementary slackness product terms are:

$$\begin{aligned}
&w_1(y_1 - y_2 - y_3 + 1) = 0 \\
&w_2(-2x_1 + y_1 - 2y_2 + 0.5y_3 + 1) = 0 \\
&w_3(-2x_2 - 2y_1 + y_2 + 0.5y_3 + 1) = 0 \\
&w_4 y_1 = 0 \\
&w_5 y_2 = 0 \\
&w_6 y_3 = 0
\end{aligned}$$

As pointed out by Bard and Moore (1990), there are six of these product terms and thus, direct search approach needs to solve $2^7 - 1 = 127$ sub-problems. By using the proposed branch-and-bound approach, the optimal solution was reached by the fourth iteration and confirmed after ten sub-problems were examined.

The actual branch-and-bound tree for solving this problem is reproduced in Fig. 2.4 from Bard and Moore (1990). The initial feasible solution for the transformed problem without the complementary slackness product terms is:

$$\begin{aligned}
&X = (x_1, x_2) = (0, 0) \\
&Y = (y_1, y_2, y_3) = (1.5, 1.5, 1) \\
&W = (w_1, w_2, w_3, w_4, w_5, w_6) = (0, 0, 0, 1, 1, 2) \\
&F_1(x, y) = 58
\end{aligned}$$

The above results should satisfy Step 2 since f_1 is set at $-\infty$ at the initiation step. However, it does not satisfy Eq. (2.17). Thus, at Step 3, select the highest value of the complementary slackness product term, which tum out to be $w_6 q_6 = 2$ with $i_1 = 6$. We obtain, at the first level, $S_1^+ = \{6\}$, $S_1^- = 0$, $S_1^0 = (1, 2, 3, 4, 5)$, $P_1 = 6$. The next level turns out to be $i = 5, 4$. Following the branch-and-bound procedure, the decision tree shown in Fig. 2.4 is produced.

The optimum results obtained are:

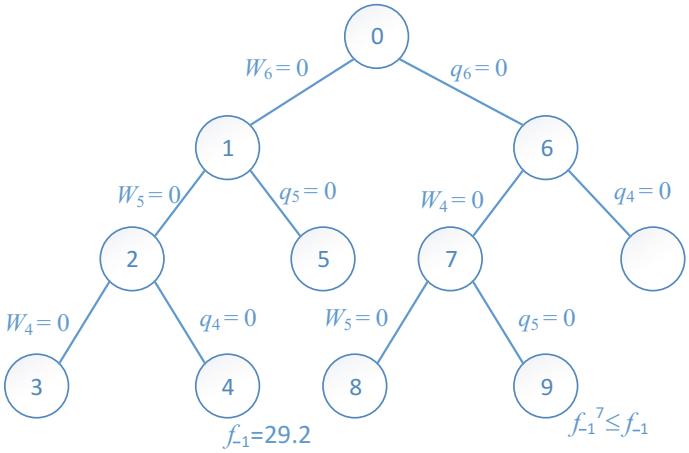

Fig. 2.4 Branch-and-bound tree, Example 2.5 (Bard and Moore 1990)

$$
\begin{aligned}
X &= (0, 0.9) \\
Y &= (0, 0.6, 0.4) \\
W &= (0,\ 1,\ 3,\ 6,\ 0,\ 0) \\
F_1 &= 29.2
\end{aligned}
$$

Bard and Moore (1990) solved a wide range of problems containing up to 60 upper-level variables, 40 lower-level variables, and 40 constraints for the linear case. They also examined the influences of the various variables and factors such as the number of lower level decision variables, ill-conditioned problems, and different branching rules. These investigators concluded that the proposed branch- and-bound approach compared favorably, both in performance and in robustness, with virtually all other contenders at that time. Furthermore, due to the very general nature of the branch-and-bound approach, the basic algorithm can be extended so that it can be used to solve non-linear problems. However, Ben-Ayed (1993) stated that, for large-scale problems, the efficiency of the approach is still constrained by the exponential growth of the branch-and-bound tree.

2.3.3.2 Algorithm of Hansen et al.

Hansen et al. (1992) formed a new branch-and-bound algorithm by examining the tightness of the follower's constraint and based on which to form the branch-and-bound tree and to fathom or simplify the sub-problem. We shall follow Hansen et al. (1992) in the following discussions and the numerical example.

For the ease of discussion, consider the following linear BLPP with constraints added to the upper-level problem:

$$\max_x f_1(x, y) = c_{11}x + c_{12}y \tag{2.18}$$

subject to:

$$B_1x + B_2y \leq d$$

$$x \geq 0$$

where y solves:

$$\max_y f_2(x, y) = c_{21}x + c_{22}y \tag{2.19}$$

subject to:

$$A_1x + A_2y \leq b$$

$$y \geq 0$$

where x and y are n_1- and n_2-dimensional vectors, respectively, and B_1 and A_1 are $m_1 \times n_1$- and $m_2 \times n_2$-dimensional matrices, respectively. For simplicity, the transpose sign T for the coefficients of the objective functions have been omitted. The above model contains first-level constraints that are binding only for the leader but also influenced by the decision of the follower. In addition to the necessary theory for the algorithm, these investigators also proved that the problem represented by Eqs. 2.18 and 2.19 is strongly NP-hard. The authors first introduced the 0–1 Boolean variable α_i, which equals 1 if constraint i is tight or is an active constraint, and equals to 0 otherwise. The following necessary conditions for optimality was established:

Theorem 2.1
In any rational solution to the BLPP represented by Eqs. (2.18) and (2.19), the tightness of the constraints in the follower's sub-problem is such that:

$$\sum_{i|[A_2]_{ij}} \alpha_i \geq 1, \quad \text{if } [c_{22}]_j > 0 \tag{2.20}$$

$$\sum_{i|[A_2]_{ij}<0} \alpha_i + \alpha_{m^2+j} \geq 1, \quad \text{if } [c_{22}]_j < 0 \tag{2.21}$$

for $j = 1, 2, \ldots, n_2$. Furthermore, the investigators also proved that in any optimal solution to this BLPP problem, the tightness of the constraints in the follower's sub-problem is such that Conditions 2.20 and 2.21 are satisfied for all $j \in (1, 2, \ldots, n_2)$ according to whether $c_{22} > 0$ and $c_{22} < 0$, respectively.

If the ith constraint is tight, that is $\alpha_i = 1$, then the ith constraint of the lower becomes an equality and it can be eliminated if it is a non-negative constraint. However, this approach does not reduce the structure constraints. Obviously, in case all the variables, x_2, for the lower level are eliminated, the problem reduced to a one-level problem and the objective function of the lower level can be ignored.

If $\alpha_i = 0$, the ith constraint in the follower's problem becomes a strict inequality if it is a non-negativity constraint. From the complementary slackness theory, the ith variable of the dual of the follower's problem must be equal to zero. Thus, the dual problem will be solved.

Based on Eqs. 2.20 and 2.21, a branch-and-bound algorithm with depth- first search was formulated. When a branch corresponding $\alpha_i = 0$ is explored, the corresponding α_i is deleted in Eqs. 2.20 and 2.21. When branching is done on the values of α_i, at most $2^{m2+n2+1} - 1$ sub-problems will be generated. This number can be reduced when $\alpha_i = 0$.

The bounds for the branch-and-bound algorithm are obtained by solving the relaxed linear programming problem. During the iteration, deleting the objective function of the follower, we obtain the leader relaxation (LR) problem. Solving this linear programming, or solving the LR problem, we obtain the upper bound.

The branch-and-bound algorithm is essentially based on the repeated solving three linear programming problems, namely, the leader's relaxed (LR) problem, the follower's relaxed (FR) problem, and the follower's sub-problem (FS). The relaxed problems are not the original problem as listed in Eqs. 2.18 and 2.19 but are problems during the iteration, where some or all of the Y have been eliminated and the initial constraints have been separated into tight, loose, or undecided. If the follower's objective function is deleted from this iterated BLPP, we obtain the LR problem. If the problem is consisted only of the iterated follower's problem with fixed or given x, we obtain the FR problem. The FS is consisted of original follower's problem, Eq. 2.19, with fixed or given x.

A subset of logical relation R is defined as:

$$\begin{aligned} R &= r_k, \quad k \in K \\ r_k &\equiv \sum_{i \in I_k} \alpha_i \geq 1, \quad I_k \subseteq \{1, 2, \ldots, m_2 + n_2\} \end{aligned} \tag{2.22}$$

where r_k is called a logical relation. The algorithm can now be summarized in the following (Hansen et al. 1992).

Step 1. Initialization. Obtain an initial solution $(x_{\text{opt}}, y_{\text{opt}})$ by a heuristic method. If no heuristic solution, use the best guessed values as $(x_{\text{opt}}, y_{\text{opt}})$ and set $z_{o\text{pt}} = -\infty$.

Step 2. First direct optimality test. Solve LR problem. Let the solution be represented by (x_L^*, y_L^*). If $z_L^* = c_{11}x_L^* + c_{12}y_L^* \leq z_{o\text{pt}}$, go to Step 13. Otherwise, go to the next step.

Step 3. First direct feasibility test. Solve the dual of FR(x) problem. If it has no feasible solution, go to Step 13. Otherwise, go to the next step.

Step 4. Direct resolution test, first part. Check if the solution of LR, (x_L^*, y_L^*) obtained in Step 2 is rational for the current sub-problem. Solve FR (x_L^*) and let the solution be y_F^*. If $c_{12}y_F^* = c_{12}y_L^*$, then (x_L^*, y_L^*) is a rational solution. Go to the next step. Otherwise, go to Step 6.

Step 5. Direct resolution test, second part. Check if the LR solution is also rational for the initial problem. Solve FS(x_L^*) and let the solution be y_{FS}^*. If $c_{12}y_{FS}^* = c_{12}y_L^*$, then (x_L^*, y_L^*) is a rational solution. Update $(x_{\text{opt}}, y_{\text{opt}})$ and $z_{o\text{pt}}$. If $z_{o\text{pt}} < c_{11}x_L^* + c_{12}y_L^*$, go to Step 13. Otherwise go to Step 6.

Step 6. Second direct optimality test. Compute all the penalties p_i for strictly positive slack variables in the iteration tableau of LR. Set p_i equal to 0 for all other variables. Then for all k such that $r_k \in R$, compute:

$$\pi_k = \min_{i \in \in} p_i$$

If $z_{opt} \geq z_L^* - \max_k \pi_k$, go to Step 13. Otherwise, go to the next step.

Step 7. Second direct feasibility test. If LR is infeasible, go to Step 13.

Step 8. First conditional optimality test. Consider again LR with the penalty p_i. For all i such that
If $z_{opt} \geq z_L^* - p_i$, fix α_i at 0 and update R.

Step 9. Third direct optimality test. If R contains a relation r_k such that $\alpha_j = 0$ for $j \in I_k$, go to Step 13.

Step 10. Relational optimality test. For all remaining y_j appearing in $f_2(x, y)$, add to R the logical relations (2.20) or (2.21), if they are non-redundant. Eliminate from R those relations that have become redundant.

Step 11. Second conditional optimality test. If R contains a relation r_k such that $\alpha_j = 0$ for *all* $j \in I_k$ except for one index i, set the corresponding $\alpha_i = 1$. Return to Step 2

Step 12. Branching. Apply the selected branching rule to choose either a free variable α_I, or, a logical relation r_k, for which all variables α_j with $i \in I_k$ are free. In the former case, branching by fixing $\alpha_i = 1$. In the latter case, branching by fix the first variable α_i in r_k equal to 1. Return to Step 2.

Step 13. Backtracking. If branching took place on a variable, find the last α_i based on which branched. Set this $\alpha_i = 0$ and free the α_j fixed at 0 after α_i was fixed at 1. Otherwise, consider the last logical relation r_k for which the branches explored are less than $m_1 + n_2$. If there is no such variable or logical relation, stop. Otherwise, update the current sub-problem and return to Step 2.

To illustrate the algorithm, consider the following example, which was solved by Hansen et al. (1992).

Example 2.6

$$\max_x f_1 = 8x_1 + 4x_2 - 4y_1 + 40y_2 + 4y_3$$

subject to:

$$x_1 + 2x_2 - y_3 \leq 1.3$$
$$x_1 \geq 0$$
$$x_2 \geq 0$$

$$\max_y f_2 = -2y_1 - y_2 - 2y_3$$

subject to:

$$-y_1 + y_2 + y_3 \leq 1 \qquad (\alpha_1)$$
$$4x_1 - 2y_1 + 4y_2 - y_3 \leq 2 \qquad (\alpha_2)$$
$$4x_2 + 4y_1 - 2y_2 - y_3 \leq 2 \qquad (\alpha_3)$$
$$y_1 \geq 0 \qquad (\alpha_4)$$
$$y_2 \geq 0 \qquad (\alpha_5)$$
$$y_3 \geq 0 \qquad (\alpha_6)$$

Step 1. The initial solution is obtained by using the following equation:

$$\max_{x,y} z = \lambda c_{11} x + (1 - \lambda) c_{22} y$$

where $\lambda = (n_1 + n_2)/(n_1 + 2n_2) = 5/7$
thus, $(x_{opt}, y_{opt}) = (1.5, 0, 1, 0, 2)$ and $z_{opt} = 9$

Step 2. Solve LR, obtain $(x_L^*, y_L^*) = (0, 0, 1.5, 1.5, 1)$ with $z_L^* = 58$ which is larger than $z_{opt} = 9$.

Step 3. Since no constraint has imposed, it is feasible.

Step 4. Solve FR, obtain $y_L^* = (0, 0, 0)$. Since $c_{12} y_L^* = -6.5 \neq y_L^* = 0$, This solution is not rational. Go to Step 6.

Step 6. The penalties p_i associated with the follower's constraints are:

$$p_1 = 0,\ p_2 = 0,\ p_3 = 0,\ p_4 = 28.8,\ p_5 = 42,\ p_6 = 22$$

Step 10. The set of R logical conditions are:

$$\alpha_1 + \alpha_2 + \alpha_3 \geq 1$$
$$\alpha_3 + \alpha_5 \geq 1$$
$$\alpha_2 + \alpha_3 + \alpha_6 \geq 1$$

Step 12. Branch on the second relation. Since $p_5 > p_3$, first consider $p_5 = 1$. This leads to $y_2 = 0$ and the problem reduced to:

$$\begin{aligned}
&\max_x f_1 = 8x_1 + 4x_2 - 4y_1 + 4y_3\\
&\text{subject to:}\\
&\quad x_1 + 2x_2 - y_3 \le 1.3\\
&\quad x_1 \ge 0\\
&\quad x_2 \ge 0\\
&\max_y f_2 = -2y_1 - y_2 - 2y_3\\
&\text{subject to:}\\
&\quad -y_1 + y_3 \le 1\\
&\quad 4x_1 - 2y_1 - y_3 \le 2\\
&\quad 4x_2 + 4y_1 - y_3 \le 2\\
&\quad y_1 \ge 0\\
&\quad y_3 \ge 0
\end{aligned}$$

Step 2. The solution of the LR problem is $(x_L, y_L^*) = (1.5, 0, 1, 2)$ with $z_L^* = 16 \le z_{opt}$. The incumbent solution remains $z_{opt} = 16$ and $(x_{opt}, y_{opt}) = (1.5, 0, 1, 2)$.

Step 13. Backtracking leads to the branches: $\alpha_5 = 0, \alpha_3 = 1$. Variable y_1 is eliminated. The sub-problem is reduced to:

$$\begin{aligned}
&\max_x f_1 = 8x_1 + 8x_2 + 38y_2 + 3y_3 - 2\\
&\text{subject to:}\\
&\quad x_1 + 2x_2 - y_3 \le 1.3\\
&\quad x_1 \ge 0\\
&\quad x_2 \ge 0\\
&\max_y f_2 = 2x_2 - 2y_2 - 2.5y_3 - 1\\
&\text{subject to:}\\
&\quad x_2 + 0.5y_2 + 0.75y_3 \le 1.5 \qquad (\alpha_1)\\
&\quad 2x_1 + x_2 + 1.5y_2 - 0.75y_3 \le 1.5 \qquad (\alpha_2)\\
&\quad 8x_2 - 0.5y_2 - 0.25y_3 \le 0.5 \qquad (\alpha_4)\\
&\quad y_2 \ge 0 \qquad (\alpha_5)\\
&\quad y_3 \ge 0 \qquad (\alpha_6)
\end{aligned}$$

Tightness of the fourth constraint, $\alpha_4 = 1$ allows y_2 to be eliminated. The sub-problem now becomes:

$$\max_x f_1 = 8x_1 + 84x_2 - 16y_3 - 40$$
subject to:
$$x_1 + 2x_2 - y_3 \leq 1.3$$
$$x_1 \geq 0$$
$$x_2 \geq 0$$
$$\max_y f_2 = -2x_2 - 1.5y_3 + 1$$
subject to:
$$2x_2 + 0.5y_3 \leq 2 \qquad (\alpha_1)$$
$$2x_1 + 4x_2 - 1.5y_3 \leq 3 \qquad (\alpha_2)$$
$$-2x_2 + 0.5y_3 \leq -1 \qquad (\alpha_3)$$
$$y_3 \geq 0 \qquad (\alpha_4)$$

The new logical relation is:

$$\alpha_2 + \alpha_6 \geq 1$$

Solving the current LR problem, we obtain: $(x_L, y_L^*) = (0, 0.833, 0.466)$ with $z_L^* = 26.73 > z_{opt} >$ and $p_1 = 0$, $p_2 = 8.333$, $p_6 = 12.333$. Solve FR yields: $y_L^* = 0.3555$ and $c_{12}y_F = -1.3 \neq c_{12}y_L = -1.466$. Thus, (x_L, y_L^*) is not rational.

According to Step 8, penalty $p_6 = 12.33$, which is such that $z_{opt} > z_L^* - p_6$. Thus, $\alpha_6 = 0$. The set R is now reduced to $\alpha_2 \geq 1$. Tightness of the second constraint enables y_3 to be removed. Thus, the problem now becomes:

$$\max_x f_1 = -13.33x_1 + 41.33x_2 - 8$$
subject to:
$$-x_1 - 2x_2 \leq -2.1$$
$$x_1 \geq 0$$
$$x_2 \geq 0$$
$$\max_y f_2 = -2x_2 - 6y_3 + 4$$
subject to:
$$2x_1 + 10x_2 \leq 9 \qquad (\alpha_1)$$
$$x_1 - x_2 \leq 0 \qquad (\alpha_5)$$
$$-2x_2 - 4x_2 \leq -3 \qquad (\alpha_6)$$

Solving the current LR problem, we obtain $x_L^* = (0.5, 0.8)$ with $z_L^* = 18.4$. Solve the FS, obtain $y_L^* = (0, 2, 8)$ with $c_{12}y_L^* = c_{12}y_F^* = -1.8$. Thus, the solution (0.5, 0.8, 0, 2, 0.8) is rational for the initial problem. Update the problem and backtracking discovers the current solution is the optimal solution. Thus, the problem is solved by only examining three nodes.

Hansen et al. (1992) carried out a fairly extensive numerical experiment. Problems with up to 150 constraints, 250 variables controlled by the leader and 150 variables controlled by the follower were solved. Experiments to compare the current approach with that of Bard and Moore (1990) and Judice and Faustino (1992) were also carried out. The current approach always outperforms than that of Bard and Moore.

2.3.4 Penalty-Function Approach

Based on the fact that when the duality gap is reduced to zero, the optimal solution of the problem will be reached; Anandalingam and White (1990) proposed a penalty-function approach by appending this gap to the upper-level objective function with a penalty. This structure leads to the decomposition of the composite problem into a series of linear programs and thus an efficient algorithm can be developed. With certain assumptions, these investigators derived the necessary theories for this approach. Again, we shall not go into details of the theoretical developments. The following discussion follows the work of Anandalingam and White (1990).

Given the upper-level control vector x_1, the lower-level problem, Eq. (2.2), is reduced to the following single-level regular linear programming problem:

$$\begin{aligned} &\max c_{22}^T x_2 \\ &\text{subject to:} \\ &\quad A_2 x_2 \le b - A_1 x_1 \\ &\quad x_2 \ge 0 \end{aligned} \tag{2.23}$$

The dual problem of the above problem is:

$$\begin{aligned} &\min w^{\mathrm{T}}(b - A_1 x_1) \\ &\text{subject to:} \\ &\quad w^{\mathrm{T}} A_2 \ge c_{22} \\ &\quad w \ge 0 \end{aligned} \tag{2.24}$$

where w is the dual row vector. The optimal solution lies in the interval $[c_{22}^{\mathrm{T}} x_2, w^{\mathrm{T}}(b - A_1 x_1)]$. That is, the optimum lies between the maximum of the primal and the minimum of the dual. When the duality gap given by the difference:

$$\Pi(x_1, x_2, w) = [w^{\mathrm{T}}(b - A_1 x_1) - c_{22}^{\mathrm{T}} x_2]$$

equals to zero, the lower-level optimal solution would be reached. Thus, use the above difference as a penalty term, the following overall problem can be obtained:

$$\max_{x1,x2,w} F(x_1, x_2, w, K) = c_{11}^{\mathrm{T}} x_1 + c_{12}^{\mathrm{T}} x_2 - K[w^{\mathrm{T}}(b - A_1 x_1) - c_{22}^{\mathrm{T}} x_2]$$
subject to:
$$A_1 x_1 + A_2 x_2 \leq b$$
$$w^{\mathrm{T}} A_2 \geq c_{22}$$
$$x_1, x_2, w \geq 0 \tag{2.25}$$

where K is a large positive constant and the above expression will reach optimum when $[w^{\mathrm{T}}(b - A_1 x_1) - c_{22}^{\mathrm{T}} x_2]$ approaches zero.

By assuming the feasible region for the primal and dual vectors, x_1, x_2, and w, to form nonempty bounded polyhedrons, Anandalingam and White (1990) developed an algorithm that is able to converge to the optimum by discretely increasing the value of the penalty K starting with $K = 0$. To form the search procedure, the algorithm requires that the following formulation to be convex:

$$\theta(w, K) = \max_{x1,x2} F(x_1, x_2, w, K)$$
subject to:
$$A_1 x_1 + A_2 x_2 \leq b$$
$$x_1, x_2 \geq 0 \tag{2.26}$$

Since the function $\theta(w, K)$ is convex, a solution to the problem:

$$\max_w \theta(w, K)$$
subject to:
$$w^{\mathrm{T}} A_2 \geq c_{22}$$
$$w \geq 0 \tag{2.27}$$

can be obtained by searching the extreme comers of the feasible set of the dual variable w. Thus, the algorithm essentially is, at each fixed value of K, searching the vertexes of the polyhedron for the dual variable w. After the desirable value has been obtained at this fixed K, the value of K can be increased and the search is performed again until the duality gap reaches zero. Based on the convexity of the function $\theta(w, K)$, the optimality condition at any iteration i can be obtained from the following parametric function with K and w as parameters:

$$\chi(w^i, K) = \min_w (w - w^i)(b - A_1 x_1(w^i, K))$$
subject to:
$$w^{\mathrm{T}} A_2 \geq c_{22}$$
$$w \geq 0 \tag{2.28}$$

Thus, Anandalingam and White (1990) proposed the following procedure with a small increment of K at each cycle:

Step 0. (Initialization) Set $i = 0$ and $K = 0$. Choose w^0 within the feasible region, and choose a small step size λ. Go to Step 1.

Step 1. Solve the following problem with fixed values of w and K to obtain $\{x_1(w^i, K), x_2(w^i, K)\}$:

$$\begin{aligned}
&\max_{x1,x2}\{c_{11}^{\mathrm{T}}x_1 + c_{12}^{\mathrm{T}}x_2 - K[w^{\mathrm{T}}(-A_1x_1) - c_{22}^{\mathrm{T}}x_2]\}\\
&\text{subject to:}\\
&\quad A_1x_1 + A_2x_2 \le b\\
&\quad w^{\mathrm{T}}A_2 \ge c_{22}\\
&\quad x_1, x_2, w \ge 0
\end{aligned} \tag{2.29}$$

Note that the maximization is with respect to x_1 and x_2 only.

Step 2. Solve Eq. (2.29) to obtain $w^*(w^i, K)$.

Step 3. (Optimality Test)

Step 3.1. If $\chi(w^i, K) < 0$, then set $w^{i+1} = w^*(w^i, K)$ and $i = i + 1$. Go to Step 1.

Step 3.2. If $\chi(w^i, K) \ge 0$, and $\pi\{x_1(w^i, K), x_2(w^i, K), w^i\}$, then set $K = K + \lambda'$ and $i = 0$. Go to Step 1.

Step 3.3. If $\chi(w^i, K) \ge 0$, and $\pi\{x_1(w^i, K), x_2(w^i, K), w^i\} = 0$, then optimality is reached. The optimal solution is $\pi\{x_1(w^i, K), x_2(w^i, K), w^i\}$, and stop.

To illustrate the above procedure, the same linear two-level programming problem solved by Anandalingam and White (1990) is used in the following example.

Example 2.7 In the following discussions, we follow the solution obtained by Anandalingam and White (1990).

$$\begin{aligned}
&\max_{x1} f_1 = x_1 + 3x_2\\
&\text{where } x_2 \text{ solves:}\\
&\max_{x2} f_2 = x_1 + 3x_2\\
&\quad \text{subject to:}\\
&\qquad -x_1 - 2x_2 \le -10\\
&\qquad x_1 - 2x_2 \le 6\\
&\qquad 2x_1 - x_2 \le 21\\
&\qquad x_1 + 2x_2 \le 38\\
&\qquad -x_1 + 2x_2 \le 18\\
&\qquad x_1, x_2 \ge 0
\end{aligned}$$

For convenience, X is used to denote the above constraint set. The dual constraints of the lower-level problem are:

$$\begin{aligned} W &= (w_1, w_2, w_3, w_4, w_5)^{\mathrm{T}} \\ &= -2w_1 - 2w_2 - w_3 + 2w_4 + 2w_5 \geq -3 \\ w_i &\geq 0, 1 = 1, \ldots, 5 \end{aligned}$$

The solution procedure proceeds as follows (Anandalingam and White 1990):

Iteration 1.

Step 0. (Initialization) Let $i = 0$, $K = 0$, $w^0 = (1.5, 0, 0, 0, 0)$ and $\lambda' = 0.1$.

Step 1. To solve Eq. (2.29), we first obtain the coefficients for the variables in the objective function. For x_1 we have $c_{11} + Kw^0A_1 = 1$ and for x_2, we have $c_{12} + Kc_{22} = c_{12} = 3$. Thus, Eq. (2.29) becomes:
$\max_{x1,x2}\{x_1 + 3x_2\}$ and subject to the constraint of X
The solution is $x_1(w^0, K = 0) = 10$ and $x_2(w^0, K = 0) = 14$. Go to Step 2.

Step 2. Substituting the values obtained in Step 1 into Eq. (2.28), we have:

$$\begin{aligned} &\chi(w^i, K) = \min_w\{-w_2 + w_3 + 28w_4 + 28w_5\} \\ &\text{subject to:} \\ &\quad -2w_1 - 2w_2 - w_3 + 2w_4 + 2w_5 \geq -3 \\ &\quad w_1, w_2, w_3, w_4, w_5 \geq 0 \end{aligned}$$

The solution for this problem is $(w_1, w_2, w_3, w_4, w_5) = (0, 1.5, 0, 0, 0)$.

Step 3. (Optimality Test)
$\chi(w^o, K = 0) = -6 < 0$, thus set $(w^{o+1})^{\mathrm{T}} = (w^1)^{\mathrm{T}} = (w^*, K = 0)^{\mathrm{T}} = (0, 1.5, 0, 0, 0)$ and $i \leftarrow i + 1 = 0 + 1 = 1$. Go to Step1.

Iteration 2.

Step 1. The coefficients are $(x_1, x_2) = (1, 3)$. Thus, Eq. (2.29) becomes:

$$\max_{x1x2}\{x_1 + 3x_2 | (x_1, x_2) \in X\}$$

The solution is $x_1(w^1, K = 0) = 10$ *and* $x_2(w^1, K = 0) = 14$. Go to Step 2.

Step 2. The solution remains unchanged. Go to Step 3.

Step 3. (Optimality Test)
$\chi(w^1, K = 0) = 0 \geq 0$, and duality gap $\pi\{x_1(w^1, K = 0), x_2(w^1, K = 0), w^1\} = 36 > 0$, then the optimal solution is not found. Set $K = 0 + 0.1 = 0.1$. Go to Step 1.

Iteration 3.

Step 1. The coefficients for x_1 and x_2 are $c_{11}+Kw^1A_1 = 1.15$ and $c_{12}+Kc_{22} = 2.7$, respectively. Thus, Eq. 2.29 becomes:

$$\max_{x1x2}\{1.15x_1 + 2.7x_2|(x_1, x_2) \in X\}$$

The solution is $x_1(w^1, K = 0) = 10$ and $x_2(w^1, K = 0.1) = 14$. Go to Step 2.

Step 2. The solution remains unchanged. Go to Step 3.

Step 3. (Optimality Test)
The result remains unchanged. Set $K = K + \lambda' = 0.1 + 0.1 = 0.2$. Go to Step 1.

Iteration 4.

Step 1. $c_{11} + Kw^1A_1 = 1.3$ and $c_{12} + Kc_{22} = 2.4$. Thus, Eq. (2.29) becomes:

$$\max_{x1x2}\{1.3x_1 + 2.4x_2|(x_1, x_2) \in X\}$$

The solution is $x_1(w^1, K = 0.2) = 16$ and $x_2(w^1, K = 0.2) = 11$. Go to Step 2.

Step 2. Substituting the values obtained from Step 1 into Eq. 2.28, we have:

$$\chi(w^i, K) = \min_w\{6w_1 - 10w_2 - 11w_3 + 22w_4 + 34w_5 + 15\}$$

subject to:

$$-2w_1 - 2w_2 - w_3 + 2w_4 + 2w_5 \geq -3$$
$$w_1, w_2, w_3, w_4, w_5 \geq 0$$

The solution for this problem is $(w_1, w_2, w_3, w_4, w_5) = (0, 0, 3, 0, 0)$. Go to Step 3.

Step 3. (Optimality Test)
$x_1(w^1, K = 0.2) = -18 < 0$, then set $(w^{1+1})^{\mathrm{T}} = (w^2)^{\mathrm{T}} = (w^*, K = 0.2)^{\mathrm{T}} = (0, 0, 3, 0, 0)$ and $i \leftarrow i + 1 = 1 + 1 = 2$. Go to Step 1.

Iteration 5.

Step 1. $c_{11} + Kw^2A_1 = 2.2$ and $c_{12} + Kc_{22} = 2.4$. Thus, Eq. (2.29) becomes:

$$\max_{x1x2}\{2.2x_1 + 2.4x_2|(x_1, x_2) \in X\}$$

The solution is $x_1(w^2, K = 0.2) = 16$ and $x_2(w^2, K = 0.2) = 11$. Go to Step 2.

Step 2. The solution remains unchanged. Go to Step 3.

Step 3. (Optimality Test)
$\chi(w^2, K = 0.2) = 0$ and the duality gap $\pi\{x_1(w^2, K = 0.2), x_2(w^2, K = 0.2), w^2\} = 0$. Thus, optimality is reached. The optimal solution is $(x_1, x_2) = (16, 11)$. ■

Anandalingam and White (1990) also provided the computational results of 50 randomly selected problems. The proposed penalty-function algorithm marginally outperforms the grid-search algorithm (Bard 1982, 1983b) and easily outperforms the kth-best algorithm (Bialas and Karwab 1982, 1984). Furthermore, White and Anandalingam (1993) extend the procedure to ten steps to obtain the global optimal solution for the linear case. Computational results show that the new ten-step penalty-function approach is easily outperformed the kth-best algorithm, and slightly worse than branch-and-bound algorithm (Bard and Moore 1990).

Aiyoshi and Shimizus (1981a, b, 1984) proposed a penalty-function approach for solving the BLPP with non-linear objective functions. However, their penalty function is convex, which is much easier to solve than the resulting non- convex function from the KKT transformations.

2.4 Other Multi-level Programming Algorithms

Although not as efficient, some of the algorithms discussed in the previous sections can also be extended or modified for solving more complicated problems such as MLDPP, MLPP, quadratic MLPP, and even non-linear BLPP. In fact, several algorithms discussed earlier in this chapter were formulated for solving quadratic or linear-quadratic problems. Furthermore, to solve non-linear and discrete problems, approaches such as gradient descend, branch-and-bound technique, and cutting-plane algorithm have been developed. Some of these algorithms are discussed in this section.

2.4.1 Linear Bi-level Distributed Programming

A linear bi-level distributed programming problem (BLDPP) is characterized by one decision center at the top level and p divisions or decision centers at the bottom level with the assumption that decision units in the lower level are independent and under the control of the upper level. In other words, the leader or top level makes the decision first and the followers must make their decisions based on the leader's decision. Depending on the actual system, different degrees of cooperation between the followers can occur. In an organization, the divisions or followers are required to cooperate for the common good in addition to optimize individually. On the other

hand, many duo-ploy systems such as the traffic system, there is no cooperation between the traffic users, who are the followers, except under special situations. The same is true for most economic duo-ploy systems.

Let $f_{k_i}(x)$ represents the objective function of the i-th division at the k-th level and c_{kij} represents the cost coefficient of the decision variable x_j for the i-th division of the k-th level, with $k = 1, 2$ and $i = 1, \ldots, s_k$. Since there is only one division at the top level, thus $s_k = 1$. Assume p divisions at level 2, then $i = 1, \ldots, s_k$ and $s_2 = p$. The BLDPP problem can be represented as:

$$\max_{x_{11}} f_{11}(x) = \sum_j c_{11j}^T x_j \quad \text{(upper level)}$$

where $x_{21}, x_{22}, \ldots, x_{2p}$ solve:

$$\begin{cases} max_{x_{21}} f_{21}(x) = \sum_j c_{21j}^T x_j \\ \qquad \ldots \qquad\qquad\qquad \text{(lower level)} \\ max_{x_{2p}} f_{2p}(x) = \sum_j c_{2pj}^T x_j \end{cases}$$

subject to:

$$\sum_{k,i} A_{ki} x_{ki} \leq b, \quad k = 1, 2 \text{ and } i = 1, \ldots, s_k$$

$$x_j \geq 0, \quad j =, \ldots, n \tag{2.30}$$

where $j = 1, \ldots, n$ represents the j-th decision variable and x_j for $j = 2, \ldots, n$, are decisions of the p divisions in the lower level. Note that $p = n - 1$.

In theory, most of the algorithms discussed in the previous sections can be extended or modified for solving the linear BLDPP. However, in the practice, those algorithms can become very complex and difficult to solve. To illustrate the approaches, two algorithms for solving the linear BLDPP are discussed in the following.

Anandalingam (1988) proposed a procedure for solving the BLDPP by using the KKT optimality conditions and the mixed-integer approach of Fortuny-Amat and McCarl (1981). Another algorithm for solving the multiple agents system was developed by Anandalingam and Apprey (1991) based on a penalty-function approach, which was developed for BLPP by Anandalingam and White (1990). Anandalingam and Apprey also discussed the existence of Nash equilibrium under different conditions and methods for coordinating conflicting agents. The approach was used to solve a conflict resolution problem of an international river between India and Bangladesh.

2.4.1.1 Mixed-Integer Problem with Complementary Slackness

Using the KKT conditions, the lower-level problems can be transformed into the constraints of the upper-level problem. The transformed problem is a one-level optimization problem. Thus, the original nested optimization problem represented by Eq. (2.30) is reduced to the following traditional non-linear programming problem:

$$\begin{aligned}
&\max_{x_{11}} f_{11}(x) = \sum_j c_{11j}^T x_j \\
&\text{subject to:} \\
&\sum_{k,i} A_{ki} x_{ki} \leq b, \quad k = 1, 2 \text{ and } i = 1, \ldots, s_k \\
&w^T (A_{ki} x_{ki} - b = 0), \quad k = 1, 2 \text{ and } i = 1, \ldots, s_k \\
&\sum_{k,i} w A_{ki} > c_{2jj}, \quad j = 1, \ldots, p \\
&w, x_j \geq 0, \quad j = 1, \ldots, n
\end{aligned} \tag{2.31}$$

where $j = 1, \ldots, n$ represents the j-th decision variable and x_j for $j = 2, \ldots, n$, are decisions of the p divisions in the lower level. Note that $p = n - 1$, and w is a dual vector.

The non-linear problem was first reduced to mixed-integer problem by using the approach due to Fortuny-Amat and McCarl (1981) and was discussed in the previous chapter. Introducing the binary variable, $\eta \in \{0, 1\}$ and let q denote the original constraint matrix, the non-linear complementary slackness constraint:

$$w^T q = w^T (A_{ki} x_{ki} - b) = 0$$

is replaced by the following two inequalities:

$$\begin{aligned}
q &\leq M\eta \\
w &\leq (1 - \eta)M
\end{aligned}$$

where M is a large positive constant. Using the above inequalities to enforce the complementary slackness conditions, a much larger mixed-integer problem is formed. Let the second constraint set in Eq. (2.31):

$$w^T (A_{ki} x_{ki} - b) = 0, \quad k = 1, 2 \text{ and } i = 1, \ldots, s_k$$

be replaced by the two inequalities:

$$w \leq (1 - \eta)M \text{ and}$$

$$\sum_{k,i} A_{ki}x_{ki} - b \leq M\eta, \quad k = 1, 2 \text{ and } i = 1, \ldots, s_k$$

where η is a binary vector and Eq. (2.31) becomes:

$$\begin{aligned}
&\max_{x_{11}} f_{11}(x) = \sum_j c_{11j}^T x_j \\
&\text{subject to:} \\
&\quad \sum_{k,i} A_{ki}x_{ki} \leq b, k = 1, 2 \text{ and } i = 1, \ldots, s_k \\
&\quad w \leq (1 - \eta)M \\
&\quad \sum_{k,i} A_{ki}x_{ki} - b \leq M\eta \\
&\quad \sum_{k,i} wA_{ki} > c_{2jj}, j = 1, 2, \ldots, p \\
&\quad \eta \in \{0, 1\} \\
&\quad w, x_j \geq 0, j = 1, \ldots, n
\end{aligned} \tag{2.32}$$

where M is a large positive constant, $j = 1, \ldots, n$ represents the j-th decision variable, and x_j for $j = 2, \ldots, n$ are decisions of the p divisions in the lower level. Equation (2.32) represents a mixed-integer problem. The following example illustrates the procedure.

Example 2.8 A problem with two lower levels considered by Anandalingam (1988) is solved in the following.

$$\begin{aligned}
&\max_{x_1} f_1 = x_1 + y_1 + 2y_2 + y_3 \\
&\text{where } y_1, y_2, \text{ and } y_3 \text{ solve:} \\
&\quad \begin{cases} \max_{y_1} f_{21} = x_1 + 3y_1 + y_2 + y_3 \\ \max_{y_2} f_{22} = x_1 + y_1 + 3y_2 + y_3 \\ \max_{y_3} f_{23} = x_1 + y_1 + y_2 + 3y_3 \end{cases} \\
&\quad \text{subject to:} \\
&\qquad 3x_1 + 3y_1 \leq 30 \\
&\qquad 2x_1 + y_1 \leq 20 \\
&\qquad y_1 + y_3 \leq 15 \\
&\qquad x_1 + 2y_1 + 2y_2 + y_3 \leq 40 \\
&\qquad y_1 \leq 10 \\
&\qquad y_2 \leq 10 \\
&\qquad x_1, y_1, y_2, y_3 \geq 0
\end{aligned}$$

Applying the KKT conditions, we have:

$$
\begin{aligned}
&\max f_1 = x_1 + y_1 + 2y_2 + y_3 \\
&\text{subject to:} \\
&\quad w_1(3x_1 + 3y_1 - 30) = 0 \\
&\quad w_2(2x_1 + y_1 - 20) = 0 \\
&\quad w_3(y_1 - 10) = 0 \\
&\quad w_4(y_1 + y_1 - 15) = 0 \\
&\quad w_5(y_3 - 10) = 0 \\
&\quad w_6(x_1 + 2y_1 + 2y_2 + y_3 - 40) = 0 \\
&\quad 3w_1 + w_2 + 2w_6 = 3 \\
&\quad w_3 + w_4 + 2w_6 = 3 \\
&\quad w_4 + w_5 + w_6 = 3 \\
&\quad w_1, w_2, w_3, w_4, w_5, w_6 \geq 0 \\
&\quad x_1, y_1, y_2, y_3 \in Y
\end{aligned}
$$

where Y represents the constraint set of the original problem. Applying the mixed-integer approach of Fortuny-Amat and McCarl (1981), we finally obtain the following formulation:

$$
\begin{aligned}
&\max f_1 = x_1 + y_1 + 2y_2 + y_3 \\
&\text{subject to:} \\
&\quad w_1 \leq (1 - \eta) \\
&\quad 3x_1 + 3y_1 - 30 \leq M\eta_1 \\
&\quad w_2 \leq (1 - \eta_2) \\
&\quad 2x_1 + y_1 - 20 \leq M\eta_2 \\
&\quad w_3 \leq (1 - \eta_3) \\
&\quad (y_1 - 10) \leq M\eta_3 \\
&\quad w_4 \leq (1 - \eta_4) \\
&\quad (y_1 + y_3 - 15) \leq M\eta_4 \\
&\quad w_5 \leq (1 - \eta_5) \\
&\quad (y_3 - 10) \leq M\eta_5 \\
&\quad w_6 \leq (1 - \eta_6) \\
&\quad (x_1 + 2y_1 + 2y_2 + y_3 - 40)M\eta_6 \\
&\quad 3w_1 + w_2 + 2w_6 = 3 \\
&\quad w_3 + w_4 + 2w_6 = 3 \\
&\quad w_4 + w_5 + w_6 = 3
\end{aligned}
$$

$$\eta_1, \eta_2, \eta_3, \eta_4, \eta_5, \eta_6 \in \{0, 1\}$$
$$w_1, w_2, w_3, w_4, w_5, w_6 \geq 0$$
$$x_1, y_1, y_2, y_3 \in Y$$

where M is a large positive number. Using the LINGO programming code (1992), the above mixed-integer problem was solved. With $M = 10{,}000$, the solution obtained $f_{11}^* = 35$, $(x_1^*, y_1^*, y_2^*, y_3^*) = (5, 5, 10, 5)$, $f_{21}^* = -5$, $f_{22}^* = 15$, and $f_{23}^* = -5$.

2.4.1.2 Penalty-Function Approach

Anandalingdam and Apprey (1991) extended the penalty-function approach of Anandalingdam and White (1990) to solve the bi-Ievel multi-agents problem. The basic idea is to add every objective of lower-level agent to the objective of upper level so that the problem becomes:

$$\max_{x,w} z(x, w, K) = \sum c_{11j}^T x_j - K\{\sum_1 [w_1(b - A_{2i-}x_{2i-}) - \sum_j c_{21j}^T x_j]\}$$

subject to:

$$\sum_{k,i} A_{ki}x_{ki} \leq b, \quad k = 1, 2 \text{ and } i = 1, \ldots, s_k$$
$$w_l^T A_l \geq c_{2ll}, \quad l = 1, \ldots, p$$
$$w_l \geq 0, \quad l = 1, \ldots, p$$
$$x_j \geq 0, \quad j = 1, \ldots, n \tag{2.33}$$

where K is a large positive constant, $j = 1, \ldots, n$ represents the j-th decision variable, and x_j for $j = 2, \ldots, n$ are decisions of the p divisions in the lower level. Note again that $p = n - 1$.

Let the vectors X and W denote the primal feasible region and the dual feasible region, respectively. The dual feasible space is assumed to be non-empty and forms a bounded polyhedral, which is further assumed to be non-degenerate at each vertex. Then, for a given value of w and a fixed K, we can define:

$$\Theta(w, K) = \max_x Z(x, w, K)$$
subject to:
$$x \in X \tag{2.34}$$

$\Theta(w, K)$ is convex in its decision space, and a solution to the problem:

$$\min_w \Theta(w, K)$$
subject to:
$$w \in W \tag{2.35}$$

will occur at some w^* belonging to the extreme points of W.

Anandalingam and Apprey (1991) formulated the following algorithm for solving the BLDPP, which is similar to that of Anandalingam and White (1990) discussed in the previous section.

Step 0. Choose fairly large value of K, and $w^0 \in \{$the extreme points of $W\}$.
Step 1. Solve Eq. (2.34) to obtain $x(w^{0*}, K)$.
Step 2. Solve Eq. (2.35) to obtain w^{0*}. If $w^{0*} = w^0$, go to Step 3. Otherwise, set $w^0 = w^{0*}$ and go to Step 1.
Step 3. (Optimality Test)

Compute the duality gap:

$$\pi\{x(w^{0*}, K), w^0 \sum_1 [w_1^{0*}(b - A_{2i.}x_{2i.}) - \sum_j c_{21j}^T x_j]\}$$

If $\pi\{x(w^{0*}, K), w^{0*}\} = 0$, then optimality is reached. The optimum solution is $\{x(w^{0*}, K), w^{0*}\}$. Otherwise, set $K = K + \lambda'$ and go to Step 1.

2.4.2 *Linear Three-Level Programming Problem*

Because of the complexity of MLPPs, only three level problems will be discussed in this section. In fact, even with the increase of only one level, the solution of the problem is much more complicated compared to the BLPP. Thus, not much research has been carried out. One fruitful approach appears to be the interactive fuzzy decision making to be discussed in later Chapters.

The three level programming problem (TLPP), whether from the standpoint of the three-ploy Stackelberg behavior or from the interactive organizational behavior, is a very practical problem and encountered frequently in actual practice. For example, Cassidy et al. (1971) considered the distribution of federal budget. Federal government, the first level, allocates the money to the various states; the states, as the second level of managers, allocate money to the various cities and the city can further allocate to different projects. The behavior of the lower hierarchies are not within the control of the higher hierarchies except through legislation and existing laws, and thus the allocation of budget of the federal government has the some of the characteristics of the Stackelburg behavior.

The three level programming problem (TLPP) can be defined as a three-person, non-zero sum game with perfect information in which players move sequentially from top to bottom level. The TLPP can be formulated as:

$$\begin{aligned}
&\max_{x_1} f_1(x) = c_{11}^T x_1 + c_{12}^T x_2 + c_{13}^T x_3 \quad \text{(1st level)}\\
&\text{where } x_2 \text{ solves:}\\
&\max_{x_2} f_2(x) = c_{21}^T x_1 + c_{22}^T x_2 + c_{23}^T x_3 \quad \text{(2nd level)}\\
&\quad \text{where } x_3 \text{ solves:}\\
&\max_{x_3} f_3(x) = c_{31}^T x_1 + c_{32}^T x_2 + c_{33}^T x_3 \quad \text{(3rd level)}\\
&\qquad \text{subject to:}\\
&\qquad\quad (x_1, x_2, x_3) \in X = \{(x_1, x_2, x_3) | A_1 x_1 + A_2 x_2 + A_3 x_3 \leq b,\\
&\qquad\quad \text{and } x_1, x_2, x_3 \geq 0\}
\end{aligned} \tag{2.36}$$

where x_1, x_2, and x_3 are the controls or decisions for the 1st, 2nd, and 3rd levels, respectively.

Equation (2.36) represents a three-level hierarchy process, whose objective is to maximize $\boldsymbol{f} = (f_1, f_2, f_3)$ by choosing the decisions $\boldsymbol{x} = (x_1, x_2, x_3)$ in a top to bottom manner. In other words, the top-level player first attempts to maximize f_1 over the convex region $\boldsymbol{X}$ by selecting x_1 where $\boldsymbol{X}$ represents the set of the original constraints. The second level player then maximize f_2 based on the already decided x_1 by selecting x_2. The last level or third player maximize f_3 based on the decisions of x_1 and x_2 by selecting x_3. Thus, in a sense, the problem of the third or the lowest level player is the simplest because once x_1 and x_2 are decided, the problem for the third level is a simple one level programming problem. Consider the third level problem with x_1 and x_2 being fixed (Bard 1984b):

$$X_3^*(x_1, x_2) = \{x_3 | x_3 \text{ solves:} \max(c_{33}^T x_3 | x_3 \in S^3)\} \tag{2.37}$$

where

$$S^3 = \{x_3 | A_3 x_3 \leq b - A_1 x_1 - A_2 x_2, x_1, x_2, x_3 \geq 0\} \tag{2.38}$$

is called the level three inducible region and X_3^* is the level three rational reaction set in reaction to x_1 and x_2. After the third level, the second level player's problem with fixed with x_i can be formulated as:

$$X_2^*(x_2, x_3) = \{x_2 | x_3 \text{ solves:} \max(c_{22}^T x_2 + c_{23}^T x_3 | (x_2, x_3) \in S^2)\} \tag{2.39}$$

where

$$S^2 = \{(x_2, x_3) | A_2 x_2 + A_3 x_3 \leq b - A_1 x_1, x_1 \geq 0, x_2 \geq 0, x_3 \in X_3^*\} \tag{2.40}$$

is called the level two inducible region and X_2^* is the level two rational reaction set. Finally, the top level or level one player's problem can be formulated as:

$$\max(c_{11}^T x_1 + c_{12}^T x_2 + c_{13}^T x_3 | (x_1, x_2, x_3) \in S^1) \tag{2.41}$$

where

$$S^1 = \{(x_1, x_2, x_3) \in X | x_1 \geq 0, (x_2, x_3) \in X_2^*\} \tag{2.42}$$

is the level one inducible region. Thus, strictly speaking, the decision variables for the second and the third levels should be expressed as $x_2(x_1)$ and $x_3(x_2(x_1))$, respectively. Notice that the upper-level DMs may restrict the options available to the lower-level DMs and the lower MDs' decisions, although cannot directly influence the decisions of the upper levels, do eventually influence the overall and each individual DM's payoffs.

From Eqs. (2.37)–(2.42), we can see the basic problems connected with the solution of the TLPP, which is very similar to the BLPP except more complicated. For example, if either X_2^* or X_3^* or both are not singletons, multiple solutions may exist at the lower levels in the hierarchy. Thus, the higher levels may not obtain the best returns if the multi-ploy decision sequence is carried out.

Bard (1984b), by the combined use of the cutting-plane method and vertex search, proposed an algorithm for solving linear TLPPs. By examining a limited number of test problems, Bard concluded that the amount of work required finding a solution seems to grow with the square of the row dimension of the underlying polyhedral set. Wen and Bialas (1986) proposed a hybrid algorithm for solving the linear TLPP. This algorithm first adopts the kth-best algorithm, which was developed for solving BLPPs, to generate the kth-best extreme point by optimizing the first-level problem, and then using a complementary-pivot algorithm to check for feasibility of the second-level problem. The procedure may pose computational problem when the number of constraints is large. Thus, Anandalingam (1988) proposed a KKT Stackelberg-equilibrium approach to reduce the computational time. In addition, Anandalingam and Apprey (1991) provide an algorithm that combines the kth-best with the penalty-function approach for solving linear TLPPs.

Wang et al. (1994) suggested a method for generating non-dominated solutions for a three level decentralized system with multiple-objective headquarters. The approach is based on a revised, two-level simplex algorithm with the search of a domination tree.

2.4.2.1 Hybrid Extreme-Point Search Algorithm

The hybrid kth-best algorithm (Wen and Bialas 1986) solves the three-level programming problem by first generate the kth-best extreme point, which is obtained by maximizing f_1 over $\boldsymbol{X}$ Then a complementary-pivot algorithm is used to check the feasibility. Assume that the entire feasible region is bounded and has a finite number of extreme points, the solution procedure can be illustrated as follows (Wen and Bialas 1986):

Step 0. Set $k = 1$ and $\alpha' = 0$.

Step 1. Use the kth-best algorithm to find the kth-best solution of the first-level problem, which is:

$$\begin{aligned}&\max f_1(x) = c_{11}^T x_1 + c_{12}^T x_2 + c_{13}^T x_3\\&\text{subject to:}\\&\quad x \in X\end{aligned}$$

Let the solution obtained be represented as: $\underline{x}_{[K]} = (\underline{x}_1, \underline{x}_2, \underline{x}_3)$ and go to Step 2.

Step 2. (S^2 check) With the fixed values of $\underline{x}_2$ and $\underline{x}_3$, solving the following problem:

$$\begin{aligned}&\max f_3(x) = c_{31}^T x_1 + c_{32}^T \underline{x}_2 + c_{33}^T \underline{x}_3\\&\text{subject to:}\\&x \in X \cap \{x \in (x_1, \underline{x}_2, \underline{x}_3)\}\end{aligned}$$

Let the solution obtained be represented as: $(\bar{x}_1, \underline{x}_2, \underline{x}_3)$, where $\bar{x}_1$ is the newly obtained value. If $\bar{x}_1 \neq \underline{x}_1$ then $x_{[K]} \notin S^2$. Thus, $\underline{x}_{[K]}$ is not in S^3. Set $K = K + 1$ and go to Step 1. If $\bar{x}_1 = \underline{x}_1$ then $\underline{x}_{[K]} \in S^2$. Go to Step 3.

Step 3. (First best check) With fix value of $\underline{x}_3$, solving the following problem:

$$\begin{aligned}&\max\ f_2(x) = c_{21}^T x_1 + c_{22}^T x_2 + c_{23}^T \underline{x}_3\\&\text{subject to:}\\&x \in X \cap \{x \in (x_1, x_2, \underline{x}_3)\}\end{aligned} \tag{2.43}$$

Again, let the just obtained solution be represented by: $(x_1^*, x_2^*, \underline{x}_3)$. Check if $(x_1^*, x_2^*, \underline{x}_3)$ is in S^2 by solving the following problem:

$$\begin{aligned}&\max\ f_3(x) = c_{31}^T x_1 + c_{32}^T x_2 + c_{33}^T \underline{x}_3\\&\text{subject to:}\\&x \in X \cap \{x \in (x_1, x_2^*, \underline{x}_3)\}\end{aligned}$$

Obtain the optimal solution $(\bar{x}_1, x_2^*, \underline{x}_3)$. If $\bar{x}_1 = x_1^*$ then the first-best solution of Eq. (2.43) is in S^2. Go to Step 4. If $\bar{x}_1 \neq x_1^*$, then the first-best solution $\underline{x}_{[K]}$ of Eq. (2.43) is not in S^2. Set:

$$\alpha = c_{21}^T \underline{x}_1 + c_{22}^T \underline{x}_2 + c_{23}^T \underline{x}_3,$$

go to Step 5.

Step 4. (S^3-Check) If $(x_1^*, x_2^*) = (\underline{x}_1, \underline{x}_2)$, then $\underline{x}_{[K]} \in S^3$. Thus, $\underline{x}_{[K]}$ is the optimal solution of the three-level system, and the algorithm stops. Otherwise, $\underline{x}_{[K]} \notin S^3$. Set $K = K + 1$ and go to Step 1.

Step 5. (CP-Check) Solve the following problem by using the restricted basic entry procedure in the parametric complementary-pivot algorithm due to Bialas and Karwan (1984):

$$\begin{aligned} & A_1x_1 + A_2x_2 + u = b - A_3x_3 \\ & -\varepsilon Hx_1 + w^T A_2 - v = c_{22} \\ & c_{11}^T x_1 + c_{12}^T x_2 \geq \alpha' + \delta \\ & uw = 0 \\ & x_1^T v = 0 \\ & x_1, x_2, u, v, w \geq 0 \end{aligned} \tag{2.44}$$

where the matrix H may be any negative definite matrix, and ε is a suitably small positive scalar, and δ is a sufficiently small positive number. If we cannot find any feasible solution for Eq. (2.44), then $(\underline{x}_2, \underline{x}_3)$ is the optimal solution for the lower two-level problem for fixed $\underline{x}_2$, and hence $\underline{x}_{[K]} \in S^3$ and $\underline{x}_{[K]}$ is the optimal solution for the three-level system, and the algorithm stops. Otherwise, $\underline{x}_{[K]} \notin S^3$; set $K = K + 1$ and go to Step 1.
Let us illustrate the algorithm using the following example from Wen and Bialas (1986).

Example 2.9

$\max_{x_3} f_1 = -4x_1 + 3x_2 + 7x_3$ (1st level)
where x_1, x_2 solve:
  $\max_{x_2} f_2 = x_2$ (2nd level)
  where x_1 solve:
    $\max_{x_1} f_3 = x_1$ (3rd level)
    subject to:
      $x_1 + x_2 + x_3 \leq 3$
      $x_1 + x_2 - x_3 \geq 1$
      $x_1 - x_2 + x_3 \geq 1$
      $-x_1 + x_2 + x_3 \leq 1$
      $x_3 \leq 0.5$
      $x_1, x_2, x_3 \geq 0$

The first step of the hybrid algorithm is to solve the following linear program:

$$\begin{aligned} & \max f_1 = -4x_1 + 3x_2 + 7x_3 \\ & \text{subject to:} \\ & \qquad x_1 + x_2 + x_3 \leq 3 \end{aligned}$$

$$
\begin{aligned}
& x_1 + x_2 - x_3 \geq 1 \\
& x_1 - x_2 + x_3 \geq 1 \\
& -x_1 + x_2 + x_3 \leq 1 \\
& x_3 \leq 0.5 \\
& x_1, x_2, x_3 \geq 0
\end{aligned}
$$

The first-best solution obtained is $\underline{x}_{[1]} = (0, 1, 0)$. Using S^2-Check to determine if $\underline{x}_{[1]}$ is an element of S^2 which solves:

$$
\begin{aligned}
& \max f_3 = x_1 \\
& \text{subject to:} \\
& \quad x_1 + x_2 + x_3 \leq 3 \\
& \quad x_1 + x_2 - x_3 \geq 1 \\
& \quad x_1 - x_2 + x_3 \geq 1 \\
& \quad -x_1 + x_2 + x_3 \leq 1 \\
& \quad x_2 = 1 \\
& \quad x_3 = 0 \\
& \quad x_1 \geq 0
\end{aligned}
$$

In this case, the resulting solution $\underline{\boldsymbol{x}} = (2, 1, 0) \neq \underline{\underline{\boldsymbol{x}}}_{[1]}$. Hence, $\underline{\underline{\boldsymbol{x}}}_{[1]} \notin \boldsymbol{S}^2$. Return to Step 1 to find the next best solution.

Among the adjacent extreme points of $\underline{\underline{\boldsymbol{x}}}_{[1]}$, which are:$\{(1, 0, 0), (1, 2, 0), (0.75, 0.75, 0.5)\}$, the second best level-three solution is $\underline{\underline{\boldsymbol{x}}}_{[2]} = (0.75, 0.75, 0.5)$. Again the algorithm goes to Step 2 and rejects $\underline{\underline{\boldsymbol{x}}}_{[2]}$ according to $\underline{\underline{\boldsymbol{x}}}_{[2]} \notin \boldsymbol{S}^2$.

Among the candidates, $\{(1, 0, 0), (1, 2, 0), (1.25, 1.25, 0.5), (1, 0.5, 0.5)\}$, the third best solution of the level-three problem is (1.25, 1.25, 0.5). In Step 2, the result is $\underline{\underline{\boldsymbol{x}}}_{[3]} = \underline{\boldsymbol{x}}$, hence $\underline{\underline{\boldsymbol{x}}}_{[3]} \in \boldsymbol{S}^2$. In Step 3 the first-best check, the algorithm finds the optimal solution of the level-two problem by solving the following problem with fixed $x_3 = 0.5$:

$$
\begin{aligned}
& \max \ f_2 = x_2 \\
& \text{subject to:} \\
& \quad x_1 + x_2 + x_3 \leq 3 \\
& \quad x_1 + x_2 - x_3 \geq 1 \\
& \quad x_1 - x_2 + x_3 \geq 1 \\
& \quad -x_1 + x_2 + x_3 \leq 1 \\
& \quad x_3 = 0.5 \\
& \quad x_1, x_2 \geq 0
\end{aligned}
$$

with the resulting solution $\boldsymbol{x}^* = (1.25, 1.25, 0.5)$, which also belongs to $\boldsymbol{S}^2$ by solving the following problem and has $\underline{x}_1 = x_1^* = 1.25$:

$$\begin{aligned}
&\max f_3 = x_3 \\
&\text{subject to:} \\
&\quad x_1 + x_2 + x_3 \leq 3 \\
&\quad x_1 + x_2 - x_3 \geq 1 \\
&\quad x_1 - x_2 + x_3 \geq 1 \\
&\quad -x_1 + x_2 + x_3 \leq 1 \\
&\quad x_2 = 1.25 \\
&\quad x_3 = 0.5 \\
&\quad x_1 \geq 0
\end{aligned}$$

In Step 4 check shows that $\underset{=[3]}{x} = \boldsymbol{x}^*$, hence: $\underset{=[3]}{x} \in \boldsymbol{S}^3$. Thus, (1.25, 1.25, 0.5) is a global solution.

These investigators also carried out numerical experiments. They discovered that the problem is easier to solve as the percentage of variables controlled by the third level DM is increased.

2.4.2.2 Mixed-Integer Problem with Complementary Slackness

To solve the three-level hierarchical system represented by Eq. 3.7, first consider the lower two-level problem, which is composed by the lowest level and the second level In other words, we first ignore the highest or first level of the original problem. With a given $\boldsymbol{x}_1$, the bottom two levels can be represented as:

$$\begin{aligned}
&\max_{x2} f_2(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) = \boldsymbol{c}_{21}^T \boldsymbol{x}_1 + \boldsymbol{c}_{22}^T \boldsymbol{x}_2 + \boldsymbol{c}_{23}^T \boldsymbol{x}_3 \\
&\text{where } \boldsymbol{x}_3 \text{ solves:} \\
&\quad max_{x3} f_3(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) = \boldsymbol{c}_{31}^T \boldsymbol{x}_1 + \boldsymbol{c}_{32}^T \boldsymbol{x}_2 + \boldsymbol{c}_{33}^T \boldsymbol{x}_3 \\
&\quad \text{subject to:} \\
&\qquad A_2 x_2 + A_3 x_3 \leq b - A_1 x_1 \\
&\qquad x_2, x_3 \geq \boldsymbol{0}
\end{aligned} \tag{2.45}$$

Applying the KKT conditions to Eq. 3.16, we have:

$$\begin{aligned}
&\max_{x2,\,x3}\ f_2(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) = \boldsymbol{c}_{22}^T\boldsymbol{x}_1 + \boldsymbol{c}_{22}^T\boldsymbol{x}_2 + \boldsymbol{c}_{23}^T\boldsymbol{x}_3\\
&\text{subject to:}\\
&\quad \boldsymbol{A}_2\boldsymbol{x}_2 + \boldsymbol{A}_3\boldsymbol{x}_3 \le \boldsymbol{b} - \boldsymbol{A}_1\boldsymbol{x}_1\\
&\quad \boldsymbol{u}^T\boldsymbol{A}_3 = \boldsymbol{c}_{33}\\
&\quad \boldsymbol{u}^T(\boldsymbol{A}_2\boldsymbol{x}_2 + \boldsymbol{A}_3\boldsymbol{x}_3 + \boldsymbol{A}_1\boldsymbol{x}_1 - \boldsymbol{b}) = \boldsymbol{0}\\
&\quad \boldsymbol{x}_2,\ \boldsymbol{x}_3,\ \boldsymbol{u} \ge \boldsymbol{0}
\end{aligned} \tag{2.46}$$

where $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ are the controls or decisions for the 2nd and 3rd level DMs, respectively, and $\boldsymbol{u}$ is a dual vector. Now, consider the problem composed of the original first level and the above one-level problem. Applying again the KKT conditions to this problem, we finally obtain the following one-level transformed problem of the original three hierarchical system:

$$\begin{aligned}
&\max\ f_1(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) = \boldsymbol{c}_{31}^T\boldsymbol{x}_1 + \boldsymbol{c}_{22}^T\boldsymbol{x}_2 + \boldsymbol{c}_{23}^T\boldsymbol{x}_3\\
&\text{subject to:}\\
&\quad \boldsymbol{A}_1\boldsymbol{x}_1 + \boldsymbol{A}_2\boldsymbol{x}_2 + \boldsymbol{A}_3\boldsymbol{x}_3 \le \boldsymbol{b}\\
&\quad \boldsymbol{u}^T\boldsymbol{A}_3 = \boldsymbol{c}_{33}\\
&\quad \boldsymbol{w}^T\boldsymbol{A}_2 = \boldsymbol{c}_{22}\\
&\quad \boldsymbol{w}^T\boldsymbol{A}_3 = \boldsymbol{c}_{23}\\
&\quad \boldsymbol{w} - \boldsymbol{v} - \lambda\boldsymbol{u} = \boldsymbol{0}\\
&\quad \boldsymbol{v}^T(\boldsymbol{A}_1\boldsymbol{x}_1 + \boldsymbol{A}_2\boldsymbol{x}_2 + \boldsymbol{A}_3\boldsymbol{x}_3 - \boldsymbol{b}) = \boldsymbol{0}\\
&\quad \boldsymbol{u}^T(\boldsymbol{A}_1\boldsymbol{x}_1 + \boldsymbol{A}_2\boldsymbol{x}_2 + \boldsymbol{A}_3\boldsymbol{x}_3 - \boldsymbol{b}) = \boldsymbol{0}\\
&\quad \boldsymbol{x}_1,\ \boldsymbol{x}_2,\ \boldsymbol{u},\ \boldsymbol{w},\ \boldsymbol{v},\ \lambda \ge \boldsymbol{0}
\end{aligned} \tag{2.47}$$

where v and λ are Lagrange multipliers.

Although we only illustrated the transformation process for three hierarchies, it is obvious that it can be extended to four, five or many hierarchies. However, the problem will be increasingly complex and non-linear as the number of hierarchy increases.

Example 2.10 Consider the following three-level problem solve by Anandalingam (1988):

$$\begin{aligned}
&\max_{x1}\ f_1 = 7x_1 + 3x_2 + 4x_3 \quad (1\text{st level})\\
&\text{where } x_2, x_3 \text{ solves:}\\
&\quad \max_{x2}\ f_2 = x_2 \quad (2\text{nd level})\\
&\quad \text{where } x_3 \text{ solves:}\\
&\quad\quad \max_{x3}\ f_3 = x_3\ (3\text{rd level})\\
&\quad\quad \text{subject to:}\\
&\quad\quad\quad x_1 + x_2 + x_3 \le 3
\end{aligned}$$

$$
\begin{aligned}
& x_1 + x_2 - x_3 \geq 1 \\
& x_1 - x_2 + x_3 \geq 1 \\
& -x_1 + x_2 + x_3 \leq 1 \\
& x_3 \leq 0.5 \\
& x_1,\ x_2,\ x_3 \geq 0
\end{aligned}
$$

Applying the KKT conditions to the lower two levels, we have:

$$
\begin{aligned}
& \max_{x1}\ f_1 = 7x_1 + 3x_2 + 4x_3 \\
& \text{where } x_2, x_3 \text{ solves:} \\
& \quad \max_{x2}\ f_2 = x_2 \\
& \quad \text{subject to:} \\
& \qquad \boldsymbol{x} \in \boldsymbol{Y} \\
& \qquad w_1 + w_2 - w_3 + w_4 + w_5 = 1 \\
& \qquad w_1(x_1 + x_2 + x_3 - 3) = 0 \\
& \qquad w_2(-x_1 - x_2 + x_3 + 1) = 0 \\
& \qquad w_3(-x_1 - x_2 - x_3 + 1) = 0 \\
& \qquad w_4(-x_1 + x_2 + x_3 - 1) = 0 \\
& \qquad w_5(x_3 - 0.5) = 0 \\
& \qquad v w_1, w_2, w_3, w_4, w_5 \geq 0
\end{aligned}
$$

Applying the KKT conditions again to the above problem, we have:

$$
\begin{aligned}
& \max\ f_1 = 7x_1 + 3x_2 + 4x_3 \\
& \text{subject to:} \\
& \quad \boldsymbol{x} \in \boldsymbol{Y} \\
& \quad w_1 + w_2 - w_3 + w_4 + w_5 = 1 \\
& \quad w_1(x_1 + x_2 + x_3 - 3) = 0 \\
& \quad w_2(-x_1 - x_2 + x_3 + 1) = 0 \\
& \quad w_3(-x_1 - x_2 - x_3 + 1) = 0 \\
& \quad w_4(-x_1 + x_2 + x_3 - 1) = 0 \\
& \quad w_5(x_3 - 0.5) = 0 \\
& \quad w_1, w_2, w_3, w_4, w_5 \geq 0 \\
& \quad v_1(x_1 + x_2 + x_3 - 3) = 0 \\
& \quad v_2(-x_1 - x_2 + x_3 + 1) = 0 \\
& \quad v_3(-x_1 - x_2 - x_3 + 1) = 0 \\
& \quad v_4(-x_1 + x_2 + x_3 - 1) = 0
\end{aligned}
$$

$$v_5(x_3 - 0.5) = 0$$
$$(v_1 - v_2 - v_3 + v_4) + w_1\lambda_1 - w_2\lambda_2 - w_3\lambda_3 + w_4\lambda_4 = 1$$
$$v_1, v_2, v_3, v_4, \lambda_1, \lambda_2, \lambda_3, \lambda_4 \geq 0$$

Let:

$$u_1 = v_1 + w_1\lambda_1$$
$$u_2 = v_2 + w_2\lambda_2$$
$$u_3 = v_3 + w_3\lambda_3$$
$$u_4 = v_4 + w_4\lambda_4$$

then the one-level problem becomes:

$$\max f_1 = 7x_1 + 3x_2 + 4x_3$$
subject to:
$$\boldsymbol{x} \in \boldsymbol{Y}$$
$$w_1 + w_2 - w_3 + w_4 + w_5 = 1$$
$$w_1(x_1 + x_2 + x_3 - 3) = 0$$
$$w_2(-x_1 - x_2 + x_3 + 1) = 0$$
$$w_3(-x_1 - x_2 - x_3 + 1) = 0$$
$$w_4(-x_1 + x_2 + x_3 - 1) = 0$$
$$w_5(x_3 - 0.5) = 0$$
$$w_1,\ w_2,\ w_3,\ w_4,\ w_5 \geq 0$$
$$v_1(x_1 + x_2 + x_3 - 3) = 0$$
$$v_2(-x_1 - x_2 + x_3 + 1) = 0$$
$$v_3(-x_1 - x_2 - x_3 + 1) = 0$$
$$v_4(-x_1 + x_2 + x_3 - 1) = 0$$
$$v_5(x_3 - 0.5) = 0$$
$$u_1 - u_2 - u_3 + u_4 = 1$$
$$u_1 = v_1 + w_1\lambda_1$$
$$u_2 = v_2 + \underline{w}_2\lambda_2$$
$$u_3 = v_3 + w_3\lambda_3$$
$$u_4 = v_4 + w_4\lambda_4$$
$$v_1,\ v_2,\ v_3,\ v_4,\ \lambda_1,\ \lambda_2,\ \lambda_3,\ \lambda_4,\ u_1,\ u_2,\ u_3,\ u_4 \geq 0$$

The above problem can be transformed into a mixed-integer problem by using the Fortuny-Amat and McCarl transformation (1981), which was discussed in Sect. 3.1.1. Using the LINGO programming code and with $M = 10{,}000$, this mixed-

integer programming was solved. The solution obtained is $f_1^* = 8.5$ at $(x_1^*, x_2^*, x_3^*) = (1.5, 0, 0.5)$, with $f_2^* = 0,\ f_3^* = 0.5$. We also solved the problem with $M = 100{,}000$, 1,000,000, 10,000,000, and 100,000,000 and the solution remains unchanged.

2.4.2.3 Simplex-Cutting-Plane Algorithm

Bard (1984b) proposed a cutting-plane algorithm employing a vertex-search procedure to solve the linear TLPP. To obtain the algorithm, Bard (1984b) first considered the following two-level problem obtained using the KKT conditions:

$$\begin{aligned}
&\max_{x1}\ f_1(\boldsymbol{x}) = \boldsymbol{c}_{11}^T\boldsymbol{x}_1 + \boldsymbol{c}_{12}^T\boldsymbol{x}_2 + \boldsymbol{c}_{13}^T\boldsymbol{x}_3\\
&\text{where } x_2, x_3 \text{ solves:}\\
&\quad \max_{x2,\,x3}\ f_2(\boldsymbol{x}_1,\ \boldsymbol{x}_2,\ \boldsymbol{x}_3) = \boldsymbol{c}_{22}^T\boldsymbol{x}_1 + \boldsymbol{c}_{22}^T\boldsymbol{x}\boldsymbol{v}_2 + \boldsymbol{c}_{23}^T\boldsymbol{x}_3\\
&\quad \text{subject to:}\\
&\qquad \boldsymbol{A}_2\boldsymbol{x}_2 + \boldsymbol{A}_3\boldsymbol{x}_3 \le \boldsymbol{b} - \boldsymbol{A}_1\boldsymbol{x}_1\\
&\qquad \boldsymbol{u}^T\boldsymbol{A}_3 = \boldsymbol{c}_{33}\\
&\qquad \boldsymbol{u}^T(\boldsymbol{A}_2\boldsymbol{x}_2 + \boldsymbol{A}_3\boldsymbol{x}_3 + \boldsymbol{A}_1\boldsymbol{x}_1^0 - \boldsymbol{b}) = \boldsymbol{0}\\
&\qquad \boldsymbol{x}_2,\ \boldsymbol{x}_3,\ \boldsymbol{u} \ge \boldsymbol{0}
\end{aligned} \tag{2.48}$$

Attaching an appropriately large weight K to the complementary equation and subtract this equation from the second-level objective function, we have:

$$\begin{aligned}
&\max_{x1}\ f_1(\boldsymbol{x}) = \boldsymbol{c}_{11}^T\boldsymbol{x}_1 + \boldsymbol{c}_{12}^T\boldsymbol{x}_2 + \boldsymbol{c}_{13}^T\boldsymbol{x}_3\\
&\text{where } \boldsymbol{x}_2, \boldsymbol{x}_3 \text{ solves:}\\
&\quad \max_{x2,\,x3,\,u}\ f_2(\boldsymbol{x}_1,\ \boldsymbol{x}_2,\ \boldsymbol{x}_3) = \boldsymbol{c}_{22}^T\boldsymbol{x}_1 + \boldsymbol{c}_{22}^T\boldsymbol{x}_2 + \boldsymbol{c}_{23}^T\boldsymbol{x}_3\\
&\quad - K\,\boldsymbol{u}^T(\boldsymbol{A}_2\boldsymbol{x}_2 + \boldsymbol{A}_3\boldsymbol{x}_3 + \boldsymbol{A}_1\boldsymbol{x}_1^0 - \boldsymbol{b})\\
&\quad \text{subject to:}\\
&\quad \boldsymbol{A}_2\boldsymbol{x}_2 + \boldsymbol{A}_3\boldsymbol{x}_3 \le \boldsymbol{b} - \boldsymbol{A}_1\boldsymbol{x}_1\\
&\quad \boldsymbol{u}^T\boldsymbol{A}_3 = \boldsymbol{c}_{33}\\
&\quad \boldsymbol{x}_2,\ \boldsymbol{x}_3,\ \boldsymbol{u} \ge \boldsymbol{0}
\end{aligned} \tag{2.49}$$

Bard proved that a necessary condition that x^* solves the linear TLPP is that there exists a $\boldsymbol{u}^*,\ \boldsymbol{w}^*,$ and $\boldsymbol{v}^* \in R^m$, such that (x^*, u^*, w^*, v^*) is feasible for the problem:

$$
\begin{aligned}
&\max f_1(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) = \boldsymbol{c}_{11}^T \boldsymbol{x}_1 + \boldsymbol{c}_{12}^T \boldsymbol{x}_2 + \boldsymbol{c}_{13}^T \boldsymbol{x}_3 \\
&\text{subject to:} \\
&\quad \boldsymbol{A}_1 \boldsymbol{x}_1 + \boldsymbol{A}_2 \boldsymbol{x}_2 + \boldsymbol{A}_3 \boldsymbol{x}_3 \leq \boldsymbol{b} \\
&\quad \boldsymbol{u}^T \boldsymbol{A}_3 = \boldsymbol{c}_{33} \\
&\quad \boldsymbol{w}^T \boldsymbol{A}_2 = \boldsymbol{c}_{22} \\
&\quad \boldsymbol{w}^T \boldsymbol{A}_3 = \boldsymbol{c}_{23} \\
&\quad K\,\boldsymbol{u} - \boldsymbol{v} + \boldsymbol{w} = \boldsymbol{0} \\
&\quad \boldsymbol{v}^T (\boldsymbol{A}_1 \boldsymbol{x}_1 + \boldsymbol{A}_2 \boldsymbol{x}_2 + \boldsymbol{A}_3 \boldsymbol{x}_3 - \boldsymbol{b}) = \boldsymbol{0} \\
&\quad \boldsymbol{u}^T (\boldsymbol{A}_1 \boldsymbol{x}_1 + \boldsymbol{A}_2 \boldsymbol{x}_2 + \boldsymbol{A}_3 \boldsymbol{x}_3 - \boldsymbol{b}) = \boldsymbol{0} \\
&\quad \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{u}, \boldsymbol{w}, \boldsymbol{v} \geq \boldsymbol{0}
\end{aligned} \tag{2.50}
$$

where $\boldsymbol{u}$, $\boldsymbol{v}$ and $\boldsymbol{w}$ are KKT multipliers associated with the constraint sets, and K is a sufficiently large finite constant.

Based on Eq. (2.50) and an upper bound for the objective function f_1, Bard developed the following simplex-cutting plane algorithm with simplex vertex enumeration:

Step 1. Solve Eq. 3.21 to obtain the trial point $\underline{\underline{\boldsymbol{x}}} = (\underline{\underline{\boldsymbol{x}}}_1, \underline{\underline{\boldsymbol{x}}}_2, \underline{\underline{\boldsymbol{x}}}_3)$, and a tight upper bound on f_1. Go to Step 2.

Step 2. Fix $\boldsymbol{x}_1$ at $\underline{\underline{\boldsymbol{x}}}_1$ and solve the second-level problem only in Eq. 3.7 to obtain $\underline{\boldsymbol{x}}_{[1]}$. If $\underline{\underline{\boldsymbol{x}}} = \underline{\boldsymbol{x}}_{[1]}$ or if $f_1(\underline{\boldsymbol{x}}_{[1]}) = f_1(\underline{\underline{\boldsymbol{x}}})$ stop. Otherwise, let $f_{-1} = f_1(\underline{\boldsymbol{x}}_{[1]})$ as a lower bound and go to Step 3.

Step 3. Search for a vertex adjacent to $\underline{\boldsymbol{x}}_{[k]}$ that both lies in the first-level inducible region and is in the direction of increase of f_1. If none is found, go to Step 4. Otherwise, move to this vertex and repeat the search until a local optimum is reached, name it $\boldsymbol{x}^*$ and update the lower bound by setting $f_{-1} = f_1(\boldsymbol{x}^*)$.

Step 4. Add the cut $f_1 \leq f_{-1} + \varepsilon$ to the feasible set of Eq. 3.21 and solve to obtain $\underline{\underline{\boldsymbol{x}}}$. $\varepsilon > 0$ is a sufficiently small constant so that no unexplored vertices-of the original constraint set are eliminated.

Step 5. Test to see whether $\underline{\underline{\boldsymbol{x}}}$ is in the region by fixing $\boldsymbol{x}_1$ at $\underline{\underline{\boldsymbol{x}}}_1$, and solve the second-level problem-only in Eq. 3.7. If the test is positive, put $\underline{\boldsymbol{x}}_{[k+1]} = \underline{\underline{\boldsymbol{x}}}$ eliminate the cut and go to Step 3. Otherwise, search adjacent vertices in the direction of increasing $\underline{\boldsymbol{x}}_{[k+1]}$ in the first-level inducible region. Repeat search if necessary starting from each of the other multiple solutions obtained at Step 4 until such a point is found; then go to Step 3. If the search fails, stop and $\boldsymbol{x}^*$ is the global optimum.

The following problem solved by Bard (1984b) is used to illustrate the approach.

Example 2.11

$$
\begin{aligned}
&\max_{x1} f_1 = -4x_1 + 2x_2 - 5x_3 \quad \text{(1st level)}\\
&\text{where } x_2, x_3 \text{ solves:}\\
&\quad \max_{x2} f_2 = -x_2 + 4x_3 \quad \text{(2nd level)}\\
&\quad \text{where } x_3 \text{ solves:}\\
&\quad\quad \max_{x3} f_3 = 2x_3 \quad \text{(3rd level)}\\
&\quad\quad \text{subject to:}\\
&\quad\quad\quad 2x_2 - x_3 \geq 2\\
&\quad\quad\quad -3x_1 + x_2 - x_3 \geq -12\\
&\quad\quad\quad -3x_2 - x_3 \geq -24\\
&\quad\quad\quad x_1 \geq 2\\
&\quad\quad\quad -x_3 \geq -6\\
&\quad\quad\quad x_1,\ x_2,\ x_3 \geq 0
\end{aligned}
$$

The corresponding Eq. (2.50) is obtained as:

$$
\begin{aligned}
&\max f_1 = 7x_1 + 3x_2 + 4x_3\\
&\text{subject to:}\\
&\quad 2x_2 - x_3 \geq 2\\
&\quad -3x_1 + x_2 - x_3 \geq -12\\
&\quad -3x_2 - x_3 \geq -24\\
&\quad x_1 \geq 2\\
&\quad -x_3 \geq -6\\
&\quad u_1 + u_2 + u_3 + u_5 = 2\\
&\quad w_1 + w_2 + w_3 + w_5 = 4\\
&\quad 2w_1 + w_2 - 3w_3 = 1\\
&\quad K\,u_1 - v_1 + w_1 = 0\\
&\quad K\,u_2 - v_2 + w_2 = 0\\
&\quad K\,u_3 - v_3 + w_3 = 0\\
&\quad K\,u_4 - v_4 + w_4 = 0\\
&\quad K\,u_5 - v_5 + w_5 = 0\\
&\quad u_1(2x_2 - x_3 - 2) = 0\\
&\quad u_2(-3x_1 + x_2 - x_3 + 12) = 0\\
&\quad u_3(-3x_2 - x_3 + 24) = 0\\
&\quad u_4(x_1 - 2) = 0\\
&\quad u_5(-x_3 + 6) = 0
\end{aligned}
$$

Table 2.5 Decisions and objective functions, Example 2.11 (Bard 1984b)

Vertex	(x_1, x_2, x_3)	F_1	f_2	F_3
Global optimum, A	(14/3, 1, 0)	−50/3	−1	0
Local optimum, B	(2, 4, 6)	−30	20	12
Feasible point, C	(10/3, 4, 6)	−106/3	20	12
High point, D	(2, 8, 0)	8	−8	0
Equation (2.44) solution, E	(2, 1, 0)	−6	−1	0

$$
\begin{aligned}
&v_1(2x_2 - x_3 - 2) = 0\\
&v_2(-3x_1 + x_2 - x_3 + 12) = 0\\
&v_3(-3x_2 - x_3 + 24) = 0\\
&v_4(x_1 - 2) = 0\\
&v_5(-x_3 + 6) = 0\\
&\boldsymbol{u},\ \boldsymbol{v},\ \boldsymbol{w},\ \boldsymbol{x} \geq 0
\end{aligned}
$$

and K is a sufficiently large finite constant.

The following discussions follow the solution carried out by Bard (1984b).

Let $K = 100$ and $\varepsilon = 1$ and apply Step 1 to the above problem, the solution obtained is $\underline{\underline{\boldsymbol{x}}} = (\underline{\underline{x}}_1, \underline{\underline{x}}_2, \underline{\underline{x}}_3) = (2, 1, 0)$, which is Point E in Table 2.5 and this solution is not in level 1 inducible region.

Fixing x_1 at 2 and solve the level 2 problem in Eq. (2.36), we obtain point B or $\underline{\boldsymbol{x}}_{[1]} = (2, 4, 6)$, which belongs to the level 1 inducible region. The search at Step 3 leads to the conclusion that point B is a local solution because there are no adjacent vertices. At Step 4, the cut $f_1 = -4x_1 + 2x_2 - 5x_3 \leq -30 + \varepsilon$ is added to the constraint region of the transformed formulation and obtains a new point $\underline{\underline{\boldsymbol{x}}} = (4.57, 6.43, 4.71)$. This point $\underline{\underline{\boldsymbol{x}}}$ is not in the level 1 inducible region; therefore, at Step 5 a search of adjacent vertices must be conducted to find an element of the level 1 inducible region that intersects the cut, should one exist. The search leads to the $\boldsymbol{x}_{[2]} = (3.65, 3.05, 4.1)$ that lies along the line from point A, which is (14/3, 1, 0) to point C, which is (10/3, 4, 6), on the level 1 inducible region. Returning to Step 3 brings us to point A, which turns out to be the local optimum $\boldsymbol{x}^* = (14/3, 1, 0)$. A second cut must now be added but the algorithm terminates at Step 5 when no new point in the level 1 inducible region can be found that produces a value of f_1 larger than −50/3, the best lower bound.

2.4.2.4 Penalty-Function Approach

Anandalingdam and Apprey (1991) proposed an algorithm that combines *k*th-best algorithm with penalty function to solve the linear three-level problem. First, the following first-level problem is solved:

$$\begin{aligned} &\max f_1(\boldsymbol{x}) = \boldsymbol{c}_{11}^T \boldsymbol{x}_1 + \boldsymbol{c}_{12}^T \boldsymbol{x}_2 + \boldsymbol{c}_{13}^T \boldsymbol{x}_3 \\ &\text{subject to:} \\ &\quad \boldsymbol{A}_1 \boldsymbol{x}_1 + \boldsymbol{A}_2 \boldsymbol{x}_2 + \boldsymbol{A}_3 \boldsymbol{x}_3 \leq \boldsymbol{b} \\ &\quad \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3 \geq \boldsymbol{0} \end{aligned} \tag{2.51}$$

Let $(\boldsymbol{x}_1^{1*}, \boldsymbol{x}_2^{1*}, \boldsymbol{x}_3^{1*})$ represent this solution. For a given $\boldsymbol{x}_1^{1*}$, the two-lower-level problem becomes:

$$\begin{aligned} &\max_{x2} f_2(\boldsymbol{x}) = \boldsymbol{c}_{21}^T \boldsymbol{x}_1^{1*} + \boldsymbol{c}_{22}^T \boldsymbol{x}_2 + \boldsymbol{c}_{23}^T \boldsymbol{x}_3 \\ &\text{where } \boldsymbol{x}_3 \text{ solves:} \\ &\quad \max_{x3} f_3(\boldsymbol{x}) = \boldsymbol{c}_{31}^T \boldsymbol{x}_1^{1*} + \boldsymbol{c}_{32}^T \boldsymbol{x}_2 + \boldsymbol{c}_{33}^T \boldsymbol{x}_3 \\ &\quad \text{subject to:} \\ &\quad\quad \boldsymbol{A}_2 \boldsymbol{x}_2 + \boldsymbol{A}_3 \boldsymbol{x}_3 \leq \boldsymbol{b} - \boldsymbol{A}_1 \boldsymbol{x}_1^{1*} \\ &\quad\quad \boldsymbol{x}_2, \boldsymbol{x}_3 \geq \boldsymbol{0} \end{aligned} \tag{2.52}$$

The global optimal solution for Eq. 3.23 is obtained through the algorithm listed in Sect. 3.1.2. Suppose the solution is $(\boldsymbol{x}_2^1, \boldsymbol{x}_3^1)$, and if $\boldsymbol{x}_2^1 = \boldsymbol{x}_2^{1*}$, and $\boldsymbol{x}_3^1 = \boldsymbol{x}_3^{1*}$, then $(\boldsymbol{x}_1^{1*}, \boldsymbol{x}_2^{1*}, \boldsymbol{x}_3^{1*})$ is the optimal solution for the overall problem. Otherwise, the second best solution $(\boldsymbol{x}_1^{2*}, \boldsymbol{x}_2^{2*}, \boldsymbol{x}_3^{2*})$ is obtained from Eq. 3.22, the process is repeated. The suggested algorithm can be summarized as follows:

Step 1. Solve Eq. 3.22 and let the solution be represented by $\boldsymbol{x}^{1*} = (\boldsymbol{x}_1^{1*}, \boldsymbol{x}_2^{1*}, \boldsymbol{x}_3^{1*})$. Go to Step 2.
Step 2. Solve Eq. 3.23 by the algorithm in Sect. 3.1.2 with $\boldsymbol{x}_1$ given as $\boldsymbol{x}_1^{1*}$ and let this solution be represented by $\boldsymbol{x}^{1*} = (\boldsymbol{x}_1^{1*}, \boldsymbol{x}_2^{1}, \boldsymbol{x}_3^{1})$. Go to Step 3.
Step 3. If $\boldsymbol{x}^1 = \boldsymbol{x}^{1*}$, the optimality is reached and stop. Otherwise, go to Step 4.
Step 4. Let $\boldsymbol{x}^{2*} \in \boldsymbol{X}_v$ be the adjacent point to $\boldsymbol{x}^{1*}$. Set $\boldsymbol{x}^{1*} = \boldsymbol{x}^{2*}$, go to Step 2.

The proposed procedure essentially follows the *k*th-best algorithm to search for *k*th extreme point, and the optimal solution is reached if the adjustments between the first-level (leader) and the two lower levels (followers) are completed.

2.5 Discussions

Because of the practical importance of the duo- or multi-ploy type decision-making process, many algorithms have been developed for solving this type of problems by using a variety of different approaches. In this chapter, several representative approaches are introduced. Nevertheless, there are still many other algorithms are not included in this chapter. Among the traditional optimization approaches, the several algorithms based on branch-and-bound appear to be most efficient. Most of these algorithms also borrow the idea of mixed integer to handle the complementary slackness term due to Fortuny-Amat and McCarl (1981). The kth-best algorithm, although not computationally efficient, does give a reasonably clear picture of the problem involved. The main complication in solving the multi-ploy type problem is due to the fact that the rational reactions of the lower levels must be singletons in order to guarantee true optimum of the upper levels. It appears that the most promising future development so that more complicated practical problem can be solved is in the use of metaheuristics, which, although cannot guarantee true optimum, does avoid the problem of exponential explosion.

The algorithms introduced in this chapter are not suited for solving interactive decision problems, where the mutual interactions between the different levels are more flexible and where the decision process is more in line with that of the hierarchy organizations such as the manufacturing or service industries. Furthermore, these algorithms are not very efficient even for the duo- or multi-ploy type systems. The approach by the use of fuzzy interactive decision making process, which is much more flexible to allow interactions between the different levels, will be discussed in later chapters. Following the same approach, various examples will be given to illustrate the solution procedures.

Chapter 3
Possibility Theory and Fuzzy Optimization

Fuzzy set theory is a powerful tool for representing vague phenomena and for linguistic representation in modern computers. The basic concept has been applied in many different areas. For the convenience of later discussions, some of the fuzzy concepts are summarized in this chapter. However, we shall not discuss the most basic aspects and assume the reader has some familiarity with fuzzy set theory. For more details, the reader can consult the many excellent books on this subject. For example, the book by Klir and Folger (1988) and especially the papers by Zadeh, which has been collected in a single book (1987), can be consulted. The book by Lee and Zhu (1995) and the papers by Zadeh are more directly connected with this chapter. In fact, we follow these works closely during most of the discussions.

3.1 Possibility Theory

The concept of possibility proposed by Zadeh (1978, 1987) forms one of the most useful foundations in fuzzy set theory. By the use of this concept, concrete meanings can be established and used for the formation of fuzzy membership functions. Furthermore, because fuzzy set is ideally suited for the representation of linguistic expressions, the frequently used linguistic expressions by management can be expressed in terms of membership functions. In order to use this concept in hierarchical decision-making, the meaning of possibility or possibility measure is first summarized in this section.

Possibility measure can be defined either based on confidence measure or based on fuzzy set (Lee and Zhu 1995). The former gives the connection between possibility measure and evidence theory and the latter gives the connection between possibility and fuzzy membership functions. Confidence measure is also known as fuzzy measure or Sugeno measure. Sugeno first proposed this unifying measure (Takagi and Sugeno 1983). Evidence theory was first proposed by Dempster and Shafer (Shafer 1976) and is also known as upper and lower probability. The advantage based on

C.-B. Cheng et al., *Fuzzy and Multi-Level Decision Making: Soft Computing Approaches*, Studies in Fuzziness and Soft Computing 368,
https://doi.org/10.1007/978-3-319-92525-7_3

confidence measure is that the concept of confidence measure can unify the various measures such as the belief measure, plausibility measure, and probability measure and thus gives a theoretical basis of possibility measure. Another reason for the better theoretical basis is because the confidence approach is based on a set of basic axioms. The advantage based on fuzzy set is that the approach offers easy means to obtain numerical values of the representation. In order to save space and also for later applications, only the approach via fuzzy set will be used to define the possibility measure.

The theory of possibility, as the name implies, deals with the possible rather than the probable values of a variable (Zadeh 1987; Lee and Zhu 1995). The term variable used here is more in a linguistic sense than a numerical one. Let us consider the statement or proposition, "X is F", where F is a fuzzy set characterized by its membership function μ_F. The variable X could be the name of an object, some attribute of an object, a variable, a proposition, or any other similar representations. For example, in "X is a small number", X is the name of a variable. In "John is tall", John is the name of an object. In "Xelo company's product has very good quality", the word "product" is the name of an attribute of the object "Xelo company". In "The company is very young is quite true", where "The company is very young" is a proposition. The expressions "small number", "tall", "very good quality", and "quite true" are fuzzy sets. Notice that all these expressions are fuzzy or vague and cannot be used to form a crisp set.

3.1.1 Possibility Distribution

One of the central concepts in possibility theory is possibility distribution, which serves the same purpose in possibility theory as probability distribution in probability theory.

To define possibility distribution, Zadeh introduced the concept of a fuzzy restriction. We first define a fuzzy concept F in the non-fuzzy universe U, where the fuzzy concept F is characterized by the membership function, $\mu_F(u)$.. Then, we consider a given variable X and estimate to what degree each of the members of the variable X can be assigned to the fuzzy set F. Thus, F is a fuzzy restriction on the variable X if F acts as an elastic constraint on the values that may be assigned to X. In other words, $\mu_F(u)$ is the degree to which the constraint represented by the fuzzy concept F is satisfied when u is assigned to X. Equivalently, $1 - \mu_F(u)$ is the degree to which the constraint must be stretched in order to allow the assignment of u to X, where u represents the generic elements of X.

Let $R(X)$ denote a fuzzy restriction associated with X. Then $R(X) = F$ is called a relational assignment equation by Zadeh, because it assigns the fuzzy set F to the fuzzy restriction $R(X)$.

Let $A(X)$ be an attribute of the variable X. For example, if $A(X)$ is the product of Xelo company and F is the fuzzy set "very good quality", the proposition "Xelo company's product has very good quality" can be expressed as:

$$R(A(X)) = F \tag{3.1}$$

In other words, the product of Xelo company is restricted by the fuzzy set "very good quality".

As another example, consider "John is tall". This proposition implied the attribute height of John. Thus, we have:

$$\text{John is tall} \rightarrow R(\text{height(John)}) = \text{tall} \tag{3.2}$$

Thus, let F be a fuzzy subset of a universe of discourse U, which is characterized by its membership function μ_F, with the grade of membership $\mu_F(u)$, interpreted as the compatibility of u with the concept label F. Let X be a variable taking on values in U and let F act as a fuzzy restriction, $R(X)$, associated with X. Then, the proposition "X is F", which translates into $R(X) = F$, associates a possibility distribution Π_x with X, which is postulated to be equal to $R(X)$. Thus, the possibility distribution function associated with X, Π_x, can be defined to be numerically equal to the membership function of *F,* that is,

$$\pi_x \equiv \mu_F \tag{3.3}$$

where "$\equiv$"stands for "is defined to be" and $\pi_x$represents the possibility distribution function of Π_x.

Therefore, $\mu_F(u)$ not only represents the grade of membership of u in U, but also represents the possibility that X is u given that X is F. Mathematically,

$$\text{Poss}(X \text{ is } u | X \text{ is } F) = \mu_F(u), \ u \in U, \tag{3.4}$$

Which is a conditional possibility expression parallel to the conditional probability expression.

Example 3.1 Let U be the universe of positive integers and F the fuzzy subset of small positive integers defined by,

$$\begin{aligned} F &= \text{small positive integers} \\ &= 1.0/1 + 1.0/2 + 0.9/3 + 0.7/4 + 0.5/5 + 0.3/6 + 0.1/7 \end{aligned} \tag{3.5}$$

where we have used the conventional method to express the membership function. The plus signs "+" should be interpreted as unions, not as additions and the "/" should not be interpreted as division. The above equation can be interpreted as:

> The membership function for the integer "1" is 1.0 in the fuzzy set F, the membership function for the integer 3 is 0.9 in the fuzzy set F, the membership function for the integer 5 is 0.5 in the fuzzy set F, and etc.

For example, we can form the following conditional possibility expressions:

$$\text{Poss}(X \text{ is a small integer} | X \text{ is } 4) = 0.7 \tag{3.6}$$

$$\text{Poss}(X \text{ is a small integer} | X \text{ is } 6) = 0.3 \tag{3.7}$$

3.1.2 Possibility Measure

Based on the above discussion and the equivalence between the membership function of a fuzzy subset and its possibility, the possibility measure can be defined in the following manner. Let A be a non-fuzzy subset of the universe of discourse U and let π_x be the possibility distribution function of Π_x; then, the possibility measure, $\Pi(A)$, of A is defined as a number in the interval (0, 1) and is given by,

$$\Pi(A) = \sup_{u \in A} \pi_x(u) \tag{3.8}$$

where Π_x is the possibility distribution associated with the variable X which takes the value in U. The possibility measure can also be interpreted as the possibility that the value of X belongs to A, or,

$$\text{poss}(x \in A) = \Pi(A) = \sup_{u \in A} \pi_x(u) = \sup_{u \in A} \mu_F(u) \tag{3.9}$$

where the last equality is obtained by using the relationship between the possibility distribution function the membership function of the fuzzy set F.

Example 3.2 Xelo manufacturing company has three divisions. Each division makes a different product for sale. The management estimates that the possibilities of increase 10% in sales for the next fiscal year are 0.7, 0.5, and 0.9 for Divisions one, two, and three, respectively.

Notice that in this example, the universe of discourse is all the divisions which increase sales 10%, and the subset A is either the three given divisions or any combinations of the divisions as long as the individual divisions' increase in sales is 10%. The divisions, which increase sales 10%, are crisp and the subset A is certainly not fuzzy. The fuzzy concept F is the possibility of increasing sales 10%. If we define the crisp subset A as {division 1 increase 10%, division 2 increase 10%, division 3 increase 10%}, then the possibility distribution is,

$$\begin{aligned} \Pi_x = {} & 0.7/(\text{division 1 increase 10\% }) + 0.5/(\text{division 2 increase 10\% }) \\ & + 0.9/(\text{division 3 increase 10\%}) \end{aligned} \tag{3.10}$$

According to Eq. (3.8) or Eq. (3.9), the possibility of increasing 10% for divisions 1, 2, and 3 as one entity is,

$$\begin{aligned} \Pi(A) &= \Pi(\{\text{division 1 increase 10\%, division 2 increase 10\%, division 3 increase 10\%}\}) \\ &= \sup\{0.7, 0.5, 0.9\} = 0.9 \end{aligned} \tag{3.11}$$

If we define subset A as composed of only two divisions, then the possibility of increasing 10% is:

$$\begin{aligned}\Pi(A) &= \Pi(\{\text{division 1 increase 10\%, division 2 increase 10\%}\}) \\ &= \sup\{0.7, 0.5\} = 0.7\end{aligned} \tag{3.12}$$

Notice that the above definition of possibility is consistent with the meaning of the word "possibility" in our usual daily usage.

Example 3.3 Suppose Xelo Company is starting to produce a new product. From the various economical data about the product, the company concluded that if the total sale during the first year is about 35 million pounds, then this product would produce a profit. But, many economic and other factors can influence this estimate. Based on past experience, plus or minus 5 million pounds should be allowed for this rough estimation. Thus, the membership function for the profit potential or making a profit can be represented as:

$$\mu_{\text{profit potential}}(u) = \begin{cases} 1, 40 \leq u \\ \frac{u-30}{10}, 30 \leq u \leq 40 \\ 0, u \leq 30 \end{cases} \tag{3.13}$$

Suppose the total sales of this new product during the first year is 35 million pounds, then the profit potential or the possibility of making a profit during the first year is:

$$\begin{aligned}\text{Poss(profit|amount produced} = 35 \text{ million pounds)} &= \text{poss(making a profit)} \\ &= \mu_{\text{profit potential}}(35) = 0.5\end{aligned} \tag{3.14}$$

3.1.3 Possibility Measure Based on Fuzzy Set with Fuzzy Subset

In the above discussions, the possibility measure was defined on a crisp or non-fuzzy subset A. When A is a fuzzy subset, the belonging of a value of X to A has no meaning. A more general definition for possibility measure with fuzzy subset A was proposed by Zadeh and can be introduced as follows.

Let A be a fuzzy subset of the universe U and let Π_x be a possibility distribution associated with a variable X which takes value in U. The possibility measure, $\Pi(A)$, of A is then defined by,

$$\text{Poss}\{X \text{ is A}\} = \Pi(\text{A}) = \sup_{u \in U} \min(\mu_A(u), \pi_x(u)) \tag{3.15}$$

where $\mu_A(u)$ is the membership of u in the fuzzy subset A. The above equation can be interpreted as the possibility that X takes fuzzy set A as its value.

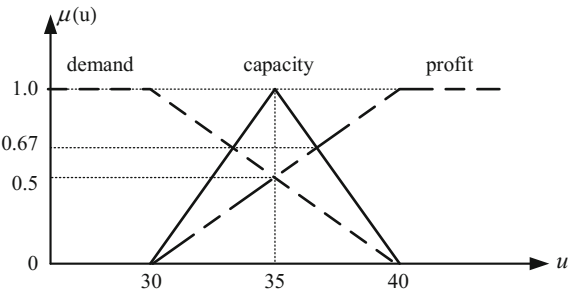

Fig. 3.1 Membership functions of Xelo Company

Example 3.4 Consider the problem in Example 3.3. In addition to the economic considerations of the new product, the sales or demand potential of the product must also be considered. From marketing survey, it was found out that the demand for this product during the first year is about 35 million pounds. Because various factors can influence this survey, plus or minus 5 million pounds should be allowed to this figure. Thus, the membership function for demand potential can be expressed as (see Fig. 3.1):

$$\mu_{\text{demand potential}}(u) = \begin{cases} 1, u \leq 30 \\ \frac{40-u}{10}, 30 \leq u \leq 40 \\ 0, u \leq 40 \end{cases} \tag{3.16}$$

Using the membership functions of profit potential, Eq. (3.8), and demand potential, Eq. (3.16), the possibility of making a profit based on Eq. (3.15) is:

$$\begin{aligned} \text{Poss(making a profit)} &= \sup \ \min(\mu_A(u), \mu_x(u)) \\ &= \sup \ \min(\mu_{\text{sales potential}}(u), \mu_{\text{profit potential}}(u)) \\ &= 0.5 \end{aligned} \tag{3.17}$$

Example 3.5 Consider the problem of Xelo Company again. Although the new plant for the new product was built for a capacity of 35 million pounds, the actual on-line capacity may be different due to many factors, which cannot be controlled. From past experience, the actual capacity may be anywhere within the interval of (30, 40 million pounds). It should be noted that because of the particular process, it is very uneconomical, in fact, may be dangerous to produce too much below capacity. Thus, the membership function for capacity should reach a value of one at 35 million pounds, or this is the most possible value. Assuming the possibility decreases linearly when the capacity moves away from 35 from either side and reaches a value of zero at either 30 or 40 million pounds. Thus, the membership function for the approximate capacity of production is (see Fig. 3.1):

$$\mu_{\text{capacity}}(u) = \begin{cases} 0, u \leq 30 \text{ or } u \geq 40 \\ \frac{u-30}{5}, 30 \leq u \leq 35 \\ \frac{40-u}{5}, 35 \leq u \leq 40 \end{cases} \tag{3.18}$$

Assume that the company has the least confidence about the profit potential and the estimated approximate capacity, then an estimate on possibility of making a profit based on these two fuzzy numbers can be obtained by using Eq. (3.15). The result is:

$$\begin{aligned} \text{Poss(making a profit)} &= \sup \min(\mu_A(u), (u)) \\ &= \sup \min(\mu_{\text{profit potential}}(u), \mu_{\text{approximate capacity}}(u)) \\ &= 0.67 \end{aligned} \tag{3.19}$$

If, on the other hand, the company is worried more about capacity and demand potential, the possibility of making a profit is:

$$\begin{aligned} \text{Poss(making a profit)} &= \sup \min(\mu_A(u), \mu_x(u)) \\ &= \text{supmin}(\mu_{\text{approximate capacity}}(u), \mu_{\text{demand potential}}(u)) \\ &= 0.67 \end{aligned} \tag{3.20}$$

Finally, if we take all three factors into consideration, that is profit potential, market demand and the approximate capacity, then the possibility of making a profit is:

$$\begin{aligned} \text{Poss(making a profit)} &= \sup \min(\mu_{\text{profit potential}}(u), \mu_{\text{demand potential}}(u), \mu_{\text{approximate capacity}}(u) \\ &= 0.5 \end{aligned} \tag{3.21}$$

3.1.4 Possibility Versus Probability

Possibility deals with the possible while probability deals with the probable. Possibility is associated with the degree of feasibility or the sense of attainment, whereas, probability relates to the degree of likelihood, frequency, or proportion. This implies that what is possible may not be probable, and what is improbable may not be impossible.

There are many types of uncertainties. Probability models randomness while possibility models fuzziness where the events have no clear boundary. The concept of possibility, unlike probability, does not involve the notion of repeated experimentation. Thus, possibility can be used to model the imprecision of uncertainties, which are not susceptible to probability analysis or characterization (Lee and Zhu 1995).

Another way to compare these two different approaches of modeling uncertainties is to consider the rules, which govern their aggregation. Possibility theory based on the max-min aggregation while probability theory requires that the summation of the

probabilities of all the possible outcomes must be equal to one. In possibility, intersection is modeled by the min-operator and union is modeled by the max-operator, while in probability, intersection of two independent events corresponds to the product operator and the union of two independent, mutually exclusive events corresponds to the sum operator. Thus, in a sense, the requirements of possibility are not as strict as that of probability. In fact, it has been proved that possibility measure reduces to probability measure if only singletons are used in assigning the amount of believe.

Many of the particular characteristics of possibility theory are intimately connected with linguistics. For example, human language is frequently vague and approximate, which is exactly the same for fuzzy or possibility. Language is intimately connected with human activities and thus it is subjective and cannot be objective, fuzzy set is also subjective, not like the objective requirements when probability is used. One example would clear the reason why language or fuzzy set must be subjective. Consider the word "young" in the phrase "a young man". Since it is intimately connected with our concept, which represents "young", it would be impossible to define the word "young" objectively.

Perhaps the most important difference is that fuzzy set is best for describing humanistic aspects and probability is more for mechanistic or scientific applications.

3.2 Knowledge Representation

Possibility theory based on fuzzy membership function or fuzzy logic forms a useful approach for knowledge representation and subsequent manipulation or aggregation on computer. It is much more flexible and capable of linguistic representation than traditional logical systems. In other words, the traditional proposition logic is too restrictive for linguistic representation and aggregation. Some of the problems with the traditional two-value logic are:

1. in the traditional logic, a proposition is either true or false. But, human language is generally vague or fuzzy and it seldom only has the two extremes;
2. the predicates in two-value logic are constrained to be crisp. But, human language frequently contains fuzzy or vague predicates;
3. traditional logic only allows two quantifiers: *all* and *some*. By contrast, human language uses many quantifiers, both fuzzy and crisp, such as *many*, *much*, *most*, *Jew*, *frequently*, etc.

Since linguistics represents human intelligence and human knowledge, knowledge or linguistic representation in computer constitutes the essential ingredient in the use of computer for solving and manipulating soft science such as management or decision making, which is frequently expressed in linguistic terms.

One of the most difficult problems encountered is the representation of common sense knowledge, some typical examples of which are:

birds fly;
snow is white;
this manufacturing process is reliable.

The first two propositions were well known and were first encountered in the investigation of expert systems. One characteristic of these propositions is that it is preponderantly, but not always, true. This type of daily encountered common sense knowledge is the most difficult problem to handle. However, they can be handled elegantly by the use of fuzzy concepts. Notice that these propositions have implicit or implied fuzzy quantifiers. For example, *birds fly* should be *most birds fly* and *snow is white* should be *snow is usually white*.

In this section, knowledge representation will be treated in terms of linguistic representation. Linguistic variable and linguistic terms will be first defined. The syntactic and semantic rules of the linguistic variable will then be discussed. And, finally, linguistic algorithm for the manipulation of the representations will be presented. Because of space limitations, only a summary can be presented. The interested reader can consult the literature (Lee and Zhu 1995; Zadeh 1987).

3.2.1 Linguistic Variable

Linguistic variable forms the first step or basis in the manipulation or aggregation of human language on computer, or, computing with words, as Zadeh proposed. Since human intelligence is embodied in the human language, in order to manipulate or aggregate human intelligence, it is essential to be able to aggregate the human language on modem computer.

Like the traditional or numerical variable, linguistic variable also has values. For example, consider the linguistic variable, *price to earnings ratio* (*PE ratio*), which is usually used to describe the expensiveness of the stock of a company. The values of this linguistic variable may be "the stock of the company is *very expensive, fairly expensive, expensive, average, cheap, fairly cheap, very cheap,* etc". It should be noted that the expensiveness of a company's stock is, in general, a very vague expression and cannot depend on its numerical value. A very high PE ratio for a very fast growing company is not necessarily expensive, and, on the other hand, a very low PE ratio for a company, who is continuously loosing its market share, may be still too expensive. Thus, instead of using numerical values to describe the expensiveness of a company's stock, it is much more reasonable to use the above mentioned linguistic terms.

Zadeh (1987) characterized a linguistic variable by the quintuple (L, $T(L)$, U, S, M) in which L is the name of the variable such as the linguistic variable *PE ratio*; $T(L)$ is the term-set of L or the collection of the linguistic values such as very expensive, fairly expensive, average, etc., in the above example; U is the universe of discourse; S is the syntactic rule, by the use of which the term-set X for L can be generated; and M is the semantic rule which associates with each linguistic value X its meaning

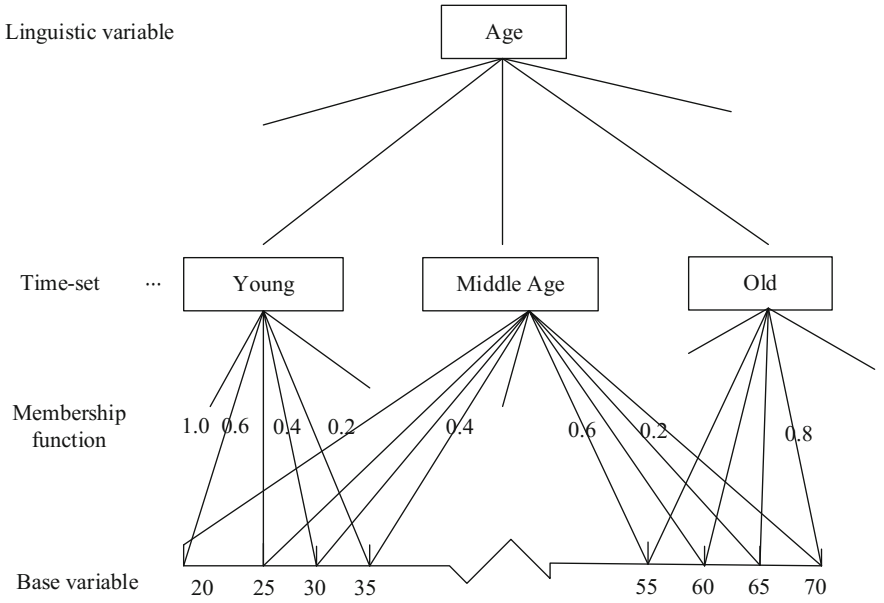

Fig. 3.2 The linguistic variable *Age* (Zadeh 1978)

$M(X)$ by the use of the possibility concept and fuzzy set. Notice that $M(X)$ is a fuzzy subset of U and the concept of linguistic variable is defined based on possibility and the fuzzy concept.

A linguistic variable can generally be represented by a three- or two-level hierarchy system. This is shown in Fig. 3.2, where the linguistic variable *Age*, as used by Zadeh, is reproduced (1978, 1987). The lowest hierarchy is the base variable, which usually is a subset of the universe of discourse. In the present case, the base variable is the actual age. The middle hierarchy is the term-set, or the linguistic value of the variable. The terms of the term-set for *Age* are *very young*, *fairly young, young, middle age, old, fairly old, very old*, etc. The membership functions for the term-set are also shown in the figure.

For Some linguistic variables, there exist no base variables and thus a two-level hierarchy must be used. One example is the linguistic variable describing the ability of the manager of a company, *Manager*, whose term-set may be *very good manager, good manager, average manager, bad manager, very bad manager,* etc. Another example is the linguistic variable appearance, whose term-set may be *very neat, fairly neat, neat, average, sloppy, fairly sloppy, very sloppy,* etc. There exist no base variables for these linguistic variables. In general, more precise and meaningful representation, in terms of membership function, can be obtained if the base variable for this linguistic variable can be established.

3.3 Fuzzy Decision Making

After formulating the fuzzy representations of the various linguistic terms, the next step is the decision-making based on these formulations and by using fuzzy optimization or decision making techniques. Fuzzy optimization approaches have been developed for most of the existing crisp optimization and decision making techniques. Fuzzy linear programming is a well-developed field and has been extended to fuzzy multiple-objective programming and fuzzy goal programming. Fuzzy multiple attribute decision making is an ideal approach to overcome the representation of the frequently encountered linguistic expressions such as *very good, good, fairly good, average, fairly poor, poor, and very poor*. For dynamic and non-linear systems, there are fuzzy dynamic programming and various fuzzy non-linear programming techniques. Many of these techniques can generally be divided into two types. One is based on the original crisp approach with the addition of fuzzy aspects and the other handles the fuzzy problem directly, which frequently forms completely new approaches. One good example is the newly formed possibility programming.

Because of the vast development in this area, it is clear impossible, nor desirable, to have even a simple comprehensive review. Only some basic concepts and the ones, which will be used in latter chapters, will be summarized. The interested readers can consult the references listed at the end of the book (Zimmermann 1996; Chang and Lee 1993).

3.3.1 Fuzzy Linear Programming

The very first contribution in fuzzy decision making is due to Bellman and Zadeh (1970), who proposed the concept of symmetry between the objective function or goal and the constraint. Many later developments are based on this concept. These investigators argued that for complicated problems in actual situations, it is more practical to consider the objective is to achieve a desired fuzzy goal, which can be treated symmetrically with the constraint. In order to simultaneously satisfy all the constraint and the goal, the minimum or the conjunction and should be used. Thus, we can formulate the following:

$$\mu_D = \mu_G \wedge \mu_C = \min\{\mu_G, \mu_C\} \tag{3.22}$$

where μ_D, μ_G, and μ_C represent respectively the membership functions of decision, goal and constraint. In order to obtain the maximum value of the membership function of the final decision, the above expression is maximized:

$$(\mu_D)\max = \max\{\min(\mu_G, \mu_C)\} \tag{3.23}$$

Based on this symmetric concept, Zimmermann (1978, 1996) formulated fuzzy linear programming in the following manner. Consider the typical linear programming problem:

$$\begin{aligned} &\max z = c^T x \\ &\text{subject to:} \\ &\quad Ax \leq b \\ &\quad x \geq 0 \end{aligned} \tag{3.24}$$

where x and c are n-dimensional vectors, b is m-dimensional and A is an $m \times n$ matrix. In many practical applications and especially for large practical systems, we frequently prefer to achieve an established goal instead of maximizing the objective function. Assuming the established aspiration level or goal as z, Eq. (3.24) reduces to:

$$\begin{aligned} &c^T x \leq z \\ &Ax \leq b \\ &x \geq 0 \end{aligned} \tag{3.25}$$

which can be stated as finding the values of x such that:

$$\begin{aligned} &Bx \leq d \\ &x \geq 0 \end{aligned} \tag{3.26}$$

where the first two parts of Eq. (3.25) have combined to form the first part of Eq. (3.26). The first vector equation in Eq. (3.26) represents $(m + 1)$ equations. In order to satisfy all the inequality constraints in the first part of Eq. (3.26), the membership function of the decision must obey the following equation:

$$\mu_D(x) = \min_i \{\mu_i(x)\} \tag{3.27}$$

where the minimization is with respect to all the equations in the first vector equation in Eq. (3.26). The results of Eq. (3.27) are represented by a fuzzy set. If the decision-maker is interested in a crisp optimum, we should take the maximum possible value of the membership function. Thus, we have:

$$\mu(x) = \max_{x \geq 0} \{\min_i (\mu_i(x)\} \tag{3.28}$$

which can also be written as:

$$\mu(x) = \max_{x \geq 0} \lambda \tag{3.29}$$

with

$$\lambda = \min_i\{\mu_i(x)\} \tag{3.30}$$

In practice, frequently, the inequality constraints in Eq. (3.24) are not strictly obeyed. Thus, we can also assume the inequalities are fuzzy and the fuzzy membership functions are:

$$\mu_i(x) = \begin{cases} 1, \text{if}(Bx)_i \le d_i \\ 1 - \frac{(Bx)_i - d_i}{p_i}, \text{if}\, d_i \le (Bx)_i \le d_i + p_i \\ 1, \text{if}(Bx)_i \ge d_i + p_i \end{cases} \tag{3.31}$$

where $i = 1, 2,\ldots, m + 1$ and p_i is the allowed violations of the objective and constraints or the allowed violations of constraint i in the first part of Eq. (3.26). This allowed violation must be determined by the decision-maker. Substituting Eq. (3.31) into Eq. (3.30), we have:

$$\lambda = \min_i\left(1 - \frac{(Bx)_i - d_i}{p_i}\right) \tag{3.32}$$

The minimization sign can be replaced by inequality, we have:

$$\lambda \le \left(1 - \frac{(Bx)_i - d_i}{p_i}\right) \tag{3.33}$$

after simplification of which, we have:

$$\lambda p_i + (Bx)_i \le d_i + p_i, \quad i = 1, 2, \ldots, m + 1 \tag{3.34}$$

Combine with Eq. (3.29), we obtain the new linear programming problem:

$$\begin{aligned} &\max \lambda \\ &\text{subject to:} \\ &\quad \lambda p_i + (Bx)_i \le d_i + p_i, \quad i = 1, 2, \ldots, m + 1 \\ &\quad x_j \ge 0, \quad j = 1, 2, \ldots, n \end{aligned} \tag{3.35}$$

In general, the solution of Eq. (3.35) gives the solution of the original fuzzified linear programming problem. By fuzzified, we mean that we allowed some tolerance to the original strict inequalities and used a predefined goal to replace the maximization operation. Computationally, Eq. (3.35) only increased one constraint as compared to the original linear problem, Eq. (3.24). Thus, this approach is used frequently due to its ease in obtaining numerical solution. Many extensions in fuzzy decision-making such as fuzzy multi-objective programming are also frequently based on this approach.

3.3.2 Fuzzy Approach to Multiple-Objective Programming

One of the difficulties in multi-objective decision making is the aggregation or compromise between the various objectives. Various approaches to carry out this aggregation have been proposed and they can be classified approximately into compensatory and non-compensatory aggregation operators. Zimmermann and coworkers (1978, 1996) have proposed a compensatory and γ operator, which was a combination of the algebraic product and the algebraic sum. However, due to the ease of computation, the most frequently used aggregate operator is the *min* operator, which is completely non-compensatory. To overcome this non-compensatory problem, Lee and Li (1993) and Li (1990) proposed a two phase approach, where the non-compensatory operator *min* was used in the first phase to obtain the optimal degree of the overall satisfaction. Then a fully compensatory operator *averaging* was introduced in the second phase to find a non-dominated solution. It was shown that, in this way, a non-dominated and balanced solution could be obtained in phase two, regardless of the uniqueness of the solution.

The general multiple-objective programming problem can be represented as:compensatory operator may be used

$$\begin{aligned} &\max\ Z = (Z_1, Z_2, \ldots, Z_l)^T = (c_1x, c_2x, \ldots, c_kx, \ldots, c_lx)^T \\ &\min\ W = (W_1.W_2, \ldots, W_r)^T = (c_1x, c_2x, \ldots, c_sx, \ldots, c_rx)^T \end{aligned} \tag{3.36}$$

subject to:

$$\begin{aligned} &Ax \le b \\ &x \ge 0 \end{aligned}$$

where c_k, c_s, and x are n-dimensional vectors with $k = 1, 2,\ldots, l$ and $s = 1,2,\ldots, r$, b is an m-dimensional vector, and A is an $m \times n$ matrix. Many different approaches have been proposed to solve this problem. Only two of the more frequently used approaches, namely, compromise programming and goal programming, will be summarized below.

Fuzzy concepts can also be used to obtain the compromise of the various objective functions of the above crisp programming problem. In fact, it has been shown that the crisp compromise programming approach with $p \Rightarrow \infty$ is equivalent to the following fuzzy approach (Lee and Li 1993). The first step is to formulate the membership functions for the objectives in the following manner:

$$\mu_k = \frac{Z_k^* - Z_k(x)}{Z_k^* - Z_k^-}, \quad k = 1, 2, \ldots, l \tag{3.37}$$

$$\mu_s = \frac{W_s(x) - W_s^*}{W_s^- - W_s^*}, \quad s = 1, 2, \ldots, r \tag{3.38}$$

where Z_k^*, W_s^* and Z_k^-, W_s^- are the ideal and anti-ideal solutions, respectively, of Eq. (3.36).

The problem now is how to aggregate the objective functions in the fuzzy sense. Extend the concept of Bellman and Zadeh (1970), Zimmermann (1996) proposed two aggregate operators, the *min* operator, a non-compensatory operator, and the *product* operator, which is compensatory. However, the resulting problem by using the *product* operator is non-linear, which is generally difficult to solve. Thus, the *product* operator is seldom used. Using the *min* operator and follows similar argument as that used in obtaining Eq. (3.35), the desired programming problem can be obtained. First notice that the final solution must satisfies all the membership functions of μ_k and μ_s in Eqs. (3.37) and (3.38). Thus, the conjunction and or the *min* operator must be used, we have:

$$\lambda = \min_i \mu_i(x) = \min_{k,s}(\mu_k, \mu_s) \tag{3.39}$$

where λ is the final membership function. The min operator can be replaced by inequality and, in addition, we would like to obtain the maximum value of the final membership function A. Thus, the final programming problem is obtained from the above equations as:

$$\begin{aligned} &\max \lambda \\ &\text{subject to:} \\ &\lambda \le \frac{Z_k(x) - Z_k^-}{Z_k^* - Z_k^-}, \quad k = 1, 2, \ldots, l \\ &\lambda \le \frac{W_s^- - W_s(x)}{W_s^- - W_s^*}, \quad s = 1, 2, \ldots, r \\ &Ax \le b \\ &x \ge 0 \end{aligned} \tag{3.40}$$

The biggest disadvantage of Eqs. (3.39) or (3.40), which is obtained by using the min operator, is due to the non-compensatory nature of the *min* operator. In other words, the results obtained by the *min* operator represent the worst situation and cannot be compensated by other members, which may be very good. To overcome this difficulty, a compensatory operator may be used. Lee and Li (1993) proposed the use of the *arithmetical average* operator and the programming problem becomes:

$$\max \xi = \frac{1}{l+r}\left(\sum_{k=1}^{l} \lambda_{\mathrm{k}} + \sum_{s=1}^{r} \lambda_{\mathrm{s}}\right) \tag{3.41}$$

subject to:

$$\begin{aligned}
&\lambda_k \leq \tfrac{Z_k(x)-Z_k^-}{Z_k^*-Z_k^-}, \quad k = 1, 2, \ldots, l \\
&\lambda_s \leq \tfrac{W_s^- - W_s(x)}{W_s^- - W_s^*}, \quad s = 1, 2, \ldots, r \\
&Ax \leq b \\
&x \geq 0
\end{aligned}$$

However, even with the compensatory operator, the results are frequently unbalanced. To obtain a balanced result, Lee and Li (1993) proposed the addition of a second phase by using the numerical results of the first phase. Assuming the solution obtained for the first phase is $(\lambda^{(1)}, x^{(1)})$, then the second phase is formulated as:

$$\max \xi = \frac{1}{l+r}\left(\sum_{k=1}^{l} \lambda_{\mathrm{k}} + \sum_{s=1}^{r} \lambda_{\mathrm{s}}\right) \tag{3.42}$$

subject to:

$$\begin{aligned}
&\lambda^{(1)} \leq \lambda_k \leq \tfrac{Z_k(x)-Z_k^-}{Z_k^*-Z_k^-}, \quad k = 1, 2, \ldots, l \\
&\lambda^{(1)} \leq \lambda_s \leq \tfrac{W_s^- - W_s(x)}{W_s^- - W_s^*}, \quad s = 1, 2, \ldots, r \\
&Ax \leq b \\
&x \geq 0
\end{aligned}$$

This two phase approach has been used by Lee and Li (1993) with satisfactory results.

Chapter 4
Fuzzy Interactive Multi-level Decision Making

The duo-ploy or multi-ploy decision-making process discussed in Chaps. 2 and 3 is too rigid and is not, in general, suited for an organization such as a manufacturing or service company. The decision-making process in a well-organized company requires better control of the top level and certainly cannot allow the ambiguity which happens if the rational reaction sets of the lower levels are not singletons. Furthermore, the lower decision should not be the final decision and certainly should allow the inspection of the higher levels. In fact, there should have mutual consultations so that the best objective for the overall company can be achieved. It is true that, depending on the management style of the top level and the organization of the company, different degrees of control of the higher levels to the lower levels may be carried out. Because of these different degrees of control, different or flexible interactions between the different levels should be allowed.

Although some of the algorithms discussed in Chaps. 2 and 3 can be extended or modified to allow this flexibility, the philosophy of the decision process would be completely different and, moreover, the computational requirements would increase tremendously. Since the algorithms are already limited to reasonably medium sized problems, the algorithms resulting from this direct modification or extension would not be efficient.

In this chapter, fuzzy interactive decision making, which allows various degrees of control over the decisions and thus is ideally suited for a manufacturing or service company, will be investigated. In addition, as we shall see later, that this approach has the following advantages:

- The problem is simplified and, at the same time, the representation is more realistic. This is because we are treating a fuzzy and not well defined problem as it is, or, not try to represent it more accurately than it really is;
- The use of fuzzy membership functions to represent the desires or goals of the DMs in different levels offers exceptional flexibility for the interactive decisions proposed;

C.-B. Cheng et al., *Fuzzy and Multi-Level Decision Making: Soft Computing Approaches*, Studies in Fuzziness and Soft Computing 368,
https://doi.org/10.1007/978-3-319-92525-7_4

- Since the problem is much more simplified, it is much easier to solve by using fuzzy multiple-objective decision-making algorithms.

Another example is the solution process of the k-th—best algorithm. It is certainly not reasonable to assume the solution is at the corner point. As mentioned before, this unreasonable requirement resulted in two different optimal solutions for the same problem in Examples 2.1 and 2.3. These different results are due to the fact that the optimum depends on which DM moves first in the search procedure. Furthermore, according to the concept of multi-criteria decision making, the Pareto optimum does not necessarily lie at the vertex. All the best results require are that a compromise non-dominated solution is obtained based on the DM's preferences. From this standpoint, the last few steps of the search procedure, where the winner of the two competing neighboring vertices is decided, are not needed extra computations. We are trying to force a decision instead of the more reasonable compromise decision by consultation.

Instead of searching through the vertices as in the k-th—best or other similar algorithms, a supervised search procedure, which, hopefully, will generate a non-dominated satisfactory solution for the multi-level decision-making process, is presented in this chapter. The use of fuzzy membership functions, which simplify the representation and the computations considerably, to represent the compromises between the different levels implicitly forms the basic idea of the approach.

From another standpoint, fuzzy interactive decision making is essentially a continuation of the duo- or multi-ploy search procedure. However, in order to carry out this continuation, considerable simplification in both the representation of the process and in the computational requirements are needed. The interactive decision approach provides these simplifications by exploring the basic vagueness and the ill-defined natures of most practically encountered hierarchy organizations. In other words, instead of taking the lower decision as the final decision in the multi-ploy approach, the search process continues until all levels involved are satisfied. However, due to the complexity of the algorithms, this continuation cannot be carried out easily with the multi-ploy type algorithms.

In this chapter, the simplest bi-level process will be discussed first. Other structurally more complicated processes such as the bi-level with multi-followers and multi-level processes will be presented afterward. To emphasize the computational aspects, numerical examples will be used to illustrate these approaches.

4.1 Fuzzy Bi-Level Interactive Decision Making

Instead of the duo-ploy type programming approach, consider the fuzzy interactive decision-making process applied to a bi-level organization with a single follower. The upper-level DM specifies the preferred values of his or her control variables and goals with certain amount of tolerance. This information is represented implicitly by the use of membership functions and passed to the lower-level DM. The lower-level DM obtains his or her optimum based on goals and preferences of the upper level and then

presents the results to the upper level. If the upper level agrees with the proposed solutions, a final decision is reached and, for the convenience of description, this decision or solution will be referred to as a satisfactory solution. If he or she rejects this proposal, the DMs in both levels will need to re-evaluate and changes the goals and decisions as well as their corresponding tolerances. This mutually interactive process is continued until a satisfactory solution is reached. This strategy is very flexible. Since the DMs in both levels first seek their optimal solutions in isolation, it does not violate the non-cooperate idea. However, the strategy does require a certain degree of coordination between the different levels.

Mathematically, the upper-level DM first solves the following problem:

$$\begin{aligned} &\max f_1(x_1, x_2) = c_{11}^T x_1 + c_{12}^T x_2 \\ &\text{subject to:} \\ &\quad (x_1, x_2) \in F_2 = \{(x_1, x_2) | A_1 x_1 + A_2 x_2 \le b, x_1, x_2 \ge 0\} \end{aligned} \tag{4.1}$$

the solution of which is assumed as (x_1^U, x_2^U, f_1^U). Independently, the lower-level DM solves:

$$\begin{aligned} &\max f_2(x_1, x_2) = c_{21}^T x_1 + c_{22}^T x_2 \\ &\text{subject to:} \\ &\quad (x_1, x_2) \in F_2 = \{(x_1, x_2) | A_1 x_1 + A_2 x_2 \le b, x_1, x_2 \ge 0\} \end{aligned} \tag{4.2}$$

the solution of which is assumed as (x_1^L, x_2^L, f_2^L). If $(x_1^U, x_2^U) = (x_1^L, x_2^L)$, an optimal or preferred solution is reached. However, the two solutions are usually different because of the conflicting nature of the two objectives. The upper-level DM understands that using the optimal decision x_1^U as a control factor for the lower-level DM is obviously not practical. It is more reasonable to have some tolerance that gives the lower-level DM a wider feasible region to search for his or her optimum. The range of the decision for the vector x_1 can be "around x_1^U" with its maximum tolerance p_1. In other words, the most preferred decision is x_1^U and the worst acceptable decision is at $(x_1^U - p_1)$ or $(x_1^U - p_1)$. The satisfaction or preference can be assumed as linearly increasing within the interval of $[x_1^U - p_1, x_1^U]$ and linearly decreasing within $[x_1^U, x_1^U + p_1]$. Decisions outside the interval $[x_1^U - p_1, x_1^U + p_1]$ are not acceptable. This information can be formulated as the following membership function of the fuzzy set theory:

$$\mu_{x1}(x_1) = \begin{cases} [x_1 - (x_1^U - p_1)]/p_1, & \text{if } x_1^U - p_1 \le x_1 \le x_1^U \\ [(x_1^U + p_1) - x_1]/p_1, & \text{if } x_1^U \le x_1 \le x_1^U + p_1 \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

The upper-level DM must also reset his or her goal with some tolerance. The upper level may adopt a leeway of f_1'. Any value of the goal $f_1 > f_1^U$ is absolutely acceptable and all values of $f_1 < f_1'$ are absolutely unacceptable. The satisfaction or

preference within the interval $[f_1', f_1^U]$ can be assumed as linearly increasing. The membership function can now be assumed as follows:

$$\mu_{f1}(f_1(x)) = \begin{cases} 1, & \text{if } f_1(x) \geq f_1^U \\ [(f_1(x) + f_1')]/[f_1^U - f_1'], & \text{if } f_1' \leq f_1(x) \leq f_1^U \\ 0, & \text{if } f_1(x) \leq f_1' \end{cases} \tag{4.4}$$

As an initial approximation, the value of f_1' is usually set equal to $f_1'(x_1^L, x_2^L)$.

The lower-level DM now optimizes her objective function under the new constraints of "x_1 is about x_1^U" and "f_1 is somehow near to or greater than f_1^U", which are modeled by the membership functions, i.e. Eqs. (4.3) and (4.4). Using Eqs. (4.2–4.4), the problem of the lower level becomes:

$$\begin{aligned} &\max_{x2} \ f_2(x_1, x_2) = c_{21}^T x_1 + c_{22}^T x_2 \\ &\text{subject to:} \\ &\quad A_1 x_1 + A_2 x_2 \leq b \\ &\quad f_1(x) \geq f_1 \\ &\quad x_1 \cong x_1^U \\ &\quad x_1, x_2 \geq 0 \end{aligned} \tag{4.5}$$

or

$$\begin{aligned} &\max_{x2} \ f_2(x_1, x_2) = c_{21}^T x_1 + c_{22}^T x_2 \\ &\text{subject to:} \\ &\quad A_1 x_1 + A_2 x_2 \leq b \\ &\quad \mu_{f1}(f_1(x)) \geq \alpha \\ &\quad \mu_{x1}(x_1) \geq \beta \\ &\quad \alpha \in [0, 1] \\ &\quad \beta \in [0, 1] \\ &\quad x_1, x_2 \geq 0 \end{aligned} \tag{4.6}$$

where scalars α and β are the minimum acceptable degrees of satisfaction for the objective $f_1(x)$ and the decision x_1, respectively. These feasible ranges are constrained by $\mu_{f1}(f_1(x))$ and $\mu_{x1}(x_1)$. The lower-level DM can analyze the various solutions corresponding to the upper-level DM's satisfactory levels α and β. These acceptable degrees of satisfaction are generally decided by the DMs.

In addition to the tolerance of the DM of the upper level, the lower-level DM may also have tolerance or preference level. This preference can again be represented by membership function so that he or she can rate the satisfaction of each potential solution. Let this membership function be defined as:

$$\mu_{f2}(f_2(x)) = \begin{cases} 1, & \text{if } f_2(x) \geq f_2^L \\ [(f_2(x) + f_2')] \,/\, [f_2^L - f_2'], & \text{if } f_2' \leq f_2(x) \leq f_2^L \\ 0, & \text{if } f_2(x) \leq f_2' \end{cases} \tag{4.7}$$

where $f_2' = f_2(x_1^T)$. Following the discussions made in this chapter, the lower-level DM must maximize his or her objective function within the interval $[f_2', f_2^L]$. Thus, we have:

$$\begin{aligned} &\max \delta = \mu_{f2}(f_2(x)) \\ &\text{subject to:} \\ &\quad A_1x_1 + A_2x_2 \leq b \\ &\quad \mu_{f1}(f_1(x)) \geq \alpha \\ &\quad \mu_{x1}(x_1) \geq \beta \\ &\quad \alpha \in [0, 1] \\ &\quad \beta \in [0, 1] \\ &\quad x_1, x_2 \geq 0 \end{aligned} \tag{4.8}$$

or

$$\begin{aligned} &\max \delta \\ &\text{subject to:} \\ &\quad A_1x_1 + A_2x_2 \leq b \\ &\quad \mu_{f1}(f_1(x)) \geq \alpha \\ &\quad \mu_{x1}(x_1) \geq \beta \\ &\quad \mu_{f2}(f_2(x)) \geq \delta \\ &\quad \alpha \in [0, 1] \\ &\quad \beta \in [0, 1] \\ &\quad \delta \in [0, 1] \\ &\quad x_1, x_2 \geq 0 \end{aligned} \tag{4.9}$$

where δ is the degree of satisfactory of the lower-level DM's objective function. In order to satisfy both upper and the lower levels or all the degrees of satisfaction, the solution must be the minimum of $\{\alpha, \beta, \delta\}$, or min $\{\alpha, \beta, \delta\}$. Let:

$$\lambda = \min\{\alpha, \beta, \delta\} \tag{4.10}$$

where λ is a fuzzy number resulting from the intersection of the three membership functions. In order to obtain the maximum degree of satisfaction of this fuzzy number, we must have:

$$\begin{aligned}
&\max \lambda = \max(\min\{\alpha, \beta, \delta\}) \\
&\text{subject to:} \\
&\quad A_1 x_1 + A_2 x_2 \leq b \\
&\quad \mu_{f1}(f_1(x)) \geq \alpha \\
&\quad \mu_{x1}(x_1) \geq \beta \\
&\quad \mu_{f2}(f_2(x)) \geq \delta \\
&\quad \alpha \in [0, 1] \\
&\quad \beta \in [0, 1] \\
&\quad \delta \in [0, 1] \\
&\quad x_1, x_2 \geq 0
\end{aligned} \tag{4.11}$$

Equation 6.10 can also be written as:

$$\begin{aligned}
&\mu_{f1}(f_1(x)) \geq \alpha \geq \lambda \\
&\mu_{x1}(x_1) \geq \beta \geq \lambda \\
&\mu_{f2}(f_2(x)) \geq \delta \geq \lambda
\end{aligned}$$

Using the above three inequalities, the three degrees of satisfaction, α, β and δ, in Eq. (4.11) can be eliminated. Thus, Eq. 6.11 becomes:

$$\begin{aligned}
&\max \lambda \\
&\text{subject to:} \\
&\quad A_1 x_1 + A_2 x_2 \leq b \\
&\quad \mu_{f1}(f_1(x)) \geq \lambda \\
&\quad \mu_{x1}(x_1) \geq \lambda \\
&\quad \mu_{f2}(f_2(x)) \geq \lambda \\
&\quad \lambda \in [0, 1] \\
&\quad x_1, x_2 \geq 0
\end{aligned} \tag{4.12}$$

or

$$\begin{aligned}
&\max \lambda \\
&\text{subject to:} \\
&\quad A_1 x_1 + A_2 x_2 \leq b \\
&\quad [f_1(x) - f_1']/[f_1^U - f_1'] \geq \lambda \\
&\quad [(x_1^U + p_1) - x_1']/p_1 \geq \lambda \\
&\quad [x_1 - (x_1^U - p_1]/p_1 \geq \lambda \\
&\quad [f_2(x) - f_2']/[f_2^L - f_2'] \geq \lambda \\
&\quad \lambda \in [0, 1] \\
&\quad x_1, x_2 \geq 0
\end{aligned} \tag{4.13}$$

where the constraint on x_1 is a vector, the dimension of which depends on the number of decisions of the upper level. Eq. (4.11) is a max-min programming problem proposed by Zimmermann (1978) based on the suggestions of Bellman and Zadeh (1970).

If the upper-level DM is satisfied with the solution of Eq. (4.11), a satisfactory solution is obtained. If not satisfied, the upper level should provide new membership functions for the control variable and the objective to the lower level. This process is continued until a satisfactory solution is reached. Combined with the set of control decisions and goals with tolerances, this solution becomes a satisfactory solution for Eqs. (4.1) and (4.2). Figure 4.1 shows the flow chart of this proposed procedure, which can be summarized in the following three steps:

Step 1. The upper and lower level's DMs solve their problems independently by solving Eqs. (4.1) and (4.2). If these two solutions coincide, the optimal or preferred solution of the system is obtained. Stop. Otherwise, go to Step 2.

Step 2. The upper-level OM decides her tolerances on the goal and the decisions in terms of membership functions. Meanwhile, the lower-level DM also decides his tolerance on the goal in terms of membership function. These membership functions will serve as extra constraints in forming the auxiliary problem, Eq. (4.13).

Step 3. Solve the auxiliary problem. If the DMs at each level are satisfied with the solution, a compromise solution is reached. Stop. Otherwise, got to Step 2 to obtain newly adjusted membership functions.

To illustrate the approach, let us consider the following example.

Example 4.1 Problem of balancing export and import solved in Examples 2.1 and 2.3 is again solved by the present approach. The problem is reproduced in the following:

$$\begin{aligned} &\max_{x_1} \ f_1 = 2x_1 - x_2 \quad \text{(upper level)} \\ &\text{where } x_2 \text{ solves:} \\ &\max_{x_2} \ f_2 = x_1 + 2x_2 \quad \text{(lower level)} \\ &\text{subject to:} \end{aligned} \tag{4.14a}$$

$$\begin{aligned} &3x_1 - 5x_2 \le 15 \\ &3x_1 - x_2 \le 21 \\ &3x_1 + x_2 \le 27 \\ &3x_1 + 4x_2 \le 45 \\ &x_1 + 3x_2 \le 30 \\ &x_1, x_2 \ge 0 \end{aligned} \tag{4.14b}$$

The proposed approach first solves Eqs. (4.14a) and (4.14b) separately subject to the above constraint. The solutions obtained are $\left(x_1^U, x_2^U\right) = (7.5, 1.5)$ with ${f_1}^U = 13.5$ and $\left(x_1^L, x_2^L\right) = (3, 9)$ with $f_2^L = 21$. Using this solution as reference, set ${f_1}^U = 13.5$ and assume $f_1' = 0$ and ${f_2}^U = 10.5$. Let the upper-level DM's decision

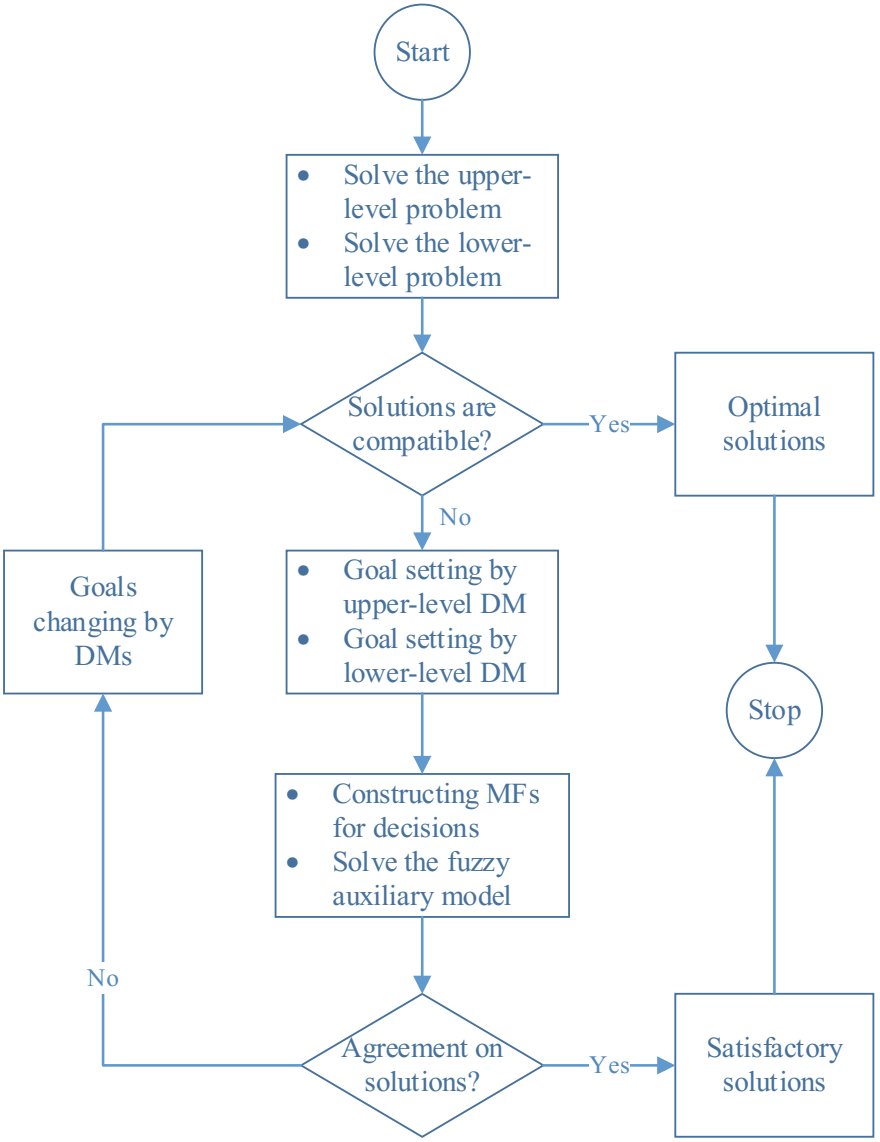

Fig. 4.1 Fuzzy interactive decision making

variable x_1, be around 7.5 with the negative- and positive-side tolerances equal to 4.5 and 0.5, respectively. Using Eqs. (4.3), (4.4) and (4.7), the membership functions $\mu_{f1}(\cdot)$, $\mu_{x1}(\cdot)$ and $\mu_{f2}(\cdot)$ are obtained. The lower-level DM then solves the following problem:

$$\max \lambda = \max(\min\{\alpha, \beta, \delta\})$$

subject to:

$$\begin{aligned}
&x \in X \\
&\mu_{f1}(f_1(x)) = (f_1 - 0)/(13.5 - 0) \geq \alpha \\
&\mu_{x1}(x_1) = (x_1 - 4.5)/(7.5 - 4.5) \geq \beta \\
&\mu_{x1}(x_1) = (8 - x_1)/(8 - 7.5) \geq \beta \\
&\mu_{f2}(f_2(x)) = (f_2 - 10.5)/(21 - 10.5) \geq \delta \\
&x_1, x_2 \geq 0 \\
&\alpha, \beta, \delta \in [0, 1]
\end{aligned}$$

Using the same argument as that used in obtaining Eq. 6.12, we can obtain the following equation:

$$\max \lambda$$

subject to:

$$\begin{aligned}
&x \in X \\
&x_1 \geq 4.5 + 3\lambda \\
&x_1 \leq 8 - 0.5\lambda \\
&f_1 = 2x_1 - x_2 \geq 13.5\lambda \\
&f_2 = x_1 + 2x_2 \geq 10.5\lambda + 10.5 \\
&\lambda \in [0, 1] \\
&x_1, x_2 \geq 0
\end{aligned}$$

the compromise solution of which is $f^* = (f_1^*, f_2^*) = (9.29, 17.72)$ with $x^* = (x_1^*, x_2^*) = (7.26, 5.23)$. This solution is a point between vertices D and C in Figs. 2.1 and 2.2. The overall satisfaction of the present solution is $\lambda = 0.688$. The satisfaction levels are $(\alpha, \beta, \delta) = (0.688, 0.947, 0.688)$. If the upper-level DM's total satisfactory level is $\lambda_1 = \min\{\alpha, \beta\}$, then the satisfaction levels for upper and lower levels are $\lambda_1 = 0.688$ and $\lambda_2 = \delta = 0.688$. On the other hand, the k-th—best solution gave f = (13, 14) with x = (8, 3), which has a satisfaction level of $(\alpha, \beta, \delta) = (0.963, 0, 0.333)$. and thus $\lambda_1 = 0$ and $\lambda_2 = 0.333$. Thus, the present solution is better balanced than that of the k-th—best approach.

If the solution is not satisfactory to anyone of the DMs, adjustment of the tolerances can be made. Again, this process can be continued until a solution, which is satisfactory to all the DMs is reached.

4.2 Fuzzy Bi-Level Interactive Decision Making with Multi-followers

The duo-ploy type decision-making process with multi-followers, or bi-level distributed programming problem (BLDPP), was discussed in Chap. 3. However, for the decision process in most organizations, more cooperation than the duo-ploy type is needed. The fuzzy interactive decision-making process discussed in the last section can be easily extended to the bi-level organization with multi-followers. For the easy of discussion, the bi-Level decentralized problem with multi-followers, Eq. (2.1), is reproduced in the following:

$$\max_{x11} \; f_{11}(x) = \sum\nolimits_j c_{21j}^T x_j \quad \text{(upper level)} \tag{4.15a}$$

where $x_{21}, x_{22}, \ldots, x_{2p}$ solve:

$$\begin{cases} \max_{x21} \; f_{21}(x) = \sum\limits_j c_{21j}^T x_j \\ \ldots \qquad\qquad\qquad\qquad \text{(lower level)} \\ \max_{x2p} \; f_{2p}(x) = \sum\limits_j c_{2pj}^T x_j \end{cases}$$

subject to:

$$\begin{aligned} &\sum_{k,i} A_{ki} x_{ki} \le b, k = 1, 2, \text{and } i = 1, 2, \ldots, s_k \\ &x_j \ge 0, \; j = 1, 2, \ldots, n \end{aligned} \tag{4.15b}$$

For convenience, let X represent the above constraint set.

First solve each DM's problem separately subject to the constraints of Equations 6.15b:

$$f_{ki}^* = f_{ki}(x_{ki}^*) = \max_{x \in X} f_{ki}(x) \tag{4.16}$$

for all k and i, where the superscript * indicates the actual solution, or the individual optimum values obtained.

Then, follow the same argument used before, that is, the minimum satisfaction level would satisfy all the DMs. Or, let:

$$f_{ki}' = \max_{k,i} \; f_{ki}(x_{ki}^*) \tag{4.17}$$

where the minimization is with respect to all the degrees of satisfaction of all the DMs. If the tolerance vector for the upper DM is given, then the membership functions, $\mu_{x1}(x_1)' = \mu_{ki}(f_{ki}(x))$, can be formulated in the same way as before. Thus the lower-level DMs' problem becomes:

$$\begin{cases} \max_{x21} \ f_{21}(x) = \sum_j c_{21j}^T x_j \\ \ldots \\ \max_{x2p} \ f_{2p}(x) = \sum_j c_{2pj}^T x_j \end{cases}$$

subject to:

$$\begin{aligned} & x \in X \\ & \mu_{f1}(f_1(x)) \geq \alpha \\ & \mu_{x1}(x_1) \geq \beta \\ & \alpha \in [0, 1] \\ & \beta \in [0, 1] \end{aligned} \tag{4.18}$$

Introducing the lower-level tolerance function and eliminate the degrees of satisfaction vector, we have:

$$\begin{aligned} & \max \lambda \\ & \text{subject to:} \\ & \quad x \in X \\ & \quad \mu_{x1}(x_1) \geq \lambda \\ & \quad \mu_{f1}(f_1(x)) \geq \lambda \\ & \quad \mu_{f2i}(f_{2i}(x)) \geq \lambda_i \\ & \quad \lambda \in [0, 1] \\ & \quad \lambda_1 \in [0, 1] \end{aligned} \tag{4.19}$$

with $i = 1, 2, \ldots, p$. Substituting the definition of the membership functions into the above equation, we finally have:

$$\begin{aligned} & \max \lambda \\ & \text{subject to:} \\ & \quad x \in X \\ & \quad [(x_1^U + p_1) - x_1]/p_1 \geq \lambda \\ & \quad [x_1 - (x_1^U - p_1)]/p_1 \geq \lambda \\ & \quad \mu_{f1}(f_1(x)) = [f_1(x) - f_1']/[f_1^T - f_1'] \geq \lambda \\ & \quad \mu_{f2i}(f_{2i}(x)) = [f_{2i}(x) - f_{2i}']/[f_{2i}^T - f_{2i}'] \geq \lambda_i \\ & \quad \lambda \in [0, 1] \\ & \quad \lambda_i \in [0, 1], i = 1, 2, \ldots, p \end{aligned} \tag{4.20}$$

Solving Eq. (4.20), the degrees of satisfaction of the various DMs can be obtained. If these degrees of satisfactions do not satisfy all the DMs, the membership functions or the tolerances of the DMs are re-adjusted. Again, this process can be continued until a satisfactory solution is reached.

In order to obtain the initial solution of Eq. (4.16), notice that the problem such as Eq. (4.15) really represents two problems, the top-level problem and the bottom-level problem. By itself, the top-level problem is fairly simple. However, since it is imbedded in the lower level, it would be ideal if we could solve the lower level first. Since all the divisions in the bottom level have the same position regarding to the decision sequence, or, they make the decisions simultaneously and in isolation based on the top level's decision, the problem of the bottom level with a given x_1 is no longer a hierarchy. In fact, the bottom level, by itself, is a one-level optimization problem. If we allow mutual interactions, the bottom-level decision process is essentially a multi-objective optimization problem. Thus, by using the fuzzy multi-objective decision-making algorithms discussed in Chap. 3, the bottom-level problem of Eq. (4.16) can be solved.

The approach is illustrated by solving the following example.

Example 4.2 Consider the following bi-Level problem with three followers, which was solved by Anandalin dam (1988) and also solved in Example 2.1.

$$\max_{x1} \ f_1 = x_1 + y_1 + 2y_2 + y_3 \tag{4.21}$$

where y_1, y_2, and y_3 solve:

$$\begin{cases} \max_{y1} \ f_{21} = -x_1 + 3y_1 - y_2 - y_3 \\ \max_{y2} \ f_{22} = -x_1 - y_1 + 3y_2 - y_3 \\ \max_{y3} \ f_{23} = -x_1 - y_1 - y_2 - y_3 \end{cases}$$

subject to:

$$\begin{aligned} & 3x_1 + 3y_1 \le 30 \\ & 2x_1 + y_1 \le 20 \\ & y_2 \le 10 \\ & y_2 + y_3 \le 15 \\ & y_3 \le 10 \\ & x_1 + 2y_1 + 2y_2 + y_3 \le 40 \\ & x_1, y_1, y_2, y_3 \ge 0 \end{aligned}$$

Individual optimal solutions of the top and bottom-level DMs subject to the constraints of Eq. (4.21) are listed in Table 4.1, where the "*" represents the optimum solution. Using these solutions as basis, we can assume that $f'_{11} = 0$ and $f'_{21} = f'_{22} = f'_{23} = 0$. Values within the interval (6, 10) for x_1 are fully satisfied to the upper-level DM. The negative and positive side tolerances for x_1 are 5 and 0, respectively. The auxiliary equation now becomes:

$$\begin{aligned} & \max \lambda \\ & \text{subject to:} \end{aligned}$$

Table 4.1 Solutions of individual objectives, Example 4.2

Vertex	f_1	f_{21}	f_{22}	f_{23}
A (10, 0, 10, 5)	35*	−25	15	−5
B (5, 5, 10, 5)	35*	−5	15	−5
C (0, 10, 0, 0)	10	30*	−10	−10
D (0, 0, 10, 0)	20	−10	30*	−10
E (0, 0, 0, 10)	10	−10	−10	30*
Fuzzy range	(10, 35)	(0, 30)	(0, 30)	(0, 30)

$$
\begin{aligned}
&\boldsymbol{x} \in \boldsymbol{X} \\
&(x_1 + y_1 + 2y_2 + y_3 - 10)/(35 - 10) \geq \alpha \\
&(x_i - 0)/(5 - 0) \geq \beta \\
&x_1 \leq 10 \\
&(-x_1 + 3y_1 - y_2 - y_3 - 0)/(30 - 0) \geq \gamma_1 \\
&(-x_1 - y_1 + 3y_2 - y_3 - 0)/(30 - 0) \geq \gamma_2 \\
&(-x_1 - y_1 - y_2 + 3y_3 - 0)/(30 - 0) \geq \gamma_3
\end{aligned}
$$

Solving the above equation, we obtain the satisfactory solution $\boldsymbol{f} = (f_1,\ f_{21},\ f_{22},\ f_{23}) = (31.07, 6.43, 6.43, 6.43, 6.43)$ with $\boldsymbol{x} = (x_1,\ y_1,\ y_2,\ y_3) = (1.07, 7.5, 7.5, 7.5)$. The satisfaction levels are (0.21, 0.21, 0.21, 0.21) for all the DMs. The solution of Example 3.1 and also Anandalingam's solution (1988) is $\boldsymbol{f} = (35, -5, 15, -5)$ with $\boldsymbol{x} = (5, 5, 10, 5)$, whose satisfaction is (1, 0, 0.5, 0). Thus, the proposed solution is much more reasonable and more evenly distributed.

4.3 Fuzzy Multi-level Interactive Decision Making

A multi-level decision-making process in an organization, in which there are K nested hierarchy levels, can be represented as:

$$\max_{x1}\ f_1(x) = \sum_j c_{1j}^T x_j \qquad \text{(1st level)} \tag{4.22}$$

where $x_2, x_3, \ldots, x_K$ solve:

$$\max_{x2}\ f_2(x) = \sum_j c_{2j}^T x_j \qquad \text{(2nd level)}$$

where $x_3, \ldots, x_K$ solve : . . . where x_K solve:

$$\max_{xk}\ f_K(x) = \sum_j c_{Kj}^T x_j \qquad (k \text{ - th level})$$

subject to:

$$\sum_k A_k x_k \leq b, k = 1, 2, \ldots, K$$
$$x_j \geq 0, j = 1, 2, \ldots, n$$

where the decision variable set $\cup_k \{x_k | k = 1, 2, \ldots, K\} = \{x_1, x_2, \ldots, x_n\} = \{x\}$ and x_K is the control variable vector of the k-th level DM.

The solution process proceeds as follows. The top-level DM provides his or her preferred ranges of f_1 and $\boldsymbol{x}_1$ to the second-level DM, who solves his or her problem based on the preference information, in terms of membership functions, of the top level. The solution process is essentially the same as that discussed in Sect. 4.1. The resulting satisfactory solution based on membership functions become additional constraints for the third-level DM. The solution of the third-level DM is presented to the upper levels. If any upper level is not satisfied with this solution, the third-level DM will then solve a new problem with new membership functions until a satisfactory solution is reached. This procedure continues until the k-th—level DM, whose solution must satisfy all DMs.

With the required membership functions represented as that of Eqs. (4.3) and (4.4), the k^{th} level DM has the following problem:

$$\begin{aligned}
&\max_{x_k} \ f_k(x) \\
&\text{subject to:} \\
&\quad x \in X \\
&\quad \mu_{f1}(f_1(x)) \geq \alpha_1 \text{ and } \mu_{x_1}(x_1) \geq \beta_1 \\
&\quad \mu_{f2}(f_2(x)) \geq \alpha_2 \text{ and } \mu_{x_2}(x_2) \geq \beta_2 \\
&\quad \cdots \\
&\quad \mu_{f(k-1)}(f_{(k-1)}(x)) \geq \alpha_{(k-1)} \text{ and } \mu_{x(k-1)}(x_{(k-1)}) \geq \beta_{(k-1)} \\
&\quad x_1, x_2 \geq 0 \\
&\quad \alpha_i \in [0, 1], i = 1, 2, \ldots, (k-1) \\
&\quad \beta_i \in [0, 1], i = 1, 2, \ldots, (k-1)
\end{aligned} \tag{4.23}$$

Following the same approach as that discussed in Sect. 4.1, we finally obtain the following expression for a problem with K levels:

$$\begin{aligned}
&\max_{xk}\ f_k(x)\\
&\text{subject to:}\\
&\qquad x \in X\\
&\qquad \mu_{f1}(f_1(x)) \geq \alpha_1 \text{ and } \mu_{x_1}(x_1) \geq \beta_1\\
&\qquad \mu_{f2}(f_2(x)) \geq \alpha_2 \text{ and } \mu_{x_2}(x_2) \geq \beta_2\\
&\qquad \cdots\\
&\qquad \mu_{f(K-1)}(f_{(K-1)}(x)) \geq \alpha_{(K-1)} \text{ and } \mu_{x(K-1)}(x_{(K-1)}) \geq \beta_{(k-1)}\\
&\qquad \mu_{fK}(f_K(x)) \geq \alpha_K\\
&\qquad \alpha_i \in [0,1], i = 1,2,\ldots,k\\
&\qquad \beta_i \in [0,1], i = 1,2,\ldots,(k-1)
\end{aligned} \tag{4.24}$$

or

$$\begin{aligned}
&\max \lambda\\
&\text{subject to:}\\
&\qquad \boldsymbol{x} \in \boldsymbol{X}\\
&\qquad \mu_{f1}(f_1(\boldsymbol{x})) \geq \alpha_1 \text{ and } \mu_{x1}(\boldsymbol{x}_1) \geq \lambda\\
&\qquad \mu_{f2}(f_2(\boldsymbol{x})) \geq \alpha_2 \text{ and } \mu_{x2}(\boldsymbol{x}_2) \geq \lambda\\
&\qquad \cdots\\
&\qquad \mu_{f(K-1)}(f_{(K-1)}(\boldsymbol{x})) \geq \lambda \text{ and } \mu_{x(K-1)}(\boldsymbol{x}_{(K-1)}) \geq \lambda\\
&\qquad \mu_{fK}(f_K(\boldsymbol{x})) \geq \lambda\\
&\qquad \lambda \in [0,1]
\end{aligned} \tag{4.25}$$

where $\lambda = \min\{\alpha_1, \alpha_2, \ldots, \alpha_{(k-1)}, \alpha_K, \beta_1, \beta_2, \ldots, \beta_{(k-1)}\}$.

Using Eq. 4.25, the $(K-1)$ auxiliary problems can be solved. If the solution does not satisfy some of the upper-level DMs, the current lowermost DM must solve the problem again with modified membership functions.

Example 4.3 Consider the following three-level problem which was solved before in Example 2.3 and also by Anandalingdam (1988):

$$\begin{aligned}
&\max_{x_1}\ f_1 = 7x_1 + 3x_2 + 4x_3\\
&\text{where } x_2 \text{ and } x_3 \text{ solve:}\\
&\quad \max_{x_2}\ f_2 = x_2\\
&\quad \text{where } x_3 \text{ solve:}\\
&\qquad \max_{x3}\ f_3 = x_3\\
&\qquad \text{subject to:}
\end{aligned}$$

$$
\begin{aligned}
&x_1 + x_2 + x_3 \leq 3\\
&x_1 + x_2 - x_3 \leq 20\\
&x_1 + x_2 + x_3 \geq 1\\
&-x_1 + x_2 + x_3 \leq 1\\
&x_3 \leq 0.5\\
&x_1, x_2, x_3 \geq 0
\end{aligned}
$$

For simplicity, $\boldsymbol{X}$ will be used to represent the above set of constraints.

First, using the constraint set $\boldsymbol{X}$, the problem is solved individually for each DM. The solutions obtained are $f^{1*} = 8.5$ with $\boldsymbol{x}^{1*} = (1.5, 0, 0.5)$; $f^{2*} = 1$ with $\boldsymbol{x}^{2*} = (0, 1, 0)$ or $(0.5, 1, 0.5)$; and $f^{3*} = 0.5$ with $\boldsymbol{x}^{3}* = (1.5, 0, 0.5), (0.5, 1, 0.5)$, or $(0, 0.5, 0.5)$. The results are also listed in Table 4.2, where the items with * represents the optimum solution. Based on this solution, the limits are set at $f_1' = 3$, $f_2' = 0$ and $f_3' = 0$. Assuming the decision variable $\boldsymbol{x}_1$ should be around 1.5 with negative and positive-sides tolerances 1.5 and 0, respectively. With $\lambda = \min(\alpha_1, \beta_1, \alpha_2)$, the auxiliary problem for the second level should be:

$$
\begin{aligned}
&\max_{x2}\ f_2 = x_2\\
&\text{subject to:}\\
&\qquad \boldsymbol{x} \in \boldsymbol{X}\\
&\qquad (7x_1 + 3x_2 + 4x_3 - 3)/(8.5 - 3) \geq \alpha_1\\
&\qquad \boldsymbol{x}_1 \geq 1.5\beta_1\\
&\qquad \boldsymbol{x}_1 \leq 1.5\\
&\qquad \alpha_1, \beta_1 \in [0, 1]
\end{aligned}
$$

or

$$
\begin{aligned}
&\max \lambda\\
&\text{subject to:}\\
&\qquad \boldsymbol{x} \in \boldsymbol{X}\\
&\qquad (7x_1 + 3x_2 + 4x_3 - 3)/(8.5 - 3) \geq \lambda\\
&\qquad \boldsymbol{x}_1 \geq 1.5\lambda\\
&\qquad \boldsymbol{x}_1 \leq 1.5\\
&\qquad \lambda \in [0, 1]
\end{aligned}
$$

Solving this second-level problem, the compromised optimal solution is $\boldsymbol{f} = (f_1, f_2) = (6.18, 0.58)$ with $\boldsymbol{x} = (x_1, x_2, x_3) = (0.92, 0.58, 0.5)$. The satisfaction level for this solution is $\lambda = 0.58$. This information is transferred to the third level. Since the information has been compromised, some modifications are needed to meet the optimum for the whole organization. After modification, the triangular

Table 4.2 Solutions of individual objectives, Example 6.3

Vertex	f_1	f_2	f_3	Note
First problem				
A (1.5, 0, 0.5)	8.5	0	(0.5)	Level I optimum
B (0, 1,0)	*3	1*	(0)	Level 2 optimum
C (0.5, 1, 0.5)	4.5	1*	(0.5)	Alternative optimum
D (0.92, 0.58, 0.5)	6.18	0.58	(0.5)	Compromise optimum
Second problem				
E (1.5, 0, 0.5)	8.5	0	0.5*	Level 3 optimum
F (0.5, 1,0.5)	4.5	1	0.5*	Alternative optimum
G (0, 0.5, 0.5)	−0.5	0.5	0.5*	Alternative optimum
Modified fuzzy range	$n=1$	$n=2$	$n=3$	
f_n	(0, 6.18, 8.5)	(0,0.58, 1)	(0, 0.5, 0.5)	Objective
x_n	(0, 0.92, 1.5)	(0, 0.58, 1)	–	Decision variable

fuzzy numbers for the objective functions and decisions are $f_1 \in [0, 6.18, 8.5]$, $f_2 \in [0, 0.58, 1]$ and $x_1 \in [0, 0.92, 1.5]$, and $x_2 \in [0, 0.58, 1]$ (see Table 4.2).

Now, the third-level problem can be formulated in the same way as before. Let $\lambda = \min(\alpha_1, \beta_1, \alpha_2, \beta_2, \alpha_3)$ and based on Eqs. (4.24) and (4.25), we have the auxiliary problem for the third level:

$$\max_{x3} \; f_3 = x_3$$

subject to:

$$\begin{aligned}
&\boldsymbol{x} \in \boldsymbol{X} \\
&(7x_1 + 3x_2 + 4x_3 - 0)/(6.18 - 0) \geq \alpha_1 \\
&(8.5 - 7x_1 - 3x_2 + 4x_3)/(8.5 - 6.18) \geq \alpha_1 \\
&(\boldsymbol{x}_1 - 0)/(0.92 - 0) \geq \beta_1 \\
&(1.5 - \boldsymbol{x}_1)/(1.5 - 0.92) \geq \beta_1 \\
&(x_2 - 0)/(0.58 - 0) \geq \alpha_2 \\
&(1 - x_2)/(1 - 0.58) \geq \alpha_2 \\
&(x_2 - 0)/(0.58 - 0) \geq \beta_2 \\
&(1 - x_2)/(1 - 0.58) \geq \beta_2 \\
&\alpha_1, \beta_1, \alpha_2, \beta_2 \in [0, 1]
\end{aligned}$$

or

$$\max \lambda$$

subject to:

$$
\begin{aligned}
&\boldsymbol{x} \in \boldsymbol{X}\\
&7x_1 + 3x_2 + 4x_3 \geq 6.18\,\lambda\\
&7x_1 - 3x_2 + 4x_3 \geq -2.32\,\lambda + 8.5\\
&\boldsymbol{x}_1 \geq 0.92\,\lambda\\
&\boldsymbol{x}_1 \geq -0.58\,\lambda + 1.5\\
&x_2 \geq 0.58\,\lambda\\
&x_2 \geq -0.42\,\lambda + 1\\
&x_3 \geq 0.5\,\lambda\\
&\lambda \in [0, 1]
\end{aligned}
$$

where $\boldsymbol{X}$ is the constraint set of the original problem.

The compromise solution obtained is $\boldsymbol{f} = (f_1,\ f_2,\ f_3) = (6.18, 0.58, 0.5)$ with $\boldsymbol{x} = (x_1, x_2, x_3) = (0.92, 0.58, 0.5)$. The satisfaction level is (1, 1, 1). In comparison with the results of Example 3.3, which is $\boldsymbol{f} = (8.5, 0, 0.5)$ with $\boldsymbol{x} = (1.5, 1, 0.5)$ and with satisfaction level of (1, 0, 1), the present solution is much better balanced.

4.4 Fuzzy Multi-level Interactive Decision Making with Multi-followers

Structurally, the most general hierarchy system is the decentralized organization, which has multi-level and with more than one division in each level except the top level. This hierarchical structure is illustrated in Fig. 4.2. Mathematically, this structure can be represented by the following set of equations:

$$\max_{x1}\ f_{11}(\boldsymbol{x}) = \sum_j c_{11j}^T \boldsymbol{x}_j,\ j = 1, 2, \ldots, n \qquad \text{(1st level)} \tag{4.26a}$$

where $\boldsymbol{x}_{21}, \boldsymbol{x}_{22}, \ldots, \boldsymbol{x}_{2p}$ individually solve:

$$
\begin{cases}
\max_{x21}\ f_{21}(\boldsymbol{x}) = \sum_j c_{21j}^T \boldsymbol{x}_j,\ j = 1, 2, \ldots, n\\
\ldots \qquad \text{(2nd level)}\\
\max_{x2p}\ f_{2p}(\boldsymbol{x}) = \sum_j c_{2pj}^T \boldsymbol{x}_j,\ j = 1, 2, \ldots, n
\end{cases} \tag{4.26b}
$$

where $\boldsymbol{x}_{31}, \boldsymbol{x}_{32}, \ldots, \boldsymbol{x}_{3q}$ individually solve:

$$\begin{cases} \max_{x31} \ f_{31}(\boldsymbol{x}) = \sum_j \boldsymbol{c}_{31j}^T \boldsymbol{x}_j, \ j = 1, 2, \ldots, n \\ \ldots \\ \max_{x3q} \ f_{3q}(\boldsymbol{x}) = \sum_j \boldsymbol{c}_{3qj}^T \boldsymbol{x}_j, \ j = 1, 2, \ldots, n \end{cases} \quad \text{(3rd level)} \tag{4.26c}$$

where $\boldsymbol{x}_{41}, \boldsymbol{x}_{42}, \ldots, \boldsymbol{x}_{4r}$ individually solve:

$$\begin{cases} \max_{x41} \ f_{41}(\boldsymbol{x}) = \sum_j \boldsymbol{c}_{41j}^T \boldsymbol{x}_j, \ j = 1, 2, \ldots, n \\ \ldots \\ \max_{x4r} \ f_{4r}(\boldsymbol{x}) = \sum_j \boldsymbol{c}_{4rj}^T \boldsymbol{x}_j, \ j = 1, 2, \ldots, n \end{cases} \quad \text{(4th level)} \tag{4.26d}$$

…

where $\boldsymbol{x}_{K1}, \boldsymbol{x}_{K2}, \ldots, \boldsymbol{x}_{Ks}$ individually solve:

$$\begin{cases} \max_{xk1} \ f_{K1}(\boldsymbol{x}) = \sum_j \boldsymbol{c}_{K1j}^T \boldsymbol{x}_j, \ j = 1, 2, \ldots, n \\ \ldots \\ \max_{xks} \ f_{Ks}(\boldsymbol{x}) = \sum_j \boldsymbol{c}_{Ksj}^T \boldsymbol{x}_j, \ j = 1, 2, \ldots, n \end{cases} \quad (k \text{ - th level})$$

subject to:

$$\begin{aligned} &\sum_j \boldsymbol{A}_j \boldsymbol{x}_j \le \boldsymbol{b}, \ j = 1, 2, \ldots, n \\ &\boldsymbol{x}_j \ge 0, \ j = 1, 2, \ldots, n \\ &n = 1 + pq \ldots s \end{aligned} \tag{4.26e}$$

where $j = 1,\ldots, n$, represents the decision variable vector of the j-th division, and $k = 1,\ldots, K$ represents the k^{th} level. It should be noted that there are p divisions at the second level. For the third level, there are q divisions under each division of the second level. Or, a total of qp divisions at the third level. For example, the third-level divisions $\boldsymbol{x}_{31},\ldots, \boldsymbol{x}_{3(q1)}$ are under the control of division 21 in the second level and the divisions $\boldsymbol{x}_{3(q1+1)},\ldots, \boldsymbol{x}_{3\,(q1+q2)}$ are under the control of division 22 in the second level. For simplicity, we did not indicate all the sub-divisions for the third level. In other words, Eq. (4.26c) should have repeated p times. Each p repetitions are under the control of one of the divisions in the second level. For the fourth level, there are r divisions under each division of the third level. Or, Eq. (4.26d) should have repeated q times. Thus, for levels $k = 3,\ldots, K$, not all the sub-divisions are listed in Eq. (4.26). We also assumed that, for the k-th level, there are s divisions under each division of the $k - 1$-th level.

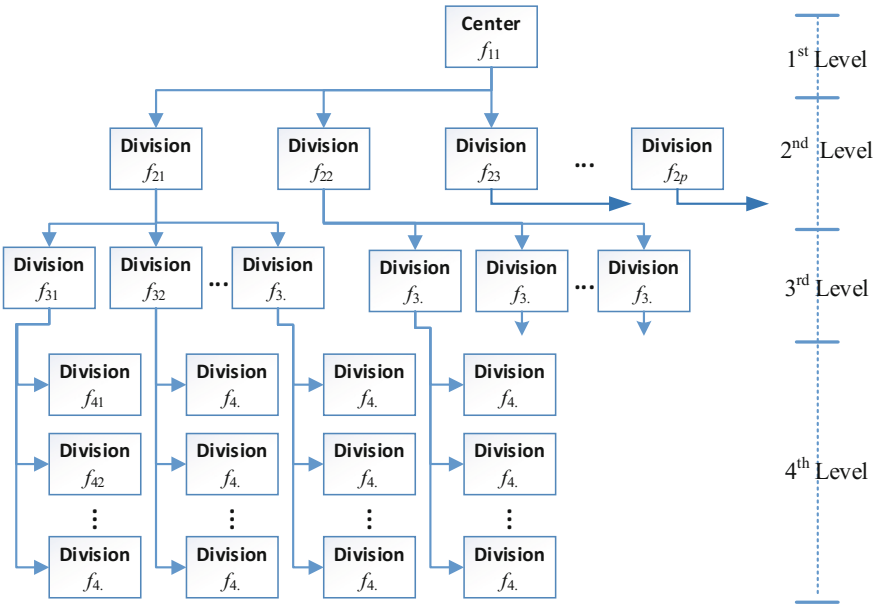

Fig. 4.2 Decentralized multi-level decision structure

Also for simplicity, the control variables, which are to be maximized, are not indicated or separated from the other variables on the right-hand sides of the maximizing equations in Eq. (4.26).

The approaches discussed in the previous two sections can be extended to solve Eq. (4.26). However, because of the complexity, it is difficult to propose a general methodology to solve this generalized equation.

Furthermore, because of the constraints at the different levels, there even is the possibility of nonexistence of a feasible solution. If the constraints can be decomposed into individual levels and divisions, then we can decompose the original structure into substructures, which are composed of $(K-1)$ single upper-level DMs, and many divisions or DMs forming the lower level in the k-th level. Then the approaches discussed in the previous sections can be applied in a straightforward manner to solve this problem.

Unfortunately, in many cases the constraints must be considered together for all the DMs in all the levels and divisions. In this case, assuming the bottom levels have feasible spaces, we can start from the bottom levels and solve the problem sequentially upward. For example, by only considering the bottom three levels, we can treat the problem as a three-level problem. First define $f_{ki}(\boldsymbol{x})$ as well as $\boldsymbol{x}_{ki}$ as the objective functions and control variable vectors of the k-th—level at the i-th division, and $\boldsymbol{c}_{kij}$ as the cost coefficient of the decision variable x_j, $j=1,\ldots,n$, for the i-th division of the k-th level. Let us assume that there is one division at the top level,

p divisions at the second level, and q divisions at the third level. Then, by extending Eqs. (4.15) and (4.22), the problem can be formulated as:

$$\max_{x1} \ f_{11}(\boldsymbol{x}) = \sum_j \boldsymbol{c}_{11j}\boldsymbol{x}_j \qquad \text{(1st level)} \tag{4.27}$$

where $\boldsymbol{x}_{21}, \boldsymbol{x}_{22}, \ldots, \boldsymbol{x}_{2p}, \boldsymbol{x}_{31}, \boldsymbol{x}_{32}, \ldots, \boldsymbol{x}_{3q}$ solve:

$$\begin{cases} \max_{x21} \ f_{21}(\boldsymbol{x}) = \sum_j \boldsymbol{c}_{21j}\boldsymbol{x}_j \\ \ldots \\ \max_{x2p} \ f_{2p}(\boldsymbol{x}) = \sum_j \boldsymbol{c}_{2pj}\boldsymbol{x}_j \end{cases} \qquad \text{(2nd level)}$$

where $\boldsymbol{x}_{31}, \boldsymbol{x}_{32}, \ldots, \boldsymbol{x}_{3q}$ solve:

$$\begin{cases} \max_{x31} \ f_{31}(\boldsymbol{x}) = \sum_j \boldsymbol{c}_{31j}\boldsymbol{x}_j \\ \ldots \\ \max_{x3q} \ f_{3q}(\boldsymbol{x}) = \sum_j \boldsymbol{c}_{3qj}\boldsymbol{x}_j \end{cases} \qquad \text{(3rd level)}$$

subject to:

$$\sum_{\mathrm{k,\,j}} \boldsymbol{A}_{kj}\boldsymbol{x}_{kj} \le \boldsymbol{b}, \ k = 1, 2, \ldots, K$$
$$\boldsymbol{x}_{kj} \ge 0, \ j = 1, 2, \ldots, n$$

where $j = 1, 2, \ldots, n$ represents the j-th decision variable. The decision variable set $\bigcup_{k,i}\{\boldsymbol{x}_{kj} | \forall i \text{ and } k\} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$,. For the second level, we have the following auxiliary problem:

$$\begin{aligned} &\max \lambda \\ &\text{subject to:} \\ &\quad \boldsymbol{x} \in \boldsymbol{X} \\ &\quad \mu_{f11}(f_{11}(\boldsymbol{x})) \ge \lambda \text{ and } \mu_{x11}(\boldsymbol{x}_{11}) \ge \lambda \\ &\quad \mu_{f2i}(f_{2i}(\boldsymbol{x})) \ge \lambda, \ i = 1, 2, \ldots, p \\ &\quad \lambda \in [0, 1] \end{aligned} \tag{4.28}$$

where $\boldsymbol{X}$ represents the original constraint of the problem. This expression is similar to Eq. (4.25). Using the satisfactory solutions of the first and the second levels in the forms of membership functions, the third-level auxiliary problem becomes:

$$\begin{aligned}
&\max \lambda \\
&\text{subject to:} \\
&\quad \boldsymbol{x} \in \boldsymbol{X} \\
&\quad \mu_{f11}(f_{11}(\boldsymbol{x})) \geq \lambda \text{ and } \mu_{\boldsymbol{x}11}(\boldsymbol{x}_{11}) \geq \lambda \\
&\quad \mu_{f21}(f_{21}(\boldsymbol{x})) \geq \lambda \text{ and } \mu_{\boldsymbol{x}21}(\boldsymbol{x}_{21}) \geq \lambda \\
&\quad \cdots \\
&\quad \mu_{f2P}(f_{2P}(\boldsymbol{x})) \geq \lambda \text{ and } \mu_{\boldsymbol{x}2P}(\boldsymbol{x}_{2P}) \geq \lambda \\
&\quad \mu_{f3I}(f_{3i}(\boldsymbol{x})) \geq \lambda,\ i = 1,\ 2,\ \ldots, q \\
&\quad \lambda \in [0,\ 1]
\end{aligned} \tag{4.29}$$

Compromise solution for the three levels with multi-followers can be obtained by solving the above equation. If the solution does not satisfy some of the DMs, an interactive process as that discussed earlier can be used.

If the constraint set or feasible domain is non-separable, DMs in the same level need to make their decisions simultaneously. Even under this condition, the problem is not necessarily reduced to a single-person decision-making problem (Bard 1983a). In fact, this problem can be treated as a multiple-objective decision-making problem. However, if parallel computer system is used, this multi-person, decision-making processes can be carried out as a one-person decision-making problem.

Although much more computation is involved, the general hierarchical structure shown in Fig. 4.2 can be solved by using the fuzzy interactive scheme represented by Eqs. (4.27)–(4.29). The process is essentially an interactive trial-and-error approach. Depending on how important is the results, various re-adjustments and re-computation can be carried out by using fuzzy membership functions.

4.5 Discussions

The proposed decision-making process proceeds from top to bottom in a natural and straightforward manner. Preference information is delivered from upper levels to lower levels sequentially, and the lower-level DM solves his or her problem under the restrictions of the upper-level DM's requirements. Since both decision variables and objective functions are considered, the proposed satisfactory solution should be more practical and reasonable. Since the solution search is based on the change of membership function instead of vertex enumeration, even a large-scale problem can be solved with reasonable amount of computation. Furthermore, for non-linear programming problem, the proposed approach is still applicable although the amount of computation can increase tremendously depending on the particular problem.

The fuzzy membership functions in our procedure are assumed to be linearly increasing or linearly decreasing. Obviously, non-linear membership function can also be used, which is important in neural network applications. Some of the non-linear membership functions are exponential or the Gaussian function.

Since different membership function and operation provide different satisfactory solution, it is important to explore various functions and operators, as well as allow the DM to change functional forms in the interactive processes. In this respect, experimental work for investigating human behavior could provide the foundation for the formulation of different membership functions and gain more understanding of the human decision-making process.

Finally, we would like to mention that input data is often imprecise or fuzzy in practice for a wide variety of hierarchical optimization problems, such as defense problems, economical analyses, government regulations, organizational management and so on. Developing methodologies and new concepts, even embedded multiple objectives in each level, for solving fuzzy, multi-level programming problems is a practical and interesting direction for future study.

Chapter 5
Aggregation of Fuzzy Systems in Multi-level Decisions

Although the various approaches proposed for fuzzy multi-level programming in the previous chapter are very useful, it is only a start in this fruitful direction. Further developments are needed to consider other aspects and thus to make the actual solution process more practical in the sense of solving more realistic problems with reduced computation. Two of the most important aspects which need to be considered are the problem of aggregation of the fuzzy representation and the representation of systems whose description is vague and not well defined. The first aspect is concerned with the basic problem of fuzzy combination, where, in the previous chapter, only the non-compensatory minimum is used. The second aspect is concerned with the handling of vague systems. Notice that the problem representing the system at each division or each level in the previous chapter is crisp, not fuzzy. The fuzziness comes from the mutual interaction or mutual competition of the different decision-makers in each level or each division. In this chapter, fuzzy systems represented by fuzzy parameters or fuzzy coefficients, not just fuzzy objectives and fuzzy decisions, will be considered.

Fuzzy information aggregation. Fuzzy number operation not only forms the basic framework of fuzzy set theory but also plays a central role in fuzzy decision making. The basic problem in fuzzy number operation is that fuzzy numbers are sets, not crisp numbers, and these sets are frequently partial ordered, rather than linear ordered. Thus fuzzy numbers can be combined or aggregated in different ways depending on the usage or the purpose of the combination. In set theoretic approach, fuzzy number operations are discussed with respect to the logical conjunction *and* and the logical disjunction *or*. Bellman and Zadeh (1970), in a seminal paper, interpreted the *and* and the *or* as the *max* and the *min* operators, respectively, in decision-making. These interpretations of the connectives *and* and *or* are generally used in the early stages of fuzzy set theory development. Bellman and Giertz (1973) and Fung and Fu (1975), among others, have examined the rational assumptions as well as the axioms of the generalized *max* and *min* operators. However, the aggregation of partial ordered fuzzy numbers is not straightforward and different emphasis or different interpretation may be needed depending on the application. Many other interpretations for the

C.-B. Cheng et al., *Fuzzy and Multi-Level Decision Making: Soft Computing Approaches*, Studies in Fuzziness and Soft Computing 368,
https://doi.org/10.1007/978-3-319-92525-7_5

connectives *and* and *or* have been proposed by various investigators. For a detailed discussion, the reader can consult the literature (Klir and Folger 1988; Zhu 1992).

As has been mentioned in the two previous chapters, the most important aspect of these interpretations is whether or not the operator is compensatory, or, in other words, whether or not trade-off between low value and high value is allowed. Since in practice, managerial decisions almost always allow some degree of compensation, compensatory operators are obviously desirable in decision making. Some of the compensatory operators are listed in Table 5.1 for problems with two operands, α and β, where the notation γ is a weighting parameter. Beside the compensatory behavior, we must also consider whether the operator is easy to use and does not introduce non-linear and other computational difficult elements. For example, Zimmermann and Zysno (1980) have shown that the Hamacher's γ-operator, which is listed as Item 3 in Table 5.1, predicts human judgment quite well, but it is a fairly complex expression, which introduces non-linear elements and causes considerable computational difficulty. Based on the γ-operator, Werners (1988) proposed the fuzzy *and* and the fuzzy *or* operators, which add a compensatory element (see Item 6 in Table 5.1) and are easy to handle. Reasonably consistent results have been obtained by using these operators. The Werners interpretation will be adopted in the following discussions.

Table 5.1 Dual pairs of common operators

Compensatory minimum operators	Compensatory maximum operators
(A.1) Algebraic product $\mu = \alpha \cdot \beta$	(B.1) Algebraic sum $\mu = \alpha + \beta - \alpha \cdot \beta$
(A.2) Bounded product $\mu = \max\{0, \alpha \cdot \beta + 1\}$	(B.2) Bounded sum $\mu = \min\{1, \alpha + \beta\}$
(A.3) Hamacher's min operator $\mu = \alpha \cdot \beta / \big[\gamma + (1-\gamma)(\alpha + \beta - 1)\big],\ \gamma \in [0, 1]$	(B.3) Hamacher's max operator $\mu = \big[(1-\gamma) \cdot \alpha \cdot \beta + \gamma(\alpha + \beta)\big]/(\gamma + \alpha \cdot \beta),\ \gamma \in [0, 1]$
(A.4) Yager's min operator $\mu = 1 - \min\Big\{1,\ \big[(1-\alpha)^q + (1-\beta)^q\big]^{1/q}\Big\},\ q \geq 1$	(B.4) Yager's max operator $\mu = \min\Big\{1,\ (\alpha^q + \beta^q)^{1/q}\Big\},\ q \geq 1$
(A.5) Dubois and Prade's min product $\mu = \alpha \cdot \beta / \max\{\alpha, \beta, \gamma\},\ \gamma \in [0, 1]$	(B.5) Dubois and Prade's max product $\mu = [(\alpha + \beta - \alpha \cdot \beta) - \min\{1-\gamma, \alpha, \beta\}\big] / \max\{\gamma, 1-\alpha, 1-\beta\},\ \gamma \in [0, 1]$
(A.6) Werners' *fuzzy and* operator $\mu = \gamma \cdot \min\{\alpha, \beta\} + (1-\gamma)(\alpha+\beta)/2,\ \gamma \in [0, 1]$	(B.6) Werners' *fuzzy or* operator $\mu = \gamma \cdot \max\{\alpha, \beta\} + (1 - \gamma, \alpha, \beta)/2,\ \gamma \in [0, 1]$
(A.7) Einstein product $\mu = \alpha \cdot \beta / [2 - (\alpha + \beta - \alpha \cdot \beta)]$	(B.7) Einstein sum $\mu = (\alpha + \beta)/(1 + \alpha \cdot \beta)$

5.1 Compensation in Bi-level Decisions

The fuzzy bi-level *programming* problem obtained in Chap. 6 and was represented by Eq. 6.13 is reproduced in the following:

$$\begin{aligned}
&\max \lambda \\
&\text{subject to :} \\
&\quad \boldsymbol{A}_1\boldsymbol{x}_1 + \boldsymbol{A}_2\boldsymbol{x}_2 \le \boldsymbol{b} \\
&\quad \left[f_1(\boldsymbol{x}) - f_1'\right] \Big/ \left[f_1^{\mathrm{L}} - f_1'\right] \ge \lambda \\
&\quad \left[(\boldsymbol{x}_1^{\mathrm{U}} + \boldsymbol{p}_1) - \boldsymbol{x}_1\right] \Big/ \boldsymbol{p}_1 \ge \lambda \\
&\quad \left[\boldsymbol{x}_1 - (\boldsymbol{x}_1^{\mathrm{U}} + \boldsymbol{p}_1)\right] \Big/ \boldsymbol{p}_1 \ge \lambda \\
&\quad \left[f_2(\boldsymbol{x}) - f_2'\right] \Big/ \left[f_2^{\mathrm{L}} - f_2'\right] \ge \lambda \\
&\quad \lambda \in [0, 1] \\
&\quad \boldsymbol{x}_1, \boldsymbol{x}_2 \ge 0
\end{aligned} \tag{5.1}$$

where $\boldsymbol{x}_1^{\mathrm{U}}$ is the preferred value of $\boldsymbol{x}_1$ and $\boldsymbol{p}_1$ is the two-side tolerance for $\boldsymbol{x}_1$ and f_1^{T} and f_1' is the upper and lower bounds of f_1, respectively.

Recall that λ was obtained by using the min operator as follows:

$$\lambda = \min(\alpha, \beta, \gamma) \tag{5.2}$$

where α, β and γ are the minimum acceptable degrees of satisfaction, respectively, for the objective function $f_1(x)$, the decision x_1, and the objective function $f_2(x)$. Instead of using the *min* operator, which is non-compensatory, we wish to use the operator proposed by Werners (1988), which is listed in Table 5.1 and is reproduced in the following:

$$\mu_{\text{and}} = \gamma \min_i(\mu_i) + (1 - \gamma) \sum_i \mu_i / m \tag{5.3a}$$

$$\mu_{\text{or}} = \gamma \max_i(\mu_i) + (1 - \gamma) \sum_i \mu_i / m \tag{5.3b}$$

where the membership functions, μ_I, $0 \le \mu_i \le 1$, $i = 1, 2, \ldots, m$, and $\gamma \in [0, 1]$ is the degree of compensation or the degree of approximation for the logical connectives *and* and *or*.

These operators are the convex combinations of the *min* or the *max* operators with the arithmetical mean. They furnish the compensatory effect without introducing too much computational requirements. Let $\lambda = \min_i(\mu_i)$, then, the connective *and* can be further simplified:

$$\mu_{\text{and}} = \gamma\lambda + (1 - \gamma)\left(\sum_i \mu_i\right)/m = \gamma\lambda + (1 - \gamma) \sum_{\mathrm{i}} ((\mu_{\mathrm{i}} - \lambda) + \lambda)/m$$

$$=\gamma\lambda + (1-\gamma)m\lambda/m + (1-\gamma)\sum_i (\mu_i - \lambda)/m$$

$$=\lambda + (1-\gamma)\sum_i (\mu_i - \lambda)/\mathrm{m}$$

where $0 \le \mu_i \le 1,\ i = 1, 2, \ldots, m, 0 \le \lambda \le 1,\ 0 \le \gamma \le 1$. Let $\lambda_i = \mu_i - \lambda$, the above expression is further reduced to:

$$\mu_{\text{and}} = \lambda + (1-\gamma)\sum_i \lambda_i/m \tag{5.4}$$

Equation 5.4 gives a compromised membership function. If we wish to obtain a crisp answer, we would like to obtain the maximum of this membership function.

Thus, we have

$$\max \mu_{\text{and}} = \lambda + (1-\gamma)\sum_i \lambda_i/m \tag{5.5}$$

Thus, a problem for the aggregation of m membership functions with given constraints $\boldsymbol{x} \in \boldsymbol{X}$ can be formulated as:

$$\max \mu_{\text{and}} = \lambda + (1-\gamma)\sum_I \lambda_i/m$$

subject to:

$$\begin{aligned} &\boldsymbol{x} \in \boldsymbol{X} \\ &\mu_i \ge \lambda + \lambda_i \\ &\lambda + \lambda_i \le 1 \\ &\gamma \in [0, 1] \\ &\lambda, \mu_i, \lambda_i \in [0, 1] \end{aligned} \tag{5.6}$$

with $i = 1, 2, \ldots, m$ and $\boldsymbol{X}$ is a crisp constraint set.

For the bi-level programming problem, we need to aggregate between the three minimum acceptable levels, α, β and γ. Thus, $\mathrm{m} = 3$. Combining Eq. 5.5 with the constraints in Eq. 5.1 for the original problem, we finally have the desired equation:

$$
\begin{aligned}
&\max \mu_{\text{and}} = \lambda + (1-\gamma)(\lambda_1 + \lambda_2 + \lambda_3)/3 \\
&\text{subject to :} \\
&\quad A_1 x_1 + A_2 x_2 \le b \\
&\quad x_1, x_2 \ge 0 \\
&\quad \mu_{f1}(f_1(\boldsymbol{x})) = \left[f_1(\boldsymbol{x}) - f_1'\right] \Big/ \left[f_1^L - f_1'\right] \ge (\lambda + \lambda_1) \\
&\quad \left[(\boldsymbol{x}_1^{\mathrm{U}} + \boldsymbol{p}_1) - \boldsymbol{x}_1\right] \Big/ \boldsymbol{p}_1 \ge (\lambda + \lambda_1) \\
&\quad \left[\boldsymbol{x}_1 - (\boldsymbol{x}_1^{\mathrm{U}} - \boldsymbol{p}_1)\right] \Big/ \boldsymbol{p}_1 \ge (\lambda + \lambda_1) \\
&\quad \mu_{f2}(f_2(\boldsymbol{x})) = \left[f_2(\boldsymbol{x}) - f_2'\right] \Big/ \left[f_2^L - f_2'\right] \ge (\lambda + \lambda_3) \\
&\quad \lambda + \lambda_i \le 1, \ i = 1, 2, 3 \\
&\quad \gamma \in [0, 1] \\
&\quad \lambda, \lambda_1, \lambda_2, \lambda_3 \in [0, 1]
\end{aligned}
\tag{5.7}
$$

where $\boldsymbol{p}_1$ is the two-sided tolerance for the decision vector x_1 and γ is the grade of compensation.

Although we specified a minimum compensation level, we could also allow a compensation level within the interval [0.1]. Since this is a linear problem, it can be solved easily. The following example resolved the problem solved in Chap. 6.

Example 5.1 Consider the problem solved in Example 6.1, where the upper and lower solutions obtained were; $\mathrm{x}^{\mathrm{U}} = (x_1^{\mathrm{U}}, x_2^{\mathrm{U}}) = (7.5, 1.5)$ for $f_1^{\mathrm{U}} = 13.5$ and $\boldsymbol{x}^{\mathrm{L}} = (x_1^{\mathrm{L}}, x_2^{\mathrm{L}}) = (3, 9)$ for $f_2^{\mathrm{L}} = 21$, which were used as reference. Now, let $f_1^{\mathrm{U}} = 13.5$ and $f_1' = 0$ instead of -3, and $f_2' = 10.5$. Further assume that the upper-level DM's control decision x_1 is around 7.5 with plus or minus tolerances 4.5 and 0.5, respectively. With these assumed values, we can establish the membership functions for μ_{x1}, μ_{f1}, and μ_{f2}. The lower-level DM-then solves the following problem, which represents an auxiliary model for the given bi-level problem in Example 6.1.

$$
\begin{aligned}
&\max \lambda \\
&\text{subject to:} \\
&\quad \boldsymbol{x} \in \boldsymbol{X} \\
&\quad \mu_{f1}(f_1(\boldsymbol{x})) = f_1/(13.5 - 0) \ge \lambda \\
&\quad \mu_{x1}(x_1) = (x_1 - 4.5)/(7.5 - 4.5) \ge \lambda \\
&\quad \mu_{x1}(x_1) = (8 - x_1)/(8 - 7.5) \ge \lambda \\
&\quad \mu_{f2}(f_2(\boldsymbol{x})) = (f_2 - 10.5)/(21 - 10.5) \ge \lambda \\
&\quad \lambda \in [0, 1]
\end{aligned}
$$

where $\boldsymbol{X}$ represents the original crisp constraint set.

Now, the corresponding compensatory formulation for the bi-level problem in Example 5.1 is:

Table 5.2 Results of Example 5.1

γ	μ_{and}	x_1	x_2
0.0	0.78	7.5	4.5
0.1	0.77	7.26	5.23
0.2	0.76	7.26	5.23
0.3	0.75	7.26	5.23
0.4	0.74	7.26	5.23
0.5	0.73	7.26	5.23
0.6	0.72	7.26	5.23
0.7	0.71	7.26	5.23
0.8	0.70	7.26	5.23
0.9	0.70	7.26	5.23
1.0	0.69	7.26	5.23

$$\max \mu_{\text{and}} = \lambda + (1-\gamma)(\lambda_1 + \lambda_2 + \lambda_3)/3$$
subject to :

$$\begin{aligned}
&\boldsymbol{x} \in \boldsymbol{X}\\
&(f_1 - 0)/(13.5 - 0) \geq (\lambda + \lambda_1)\\
&(x_1 - 4.5)/(7.5 - 4.5) \geq (\lambda + \lambda_2)\\
&(8 - x_1)/(8 - 7.5) \geq (\lambda + \lambda_2)\\
&(f_2 - 10.5)/(21 - 10.5) \geq (\lambda + \lambda_3)\\
&\lambda, \lambda_1, \lambda_2, \lambda_3 \in [0, 1]
\end{aligned}$$

where $\gamma \in [0, 1]$ represents the degree of compensation.

This compensatory formulation can be solved by any linear programming code. The compromise solution is $\boldsymbol{f}^* = (f_1^*, f_2^*) = (9.28, 17.72)$ at $\boldsymbol{x}^* = (x_1^*, x_2^*) = (7.26, 5.23)$, with a total degree of satisfaction $\mu_{\text{and}} = 0.73$ at $\gamma = 0.5$. Compared to the solution obtained in Chap. 6, where $\lambda = 0.69$ with the same objective function and decision values, the degree of satisfaction has been improved due to the use of better aggregation operator. To investigate the effect of compensation in more detail, 11 different cases with γ varies from 0 to 1.0 were solved and the results are listed in Table 5.2. The value of μ_{and} varies within a range from 0.69 to 0.78 and λ remains at 0.69, except at $\lambda = 0$, where $\lambda = 0.57$.

5.2 Compensation in Multiple-Level Problems

Based on the discussions in Chap. 6 and the use of the min operator, a three-level programming problem can be transformed into the following auxiliary form:

$$
\begin{aligned}
&\max \lambda \\
&\text{subject to :} \\
&\quad A_1x_1 + A_2x_2 + A_3x_3 \le b \\
&\quad x_1,\ x_2,\ x_3 \ge 0 \\
&\quad \mu_{f1}(f_1(\boldsymbol{x})) = \left[f_1(\mathbf{x}) - f_1'\right] \Big/ \left[f_1^L - f_1'\right] \ge \lambda \\
&\quad \left[(\boldsymbol{x}_1^{\mathrm{U}} + \boldsymbol{p}_1) - \boldsymbol{x}_1\right] \Big/ \boldsymbol{p}_1 \ge \lambda \\
&\quad \left[\boldsymbol{x}_1 - (\boldsymbol{x}_1^{\mathrm{U}} - \boldsymbol{p}_1)\right] \Big/ \boldsymbol{p}_1 \ge \lambda \\
&\quad \mu_{f2}(f_2(\boldsymbol{x})) = \left[f_2(\boldsymbol{x}) - f_2'\right] \Big/ \left[f_2^L - f_2'\right] \ge \lambda \\
&\quad \left[(\boldsymbol{x}_2^{\mathrm{U}} + \boldsymbol{p}_2) - \boldsymbol{x}_2\right] \Big/ \boldsymbol{p}_2 \ge \lambda \\
&\quad \left[\boldsymbol{x}_2 - (\boldsymbol{x}_2^{\mathrm{U}} - \boldsymbol{p}_2)\right] \Big/ \boldsymbol{p}_2 \ge \lambda \\
&\quad \mu_{f3}(f_3(\boldsymbol{x})) = \left[f_3(\boldsymbol{x}) - f_3'\right] \Big/ \left[f_3^L - f_3'\right] \ge \lambda
\end{aligned}
\tag{5.8}
$$

where $\boldsymbol{x}_1^{\mathrm{U}}$ is the preferred value of $\boldsymbol{x}_1$, $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ are the two-sided tolerances for $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, respectively, $\boldsymbol{x}_2^{\mathrm{U}}$ is the preferred value for $\boldsymbol{x}_2$, f_1^L and f_1' are the upper and lower bounds for f_1, respectively.

Replacing the min in Eq. 5.8 by the Werners operator, Eq. 5.8 can be transformed into the following auxiliary model for the tri-level problem:

$$
\begin{aligned}
&\max \mu_{\text{and}} = \lambda + (1-\gamma)(\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5)/5 \\
&\text{subject to :} \\
&\quad A_1x_1 + A_2x_2 + A_3x_3 \le b \\
&\quad x_1,\ x_2,\ x_3 \ge 0 \\
&\quad \mu_{f1}(f_1(\boldsymbol{x})) = \left[f_1(\boldsymbol{x}) - f_1'\right] \Big/ \left[f_1^L - f_1'\right] \ge \lambda + \lambda_1 \\
&\quad \left[(\boldsymbol{x}_1^{\mathrm{U}} + \boldsymbol{p}_1) - \boldsymbol{x}_1\right] \Big/ \boldsymbol{p}_1 \ge (\lambda + \lambda_2) \\
&\quad \left[\boldsymbol{x}_1 - (\boldsymbol{x}_1^{\mathrm{U}} - \boldsymbol{p}_1)\right] \Big/ \boldsymbol{p}_1 \ge (\lambda + \lambda_2) \\
&\quad \mu_{f2}(f_2(\boldsymbol{x})) = \left[f_2(\boldsymbol{x}) - f_2'\right] \Big/ \left[f_2^L - f_2'\right] \ge \lambda + \lambda_3 \\
&\quad \left[(\boldsymbol{x}_2^{\mathrm{U}} + \boldsymbol{p}_2) - \boldsymbol{x}_2\right] \Big/ \boldsymbol{p}_2 \ge (\lambda + \lambda_4) \\
&\quad \left[\boldsymbol{x}_2 - (\boldsymbol{x}_2^{\mathrm{U}} - \boldsymbol{p}_2)\right] \Big/ \boldsymbol{p}_2 \ge (\lambda + \lambda_4) \\
&\quad \mu_{f3}(f_3(\boldsymbol{x})) = \left[f_3(\boldsymbol{x}) - f_3'\right] \Big/ \left[f_3^L - f_3'\right] \ge \lambda + \lambda_5 \\
&\quad \lambda + \lambda_i \le 1,\ i = 1, 2, 3, 4, 5 \\
&\quad \lambda, \lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5 \in [0, 1]
\end{aligned}
\tag{5.9}
$$

where γ is the degree of compensation with $\gamma \in [0, 1]$.

The supervised search procedure will be defined in the same way as in Chap. 4, that is, from the first level to the second level and then to the three levels. If the solution of Eq. 5.9 does not satisfy the upper-level, modification of the membership functions will be needed with new preference information. Furthermore, it is noted that if there is a constraint, which makes the problem infeasible, this constraint should be examined carefully and may be allow some tolerance so that the original problem becomes feasible.

Example 5.2 Consider the following tri-level problem, which was solved by Anandalingdam (1988) and also in Example 4.3.

$$
\begin{aligned}
&\max_{x_1}\ f_1 = 7x_1 + 3x_2 - 4x_3\\
&\text{where } x_2 \text{ and } x_3 \text{ solves:}\\
&\quad \max_{x_2}\ f_2 = x_2\\
&\quad \text{where } x_3 \text{ solves:}\\
&\quad\quad \max_{x_3}\ f_3 = x_3\\
&\quad\quad \text{subject to:}\\
&\quad\quad\quad x_1 + x_2 + x_3 \le 3\\
&\quad\quad\quad x_1 + x_2 - x_3 \le 1\\
&\quad\quad\quad x_1 + x_2 + x_3 \ge 1\\
&\quad\quad\quad -x_1 + x_2 + x_3 \le 1\\
&\quad\quad\quad x_3 \le 0.5\\
&\quad\quad\quad x_1, x_2, x_3 \ge 0
\end{aligned}
$$

This problem was first solved for each individual decision-maker. The solution was (see also Problem 4.3): $f_1^{1\mathrm{T}} = 8.5$ at $\boldsymbol{x}_1^{1\mathrm{U}} = (1.5, 0, 0.5)$, $f_2^{1\mathrm{T}} = 1$ at $\boldsymbol{x}_2^{1\mathrm{U}} = (0, 1, 0)$ or $(0, 1, 0.5)$, and $f_3^{1\mathrm{T}} = 0.5$ at $\boldsymbol{x}_3^{1\mathrm{U}} = (1.5, 0, 0.5)$, $(0.5, 1, 0.5)$, or $(0, 0.5, 0.5)$. In addition, f_1' instead of -0.5 (only positive value is meaningful), $f_2' = f_3' = 0$. Also, assume that control decision x_1 is around 1.5 with a tolerance plus and minus 1.5. The compensatory auxiliary problem of the second level becomes:

$$
\begin{aligned}
&\max \mu_{\text{and}} = \lambda + (1-\gamma)(\lambda_1 + \lambda_2 + \lambda_3)/3\\
&\text{subject to :}\\
&\quad \boldsymbol{x} \in \boldsymbol{X}\\
&\quad 7x_1 + 3x_2 - 4x_3 \ge 8.5(\lambda + \lambda_1)\\
&\quad x_1 \ge 1.5(\lambda + \lambda_2)\\
&\quad x_1 + 1.5(\lambda + \lambda_2) \le 3\\
&\quad f_2 = x_2 \ge 0.6(\lambda + \lambda_3)\\
&\quad \lambda + \lambda_i \le 1,\ i = 1, 2, 3\\
&\quad \lambda, \lambda_1, \lambda_2, \lambda_3 \in [0, 1]
\end{aligned}
$$

where $\boldsymbol{X}$ represents the constraint set of the original problem and γ is the degree of compensation with $\gamma \in [0, 1]$.

The compromise solutions obtained for the first- and second-level problem are:

$$\boldsymbol{f}^{2\mathrm{T}} = (f_1^{2\mathrm{T}}, f_2^{2\mathrm{T}}) = (6.1, 0.6),\ \boldsymbol{x}^{2\mathrm{U}} = (x_1^{2\mathrm{U}}, x_2^{2\mathrm{U}}, x_3^{2\mathrm{U}}) = (0.9, 0.6, 0.5),\ \gamma = 0.5$$

with a degree of satisfaction $\mu_{\text{and}} = 0.62$. Assume that the top two DMs are satisfied with this solution, we then go to the. third level of this problem. Since the information has been compromised, some modifications are needed to obtain the optimum of the overall organization. Modify the intervals of decision variables and the goals as:

$x_1 = [0, 0.9, 1.8],\ x_2 = [0, 0.6, 2],\ f_1 = [0, 6.1),\ f_2 = [0, 0.6)$ and $f_3 = [0, 0.5)$.

The auxiliary problem of the third level is thus formulated as the following crisp problem:

$$\begin{aligned}
&\max \mu_{\text{and}} = \lambda + (1-\gamma)(\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5)/5 \\
&\text{subject to :} \\
&\quad \boldsymbol{x} \in \boldsymbol{X} \\
&\quad 7x_1 + 3x_2 - 4x_3 \geq 6.1(\lambda + \lambda_1) \\
&\quad x_1 \geq 0.9(\lambda + \lambda_2) \\
&\quad x_1 + 0.9(\lambda + \lambda_2) \leq 1.8 \\
&\quad f_2 = x_2 \geq 0.6(\lambda + \lambda_3) \\
&\quad x_2 + (\lambda + \lambda_4) \leq 2 \\
&\quad f_3 = x_3 \geq 0.5(\lambda + \lambda_5) \\
&\quad \lambda + \lambda_i \leq 1,\ i = 1, 2, 3, 4, 5 \\
&\quad \lambda, \lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5 \in [0, 1]
\end{aligned}$$

where $\boldsymbol{X}$ is the constraint set of the original problem and γ is the degree of compensation with $\gamma \in [0, 1]$.

After solving the third-level auxiliary problem, the compensatory solution for the whole system is:

$$\boldsymbol{f}^* = (f_1^*, f_2^*, f_3^*) = (6.1, 0.6, 0.5),\ \boldsymbol{x}^* = (x_1^*, x_2^*, x_3^*) = (0.9, 0.6, 0.5),\ \gamma = 0.5.$$

with a satisfaction level 1.0, that is, complete satisfaction for all the DMs. The crisp solution is $\boldsymbol{f}^* = (4.5, 1, 0.5)$ at $\boldsymbol{x}^* = (0.5, 1, 0.5)$ with satisfaction (1.0, 0, 1.0).

Obviously, the above procedure can be directly extended to the kth-level programming problem, with $k \geq 4$. In this case, we will need to solve $k - 1$ auxiliary problems. However, if the decision-maker is not satisfied, some interactive process will be needed to modify the degree of satisfaction

5.3 Bi-level Decentralized Problem with Equally Important Objectives

We have developed a compensatory fuzzy approach for efficiently solving multi-level mathematical programming problem. In this and the next two sections, we shall extend the procedure to hierarchy decentralized problems. These planning problems include bi-level decentralized programming problem, three-level decentralized programming problem, and, more generally, multi-level decentralized programming problem. The decentralized problem comprises of a decision center at the top level, several divisions at each of the lower levels. This is the generalized multi-level

hierarchy structure of Anandalingam's decentralized system (Anandalingam 1988). However, this generalized structure is too complex to be managed both in theory and in application. Furthermore, there is no explicit relationship among the various decision units at the same level and thus it is also difficult to investigate the problem from the standpoint of the Stackelberg game. As a result, very few investigations in this area have been carried out. As we have discussed in Chap. 4, the proposed fuzzy approach can solve these large decentralized problems efficiently. Now, we would like to examine how compensatory operators can be incorporated and thus extend the compensatory approach to this decentralized system.

Using the min operator, the auxiliary equations for the bi-level problem with s followers were obtained in Eq. 6.20, which are repeated in the following:

$$
\begin{aligned}
&\max \lambda \\
&\text{subject to :} \\
&\quad \boldsymbol{x} \in \boldsymbol{X} \\
&\quad \mu_{f1}(f_1(\boldsymbol{x})) = \left[f_1(\boldsymbol{x}) - f_1'\right] \Big/ \left[f_1^L - f_1'\right] \geq \lambda \\
&\quad \left[(\boldsymbol{x}_1^{\mathrm{T}} + \boldsymbol{p}_1) - x_1\right] \Big/ \boldsymbol{p}_1 \geq \lambda \\
&\quad \left[\boldsymbol{x}_1 - (\boldsymbol{x}_1^{\mathrm{T}} - \boldsymbol{p}_1)\right] \Big/ \boldsymbol{p}_1 \geq \lambda \\
&\quad \mu_{f2i}(f_{2i}(\boldsymbol{x})) = \left[f_{2i}(\boldsymbol{x}) - f_{2i}'\right] \Big/ \left[f_{2i}^L - f_{2i}'\right] \geq \lambda, \ i = 1, 2, \ldots, s \\
&\quad \lambda \in [0, 1]
\end{aligned}
\tag{5.10}
$$

where $\boldsymbol{X}$ represents the original constraint set, $\boldsymbol{x}_1^{\mathrm{U}}$ is the preferred value of $\boldsymbol{x}_1$ and $\boldsymbol{p}_1$ represents the two-sided tolerance of $\boldsymbol{x}_1$, and f_1^{L} and f_1' is the upper and lower bounds of f_1, respectively.

Using Werners compensatory operator, the above expression can be transformed into the following compensatory model:

$$
\begin{aligned}
&\max \mu_{\text{and}} = \lambda + (1 - \gamma)(\lambda_1 + \lambda_2 + \cdots + \lambda_{s+2})/(s + 2) \\
&\text{subject to :} \\
&\quad \boldsymbol{x} \in \boldsymbol{X} \\
&\quad \mu_{f1}(f_1(\boldsymbol{x})) = \left[f_1(\boldsymbol{x}) - f_1'\right] \Big/ \left[f_1^L - f_1'\right] \geq \lambda + \lambda_1 \\
&\quad \left[(\boldsymbol{x}_1^{\mathrm{T}} + \boldsymbol{p}_1) - \boldsymbol{x}_1\right] \Big/ \boldsymbol{p}_1 \geq (\lambda + \lambda_2) \\
&\quad \left[\boldsymbol{x}_1 - (\boldsymbol{x}_1^{\mathrm{T}} - \boldsymbol{p}_1)\right] \Big/ \boldsymbol{p}_1 \geq (\lambda + \lambda_2) \\
&\quad \mu_{f21}(f_{21}(\boldsymbol{x})) = \left[f_{21}(\boldsymbol{x}) - f_{21}'\right] \Big/ \left[f_{21}^L - f_{21}'\right] \geq (\lambda + \lambda_3) \\
&\quad \cdots \\
&\quad \mu_{f2s}(f_{2s}(\boldsymbol{x})) = \left[f_{2s}(\boldsymbol{x}) - f_{2s}'\right] \Big/ \left[f_{2s}^L - f_{2s}'\right] \geq (\lambda + \lambda_{s+2}) \\
&\quad \lambda + \lambda_i \leq 1, \ i = 1, 2, \ldots, s + 2 \\
&\quad \lambda, \lambda_1, \lambda_2, \ldots, \lambda_{s+2} \in [0, 1]
\end{aligned}
\tag{5.11}
$$

with $i = 1, 2, \ldots, s$ and γ is the degree of compensation.

Example 5.3 Consider the bi-level problem solved in Example 4.2, which was also solved by Anandalingdam (1988).

The individual solutions of the DMs in the top and bottom levels are: $f_{11}^{\mathrm{T}} = 35$ at $\boldsymbol{x}_{11}^{\mathrm{U}} = (x_1^{\mathrm{U}}, y_1^{\mathrm{U}}, y_2^{\mathrm{U}}, y_3^{\mathrm{U}}) = (10, 0, 10, 0)$ or $(5, 5, 10, 3)$, $f_{21}^{\mathrm{T}} = 30$ at $\boldsymbol{x}_{21}^{\mathrm{U}} = (0, 10, 0, 0)$, $f_{22}^{\mathrm{T}} = 30$ at $\boldsymbol{x}_{22}^{\mathrm{U}} = (0, 0, 10, 0)$, and $f_{23}^{\mathrm{T}} = 30$ at $\boldsymbol{x}_{23}^{\mathrm{U}} = (0, 0, 0, 10)$. Since negative values are not allowed, let $f'_{11} = f'_{21} = f'_{22} = f'_{23} = 0$. Assume that for x_1 the upper-level DM is satisfied for any value between 5 and 10, with negative side tolerance of 5 and positive tolerance of 0. Based on these assumptions, the membership functions can be established. Using Eq. 5.11, we obtain:

$$\max \mu_{\text{and}} = \lambda + (1-\gamma)(\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5)/5$$

subject to :

$$\begin{aligned}
&\boldsymbol{x} \in \boldsymbol{X}\\
&\mu_{f11}(f_{11}(\boldsymbol{x}) = \left[f_{11}(\boldsymbol{x}) - f'_{11}\right] \Big/ \left[f_{11}^{L} - f'_{11}\right] = (x_1 + y_1 + 2y_2 + y_3)/35 \geq \lambda + \lambda_1\\
&\left[x_1 - (x_1^{\mathrm{U}} - p_1)\right] \Big/ p_1 = x_1/5 \geq (\lambda + \lambda_2)\\
&x_1 \leq 10\\
&\mu_{f21}(f_{21}(\boldsymbol{x})) = \left[f_{21}(\boldsymbol{x}) - f'_{21}\right] \Big/ \left[f_{21}^{L} - f'_{21}\right]\\
&\qquad = (-x_1 + 3y_1 - 2y_2 - y_3)/30 \geq (\lambda + \lambda_3)\\
&\mu_{f22}(f_{22}(\boldsymbol{x})) = \left[f_{22}(\boldsymbol{x}) - f'_{22}\right] \Big/ \left[f_{22}^{L} - f'_{22}\right]\\
&\qquad = (-x_1 - y_1 + 3y_2 - y_3)/30 \geq (\lambda + \lambda_4)\\
&\mu_{f23}(f_{23}(\boldsymbol{x})) = \left[f_{23}(\boldsymbol{x}) - f'_{23}\right] \Big/ \left[f_{23}^{L} - f'_{23}\right]\\
&\qquad = (-x_1 - y_1 - 3y_2 + 3y_3)/30 \geq (\lambda + \lambda_5)\\
&\lambda + \lambda_i \leq 1,\ i = 1, 2, 3, 4, 5\\
&\lambda \in [0, 1]
\end{aligned}$$

Solving the above compensatory model, we obtained a satisfactory solution: $\boldsymbol{f}^* = (f_{11}^*, f_{21}^*, f_{22}^*, f_{23}^*) = (31.07, 6.43, 6.43, 6.43)$ with satisfaction $\mu_{\text{and}} = 0.28$ at x*= (1.07, 7.5, 7.5, 7.5) with $\gamma = 0.5$.

To investigate the influence of the compensatory operator, 11 cases were solved. The value of the parameter γ varies from 0 to 1. The results are listed in Table 5.3. From this table, we can see that the non-compensatory solution, or $\gamma = 1$, gives a value of $\lambda = 0.21$, which is very low. Thus, compensation improves the total degree of satisfaction.

5.4 Bi-level Decentralized Problem with Unequally Important Objectives

Consider a bi-level problem with s decision units at the second level and assign the weight of each decision unit according to the degree of importance of the unit. Let these assigned weights be represented by $w_{21}, w_{22}, \ldots, w_{2s}$ where the second subscript represents the decision unit in the second level and, in order to be unbiased, the summation of all the weights must be equal to one, or, $w_{21} + w_{22} + \cdots + w_{2s} = 1$. Using these weights in Eq. 5.11, we obtain the following expression:

Table 5.3 Results of Example 5.3

γ	μ_{and}	λ	x_1	y_1	y_2	y_3	f_{11}	f_{21}	f_{22}	f_{23}
0.0	0.41	0.0	3.75	6.25	8.75	6.25	33.75	0	10	0
0.1	0.37	0.0	3.75	6.25	8.75	6.25	33.75	0	10	0
0.2	0.34	0.17	2.5	7.5	7.5	7.5	32.5	5	5	5
0.3	0.32	0.17	2.5	7.5	7.5	7.5	32.5	5	5	5
0.4	0.30	0.17	2.5	7.5	7.5	7.5	32.5	5	5	5
0.5	0.28	0.21	1.07	7.5	7.5	7.5	31.07	6.43	6.43	6.43
0.6	0.27	0.21	1.07	7.5	7.5	7.5	31.07	6.43	6.43	6.43
0.7	0.25	0.21	1.07	7.5	7.5	7.5	31.07	6.43	6.43	6.43
0.8	0.24	0.21	1.07	7.5	7.5	7.5	31.07	6.43	6.43	6.43
0.9	0.23	0.21	1.07	7.5	7.5	7.5	31.07	6.43	6.43	6.43
1.0	0.21	0.21	1.07	7.5	7.5	7.5	31.07	6.43	6.43	6.43

$$\max\,(\mu_{\text{and}})_2 = \lambda + (1-\gamma)[\lambda_1 + \lambda_2 + s(w_{21}\lambda_3 + w_{22}\lambda_4 + \cdots + w_{2s}\lambda_{s+2})/(s+2)$$

subject to :

$$\begin{aligned} &\boldsymbol{x} \in \boldsymbol{Y} \\ &w_{21} + w_{22} + \cdots + w_{2s} = 1 \\ &0 \le w_{2\text{Ii}} \le 1 \end{aligned} \tag{5.12}$$

where $\boldsymbol{Y}$ represents the original constraint set of Eq. 5.11.

If we set the weights $w_{21} = w_{22} = \cdots = w_{2s} = 1/(s+2)$, this general system reduces to Eq. 5.11. One of the problems in assigning weight is that the assigned weights must be consistent. Various techniques, such as the simple additive weighting method, eigenvector method, weighted least square method, and entropy method, have been developed for assigning weight. For a detailed discussion, the reader can consult the various books on multiple attribute decision making (Hwang and Yoon 1981). Thus, after the importance of the objectives is ranked or ordered, we can use any of the above mentioned approaches to assign the weights.

Example 5.4 Consider the same problem solved in Example 5.3. Except, now, that the upper-level DM assigns different degrees of importance for the different divisions in the second level. The upper-level DM has decided that objective f_{21} is three times more important than either the second objective f_{22} or the third objective f_{23} and that the second objective is twice as important as the third objective. A matrix is usually used to represent these paired comparisons of importance and the relative weights can be obtained by using one of the methods listed above. In the present example, Saaty's eigenvector method (Saaty 1990) will be adopted. The matrix based on assigned importance can be formed as:

$$A = [a_{ij}] = \begin{bmatrix} \frac{w_{21}}{w_{21}} & \frac{w_{21}}{w_{22}} & \frac{w_{21}}{w_{23}} \\ \frac{w_{22}}{w_{21}} & \frac{w_{22}}{w_{22}} & \frac{w_{22}}{w_{23}} \\ \frac{w_{23}}{w_{21}} & \frac{w_{23}}{w_{22}} & \frac{w_{23}}{w_{23}} \end{bmatrix} = \begin{bmatrix} \frac{1}{1} & \frac{3}{1} & \frac{3}{1} \\ \frac{1}{3} & \frac{1}{1} & \frac{2}{1} \\ \frac{1}{3} & \frac{1}{2} & \frac{1}{1} \end{bmatrix} \tag{5.13}$$

where a_{ij} represents the number of times greater is the importance of objective i than that of objective j.

The relative weights of the different objectives can be obtained by solving the matrix equation

$$(\boldsymbol{A} - \lambda \mathbf{I})w = \mathbf{0} \tag{5.14}$$

where $\boldsymbol{A}$ is the 3×3 matrix defined in Eq. 5.13 and λ, is the eigenvalue of the matrix. The solution of Eq. 5.14 results in three λ values. Substituting the maximum value of these three solutions into the matrix equation, we obtain three linear algebraic equations. The weights, w_{21}, w_{22}, and w_{23} can be obtained by solving this system of linear equations.

For the current problem, the weights obtained are: $(w_{21}, w_{22}, w_{23}) = (0.59, 0.25, 0.16)$. Substituting these weights into Eq. 7.12, we have:

$$\max\,(\mu_{\text{and}})_2 = \lambda + (1-\gamma)[\lambda_1 + \lambda_2 + 3(0.59\lambda_3 + 0.25\lambda_4 + 0.16\lambda_5) + /5$$

subject to :

$$\begin{aligned}
&\boldsymbol{x} \in \boldsymbol{Z}\\
&\mu_{f11}(f_{11}(\boldsymbol{x})) = \left[f_{11}(\boldsymbol{x}) - f'_{11}\right] \Big/ \left[f^L_{11} - f'_{11}\right] = (x_1 + y_1 + 2y_2 + y_3)/35 \geq \lambda + \lambda_1\\
&\left[x_1 - (x_1^{\text{U}} - p_1)\right] \Big/ p_1 = x_1/5 \geq (\lambda + \lambda_2)\\
&x_1 \leq 10\\
&\mu_{f21}(f_{21}(\boldsymbol{x})) = \left[f_{21}(\boldsymbol{x}) - f'_{21}\right] \Big/ \left[f^L_{21} - f'_{21}\right]\\
&\qquad = (-x_1 + 3y_1 - 2y_2 - y_3)/30 \geq (\lambda + \lambda_3)\\
&\mu_{f22}(f_{22}(\boldsymbol{x})) = \left[f_{22}(\boldsymbol{x}) - f'_{22}\right] \Big/ \left[f^L_{22} - f'_{22}\right]\\
&\qquad = (-x_1 - y_1 + 3y_2 - y_3)/30 \geq (\lambda + \lambda_4)\\
&\mu_{f23}(f_{23}(\boldsymbol{x})) = \left[f_{23}(\boldsymbol{x}) - f'_{23}\right] \Big/ \left[f^L_{23} - f'_{23}\right]\\
&\qquad = (-x_1 - y_1 - 3y_2 + 3y_3)/30 \geq (\lambda + \lambda_5)\\
&\lambda + \lambda_i \leq 1,\ i = 1, 2, 3, 4, 5\\
&w_{21} + w_{22} + \cdots + w_{2s} = 1\\
&\lambda \in [0, 1]
\end{aligned}$$

where $\boldsymbol{X}$ represents the original constraint set, f^L_{11} and f'_{11} are the upper and lower bounds of f_{11}, and γ is the grade of compensation.

The satisfactory solution obtained is:

$$\begin{aligned}
&\boldsymbol{f}^* = (f^*_{11}, f^*_{21}, f^*_{22}, f^*_{23}) = (31.73, 8.85, 5.77, 5.77),\ \mu^+_{\text{and}} = 0.28,\\
&\boldsymbol{x}^* = (0.96, 8.27, 7.5, 7.5),\ \gamma = 0.5.
\end{aligned}$$

5.5 Multiple-Level Decentralized Problem

The same procedure can be applied to multi-level problem with multiple followers. To simplify the discussion, let us consider a problem with three levels. Assume that there are s decision units in the second level and q decision units in the third level, then the objective function of the third-level auxiliary problem with equally important objectives can be illustrated as:

$$\begin{aligned}
(\mu_{\text{and}})_3 = \lambda + (1-\gamma)[&\lambda_1 + \lambda_2 + (\lambda_3 + \cdots + \lambda_{s+2}) + (\lambda_{s+3} + \cdots + \lambda_{2s+2})\\
&+ (\lambda_{2s+3} + \cdots + \lambda_{2s+q+2})/(2s + q + 2)
\end{aligned} \tag{5.15}$$

where $(\lambda_3 + \cdots + \lambda_{s+2})$ represent the goals of the second levels, $(\lambda_{s+3} + \cdots + \lambda_{2s+2})$ represent the decisions of the second level, and $(\lambda_{2s+3} + \cdots + \lambda_{2s+q+2})$ represent the goals of the third level.

For unequally important objectives, assuming the assigned weights are: $w_{21}, w_{22}, \ldots, w_{2s}$ where $w_{21} + w_{22} + \cdots + w_{2s} = 1$, for the second level, and $w_{31}, w_{32}, \ldots, w_{3q}$, where $w_{31} + w_{32} + \cdots + w_{3q} = 1$, for the third level. The auxiliary model of the objective function for the third level can be represented as:

$$\begin{aligned}(\mu_{\text{and}})_4 = \lambda + (1-\gamma)[&\lambda_1 + \lambda_2 + s(w_{21}\lambda_3 + \cdots + w_{2s}\lambda_{s+2})/2 \\ &+ s(w_{21}\lambda_{s+3} + \cdots + w_{2s}\lambda_{2s+2})/2 \\ &+ q(w_{31}\lambda_{2s+3} + \cdots + w_{3q}\lambda_{2s+q+2})]/(2s+q+2)\end{aligned} \tag{5.16}$$

where $(\lambda_3 + \cdots + \lambda_{2s+2})$ represent the goals of the second level, $(\lambda_{s+3} + \cdots + \lambda_{2s+2})$ represent the decisions of the second level, and $(\lambda_{2s+3} + \cdots + \lambda_{2s+q+2})$ represent the goals of the third level.

It is readily to see that the proposed approach can be extended easily to problem with k levels.

5.6 Fuzzy Multi-level Problem

The multi-level problems considered in the previous chapters are really crisp problems and the problem itself is not fuzzy. The fuzziness comes from the mutual interactions between the various levels or between the divisions in the same level. In this section, we wish to consider fuzzy multi-level problems, where the coefficients are fuzzy or not well defined.

As discussed in Chap. 3, many different approaches have been proposed to solve programming problems with fuzzy coefficients. In principle, any of these approaches can be incorporated into the algorithms discussed for multi-level decision making in the previous chapters. However, because of space limitations, we shall only consider the approach discussed in Sect. 3.3.2 and combine it with the compensatory algorithm discussed early in this chapter. This is best illustrated by an example.

Example 5.5 Consider two objectives as in two different levels. Furthermore, we shall assume that DM in level one controls x_1 and DM in level two controls x_2. Thus, we have:

$\max_{x_1} f_1 = (1.5, 2, 2, 2.5)x_1 + (-1.5, -1, -1, -0.5)x_2$ (upper level)

where x_2 solves:

$\max_{x_2} f_2 = (0.8, 1, 1, 1.4)x_1 + (1.5, 2, 1.5, 2.5)x_2$ (lower level)

subject to :

$$(2.5, 3, 3, 3.5)x_1 + (-5.5, -5, -5, -4.5)x_2 \leq (10, 25, 25, 26)$$
$$(2.8, 3, 3.2, 3.5)x_1 + (-1.5, -1, -1, -0.5)x_2 \leq (15, 20, 25, 35)$$
$$(2.75, 3, 3.2, 3.2)x_1 + (0.8, 1, 1, 1.3)x_2 \leq (20, 25, 25, 42)$$
$$(2.5, 3, 3, 3.5)x_1 + (3.6, 4, 4, 4)x_2 \leq (30, 32, 35, 50)$$
$$(0.8, 1, 1, 1.2)x_1 + (2.6, 3, 3, 3.4)x_2 \leq (14, 15, 16, 25)$$
$$x_1, x_2 \geq 0$$

The auxiliary problem is formulated in the following:

$$\min \left\{\theta_2^1, \theta_2^2, \delta_1, \delta_2, \delta_3, \delta_4, \delta_5\right\}$$
subject to :

$$\theta_{12} = (2.5x_1 - 0.5x_2 - f_1)/(0.5x_1 + 0.5x_2) \geq \alpha$$
$$2x_1 - x_2 \leq f_1 \leq 2.5x_1 - 0.5x_2$$
$$\theta_{22} = (1.4x_1 - 2.5x_2 - f_2)/(0.4 - x_1) \geq \alpha$$
$$x_1 + 2.5x_2 \leq f_2 \leq 1.4x_1 - 2.5x_2$$
$$\delta_1 = (26 - 2.5x_1 + 5.5x_2)/(11 + 0.5x_1 + 0.5x_2) \geq \alpha$$
$$15 \leq 3x_1 - 5x_2$$
$$26 \geq 2.5x_1 - 5.5x_2$$
$$\delta_2 = (35 - 2.8x_1 + 1.5x_2)/(10 + 0.2x_1 + 0.5x_2) \geq \alpha$$
$$25 \leq 3x_1 - x_2$$
$$35 \geq 2.8x_1 - 1.5x_2$$
$$\delta_3 = (42 - 2.75x_1 + 0.8x_2)/(17 + 0.25x_1 + 0.2x_2) \geq \alpha$$
$$25 \leq 3x_1 + x_2$$
$$42 \geq 2.75x_1 + 0.8x_2$$
$$\delta_4 = (50 - 2.5x_1 + 3.6x_2)/(15 + 0.5x_1 + 0.4x_2) \geq \alpha$$
$$35 \leq 3x_1 + 4x_2$$
$$50 \geq 2.5x_1 + 3.6x_2$$
$$\delta_4 = (25 - 0.8x_1 - 2.6x_2)/(9 + 0.2x_1 + 0.4x_2) \geq \alpha$$
$$16 \leq x_1 + 3x_2$$
$$25 \geq 0.8x_1 + 2.6x_2$$
$$\theta_{12}, \theta_{22}, \delta_1, \delta_2, \delta_3, \delta_4, \delta_5 \in [0, 1]$$
$$x_1, x_2 \geq 0$$

In order to solve the above two-level problem, let us assume that the fuzzy ranges for the two objectives are:

$$f_1 \in [0, 23.47], \quad f_2 \in [0, 21.12]$$

We shall further assume that the first-level decision x_1 is around 11.6 with maximum uncertainty on the plus side of 2.06 and maximum uncertainty on the minus side of 1.94. Now, following the compensatory approach, the auxiliary problem can be obtained as:

$$\max \mu_{\text{and}} = \lambda + (1-\gamma)(\lambda_1 + \lambda_2 + \lambda_3)/3$$
subject to :

$$\begin{aligned}
&\boldsymbol{x} \in \boldsymbol{Z}\\
&(f_1 - 0)/(23.47 - 0) \geq \lambda + \lambda_1\\
&(x_1 - 9)/(11.6 - 9) \geq \lambda + \lambda_2\\
&(13 - x_1)/(13 - 11.6) \geq \lambda + \lambda_2\\
&(f_2 - 0)/(21.12 - 0) \geq \lambda + \lambda_3\\
&\gamma, \lambda, \lambda_1, \lambda_2, \lambda_3 \in [0, 1]
\end{aligned}$$

where $\boldsymbol{x} \in \boldsymbol{Z}$ represents the set of original constraint, and γ is the degree of compensation.

This problem can be solved easily and the solution is:

$$\begin{aligned}
&\boldsymbol{f}^* = (f_1^*, f_2^*) = (22.62, 19.24)\\
&\boldsymbol{x}^* = (x_1^*, x_2^*) = (10.88, 2.48)\\
&\mu_{\text{and}} = 0.920,\ \gamma = 0.5\\
&\alpha = 0.5
\end{aligned}$$

5.7 Discussions

The mutually interactive decision concept is a powerful approach. However, only a beginning has been accomplished. Many more detailed investigations are needed. Two of the most crucial issues are how to compromise or aggregate between the different DMs and how to represent the information of the different levels if this information is vague, not well defined, or even conflicting. The first problem is handled by the use of different aggregation operators and the second one used the possibility concept. Only the Werners aggregation operator is used. Obviously, many other existing aggregation operators should also be investigated.

Even within the Werners operator, further investigation is needed. For example, the value of γ, which controls the size of the compensatory part, needs more detailed study.

In order to reduce the problem to linear form, we have assumed that the DMcan furnish the desired possibility level. If the possibility is assumed unknown, then we must solve a non-linear problem. One way to solve this non-linear problem is by the use of exhaustive search for all the possibility levels. In other words, first, assume all possible values of possibility and, then, form different linear problems with each assumed possibility value. In this way, we are essentially solving a series of linear problems.

Chapter 6
Multi-level Optimization by Fuzzy Dynamic Programming

Traditional MLP concerns decentralized planning problems with multiple DMs in a multi-level or hierarchical organization where decisions interact with one another (Wen and Hsu 1991). Motivated by the concept of dynamic programming (DP) and the operation of fuzzy dynamic programming (Bellman 1957; Kacprzyk 1983), this chapter extends the decentralized planning problems to a temporal fashion where the hierarchical decisions can be made stage by stage.

Kacprzyk (1983) provided a general view of multi-stage decision-making under fuzziness and derived a general structure for solving fuzzy DP problems. The max-min operation for fuzzy decision making proposed by Bellman and Zadeh (1970) is applied to the decisions in the dynamic environment. Kacprzyk and Esogbue (1996) provided a survey of major development and applications of fuzzy DP.

When the hierarchical control feature is ignored, MLP is reduced to a multi-objective decision-making (MODM) problem. For solving the multi-objective knapsack problem (MOKP), approaches have been proposed to find efficient solutions under the DP structure. Cho and Kim (1997) developed an interactive hybrid method to adjust DM preference information through a scaling constant among objectives. Klamroth and Wiecek (2000) proposed DP-based approaches to obtain all non-dominant solutions.

The mathematical structure shared by MOKP and MLKP allows the two problems to be embedded in a common DP structure. For convenience, a fuzzy MODP is presented first in this chapter and then the approach is extended to MLP. A multi-objective knapsack problem (MOKP) and a multi-level knapsack (MLKP) problem are used to demonstrate the fuzzy dynamic programming approach.

Part of the content of this chapter is reproduced from our previous work (Shih 2005).

C.-B. Cheng et al., *Fuzzy and Multi-Level Decision Making: Soft Computing Approaches*, Studies in Fuzziness and Soft Computing 368,
https://doi.org/10.1007/978-3-319-92525-7_6

6.1 Fuzzy Multi-objective Dynamic Programming

This section discusses the multi-objective knapsack problem and then presents the fuzzy dynamic programming for solving the problem.

6.1.1 Multi-objective Knapsack Problems

A knapsack problem (KP) is defined whereby a hiker must decide which goods to include in her/his knapsack on a forthcoming trip. She/he must choose a set of objects from N objects, given each object i, $i = 1, 2, \ldots, N$, with weight w_i and utility c_i to the hiker. The goal of the hiker is to maximize the utility of her/his trip subject to the overall weight limitation W (Ahuja et al. 1993). This problem has been one of the most extensively studied and has been applied to many areas, such as, capital budgeting, cargo loading, cutting stock, flyaway kit, and project selection (Martello and Toth 1990; Salkin and De Kluyver 1975).

In the past decades, there has been an increasing concentration on the need to identify and consider several simultaneous objectives, usually conflicting, in the analysis and solution of some practical problems (Goicoechea et al. 1982). In such a case, the formulation of the multi-objective knapsack problem (MOKP) can be expressed as (Visee et al. 1998):

$$\begin{aligned} &\text{Max } z_k = \sum_{i=1}^{N} c_i^k x_i, k = 1, \ldots, K \\ &\text{s.t.} \\ &\sum_{i=1}^{N} w_i x_i \le W \quad \text{(capacity constraint)} \\ &x_i \text{ is integer, } i = 1, 2, \ldots, N. \end{aligned} \tag{6.1}$$

where x_i is the number of object i to be selected. The above expression is referred as the unbounded knapsack problem with one knapsack. When each object can be either selected by one unit or not selected, i.e. $x_i \in \{0, 1\}$, the problem is called a 0–1 KP (Martello and Toth 1990). Apparently, the MOKP is a generalization of the classic KP (Erlebach et al. 2002).

For solving crisp MOKPs, popular solution techniques are dynamic programming and integer programming (Salkin and De Kluyver 1975). Cho and Kim (1997) developed an improved interactive hybrid method to adjust DM preference information through a scaling constant among objectives. Klamroth and Wiecek (2000) proposed DP-based approaches to obtain all the non-dominated solutions. As for integer programming-based approaches, a representative example is the work of Salman et al. (2002).

6.1.2 Fuzzy Multi-objective Dynamic Programming

The knapsack problem paraphrases a general resource allocation model in which a single resource is assigned to a number of alternatives with the objective of maximizing total return (Martello and Toth 1990). Although many techniques could meet the requirement, this problem is ideally suited to the DP approach (Bellman and Lee 1978). Assume there is an N-stage process from stage 1 to stage N, and the total amount of capacity W is given in the beginning. Let s^j denote the j-th stage state variable and indicate the amount of capacity remaining at stage j, and thus the initial state s^0 represents the overall capacity W to be allocated in later stages. Each stage consumes a certain amount of capacity, and the consumption at stage j is readily known as $(s^{j-1} - s^j)$, $j = 1, 2, \ldots, N$. The consumed capacity contributes to each stage in the system which is characterized by a return function (utility) R^l. Let W^j be the consumed capacity before stage j, and thus, $0 \le s^j \le W - W^j$.

The different policy control (variables) at each stage forms a feasible space in the DP structure. Following the computational procedure, the consumed capacity at any stage yields a contribution to its objective, and this contribution will be accumulated to the total return at the end. Let $g^j\left(s^{j-1}\right)$ be the maximal accumulated return obtained from stages j through N, given the available capacity s^{j-1} remaining from stage $j-1$. The DP formulation of KP is presented as follows:

$$
\begin{aligned}
& g^j\left(s^{j-1}\right) = \max_{x^j \in F}\left[R^j + g^{j+1}\left(s^j\right)\right], \text{ and } g^N\left(s^{N-1}\right) = \max_{x^N \in F}[R^N] \\
& \text{s.t.} \\
& \quad x^j \in F \quad \text{(feasible region of policy control variables } x^j\text{)}, \\
& \quad s^j \in S \quad \text{(capacity availability constraint)}, \\
& \quad s^0 = W \quad \text{(initial resource capacity)}, \\
& \quad s^j \text{ are positive integers, } \; j = 1, 2, \ldots, N,
\end{aligned}
\tag{6.2}
$$

where F is a feasible region of all possible policy control variables at all stages, and S is a resource availability set including budget, capacity, and other constraints throughout the stages. The objective function $g^j\left(s^{j-1}\right)$ is a recursive expression at each stage, except the last stage N.

When there is more than one type of resource, multiple state variables are required to describe the resource utilizations. However, for simplicity, people often use a weighted summation of all resources and treat it as a single one. When the resource can be divided in only integral shares and the number of stages is finite, the problem is categorized as a discrete (time) or a finite-stage deterministic DP problem (Chen et al. 2003; Taha 2016).

In a DP structure, the state variable equation implicitly transfers a policy control variable and an input state to an output state at each stage, where the constraint is imposed on the policy control variable and the goal is imposed on the output state. The fuzzification of DP is carried out by considering constraint satisfaction and goal

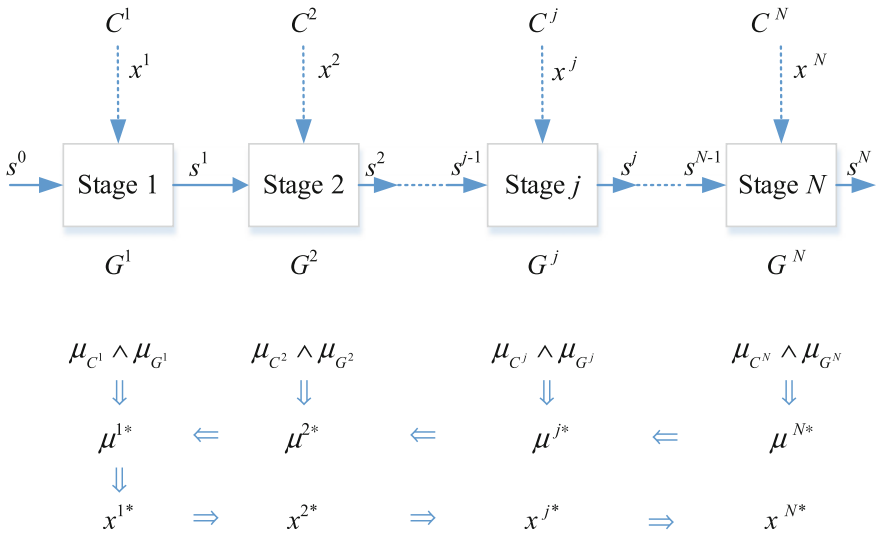

Fig. 6.1 Structure of fuzzy dynamic programming

achievement as fuzzy measures. For a specific stage j, an input state s^{j-1} is applied to a policy control variable x^j, which is subject to a fuzzy constraint $\mu_{C^j}(x^j)$, where C^j is the constraint to stage j and $\mu_{C^j}(\cdot)$ is the membership function defined on the satisfaction of C^j; the fuzzy goal $\mu_{G^j}(s^j)$ is established based on the output state s^j, where $\mu_{G^j}(\cdot)$ is the membership value of the goal achievement and is defined through some known cause/effect relations. The fuzzy DP structure is illustrated by Fig. 6.1. The fuzzy decision becomes an aggregation of the fuzzy constraints and fuzzy goals at a particular stage, but the decision will be also affected by the decisions from other stages. The optimality of fuzzy multi-stage decision making expresses how well the subsequent fuzzy constraints and fuzzy goals are satisfied by the policy controls and states.

Mathematically, the fuzzy decision in the dynamic environment is defined as (Kacprzyk 1983):

$$\mu(x^1, \ldots, x^{(N-1)}|s^0) = \mu_{C^1}(x^1) \ldots \otimes \mu_{C^N}(x^N) \otimes \mu_{G^N}(s^N) \tag{6.3}$$

where $\otimes$ is a fuzzy conjunction operator.

The problem is to find an optimal sequence of control variables $k^1*,\ldots, k^{N-1}*$, such that:

$$\mu(x^{1*}, \ldots, x^{(N-1)*}|s^0) = \max_{x^1,\ldots,x^N \in F} \left[\mu_{C^1}(x^1) \ldots \otimes \mu_{C^N}(x^N) \otimes \mu_{G^N}(s^N)\right]$$

In Fig. 6.1, the solid lines indicate the stage transitions starting from stage 1 to stage N; state variable s^0 is the initial state and the input of stage 1 as well, state

variable s^1 is the output of stage 1 and the input of stage 2, and the rest [of the??] states are defined in the same manner; policy control variable x^1 represents the possible alternatives at stage 1 and likewise for the rest of the state variables; the influence of the policy control variable on each stage is indicated by the dotted line; C^j and G^j are the constraint and goal, respectively, at stage j; the constraint is imposed on the policy control and the goal is measured based on the output state at each stage. The decision is the confluence of constraints and goals at each stage and is accumulated backward, and thus, the overall maximal satisfaction μ^{j*} represents the current accumulated degree of satisfaction from stage N backward to stage j.

A group of recursive equations can be obtained for the backward iteration at a specific stage:

$$\mu_{G^j}\left(s^{j-1}\right) = \max_{k^j \in F}[\mu_{C^j}\left(x^j\right) \otimes \mu_{G^{j+1}}\left(s^j\right) \text{ and } s^j = g\left(s^{j-1}, x^j\right), \quad j = 1, \ldots, N, \tag{6.4}$$

where $\mu_{G^j}\left(s^{j-1}\right)$ is the fuzzy goal at stage j induced by the fuzzy goal at stage $j - 1$. Though the aggregation of fuzzy constraints and fuzzy objective functions is carried out by a max–min operation (Bellman and Zadeh 1970) here, there are other alternative fuzzy operators, e.g. compensatory operator (Shih and Lee 2001) and product operator (Lai and Li 1999). The measuring function of the satisfactions of goals and constraints by DMs can be constructed based on the concept of ideal point discussed by Salukvadze (1982) and Li and Haimes (1989), where the concept of the positive ideal solution (PIS) and negative ideal solution (NIS) (Hwang and Yoon 1981) is used to define the optimistic expectation and the pessimistic expectation, respectively, of the DM. The satisfactory measures by DMs regarding the goals and constraints are construct based on the concept of positive ideal solution (PIS) and negative ideal solution (NIS) and are realized via fuzzy membership functions.

The embedded MODM problem in the DP structure at each stage is to seek a maximum degree of satisfaction through fuzzy operation under each control. Later, these temporal results are accumulated in the structure of the multi-stage decision-making process under fuzziness (Esogbue and Bellman 1984). The process to solve the embedded MODM problem is analogous to the concept of the global criterion in MODM (Hwang and Masud 1979), except the decision space of the former is discrete. Since the exact amount of budget at any specific stage is unknown, the backward procedure of DP can assist in enumerating all temporary optimal cases under available budget at each intermediate stage. When the backward procedure reaches the first stage, the optimal global evaluation is obtained; and then the optimal policy controls are picked out by tracing forward from stage 1 to stage N.

Based on the fuzzy DP structure described above, the MOKP is formulated as follows. There are W units of resource to be distributed among N activities to maximize K objectives, and $f_{jm}(x), j = 1, 2, \ldots, N, m = 1, 2, \ldots, K$, denoting the return realized by the m-th objective from an allocation of x units ($x = 0, 1, 2, \ldots, W$) to the j-th activity. The decision-making allocates the total capacity (W) to the N activities, such that the returns of K objectives from the activities are maximized (Lai and Li 1999). Shih (2005) proposed a fuzzy DP solution procedure for solving MOKPs based on

backward recursion. The procedure begins with establishing the payoff tables for all objectives, constructing the fuzzy membership functions for measuring satisfactory degrees of objectives at all stages based on the concept of PIS and NIS. Sequential forward and backward tracking of decisions along the stages is then conducted to find the best solution.

6.2 Fuzzy Multi-level Dynamic Programming

If we consider the DMs in the MLP make decisions in sequence or, equivalently, the decisions are made over time, the problem of MLP can be treated as an imbedded MLP problem in a DP structure, i.e. a multi-level dynamic programming (MLDP), where the resources are allocated over time. In the literature, relevant approaches mainly concentrated on discrete (time) resource-allocation processes for easy comprehension; in such a case, the term "stage" replaces the term "time" for a clear definition.

For resource allocation based on fuzzy MODM, Esogbue and Bellman (1984) presented a fuzzy mathematical model of the allocation process by decomposing the system into three levels and processing these hierarchical levels by the concept of stages of DP. Unlike MODP, MLDP treats the stage as a time horizon rather than simply decision-making levels. MLDP is suitable for long range planning, such as production planning, social-economic policy development, and resource allocation in planning, programming and budget systems.

Again, using the knapsack problem for demonstration, the solution procedure of MLKP by (Shih 2005) is presented next. The procedure is similar to those for solving the MOKP, where the only difference resides in the imbedded system.

Step 1. (Obtain reference information)

Step 1.1 Construct a payoff table for each objective of the DM(s) based on the possible capacity allocation under policy control (variables).
Step 1.2 Construct the fuzzy membership functions for measuring satisfactory degrees of objectives at all stages based on the concept of PIS and NIS at each stage.

Step 2. (Forward Phase)

Seek compromise solutions among multiple objectives under all possible capacity allocation at each stage by performing the minimum operation.
Let s_i^{j-1} be the i-th possibly available capacity after stage $j-1$, and $i \in I_j$, where I_j is the index set of i.
Set stage $j \leftarrow 1$
Step 2.1 If $j>N$, go to *Step 3*.
For all $i \in I_j$:
For all $x^j \in F_i = \{\text{feasible space under } s_i^{j-1}\}$, compute:

$\mu_{x^j}^j = \min_m \left\{ \mu_{R_m^j}(x^j) \right\}$, where R_m^j is the return for the m-th level DM at the j-th stage.

Step 2.2 Choose optimal decisions under the capacity at the current stage.
$\mu_i^{j*} = \max_{x^j \in F_i} \left\{ \mu_{x^j}^j \right\}$,
$J \leftarrow j + 1$, go to *Step 2.1*.

Step 3. (Backtracking Phase)

Recall all non-inferior outcomes obtained from Step 2: $\mu_i^{j*}, \forall\, i, j = 1, \ldots, N$.
Let M_j be the number of non-inferior outcomes at stage j.
Set $j \leftarrow N$.
Denote the aggregated satisfaction at stage j and its subsequent stages under various capacity constraints by $\hat{\mu}_i^j$, and $\hat{\mu}_i^N = \mu_i^{N*}, \forall\, i$.
Step 3.1 Set $j \leftarrow j - 1$.
If $j = 0$, go to *Step 4*; otherwise go to *Step 3.2*.
Step 3.2 Compute aggregated satisfaction.
For all $i \in I_j$:
For all $x^j \in F_i$, compute the aggregated satisfaction of x^j:
$\hat{\mu}_i(x^j) = \min \left\{ \mu_i^{j*}, \mu^{(j+1)*}(x^{j+1} | s_i^j) \right\}$,
where $\mu^{(j+1)*}(x^{j+1} | s_i^j)$ is the non-inferior satisfaction at stage $j+1$ given the available capacity s_i^j.
Step 3.3 Obtain the maximum satisfaction under s_i^{j-1}:
$\hat{\mu}_i^{j*} = \max_{x^j \in F_i} \{ \hat{\mu}_i(x^j) \}$.
Go to *Step 3.1*.

Step 4. (Termination of algorithm)

Obtain the optimum solution:

$$\hat{\mu}^{1*} = \max_{i \in I_1} \{ \hat{\mu}_i^{1*} \}.$$

Stop the algorithm.

Step 1 of the algorithm constructs the membership functions for evaluating the satisfaction degrees of DMs at all levels based on the information of PIS and NIS. Step 2 is the forward phase, where the satisfaction degrees of all levels according to each policy control under each possible capacity are computed first. Then the compromise of the solutions from all DMs given a policy control is obtained by performing a minimum operation on their respective satisfactions, and finally the optimum satisfaction for a given capacity at a stage is obtained by performing a maximum operation on all possible policy controls. Step 3 is the backward phase, which traces backward stage by stage from the last stage to find the aggregated satisfaction at each stage under all possible policy controls and capacity availabilities. The algorithm stops when the backward tracking reaches the first stage, and the maximum aggregated satisfaction

at the first stage under all possible capacity availabilities is chosen as the optimum solution.

The following example demonstrates the above solution procedure.

Example 6.1 A bi-level knapsack problem (Shih 2005).

$$\begin{aligned}
&\max_{x_1,x_2} \; z_1 = 4x_1 + 7x_2 + 2x_3 + x_4 \quad \text{(upper level)}\\
&\text{s.t.}\\
&\quad x_1 \geq 1,\\
&\quad x_1 \geq 2,\\
&\quad \text{where } x_3 \text{ and } x_4 \text{ solve:}\\
&\quad \max_{x_3,x_4} \; z_2 = 2x_1 + x_2 + 4x_3 + 3x_4 \quad \text{(lower level)}\\
&\quad \text{s.t.}\\
&\qquad 2x_1 + 2x_2 + x_3 + x_4 \leq 15,\\
&\qquad x_3 \geq 2,\\
&\qquad x_4 \geq 2,\\
&\qquad x_1, x_2, x_3, \text{ and } x_4 \text{ are positive integers.}
\end{aligned}$$

Assume the capacity of the knapsack is 15 units. The decisions are controlled by a team with bi-level objectives, z_1 (the team leader) and z_2 (the team member). The entire decision process is executed in four stages, where each stage corresponds to a control variable. As discussed earlier, at stage j, the available resource amount is s^{j-1} to allocate at this and the remainder stages. The allocation is restricted by the variables controlled by both levels. The allocated resource yields the objective increases at both levels, i.e. $z_1^j(x_i)$ and $z_2^j(x_i)$, at all stage $j = 1, 2, 3$ and 4. With the given the initial state s^0, the imbedded bi-objective problem maximizes its objectives at each stage.

The returns at Stage 1 for both levels maximize $R_1^1 = 4x_1$ and $R_2^1 = 2x_1$; for Stage 2, Max $R_1^2 = 7x_2$ and Max $R_2^2 = x_2$; for Stage 3, Max $R_1^3 = 2x_3$ and Max $R_2^3 = 4x_3$; and for Stage 4, Max $R_1^4 = x_4$ and Max $R_2^4 = 3x_2$. At each stage the imbedded return function $R^j = \left(R_1^j, R_2^j\right)$ forms the optimal goals in the DP structure with the given input states and control variables. The resource allocated to stage j is $W^j = s^{j-1} - s^j$, $j = 1, 2, 3$ and 4.

The PIS are $z_1 = 51$ and $z_2 = 60$, respectively, and the NIS are set as $z_1 = 0$ and $z_2 = 0$, respectively, by assuming no resource is allocated to any of the levels. DMs' preferences on goal achievement in terms of fuzzy membership functions are thus constructed based on the above PIS and NIS. Relevant information and the decisions made at all stages are presented in Tables 6.1, 6.2, 6.3 and 6.4.

The first column of these tables is a policy index which indicates the under a certain capacity availability of the action adopted by the DM. For example, in Table 6.1, policy index 1 is associated with a capacity of 10 and the action adopted by the DMs at this stage (i.e. stage 1) is $x_1 = 1$, resulting in the returns for the two-levels DMs being 4.00 and 2.00 respectively; these two returns provide satisfaction to the

Table 6.1 The degree of satisfaction under the capacity available for x_1 at stage 1

Policy index	Cap.	Act. acqed.	Cap. left	Return functions of both levels		Fuzzy goals		Max–min of satisfaction		Remarks
	s^0	x_1	s^1	R_1^1	R_2^1	$\mu_{R_1^1}$	$\mu_{R_2^1}$	$\min\{\mu_{R_1^1}, \mu_{R_2^1}\}$	μ_i^{1*}	Best policy
1	10	1	8	4.00	2.00	0.078	0.033	0.033	0.033	1
2	11	1	9	4.00	2.00	0.078	0.033	0.033	0.033	2
3	12	1	10	4.00	2.00	0.078	0.033	0.033	0.067	4
4		2	8	8.00	4.00	0.157	0.067	0.067		
5	13	1	11	4.00	2.00	0.078	0.033	0.033	0.067	6
6		2	9	8.00	4.00	0.157	0.067	0.067		
7	14	1	12	4.00	2.00	0.078	0.033	0.033	0.100	9
8		2	10	8.00	4.00	0.157	0.067	0.067		
9		3	8	12.0	6.00	0.235	0.100	0.100		
10	15	1	13	4.00	2.00	0.078	0.033	0.033	0.100	12
11		2	11	8.00	4.00	0.157	0.067	0.067		
12		3	9	12.0	6.00	0.235	0.100	0.100		

Note
1. Cap.: capacity, Act. acqed.: activity acquired
2. The capacity allocated to stage 1 is designated by W^1, or x_1, and left with $s^0 - x_1$
3. s^1 must be equal to or greater than 8 for the latter use

Table 6.2 The degree of satisfaction under the capacity available for x_2 at stage 2

Policy index	Cap.	Act. acqed.	Cap. left	Return functions of both levels		Fuzzy goals		Max–min of satisfaction		Remarks
	s^1	x_2	s^2	R_1^2	R_2^2	$\mu_{R_1^2}$	$\mu_{R_2^2}$	$\min\{\mu_{R_1^2}, \mu_{R_2^2}\}$	μ_i^{2*}	Best policy
1	8	2	4	14.00	2.00	0.275	0.033	0.033	0.033	1
2	9	2	5	14.00	2.00	0.275	0.033	0.033	0.033	2
3	10	2	6	14.00	2.00	0.275	0.033	0.033	0.050	4
4		3	4	21.00	3.00	0.412	0.050	0.050		
5	11	2	7	14.00	2.00	0.275	0.033	0.033		
6		3	5	21.00	3.00	0.412	0.050	0.050	0.050	6
7	12	2	8	14.00	2.00	0.275	0.033	0.033		
8		3	6	21.00	3.00	0.412	0.050	0.050		
9		4	4	28.00	4.00	0.549	0.067	0.067	0.067	9
10	13	2	9	14.00	2.00	0.275	0.033	0.033		
11		3	7	21.00	3.00	0.412	0.050	0.050		
12		4	5	28.00	4.00	0.549	0.067	0.067	0.067	12

Note s^2 must be equal to or greater than 4 for the latter use

Table 6.3 The degree of satisfaction under the capacity available for x_3 at stage 3

Policy index	Cap.	Act. acqed.	Cap. left	Return functions of both levels		Fuzzy goals		Max–min of satisfaction		Remarks
	s^2	x_3	s^3	R_1^3	R_2^3	$\mu_{R_1^3}$	$\mu_{R_2^3}$	$\min\{\mu_{R_1^3}, \mu_{R_2^3}\}$	μ_i^{3*}	Best policy
1	4	2	2	4.00	8.00	0.078	0.133	0.078	0.078	1
2	5	2	3	4.00	8.00	0.078	0.133	0.078		
3	6	3	2	6.00	12.00	0.118	0.200	0.118	0.118	3
4		2	4	4.00	8.00	0.078	0.133	0.078		
5		3	3	6.00	12.00	0.118	0.200	0.118		
6		4	2	8.00	16.00	0.157	0.267	0.157	0.157	6
7	7	2	5	4.00	8.00	0.078	0.133	0.078		
8		3	4	6.00	12.00	0.118	0.200	0.118		
9		4	3	8.00	16.00	0.157	0.267	0.157		
10		5	2	10.00	20.00	0.196	0.333	0.196	0.196	10
11	8	2	6	4.00	8.00	0.078	0.133	0.078		
12		3	5	6.00	12.00	0.118	0.200	0.118		
13		4	4	8.00	16.00	0.157	0.267	0.157		
14		5	3	10.00	20.00	0.196	0.333	0.196		
15		6	2	12.00	24.00	0.235	0.400	0.235	0.235	15
16	9	2	7	4.00	8.00	0.078	0.133	0.078		
17		3	6	6.00	12.00	0.118	0.200	0.118		
18		4	5	8.00	16.00	0.157	0.267	0.157		
19		5	4	10.00	20.00	0.196	0.333	0.196		
20		6	3	12.00	24.00	0.235	0.400	0.235		
21		7	2	14.00	28.00	0.275	0.467	0.275	0.275	21

Note s^3 must be equal to or greater than 2 for the latter use

Table 6.4 The degree of satisfaction under the capacity available for x_4 at stage 4

Policy index	Cap.	Act. acqed.	Cap. left	Return functions of both levels		Fuzzy goals		Max-min of satisfaction		Remarks
	s^3	x_4	s^4	R_1^4	R_2^4	$\mu_{R_1^4}$	$\mu_{R_2^4}$	$\min\{\mu_{R_1^4}, \mu_{R_2^4}\}$	μ_i^{4*}	Best policy
1	2	2	0	2.00	6.00	0.039	0.100	0.039	0.039	1
2	3	2	1	2.00	6.00	0.039	0.100	0.039		
3		3	0	3.00	9.00	0.059	0.150	0.059	0.059	3
4	4	2	2	2.00	6.00	0.039	0.100	0.039		
5		3	1	3.00	9.00	0.059	0.150	0.059		
6		4	0	4.00	12.00	0.078	0.200	0.078	0.078	6
7	5	2	3	2.00	6.00	0.039	0.100	0.039		
8		3	2	3.00	9.00	0.059	0.150	0.059		
9		4	1	4.00	12.00	0.078	0.200	0.078		
10		5	0	5.00	15.00	0.098	0.250	0.098	0.098	10
11	6	2	4	2.00	6.00	0.039	0.100	0.039		
12		3	3	3.00	9.00	0.059	0.150	0.059		
13		4	2	4.00	12.00	0.078	0.200	0.078		
14		5	1	5.00	15.00	0.098	0.250	0.098		
15		6	0	6.00	18.00	0.118	0.300	0.118	0.118	15
16	7	2	5	2.00	6.00	0.039	0.100	0.039		
17		3	4	3.00	9.00	0.059	0.150	0.059		
18		4	3	4.00	12.00	0.078	0.200	0.078		
19		5	2	5.00	15.00	0.098	0.250	0.098		
20		6	1	6.00	18.00	0.118	0.300	0.118		
21		7	0	7.00	21.00	0.137	0.350	0.137	0.137	21

Table 6.5 All possible non-inferior outcomes under the available capacity at stage 4

Capacity available s^3	Activity acquired x_4	Capacity left s^4	Maximum satisfaction $\hat{\mu}_i^{4*}$	Best policy (Index)
2	2	0	0.039	1
3	3	0	0.059	$3^{\#}$
4	4	0	0.078	6
5	5	0	0.098	10
6	6	0	0.118	15
7	7	0	0.137	21

Note
1. The data are obtained from Table 6.4
2. "#" indicates the optimal policy from the backtracking process

two DMs as 0.078 and 0.033, and a compromise solution is sought by performing a minimum operation on the respective results and obtained as 0.033, which is the result from Step 2.1 of the solution procedure. Policy indexes 3 and 4 are both under the case of capacity = 12, associated with each policy there is the action adopted and the resulting compromise satisfaction as those for policy index 1; however, we would like to choose the best results from the same capacity constraint, that is, perform a maximum operation on both the compromise satisfactions from the two policies and obtain the best satisfaction (=0.067) under capacity = 12, which is the procedure in Step 2.2.

Tables 6.1, 6.2, 6.3 and 6.4 establish the possible results can be obtained at different stages given the capacity left from their precedent stages. The backward procedure is based on these results and traces backward from the final stage (stage 4) to find the optimum solution of the problem. The backtracking process from stage 4 to stage 1 is presented in Tables 6.5, 6.6, 6.7 and 6.8. In Table 6.5, since stage 4 is the final stage, the aggregated satisfactions at this stage are exactly the maximum satisfactions obtained at this stage from the forward procedure. Policy index 3 is indicated as an optimal policy in Table 6.5. However, it is so far not known, and is found as an optimal policy when the backward procedure reaches the first stage in Table 6.8.

In Table 6.6, considering the available capacity s2, the actions (x_3) that can be adopted are enumerated and the best subsequent actions at stage 4 are discovered based on the action at stage 3. For example, when the capacity for stage 3 is 5 units, the actions that can be adopted are $x_3 = 3$ or 2. If $x_3 = 3$ is adopted, then there will be 2 units left for stage 4 and the best result can be obtained from stage 4 under that constraint is to take $x_4 = 2$ that provides a maximum satisfaction 0.039 for stage 4, as shown in Table 6.5. Consequently, action $x_3 = 3$ provides a best single-stage satisfaction at stage 3 as 0.118, and results in an aggregated satisfaction of stages 3 and 4 as 0.039. Alternatively, if $x_3 = 2$ is taken, it results in an aggregated satisfaction of 0.059. The example so far is the results from Step 3.2 of the solution procedure. Continuing this example, the best satisfaction at stage 3 under capacity = 5 is thus

Table 6.6 All possible non-inferior outcomes under the available capacity at stage 3

Capacity available s^2	Activity acquired x_3	Capacity left s^3	Maximum satisfaction (stage 3) μ_i^{3*}	Maximum satisfaction (stage 4) μ_i^{4*}	Aggregated satisfaction $\hat{\mu}_i(x^j)$	Maximum satisfaction $\hat{\mu}_i^{3*}$	Best policy (Index)
4	2	2	0.078	0.039	0.039	0.039	1
5	3	2	0.118	0.039	0.039	0.059	$2^{\#}$
	2	3	0.078	0.059	0.059		
6	4	2	0.157	0.039	0.039	0.078	4
	3	3	0.118	0.059	0.059		
	2	4	0.078	0.078	0.078		
7	5	2	0.196	0.039	0.039	0.078	8
	4	3	0.157	0.059	0.059		
	3	4	0.118	0.078	0.078		
	2	5	0.078	0.098	0.078		
8	6	2	0.235	0.039	0.039	0.098	12
	5	3	0.196	0.059	0.059		
	4	4	0.157	0.078	0.078		
	3	5	0.118	0.098	0.098		
	2	6	0.078	0.118	0.078		
9	7	2	0.275	0.039	0.039	0.118	17
	6	3	0.235	0.059	0.059		
	5	4	0.196	0.078	0.078		
	4	5	0.157	0.098	0.098		
	3	6	0.118	0.118	0.118		
	2	7	0.078	0.137	0.078		

Note
1. The data are derived from Tables 6.3 and 6.5
2. "#" indicates the optimal policy

the maximum of the two respective aggregated satisfactions and is obtained as 0.059, which is the result from Step 3.2.

Table 6.7 presents the backward tracking at stage 2, and Table 6.8 is the backward tracking at stage 1 with the tracking stopping at this stage. Table 6.8 shows the action $x_1 = 2$ can provide the best aggregated satisfaction 0.05. This policy leaves 11 units of capacity to later stages. Tracing stage 2 in Table 6.7, we find the best policy under this capacity constraint is $x_2 = 3$. With the same manner, we find the best policies at stages 3 and 4 are $x_3 = 2$ and $x_4 = 3$. Thus, $X = [2, 3, 2, 3]$ is the optimum solution of this bi-level problem.

Table 6.7 All possible non-inferior outcomes under the available capacity at stage 2

Capacity available	Activity acquired	Capacity left	Maximum satisfaction (stage 2)	Maximum satisfaction (stage 3)	Aggregated satisfaction	Maximum satisfaction	Best policy
s^1	x_2	s^2	μ_i^{2*}	μ_i^{3*}	$\hat{\mu}_i(x^j)$	$\hat{\mu}_i^{2*}$	(Index)
8	2	4	0.033	0.039	0.033	0.033	1
9	2	5	0.033	0.059	0.033	0.033	2
10	3	4	0.050	0.039	0.039	0.039	4
	2	6	0.033	0.078	0.033		
11	3	5	0.050	0.059	0.050	0.050	$6^{\#}$
	2	7	0.033	0.078	0.033		
12	4	4	0.067	0.039	0.039		
	3	6	0.050	0.078	0.050	0.050	8
	2	8	0.033	0.098	0.033		
13	4	5	0.067	0.059	0.059	0.059	12
	3	7	0.050	0.078	0.050		
	2	9	0.033	0.118	0.033		

Note
1. The data are derived from Tables 6.2 and 6.6
2. "#" indicates the optimal policy

Table 6.8 All possible non-inferior outcomes under the available capacity at stage 1

Capacity available	Activity acquired	Capacity left	Maximum satisfaction (stage 1)	Maximum satisfaction (stage 2)	Aggregated satisfaction	Maximum satisfaction	Best policy
s^0	x_1	s^1	μ_i^{1*}	μ_i^{2*}	$\hat{\mu}_i(x^j)$	$\hat{\mu}_i^{1*}$	(Index)
15	3	9	0.100	0.033	0.033	0.050	$10^{\#}$
	2	11	0.067	0.050	0.050		
	1	13	0.033	0.059	0.033		

Note
1. The data are derived from Tables 6.1 and 6.7
2. "#" indicates the optimal policy

6.3 Discussions

The supervised search approach presented in this chapter is useful for solving multi-level programming problems. The approach makes a trade-off among objectives and decisions of DMs in multi-level programming, thus reducing the complexity of solving the problem and improving the flexibility and robustness of the solutions. The compromise solutions of MLP problems are obtainable with a few iterations under

the supervision by upper-level DMs from top-down. The approach is rather efficient and gives DMs more confidence in the solution through interactions between the DMs and the analysts. Though only a bi-level example is illustrated in this chapter, the algorithm can be easily extended to multi-level cases without increasing their computational complexity.

The approach used Zadeh's fuzzy membership function for DMs' preference or degree of satisfaction and took a possibility distribution for uncertain or imprecise coefficients to solve the problems. Note, other types of fuzzy sets such as the intutionistic fuzzy set (IFS) may provide more flexibility in eliciting DMs' preferences. IFS as one form of generalized fuzzy sets is proposed by Atanassov (1986), considering both satisfaction and dissatisfaction of DMs. Moreover, though the max-min operator is particularly used in the approach for aggregating fuzzy measures, many other fuzzy operators can serve the same purpose, such as those presented in Table 5.1 in Chap. 5.

Chapter 7
Auction Mechanisms for Solving Multi-level Programming

Auction mechanism has been applied to distributed resource scheduling problems in recent years. The demand for distributed resource scheduling is more evident in manufacturing systems with complex supply structure and where customer service is prioritized (Kutanoglu and Wu 1999). In such an environment, companies usually have multiple product (project) mangers and each of them is responsible for a product or a particular project. Typically, each product/project manager is motivated to satisfy his/her customers' demands, and usually competes with each other for resource usages since there is seldom duplicated resources for dedicated product line or project due to economic consideration. The rapid advance of computing and communication technologies also enable making decisions in such a heterogeneous and distributed fashion. The resource allocation decision is usually at the hand of the decision-maker above the set of product/project managers, and hence the multi-level decentralized programming is ideally to model the problem. Auction mechanism that deals with the assignment of items to bidders coincides the resource competing scenario in the problem described above, and thus provides a way to solve the problem.

7.1 Auction Mechanism

Auction theory is important since a huge volume of goods and services, property, and financial instruments, are transacted via auctions, examples including mobile-phone licenses (Milgrom 2004), electricity (Green and Newbery 1992), and pollution permits (Dormady 2016). Auction is a market mechanism with an explicit set of rules to determine resource allocation and prices on the basis of bids from market participants (McAfee and McMillian 1987). Since the seminal paper by Vickrey (1961) on the study of auction theory, there have been many types of auctions proposed and

Part of the content of this chapter is reproduced from our previous work (Cheng 2011).

C.-B. Cheng et al., *Fuzzy and Multi-Level Decision Making: Soft Computing Approaches*, Studies in Fuzziness and Soft Computing 368,
https://doi.org/10.1007/978-3-319-92525-7_7

discussed in the literature and practice. This section first introduces some basic types of auctions: the ascending-bid auction (also called the open, oral, or English auction), the descending-bid auction (also called the Dutch auction), the first-price sealed-bid auction, and the second-price sealed-bid auction (also called the Vickrey auction), and then discusses two particular auction format: reverse auction and combinatorial auction, which are applied to two resource scheduling problems presented in this chapter.

7.1.1 Four Basic Auction Types

There are four standard types of auctions when only a single item is involved in the auction:

- Ascending-bid auctions, also called the open, oral, or English auctions. This type of auctions performs interactively in real time between the auctioneer and the bidders, either physically or electronically. During the process, the auctioneer gradually raises the price, bidders drop out until only one bidder remains, and the final bidder wins the object at this final price.
- Descending-bid auctions were originally used in the sale of flowers in the Netherlands and hence also called the Dutch auction by economists. This type of auctions has an interactive format as well, in which the auctioneer gradually lowers the price from an initial value until the moment when the first bidder accepts and pays the current price. Dutch auctions are generally used in the circumstance where the products are easily perishable such as flowers or fisheries.
- First-price sealed-bid auctions. In this type of auction, bidders submit simultaneous sealed bids to the auctioneer, where bids are written down and given in sealed envelopes. The auctioneer opens the bids altogether and announces the highest bidder who wins the object and pays the value of his/her bid.
- Second-price sealed-bid auctions, also called Vickrey auctions in honor of William Vickrey, who wrote the first game-theoretic analysis of auctions in which the second-price auction was also proposed (Vickrey 1961). In this type of auction, bidders submit simultaneously sealed bids to the auctioneer; the highest bidder wins the object and pays the value of the second-highest bid. This design is to remedy a phenomenon frequently occurring in common value auctions called winner's curse where the winning bidder tends to overpay due to incomplete information.

7.1.2 Reverse Auction

Reverse auctions are simply traditional auctions in reverse (Smart and Harrison 2002). In traditional (i.e. forward) auctions, a seller offers a product or service for sale to the highest bidder. By contrast, in a reverse auction, a buyer invites a tender for the

supply of a specific quantity of goods or services at a particular date. A reverse auction starts with the creation of a request for quotation (RFQ) which describes the buyer's specific requirements. Multiple qualified suppliers are then invited to participate in a bidding process. Many firms within the business-to-business (B2B) sector have recognized the possibility for lowering their costs by participating in online reverse auctions. The use of such auctions, also known as *electronic auctions* (Emiliani 2005), has emerged as a common technique for sourcing goods and services in many of the Fortune 2000 companies (Tully 2000). Broadly speaking, reverse auctions can be classified as either sealed bid, open bid, or semi-sealed bid, depending on the visibility granted to the individual suppliers of the competing bids. In auctions of the first type, the bidders have no visibility of the price submitted by their competitors. However, in open-bid auctions, full price visibility is made available to all participants in the bidding process. In the type of semi-sealed bid, the bidder is provided with price information that will make his bid active in the event, but no listing of the bids themselves is given; or another case is each bidder knows the current rank of their own bid in the bid-stream, but not the contents of rival bids (Teich 2004).

Reverse auctions in which the buyer awards the contract to a single bidder are referred to as sole-sourcing auctions (Bichler and Kalagnanam 2005). Che (1993) performed a pioneering study in which a mechanism was developed to support governmental procurement processes by using a utility function to score bids in terms of their price and quality issues. Branco (1997) extended this mechanism to a two-stage auction model, in which the procurer selected one of the competing firms on the basis of the bidding price, and then bargains with the selected firm to finalize the quality aspects. In implementing reverse auctions, one of the most critical factors in determining the success of the final outcome is the elicitation of the auctioneer's exact preferences regarding the various issues, e.g. price, delivery time, and quality level. Several researchers have argued that this is best achieved using some form of value function (Beil and Wein 2003; Bichler 2000, 2001). Meanwhile, Bichler et al. (1999) advocated the use of multiple-attribute utility theory (MAUT) to carry out bid evaluation in the reverse auction process. Note that a detailed review of reverse auctions and their design features can be found in (Teich et al. 2004).

As the use of reverse auctions has become increasingly common in the B2B domain, many companies have moved toward a multiple-sourcing policy, in which large amounts of less critical goods such as office furniture are procured from multiple rather than single suppliers. For example, Bichler and Kalagnanam (2005) reported that IBM had run a multiple-sourcing reverse auction for the procurement of a large quantity of chairs for one of their office buildings. According to Bichler and Kalagnanam (2005), the objective of the winner determination problem in the context of multiple-sourcing reverse auctions is to find the optimum combination of suppliers' offers which satisfies the buyer's demand and maximizes a pre-determined scoring function. The authors formulated this problem as a 0–1 knapsack problem with constraints imposed on the number of winning bids and the homogeneity of the purchase. In this formulation, bidders determine the quantities supplied to the buyer and the buyer finds an optimum combination of these quantities to satisfy the demand. Thought it is reasonable to assume the bidders have submitted their own

ideal quantity in the auction, some bidders might be able to yield a greater quantity with a smaller marginal cost due to different cost structure and production capacity availability. Thus it is possible to reduce the overall procurement cost of the buyer by exploring other quantities that can be provided by bidders than those are specified in their current bids.

Bi-level decentralized programming is suitable to model the interactive decision-making process between the buyer and the suppliers, in which the buyer is at the upper level and the individual suppliers are at the lower level. Cheng (2011) modeled the business procurement as a bi-level decentralized programming problem and solve the problem with a reverse auction mechanism, which took account of the vested interests of both the buyers and the suppliers.

7.1.3 Combinatorial Auction

Combinatorial auction was proposed as early as 1976 (Jackson 1976) for radio spectrum rights, and later Rassenti et al. (1982) proposed such auctions to allocate airport time slots (de Vries and Vohra 2003). With the combinational bids, CA enables bidders to express complementarities between items, and thus, the bidder does not have to speculate on an item's valuation under the risk of not getting other complementary items (Sandholm 2002). The winner determination problem in combinatorial auctions is an NP-complete problem (Sandholm 2000), and thus the feasibility of applying combinatorial auctions to real-world problems had been debated (McMillian 1994). To reduce the computational complexity of combinatorial auctions, Rothkopf et al. (1998) and Park and Rothkopf (2001) suggested limiting the number of bid combinations with some restriction strategies to make the winner determination problem computationally manageable. In 2008, the FCC formally adopted a combinatorial auction to sell the 700-MHz radio spectrum rights by utilizing the bid combination reduction approach of Rothkopf et al. (1998) to constrain the number of bids, and obtained $19 billion revenue for the American government. Epstein et al. (2002) also successfully implemented combinatorial auctions to help the Chile government improve the quality of school meal catering contract assignment. Their solution resulted in savings of around $40 million per year. Cheng et al. (2012) also suggested a combinatorial auction mechanism to solve a television advertising time slot allocation problem.

Recently, Cheng and Lo (2017) modeled the resource scheduling for a multi-project, multi-mode, and resource-constrained decision environment as a bi-level decentralized problem, and proposed a combinatorial auction mechanism to solve the problem. The so-called *multi-mode* in the problem means that the activities in a project can be accomplished in one out of several execution modes, each of which represents an alternative combination of resource requirement of the activity. The decision-making process has a hierarchical nature, where the resources are allocated first to individual projects by the upper-level manager, and then the project manager of each project schedules the project to optimize its outcome with the allo-

cated resources. The proposed solution procedure employs combinatorial auction mechanisms to determine resource allocations to projects, with the upper-level manager serving as the auctioneer and the project managers at the low-level bidding for resources.

7.2 Auction Mechanism for Bi-level Decentralized Programming

In many economic duo-ploy systems, it is common to see multiple followers compete with each other for resources that are distributed from the leader. The multi-sourcing and the multi-project scheduling problems mentioned in the previous section both fall in this category.

Let x be the control variable of the leader with objective function F, $y_1, y_2, \ldots, y_P$ are the controls of the p followers, and f_i is the objective function of the i-th follower, the bi-level decentralized programming is formulated as:

$$\begin{aligned}
&\max_x F\left(x, y_1, y_2, \ldots, y_p\right) \\
&\text{subject to:} \\
&\qquad G(x) \leq 0 \qquad (7.1) \\
&\text{where each } y_i, i = 1, 2, \ldots, p, \text{ solves:} \\
&\quad \max_{y_i} f_i\left(x, y_1, y_2, \ldots, y_p\right) \\
&\quad \text{subject to:} \\
&\qquad g_i\left(x, y_1, y_2, \ldots, y_p\right) \leq 0 \qquad (7.2)
\end{aligned}$$

When the decision x is made by the leader, each follower will strive to maximize his/her own interest according to Eq. (7.2) and submit his/her action y_i to the leader; the leader in turn evaluate the result based on Eq. (7.1).

Classical solution approaches for BLDPP generally assume the followers know the leader's strategy and the followers reveal their strategies simultaneously to each other as well. Such an assumption is not always true in many economic duo-ploy systems such as the multi-sourcing and the multi-project scheduling problems mentioned above. Taking the multi-sourcing problem for example, the buyer publishes his her request to suppliers and may reveal his/her criteria for quotations; however, the buyer is not aware of the profit/cost structure of the suppliers, and suppliers certainly would not reveal such information to each other since they are competing for the same award. In such a case, classical approaches are not applicable to solve the problem. Alternatively, auction mechanism that communicates between parties purely via the bids does not need to assume that the preferences and constraints of all parties are known.

Assuming x a resource allocation decision that divided the total resources to individual followers, each follower then determine the utilization (i.e. y_i) of the resources assigned to him/her to maximize his/her performance. Considering the leader as an auctioneer, individual followers are bidders submitting their bids y_i, $i = 1, 2, \ldots, p$, which maximize their utility functions, f_i, $i = 1, 2, \ldots, p$. In this auction design, we allow the auctioneer to re-evaluate his/her utility based on the received bids and adjust his/her resource allocation decision x. This process is performed in an interactive and iterative manner.

Based on the bids y_i, $i = 1, 2, \ldots, p$, submitted to the auctioneer, the auctioneer needs to have a mean to modify his/her decision x in the strive for receiving bids in the next round that will improve his/her utility F. The way to determine is called a winner determination problem. The problem can coincide with the original maximization of F, or solve an auxiliary problem instead. The auxiliary problem may modify its objective or constraints or incorporate new constraints as new information revealing in the on-going interaction. The winner determination problem can be represented by the following form:

$$\begin{aligned} &\max_x F'_t\left(x, y_{1,t-1}, y_{2,t-1}, \ldots, y_{p,t-1}\right) \\ &\text{subject to:} \\ &\quad G'_t(x) \leq 0 \end{aligned} \tag{7.3}$$

where F'_t is the objective of the problem at iteration t, G'_t is the constraint sets at iteration t, and $y_{i,t-1}$, $i = 1, 2, \ldots, p$, are the bids submitted in the previous iteration. F'_t can be identical to F or take a different form, likewise the G'_t. The way individual bidders to determine their bids is called a bid formulation problem and can be represented by:

$$\begin{aligned} &\max_{y_i} f'_{i,t}\left(x_{t-1}, y_{1,t-1}, y_{2,t-1}, \ldots, y_{p,t-1}\right) \\ &\text{subject to:} \\ &\quad g'_{i,t}\left(x_{t-1}, y_{1,t-1}, y_{2,t-1}, \ldots, y_{p,t-1}\right)0 \end{aligned} \tag{7.4}$$

where $f'_{i,t}$ is the objective of the i-th bidder at iteration t, $g'_{i,t}$ is the constraint sets of the i-th bidder at iteration t. Similarly, $f'_{i,t}$ and $g'_{i,t}$ can be identical to $f_{i,t}$ and $g_{i,t}$, respectively, or reformulated for convenience in determining the bid.

7.3 Multi-sourcing by BLDP with Reverse Auction

A business procurement usually starts with the request for quotation (RFQ) that describes the buyer's specific requirements. The buyer issues a RFQ to multiple qualified suppliers. Upon receipt of the RFQ, suppliers response with their quotations which maximize the benefits to themselves while simultaneously satisfying the buyer requirements specified in the RFQ. The buyer then evaluates the various quotations

in terms of their ability to optimize his/her benefit. The RFQ generally includes such details as the required quantity, the delivery time, the product specification, the quality grade, the warranty terms, the payment and delivery terms, the insurance requirements, and so on. Many of the conditions specified in the RFQ are negotiable, and hence the scope for mutual decision-making exists.

For the sake of simplicity, assume that there are only three issues in the quotation, namely the unit price, the delivery time and the delivery quantity. The unit price and delivery time are generally regarded as being the key factors in determining the success or otherwise of a supplier's quotation (Easton and Moodie 1999; Hendry and Kingsman 1994; Kingsman et al. 1996). However, in implementing a multiple-sourcing auction, the delivery quantity is also of significant interest since in practice, the demand allocation decision made by the buyer has a direct effect upon the suppliers' offer.

7.3.1 Bi-level Decentralized Formulation

As discussed above, the buyer's allocation decision affects the supplier's offer, and is affected in turn by the supplier's response. This circular-type relationship can be conveniently formulated using a bi-level decentralized programming (BLDP) model, in which the decisions made by the upper-level decision maker (i.e. the buyer in the current context) act as constraints upon the decisions made by the lower-level decision makers (i.e. the suppliers in the current context), which in turn prompt a re-evaluation of the decision made at the upper level. The BLDP model for multi-sourcing and its solution procedure by Cheng (2011) are introduced as follows.

In constructing the BLDP model for the multiple-sourcing reverse auction problem, the following notations apply:

Upper-level decision maker (the buyer):

Decision variables

- t delivery time.
- x_i binary variable, equal to 1 if a quantity is allocated to the i-th supplier; 0 otherwise.
- q_i quantity allocated to supplier i

Parameters

- $\bar{p}$ average unit cost to buyer.
- t' the initial delivery time demanded by the buyer.
- d difference between t and t'.
- Q total demanded quantity.
- $Q_i^{\min}$ minimum order quantity for supplier i.
- $Q_i^{\max}$ maximum order quantity for supplier i.
- S_i ordering cost of the buyer when dealing supplier i.

n total number of invited suppliers

Lower-level decision maker (the suppliers):

Decision variables

p_i unit bidding price of supplier i.

y_{ij} quantity acquired from j-th production method by supplier i, $j = 1, 2, 3, 4$; where y_{i1} is the amount fulfilled by the initial inventory, y_{i2} is acquired from the current master production schedule (MPS); y_{i3} is obtained from new production lots manufactured using the existing production capacity; and y_{i4} is obtained from crash production, i.e. new production lots manufactured using an overtime capacity

Parameters

π_i gross profit of supplier i.

τ_ι total production cost of bid of supplier i.

c_{ij} unit cost associated with j-th production method of supplier i.

$\tilde{u}_i$ upper bound of unit bidding price perceived by supplier i, i.e. the price above which the supplier considers the bid to have no chance of success.

I_i^0 initial inventory of supplier i.

ATP_{ik} cumulative available-to-promise inventory to period k based on current production schedule of supplier i.

κ_{ik} regular capacity (expressed in time) available in period k of supplier i.

λ_{ik} overtime capacity (expressed in time) available in period k of supplier i.

h_i processing time per unit product of supplier i

The bi-level decentralized programming problem is formulated as follows:

$$\text{Minimize } \bar{p} \tag{7.5}$$

$$\text{Minimize } d \tag{7.6}$$

Subject to:

$$\bar{p} = \sum_{i=1}^{n} (p_i q_i + x_i S_i)/Q \tag{7.7}$$

$$\sum_{i=1}^{n} q_i = Q \tag{7.8}$$

$$x_i\, Q_i^{\min} \le q_i \le x_i\, Q_i^{\max}, \forall i \tag{7.9}$$

$$d = t - t' \tag{7.10}$$

$$t \ge t' \tag{7.11}$$

$q_i, \forall i$, and t are integers,

$x_i \in \{0, 1\}, \forall i$

where p_i solves the i-th supplier's bid determination problem:

$$\text{Maximize } \pi_i \tag{7.12}$$

Subject to:

$$p_i \leq \tilde{u}_i \tag{7.13}$$

$$\pi_i = p_i - (\tau_i / q_i) \tag{7.14}$$

$$\tau_i = \sum_j c_{ij} y_{ij} \tag{7.15}$$

$$\sum_j y_{ij} = q_i \tag{7.16}$$

$$y_{i1} \leq I_i^0 \tag{7.17}$$

$$y_{i1} + yi_2 \leq AT P_{it} \tag{7.18}$$

$$h_i y_{i3} \leq \sum_{k \leq t} \kappa_{ik} \tag{7.19}$$

$$h_i y_{i4} \leq \sum_{k \leq t} \lambda_{ik} \tag{7.20}$$

$$p_i \geq 0,\ y_{ij}, \forall j, \text{ are integers}$$

As shown in Eqs. (7.5) and (7.6), the upper-level decision maker, i.e. the buyer, has two objectives in the reverse auction, namely (1) to minimize the average unit cost, i.e. the average bidding price and ordering cost per unit, and (2) to minimize the delay, i.e. the difference between the demanded delivery time and the actual delivery time (see Eqs. 7.10 and 7.11). Equation (7.7) illustrates the method used to compute the average unit cost in the solution procedure, while Eq. (7.8) ensures that the sum of all the individual quantities allocated to the different suppliers does not exceed the total demand Q. Finally, Eq. (7.9) expresses the constraint that the order placed on a particular supplier must be greater than or equal to a certain supplier-specific minimum quantity $Q^{\min}$, but less than or equal to the maximum order quantity $Q^{\max}$. $Q^{\min}$ is obtained based on business practice that the minimum quantity a supplier would be willing to deliver, while $Q^{\max}$ is conceived by the buyer based on the supplier's size. Note that this constraint ensures that the final allocation decision is amenable to both the buyer and the suppliers.

Once the allocated quantity q_i and delivery time t have been determined by the upper-level decision maker, each lower-level decision maker (i.e. each supplier) responds by computing an appropriate bidding price. In doing so, their objective is to maximize their profit (Eq. 7.12) by achieving an acceptable compromise between the bidding price and the production cost. The constraint given in Eq. (7.13) expresses the subjective judgment of the i-th supplier regarding the maximum price which the buyer is liable to tolerate. In practice, it is difficult to assign a precise value to this parameter, and therefore in the current study, it is evaluated using a fuzzy

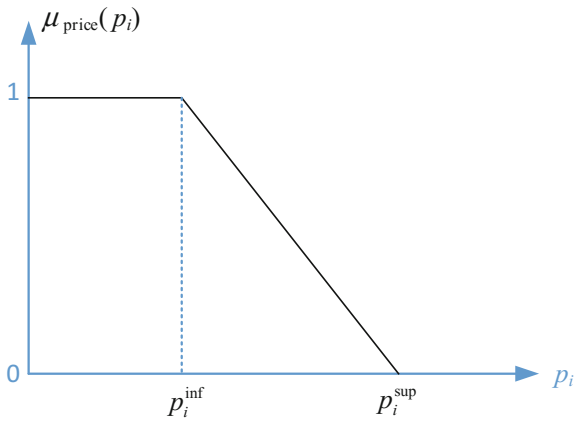

Fig. 7.1 Membership function of supplier's offer price

approach. As shown in Fig. 7.1, the fuzzy membership function of the price variable is defined as $\mu_{\text{price}}(p_i) \in [0, 1]$. When we assign a membership function to a variable, we obtained a possibility distribution of this variable (Yen and Langari 1999, p. 31). Hence, the membership function in Fig. 7.1 models the supplier's confidence in the possibility of winning the contract with an offer price p_i. Note that the values of p_i^{inf} and p_i^{sup} in Fig. 7.1 are assigned by the supplier in accordance with his experience and knowledge of the market. This concept of market-awareness price offer has been proposed in the bilateral contracts negotiation in electricity markets by Khatib and Galiana (2007). In the mixed pool/bilateral electricity market, the participant who initiates the negotiation process proposes a contract price that meets its risk/benefit criteria over an acceptable range of the length of the contract. This proposed price is a compromise between the participant's desire to maximize its benefit, yet granting enough flexibility to the other party to accept this proposal (Khatib and Galiana 2007).

Equation (7.14) computes the unit profit to the supplier, while Eq. (7.15) calculates the total production cost of the corresponding production plan. Meanwhile, Eqs. (7.17) and (7.18) describe the total number of units currently available to meet the allocated demand in the ATP cumulated to period t, i.e. the due date. [Note that the ATP inventory is the uncommitted portion of a company's inventory and planned production which is maintained in the master schedule to support customer order promises (American Production and Inventory Control Society 2004)]. Finally, Eqs. (7.19) and (7.20) express the constraint that any additional production required to satisfy the allocated demand should not exceed the sum of the regular and overtime capacities, respectively.

In the BLDP model described above, the respective decisions of the upper- and lower-level decision makers have a direct influence upon one another. For example, the buyer's decision regarding the delivery time and quantity allocation constrains the suppliers' choice of production plans and therefore affects their optimum price decision. Similarly, the offer prices submitted by the suppliers may cause the buyer

to revisit the quantity allocation decision or delivery time requirement in order to achieve a lower price.

Considering that the objectives and decisions of the upper-level decision maker usually have allowable tolerances which can be described by fuzzy membership functions and constrain the feasible space for the decisions made by the lower-level decision units. Adopting this paradigm, the search for a solution to the BLDP problem involves establishing the values of the membership functions which result in an acceptable degree of satisfaction to both the upper- and the lower-level decision makers. Pramanik and Roy (2007) also adopted the concept of membership functions to formulate MLP problems as a fuzzy goal programming problem. In their approach, the objectives and decision vectors of the MLP problem were transformed into fuzzy goals via the assignation of appropriate aspiration levels. The fuzzy goals were then characterized by membership functions obtained by defining tolerance limits for the achievement of the aspiration levels of the individual goals.

7.3.2 Problem Transformation by Max-Min Decision Approach

The max-min decision concept of Bellman and Zadeh (1970) presented in Chap. 3 is applied to solve the problem. A direct reason of using the fuzzy approach is that it provides the convenience of dealing with the fuzzy constraint (9.13) in the model. Furthermore, under the consideration that the cost information of suppliers is not revealed to the buyer, the fuzzy approach enables the separate decisions by the buyer and suppliers yet at the same time they can jointly search a satisfactory solution to all parties through the interactions of decision evaluation. When the KKT optimality conditions are used to transform the bi-level problem to a single level problem of the buyer, it also implies that such cost information is fully revealed to the buyer. It is also noted by Lai (1996) that in the hierarchy structure no one can have his individual optimum decision while his existing competitor has conflicting objectives, and thus, a satisfactory decision (defined as the membership of optimality of the objective function) is meaningful and rational for all players trying to maximize their individual objectives as much as possible.

Recall the max-min decision approach. Let $G_i, i = 1,\ldots, m$, be the m fuzzy goals and let $L_j, j = 1,\ldots,n$, be the n constraints imposed on the decision space Z. The degree of suitability (i.e. the membership value) of any potential solution to the fuzzy decision problem is defined as the conjunction of the satisfaction (also defined in terms of a membership function) it achieves of each of the fuzzy goals and constraints within the problem, i.e.

$$\mu_D(z) = \mu_{G_1}(z) \otimes \mu_{G_2}(z) \otimes \cdots \otimes \mu_{G_m} \otimes \mu_{L_1}(z) \otimes \mu_{L_2}(z) \otimes \cdots \otimes \mu_{L_n}(z), \tag{7.21}$$

where $z \in Z$ and $\otimes$ is a conjunction operator. The essence of the decision-making problem is therefore to find the solution which maximizes $\mu_D(z)$. In other words, if there exists a z^* such that $\mu_D(z^*) = \max_z\{\mu_D(z)\}$, then z^* represents the optimum solution of the decision problem D. When the conjunction operator in Eq. (7.21) is defined as a min operator, the solution can be easily obtained using a max-min approach.

7.3.3 Solution Procedure Based on Reverse Auction

When considering the interaction between the upper- and lower-level decision makers from the aspect of reverse auction, the principle of the auction mechanism presented in Sect. 7.2 can be applied to solve the problem. The solution process commences with the upper-level decision maker specifying the quantities to be allocated to each of the lower-level decision units and the associated delivery time. Each lower-level decision unit then determines an appropriate bidding price by optimizing the lower-level decision problem. The lower-level decisions are submitted to the upper-level decision maker and are evaluated using a max-min criterion based on the integrated objective of optimizing both the average unit price and the delivery time. If the result is deemed to be satisfactory then the decision process terminates; otherwise, the upper-level decision maker modifies the quantity allocation and/or delivery time and passes the amended decision to the lower-level decision units in the form of a new RFQ. Figure 7.2 illustrates the implementation of this iterative solution procedure.

7.3.4 Winner Determination Problem for Reverse Auction

When applying the max-min decision approach, the winner determination problem at the upper level can be reformulated as the single objective optimization problem:

$$\begin{aligned}
&\text{Maximize}\,\alpha \quad (\text{WDRA}-1)\\
&\text{Subject to:} && (7.22)\\
&\qquad \mu_{\bar{p}}(\bar{p}) \geq \alpha, && (7.23)\\
&\qquad \mu_d(d) \geq \alpha, && (7.24)\\
&\qquad 0 \leq \alpha \leq 1, && (7.25)\\
&\qquad \text{Equations}(7.7)-(7.11)\\
&\qquad q_i, \forall i, \text{and } t \text{ are integers},\\
&\qquad x_i \in \{0,1\}, \forall i
\end{aligned}$$

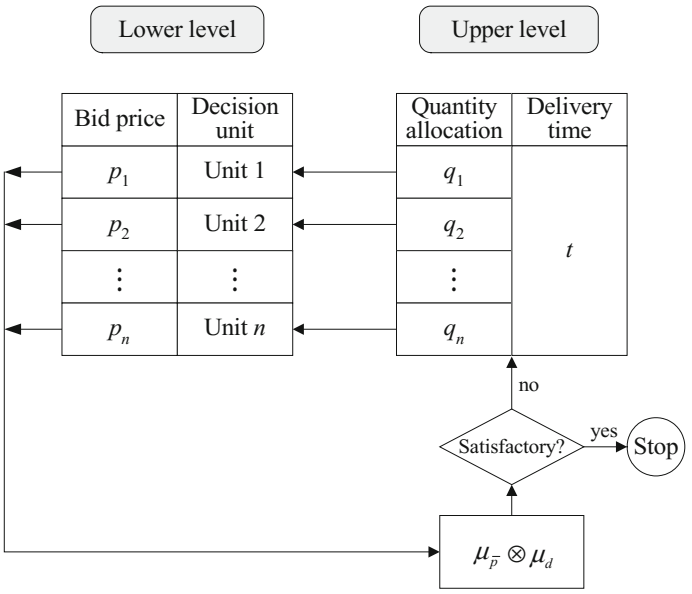

Fig. 7.2 Iterative solution procedure based on reverse auction

The formulation above implies that $\alpha = \min\{\mu_{\bar{p}}(\bar{p}), \mu_d(d)\}$. α can be considered as the degree of intersection of the two objectives $\bar{p}$ and d, and the maximization of α leads to a chance of optimum compromise between the two objectives. The membership functions $\mu_{\bar{p}}(\bar{p})$ and $\mu_d(d)$ can be interpreted as the degree of the buyer's satisfaction with $\bar{p}$ and d respectively. The average unit price to the buyer is a smaller-the-better attribute, and thus a reasonable and simple membership function design for $\mu_{\bar{p}}(\bar{p})$ has the form shown in Fig. 7.3a, in which the decision maker need only designate the preferred price and maximum acceptable price, i.e. $\bar{p}^{\inf}$ and $\bar{p}^{\sup}$, respectively. Similarly, Fig. 7.3b illustrates the membership function for the delivery time $\mu_d(d)$, in which $d^{\sup}$ indicates the maximum acceptable delay.

Assume that the solution of the above formulation is α^0. It is noted that there may be multiple decisions which will yield the same α^0. In such a case, the solution might not be unique nor efficient because the solution procedure randomly selected a solution which might be dominated by another solution with the same satisfaction degree α^0 (Lai and Hwang 1993). To resolve this problem, Lai and Hwang (1993) proposed to solve an auxiliary problem that maximizes a weighted average of the individual objectives. Accordingly, an auxiliary problem based on this concept for the WDRA-1 can be formulated as follows:

$$\text{Maximize } w_1 \mu_{\bar{p}}(\bar{p}) + w_2 \mu_d(d) \tag{7.26}$$

Subject to:

$$\mu_{\bar{p}}(\bar{p}) \geq \alpha^0, \tag{7.27}$$

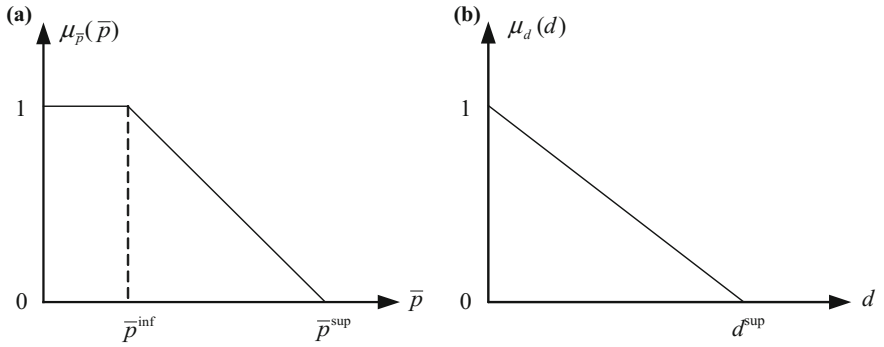

Fig. 7.3 Membership functions of **a** average unit price, and **b** delay

$$\mu_d(d) \geq \alpha^0, \tag{7.28}$$
$$\text{Equations (7.7)} - \text{(7.11)}$$

where w_1 and w_2 are the weights of the two objectives $\mu_{\bar{p}}(\bar{p})$ and $\mu_d(d)$ respectively, and $w_1 + w_2 = 1$. The weighted average of objectives allows possible compensation among objectives. Lai and Hwang (1993) further showed that the max-min problem and the auxiliary problem can be solved in one step by unifying the objectives in the two formulations. That is, the problem is reformulated as:

$$\begin{aligned} &\text{Maximize } \alpha + \delta(w_1\mu_{\bar{p}}(\bar{p}) + w_2\mu_d(d)) \quad \text{(WDRA} - 2) \\ &\text{Subject to:} \\ &\qquad \text{Equations (7.7)} - \text{(7.11)}, \ \text{(7.23)} - \text{(7.25)} \\ &\qquad q_i, \forall i, \text{ and } t \text{ are integers,} \\ &\qquad x_i \in \{0,\ 1\}, \forall i \end{aligned} \tag{7.29}$$

where δ is a sufficient small positive number. The above formulation is referred to as an augmented max-min approach.

7.3.5 Bid Formulation Problem for Reverse Auction

In a reverse auction, the suppliers are generally unsure of the buyer's exact preferences regarding the delivery time and unit cost. Thus, in the solution procedure, the preferences of the upper-level decision maker are not formally imposed on the lower-level decision units. Nevertheless, such preferences are still considered by each lower-level decision unit through Eq. (7.13) though it is just a best-guess estimate rather than true information.

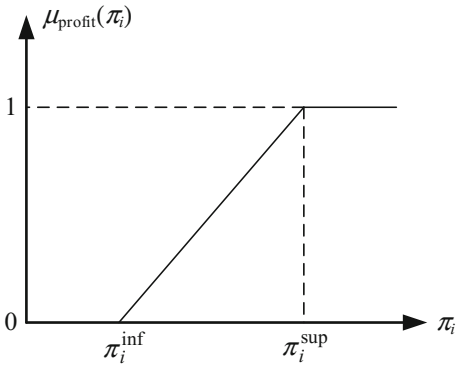

Fig. 7.4 Membership function of gross profit of supplier i

When formulating a bid, the supplier must balance the opposing requirements of securing the contract whilst simultaneously enhancing their own financial position. In other words, the bidding decision represents a trade-off between the supplier's level of profitability and the likelihood of winning the business. Similar to the upper-level decision making problem, the augmented max-min model of the lower-level decision making problem can be formulated as:

$$\text{Maximize } \beta_i + \gamma(v_{i_1}\mu_{\pi_i}(\pi_i) + v_{i_2}\mu_{p_i}(p_i)) \quad \text{(BFRA)} \tag{7.30}$$

Subject to:

$$\mu_{\pi i}(\pi_i) \geq \beta_i \tag{7.31}$$

$$\mu_{pi}(p_i) \geq \beta_i \tag{7.32}$$

$$0 \geq \beta_i \geq 1 \tag{7.33}$$

Equations (7.14) − (7.20)

$p_i \geq 0$

$y_{ij}, \forall j$, are integers

where $\beta_i = \min\{\mu_{\pi_i}(\pi_i), \mu_{p_i}(p_i)\}$, γ is a sufficient small positive number, and v_{i_1} and v_{i_2} are weights of $\mu_{\pi_i}(\pi_i)$ and $\mu_{p_i}(p_i)$ respectively and $v_{i_1} + v_{i_2} = 1$. The membership function $\mu_{pi}(p_i)$ is identical to that defined in Fig. 7.1, while π_ι as shown in Fig. 7.4 is a larger-the-better attribute, and thus its membership function can be constructed in accordance with the values of the minimum acceptable profit, π_i^{inf}, and preferred profit, π_i^{sup}, established by the supplier.

Determining the optimal solution to the BFRA problem described above involves finding the production plan which satisfies the buyer's request in the most economic manner possible. In general, the production options available to a supplier for satisfying the buyer's request can be sequenced in order of increasing cost as follows (Cheng 2008): (1) make no change to the current production plan (and utilize any inventory remaining from the previous period); (2) add new production lots to the current production plan subject to normal capacity constraints; and (3) implement a

crash production plan using overtime capacity. Clearly, if the first production option is sufficient to fulfill the buyer's request in its entirety, the other two options are redundant. In the solution procedure, the feasible production plans are directly obtained by utilizing the prioritized sequence given above to search the supplier's Master Production Schedule (MPS) and Available-to-Promise (ATP) inventory. With this method, the constraints given in Eqs. (7.16)–(7.20) can be waived and hence the BFRA problem can be solved more easily.

7.3.6 Reverse Auction Algorithm

The objective of the upper-level winner determination problem (WDRA-1) is to find the optimum combination of the quantity allocation $\boldsymbol{q} = [q_1,\ldots, q_n]$ and the delivery time t. The solution procedure mimics a real-world buyer-supplier negotiation process in that it commences by specifying a deliberately tight delivery time and then searching for the corresponding optimum quantity allocation. If an unsatisfactory result is obtained (i.e. the average unit price is too high), the buyer eases the delivery time with the anticipation of receiving an improved (i.e. cheaper) offer. The basic steps of the iterative reverse auction algorithm (denoted by IRA) can be summarized as follows.

IRA Algorithm.

Step 0: $t = t'$

Step 1: Set iteration counter $r = 1$, $q_i(r) = Q/n$, for $i = 1,\ldots,n$, where $q_i(r)$ denotes the quantity allocated to supplier i in the r-th iteration.

Step 2: Solve the BFRA with $q_i(r)$, $i = 1,\ldots,n$, and t for all n suppliers to obtain $p_i(r)$, $i = 1,\ldots,n$

Step 3: Compute $\alpha(r) = \min\{\mu_{\bar{p}}(\bar{p}), \mu_d(d)\}$.

Step 4: Set $r = r + 1$.
Solve the WDRA-2 to obtain the new quantity allocation $q_i(r)$.

Step 5: If $\sum_{i=1}^{n} |q_i(r) - q_i(r-1)|/Q < \varepsilon$ or $r > r^{\max}$, where ε is a small value and $r^{\max}$ is a pre-specified maximum number of iterations, then go to Step 6; otherwise, go to Step 2.

Step 6: Let $h = \arg\max_{r}\{\alpha(r)\}$, and
$\alpha^*(t) = \alpha(h)$, $q^*(t) = [q_1(h),\ldots, q_n(h)]$, $p^*(t) = [p_1(h),\ldots, p_n(h)]$.
Reset r.
Set $t = t + 1$.

Step 7: If $\mu_d(d) \leq 0$ then go to Step 8; otherwise go to Step 1.

Step 8: Let $g = \arg\max_{t}\{\alpha^*(t)\}$, and
$\alpha^{**} = \alpha^*(g)$, $q^{**} = q^*(g)$, $p^{**} = p^*(g)$.
Stop.

As shown, the algorithm commences by setting the requested delivery time equal to the initial time demanded by the buyer. In Step 1, the total buyer demand is evenly

allocated amongst all the suppliers. The suppliers respond by computing an appropriate bidding price (Step 2) and submitting these prices to the buyer for evaluation (Step 3). In the event that the buyer is not satisfied, he/she solves the WDRA-2 to obtain a new quantity allocation (Step 4). In Step 5, if the new quantity allocation is not significantly different from the previous version, or the search process has exceeded a pre-specified number of iterations, the search for alternative allocation distributions under the current requested delivery time is terminated. In this event, the best (i.e. cheapest) allocation pattern under the current delivery time constraint is recorded and the delivery time is then increased by one unit of time to search for potentially improved solutions (Step 6). The increasing of delivery time step by step carries out a tradeoff between the buyer's two objectives (i.e. average unit price and delivery time). A short delivery time is likely to disturb the supplier's production schedule and increases his cost, which in turn creates a higher unit price to the buyer. When the buyer increases the delivery time, the supplier would be able to utilize the scheduled production lots to fulfill the buyer's order and deliver it with lower cost. In such a case, the supplier can offer a more competitive price to win the contract and thus creates a win-win situation between the buyer and the supplier. Step 7 terminates the algorithm under the condition that $\mu_d(d) \leq 0$, i.e. the delivery time is absolutely unacceptable to the buyer, and thus no merit exists in further extending the delivery time to search for a cheaper solution. In the final step (Step 8), the optimum solution to the reverse auction problem is determined by comparing all the solutions found in Step 6 during the course of the iterations.

Example 7.1 A buyer requires 450 automobile rain brushes to be received in six week time. It is further assumed that the buyer intends to award the contract to multiple suppliers. Thus, three suppliers, namely SUPPLIER-1, SUPPLIER-2 and SUPPLIER-3, are invited to bid for the contract via a sealed-bid reverse auction process. The preferences of the buyer and the three suppliers are described by membership functions with the parameters shown in Table 7.1. Note that these parameters are confidential to the buyer and the individual suppliers. Table 7.2 summarizes the cost parameters of the three suppliers. Note that neither the buyer nor the competing suppliers have any knowledge of the cost structures of the individual suppliers. The satisfactions of the buyer's two objectives are equally important, i.e. $w_1 = w_2 = 0.5$.

Assuming there are four production methods, and the search of feasible production plans is conducted according to the following production plan enumeration procedure:

Case 1: If the current MPS is sufficient to meet the buyer's demand, i.e. the cumulative ATP prior to the delivery time requested by the buyer is greater than or equal to the buyer's demand, then no action need be taken. This part contains the initial inventory ($y_{i1} = I_i^0$) and the amount produced by the current MPS (y_{i2}). Since this plan is the most economic solution possible from the supplier's perspective, a search for alternative production plans is not required and the buyer's demand is simply satisfied from the current MPS.

Table 7.1 Membership function parameters of buyer and competing suppliers

Panel A: Buyer

Average Price		Delay
$\bar{p}^{inf}$	$\bar{p}^{sup}$	d^{sup}
22	27	4

Panel B:

Supplier	Price		Profit	
	p_i^{inf}	p_i^{sup}	π_i^{inf}	π_i^{sup}
SUPPLIER-1	21	26	3	6
SUPPLIER-2	20	26	2.5	5
SUPPLIER-3	20.5	26.5	2	6

Table 7.2 Cost parameters of competing suppliers

Supplier	c_1	c_2	c_3	c_4
SUPPLIER-1	17	18.22	22	24.5
SUPPLIER-2	18	19	22.5	25
SUPPLIER-3	20	21	23	26

Case 2: In the event that the current MPS is insufficient to meet the buyer's demand, the search process examines the existing capacity to determine whether or not sufficient surplus capacity exists to increase the MPS. If sufficient capacity does indeed exist, additional production lots are added to the MPS to make up for the short fall in the customer order. The amount produced by this new production plan is y_{i3}. Note that the additional lots are added to the tail end of the MPS in order to avoid impacting the current production arrangement. If insufficient capacity exists, the search process enters the Case 3 scenario described below.

Case 3: In the event that the current capacity availability is insufficient to meet the buyer's requirements, the buyer's demand is satisfied by invoking an overtime facility and the produced amount is y_{i4}. (Note that as in Case 2, the additional production lots are added to the tail end of the MPS in order to avoid disruption to the current production arrangement.)

The solution procedure commences with the buyer sending identical RFQs to each supplier requesting a delivery of 150 units (i.e. 450 units/3) in six week time. Upon receipt of these RFQs, each supplier examines its MPS and solves the corresponding BFRA problem in accordance with the procedures described in the following.

The current MPS of SUPPLIER-1 is shown in Table 7.3. As shown, the cumulative ATP to Week 6 is 165 units, which is greater than the buyer's request (150 units). Thus, the current MPS is sufficient to fulfill the order, i.e. $y_{11} = 20$, $y_{12} = 130$ and $y_{13} = y_{14} = 0$. In this scenario, the order can be fulfilled with no delay, and hence the supplier

need only decide upon a suitable unit price. From Table 7.4, the unit production cost is determined to be (20units × \$17/unit + 130units × \$18.22/unit)/150 = \$18.06. Inserting this production cost into the BFRA problem and utilizing optimization software, the optimum solution for SUPPLIER-1 is found to be $p_1 = 22.91$ and $\pi_1 = 4.85$.

The current MPS of SUPPLIER-2 is shown in Panel B of Table 7.3. From inspection, the cumulative ATP to Week 6 is 100 units, which is clearly less than the buyer's request for 150 units. Therefore, additional production lots must be launched to satisfy the customer order. Producing the additional 50 units requires a total of 0.4 h/unit × 50units = 20 h of additional capacity (i.e. $h_2 = 0.4$). However, Table 7.4 shows that SUPPLIER-2 has only 16 h of regular capacity available in Week 6, and thus an additional 4 h of overtime capacity are required. As a result, the total order requirement is fulfilled by $y_{21} = 10$, $y_{22} = 90$, $y_{23} = 40$ and $y_{24} = 10$, and thus the total cost is given by 10units × \$18/unit + 90units × \$19/unit + 40units × \$22.5/unit + 10units × \$25/unit = \$3040. In other words, the unit cost of the order is \$20.27. Solving the corresponding BFRA problem for SUPPLIER-2, the optimum decision is found to be $p_2 = 23.72$ and $\pi_2 = 3.45$.

Adopting a similar approach to that described above, the optimum decision for SUPPLIER-3 is determined to be $p_3 = 24.16$ and $\pi_3 = 3.56$. The optimum decisions for each of the three suppliers are shown in the upper row of Table 7.5. The bid prices of the three suppliers are submitted to the buyer, and the corresponding objective value of the WDRA-2 is computed by Step 3 of the algorithm in Sect. 3.3 to be 0.68. Solving the WDRA-2 using the price information submitted by the suppliers, the new quantity allocation is determined to be $q_1 = 250$, $q_2 = 200$ and $q_3 = 0$. New RFQs are prepared in accordance with this amended quantity allocation and are sent to the first two suppliers. Step 2 is then repeated.

Table 7.5 summarizes the complete set of computational results obtained for this illustrative example. Note that the relevant parameter settings used in the solution procedure are indicated beneath the table. In this particular example, the maximum number of iterations $r^{\max}$ for each designated delivery time is specified as 3. In other words, for each delivery time, the algorithm identifies three optimum quantity allocations. Note that when the delivery time was extended to 10, the degree of buyer satisfaction with the delivery time was equal to zero, resulting in a value of $\alpha = 0$. Thus, there was no point in exploring further solutions, and the algorithm was terminated. As shown in Table 7.5, the optimum solution to the illustrative reverse auction problem was found to be ($\bar{p} = 23.62$, and $\alpha = 0.68$ and the objective value is 0.69) in the first iteration of $t = 7$. Note that while an apparently lower average-unit-price exists in the first iteration associated with $t = 8$, i.e. $\bar{p} = 23.35$, the longer delivery time is less satisfactory to the buyer, and hence the overall buyer satisfaction is reduced to a value of $\alpha = 0.5$ and an objective value of 0.51.

Table 7.3 MPS of three suppliers

Panel A: SUPPLIER_1												
Initial inventory 20	Week											
	1	2	3	4	5	6	7	8	9	10	11	12
MPS	90		90			90		90				90
Committed orders	45	30	15	10	5	5	15	20	15	20	10	5
ATP	35		95			165		190				275
Panel B: SUPPLIER_2												
Initial inventory 10	Week											
	1	2	3	4	5	6	7	8	9	10	11	12
MPS	90			90				90		90		
Committed orders	20	20	5	5	15	15	10	20	10	10	10	20
ATP	55			100				160		210		
Panel C: SUPPLIER_3												
Initial inventory 60	Week											
	1	2	3	4	5	6	7	8	9	10	11	12
MPS			90		90		90			90		
Committed orders		20	45	5	10	10	15	20	10	10	10	20
ATP	40		80		150		195			245		

Table 7.4 Capacity availability at three suppliers

SUPPLIER		Week											
		1	2	3	4	5	6	7	8	9	10	11	12
1	Regular	0	0	0	30	45	50	60	60	60	60	70	80
	Overtime	0	0	10	20	30	30	30	30	30	30	30	30
2	Regular	0	0	0	0	0	16	40	40	60	40	80	80
	Overtime	5	5	5	5	10	10	10	10	20	20	20	20
3	Regular	0	0	0	0	20	20	30	50	50	60	60	70
	Overtime	0	0	0	10	20	20	30	30	30	30	30	30

Table 7.5 Results of illustrative example

Delivery time	r	Upper level			Lower level						$\bar{p}$	α	Obj
		q_1	q_2	q_3	p_1	π_1	p_2	π_2	p_3	π_3			
$t=6$ ($d=0$)	1	150	150	150	22.91	4.85	23.72	3.45	24.16	3.56	23.66	0.67	0.68
	2	250	200	0	23.75	4.35	24.55	3.10	–	–	24.16	0.57	0.58
	3	250	200	0	–	–	–	–	–	–	–	–	–
$t=7$ ($d=1$)	1	150	150	150	22.91	4.85	23.60	3.50	24.16	3.56	23.62	**0.68**	**0.69**
	2	250	190	10	23.75	4.35	23.96	3.35	23.80	3.80	23.91	0.62	0.63
	3	250	10	190	23.75	4.35	22.12	4.12	24.21	3.53	23.97	0.61	0.62
$t=8$ ($d=2$)	1	150	150	150	22.91	4.85	22.78	3.84	24.16	3.56	23.35	0.50	0.51
	2	250	200	0	23.52	4.49	23.28	3.63	–	–	23.47	0.50	0.51
	3	250	190	10	23.52	4.49	23.18	3.68	23.80	3.80	23.45	0.50	0.51
$t=9$ ($d=3$)	1	150	150	150	22.91	4.85	22.78	3.84	24.16	3.56	23.35	0.25	0.26
	2	250	200	0	23.52	4.49	23.28	3.63	–	–	23.47	0.25	0.26
	3	190	250	10	22.93	4.84	23.68	3.46	23.80	3.80	23.43	0.25	0.26

Parameters: $S_1 = S_2 = S_3 = 5$, $Q_1^{\min} = Q_2^{\min} = Q_3^{\min} = 10$, $Q_1^{\max} = Q_2^{\max} = Q_3^{\max} = 250$
$h_1 = 0.5$, $h_2 = 0.4$, $h_3 = 0.5$, $\delta = \gamma = 0.02$, and $r^{\max} = 3$. *Obj*: objective value of WDRA-2

7.4 Discussions

The auction mechanism presented in this chapter for solving the BLDP problems is in fact a special kind of interactive approach. Unlike the fuzzy interactive approaches presented in Chap. 5, players in this auction mechanism did not need to reveal their exact preference functions to each other, and the communications between them are performed via bids. Such an auction mechanism allows this bi-level decision making to be carried out in a remote, virtual, and cross organizational environment.

The auction mechanism is generally an interactive framework for BLDP problems. The optional decisions including winner determination and bid formulation have to be formulated based on the characteristics of the problem and the preferences of players.

This chapter presented the auction mechanism for bi-level problems only; nevertheless, the mechanism could be extended to problems with more than two levels. Considering a three-tier supply chain, each tier procures material or parts from its suppliers to make its own production. There are multiple entities in each tier excepting the first tier has only one entity. The decision interactions between the first tier decision unit and the second tier decision units are the same as the bi-level environment, while the interactions between the second and the third tiers are more complicated. Each decision unit in the second tier needs to determine the distribution of its orders to multiple suppliers in the third tier, and the units in the third tier has to allocate its production resources to fulfill the orders from multiple customers in the second tier. Winner determination decisions are made in sequence from the top (i.e. the first tier) and bids are submitted in sequence as well reversely from the bottom (i.e. the third tier). The decision process could be inefficient when it is difficult to reach an agreement between the top-down and bottom-up decisions.

Chapter 8
Metaheuristics for Multi-level Optimization

For the intrinsic complexity of multi-level programming problems, metaheuristic algorithms, such as genetic algorithm (GA), particle swarm optimization (PSO) and tabu search, have been used to solve the problems. Based on the strategies used, Talbi (2013) classified approaches in this domain into four types as shown in Fig. 8.1:

- Nested sequential approach
  The lower-level decision-making problem is solved in a nested and sequential manner to evaluate the solutions provided by the upper-level decision-maker. This approach is further divided to two types: repairing approach and constructing approach. The repairing approach treats the lower-level problem as constraints and solves it in the evaluation step, while the constructing approach applies improving algorithm for each level sequentially until meeting a stopping criterion.
- Single-level transformation approach
  The original multi-level optimization problem is reformulated as a single-level optimization problem, and then metaheuristics are applied to solve the single-level problem.
- Multi-objective approach
  The original multi-level optimization is transformed to a multi-objective optimization problem, and multi-objective metaheuristics are then used to solve the problem.
- Co-evolutionary approach
  Metaheuristics are used to solve different levels of the problem, and they co-evolve in parallel and exchange information.

C.-B. Cheng et al., *Fuzzy and Multi-Level Decision Making: Soft Computing Approaches*, Studies in Fuzziness and Soft Computing 368,
https://doi.org/10.1007/978-3-319-92525-7_8

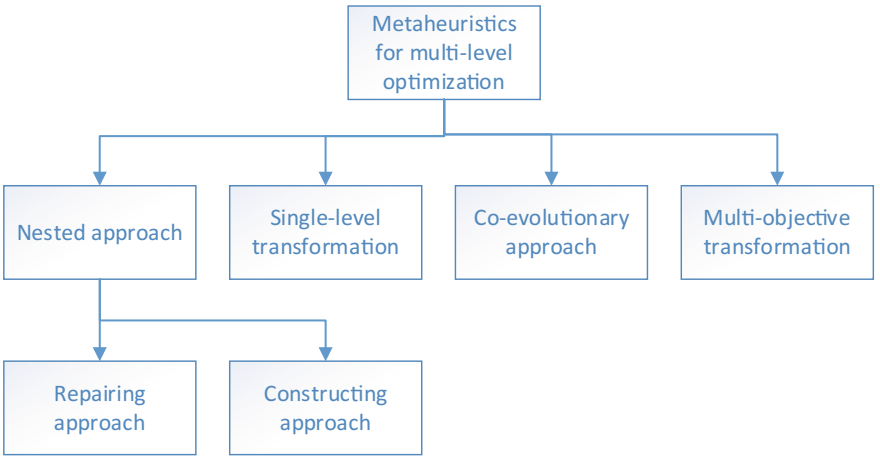

Fig. 8.1 Taxonomy of metaheuristics for multi-level programming (Talbi 2013)

8.1 Metaheuristics

Three popular metaheuristics are introduced in this section, including the genetic algorithm, particle swarm optimization and tabu search technique.

8.1.1 Genetic Algorithm

Genetic algorithm (GA), first proposed by Holland (1975), is a derivative-free stochastic optimization approach based on the concept of biological evolutionary processes. GA encodes each point in a solution space into a binary bit string called a chromosome, and each chromosome is evaluated by a fitness function, which corresponds to the objective function of the original problem. Usually, GA keeps a pool of chromosomes at the same time, and these chromosomes can evolve with the operations of selection, crossover, and mutation. After a number of generations, the population will contain, hopefully, chromosomes with better fitness values. Even under the best conditions, only local optimal solution can be expected. The GA procedure includes the following steps.

Encoding

Assume the solution space $\boldsymbol{x} = (x_1, \ldots, x_p)^T$ is a real vector and is transformed into binary strings as the example shown below:

The length of the string depends on the required precision. For example, if the required precision for x_1 is three places after the decimal point and the feasible region for x_1 is $[b_1^l, b_1^u]$, then the number of bits N^b required to represent x_1 is determined as follows:

x_1	x_2	…	x_p
1001	0101	…	1100

$$2^{N_b-1} < \left(b_1^u - b_1^l\right) \times 10^3 \leq 2^{N_b} - 1$$

Fitness Evaluation

The next step after the chromosomes are generated is to calculate the individual fitness value of each chromosome. The fitness function is usually design to comply with the objective function of the original optimization problem.

Selection

Selection operation mimics the principle of *survival of the fitness*. The purpose is to choose chromosomes from the current generation to produce offspring for the next generation based on probability. Those chromosome which have higher fitness values will be chosen with higher probabilities. The basic form of this operation is the so-called *roulette wheel* approach. The selection probability for each chromosome in the approach is set equal to:

$$\pi_h = \frac{f_h}{\sum_{r=1}^{N_c} f_r} \tag{8.1}$$

where f_h is the fitness value of the h-th chromosome and N_c is the total number of chromosomes in the current generation. A cumulative probability for each chromosome is then calculated by

$$Q_h = \sum_{r=1}^{h} f_h \tag{8.2}$$

The selection procedure is operated by randomly generating a number d within [0, 1], and if $d \leq Q_1$ then select the first chromosome; otherwise, select the h-th chromosome such that $Q_{h-1} \leq d \leq Q_h$. This procedure is replicated N_c times.

Crossover

The purpose of crossover is to generate new chromosomes that we hope will retain good features such as with higher fitness from the previous generation. This procedure is carried out by selecting pairs of parent chromosomes with a probability equal to a given crossover rate. A chromosome is chosen for crossover when the random number generated for it is less than or equal to the crossover rate. A basic crossover operation called one-cut-point method is commonly used. This method sets a crossover point on the chromosome strings randomly and two parent chromosomes are interchanges at this point.

Mutation

The operation of mutation creates a new chromosome which is very different from the current gene pool; therefore, it can provide a new search direction and prevent the population from converging to a local optimum prematurely. This operation is carried out by flipping bits of the chromosome strings randomly. A bit is chosen to flip if the random number generated for it is less than or equal to a predefined mutation rate.

The algorithm repeats with the process of selection, crossover, mutation, and fitness evaluation until no significant improvement can be obtained.

8.1.2 Particle Swarm Optimization

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Eberhart and Kennedy (1995), which was inspired by social behaviour of bird flocking or fish schooling. A PSO algorithm works by having a population, i.e. a swarm, of candidate solutions called particles to move around in the search-space to find the optimal solution of the concerned problem. The movements of the particles are guided by their own best known position in the search-space and the entire swarm's best known position.

Similar to GA, the solutions of the original problem are encoded as particles in PSO as well. The format of the particle is generally determined by the user depending on the problem on hand. The optimality of a particle is evaluated by its fitness function, which again is designed to comply with the original objective function. Each particle memorizes its best fitness and position, called individual best and denoted by *pbest*; the best fitness among all individual best is called global best and is denoted by *gbest*. Each particle updates its position based on the current pbest and gbest according to the rule suggested by Eberhart and Kennedy (1995). Let x_k^τ be the position of the k-th particle at the τ-th generation. The particle is updated by:

$$x_k^{\tau+1} = x_k^\tau + v_k^\tau \tag{8.3}$$

where v_k^τ is the moving velocity of the particle and is computed by

$$v_k^\tau = v_k^{\tau-1} + \lambda_1 Rand()(x_k^{pbest} - x_k^\tau) + \lambda_2 Rand()(x^{gbest} - x_k^\tau), \tag{8.4}$$

where x_k^{pbest} and x_k^{gbest} are the positions of the individual best and the global best respectively, λ_1 and λ_2 are acceleration constants used to regulate the impact between the individual best and the global best, and $Rand()$ is a random number in [0,1] to enable the diversity of particles.

The steps of the PSO algorithm are as follows.

Step 0 Initial particle generation:
Initialize generation counter, $\tau \leftarrow 0$.
Determine number of particles.
Step 1 For each particle do
Fitness evaluation.
Update individual best *pbest*.
Update position according to Eqs. (8.3) and (8.4).
Step 2 Update global best *gbest*.
Step 3 If the stop criterion is not satisfied,
$\tau \leftarrow \tau + 1$, and go to *Step 1*; otherwise go to *Step 4*.
Step 4 Select *gbest* as the solution of the problem.

8.1.3 Tabu Search

The concept of tabu search (TS) is to guide alocal heuristic search procedure to explore the solution space beyond local optimality. TS techniques were developed by Glover (1986, 1989), and are mainly used to solve combinatorial optimization problems. Application domains of these techniques include graph coloring (Hertz and de Werra 1987), traveling salesman problem (Malek et al. 1989), path assignment (Anderson et al. 1993), flow shop sequencing (Barnes and Laguna 1993; Widmer and Hertz 1989), job shop scheduling (Dell'Amico and Trubian 1993; Widmer 1991), and neural networks learning (de Werra and Hertz 1989).

The principle of tabu search is described by Glover et al. (2008) as follows. Similar to neighborhood search, TS proceeds iteratively from one point (solution) to another until a predefined termination criterion is met. Each solution x has an associated neighborhood $N(x) \subset X$, and each solution $x\prime \in N(x)$ is reached from x by an operation called a *move.* TS accepts moves that deteriorate the current objective function value but the moves are chosen from a modified neighborhood $N^*(x)$. Short-term and long-term memory structures are foundations of the composition of $N^*(x)$. TS uses attributive memory for guiding the formation of $N^*(x)$. Rather than recording full solutions, attributive memory keeps records of solution attributes that change in moving from one solution to another. The most commonly used memory structure in TS is the recency-based memory. This type of memory keeps track of solutions attributes that have changed during the recent past. To exploit this memory, attributes that found in recent solutions are labeled as *tabu-active*, and solutions that contain tabu-active elements become tabu. In the TS strategies based on short-term considerations, $N^*(x)$ is a subset of $N(x)$, and the tabu list serves to identify elements of $N(x)$ excluded from $N^*(x)$.

A fundamental component of tabu search methods that utilizes is a recency-based memory is called Simple Tabu Search in Glover (1989). Simple Tabu Search is a local search procedure, where the procedure initiates from any feasible solution x' in X,

and a neighborhood $N(x')$ is defined for each x'. The move from x' to a neighboring solution x'' in $N(x')$ is obtained by optimizing the objective function $f(x)$ over $N(x')$. As discussed earlier, a move to a solution x'' that is worse than the current solution x' may be accepted. Such a strategy enables the chance of escaping from local optima, but, on the other hand, might be trapped in a cycle. A short term memory is introduced, by embedded in a tabu list T; for example, T can be designed as the past k moves, where a move remains tabu only during k iterations, so that whenever a move $x' \rightarrow x''$ is made, the opposite move $x'' \rightarrow x'$ is added to the end of T, with the oldest move in T being removed.

An important feature of the Simple Tabu Search is the use of an aspiration level $A(f(x))$, of which the value depends on a specified move and/or the value of objective function f. As the searching path may be altered by the effect of T, some solutions might be forbidden but would be generated from the present solution by a tabu move. Without limiting too much the degree of freedom in the choice of a solution in $N(x')$, canceling the tabu status of a move may be acceptable. Such a case would be accepted if solution x'' obtained from x' is sufficiently good. This intention can be implemented by comparing the objective value of x'' with the aspiration level, i.e. if $f(x'') > A(f(x'))$, the move is accepted.

The stop criterion of the algorithm can set as: (1) significant improvement of the solution is not observed after a fixed maximum number of consecutive iterations is reached, or (2) a closer estimation of the maximum value in the objective function f is achieved.

8.2 Bi-level Optimization Problem Solved by Genetic Algorithm[1]

GA has been applied to solve multi-level programming problem by a few researchers. Liu (1998) developed a GA for multilevel programming with multiple followers. Oduguwa and Roy (2002) applied GA to solve a BLP aiming to encouraging the cooperation between two players. Kuo and Han (2011) modeled a supply chain distribution problem as a BLP and proposed a solution procedure based on GA and PSO. Lin et al. (2009) presented a modified GA to solve a BLP network design problem. Li (2015) proposed a genetic algorithm with a novel coding scheme and global convergence to solve nonlinear BLP problems.

When applying GA to solve a transformed single-level problem from the original multi-level problem, the algorithm generally works like it is used to solve a regular single-objective optimization problem. However, GA with its stochastic search ability could provide advantages in exploring solutions that are unable to be found by corner search approaches if the interactive and hierarchical nature of the problem is well utilized. Recall the multi-sourcing problem modeled by BLDP and solved by reverse auction in Sect. 7.3 of Chap. 7. In the presented solution procedure, the upper-level

[1] Part of the content of this section is reproduced from our previous work (Cheng et al. 2011).

decision maker alters the quantity allocation to suppliers at each iteration in an attempt to obtain a new bid combination which has a lower average price than the previous one. Such a quantity reallocation is solved by a corner-search based linear programming solver. Thus, the solutions obtained by the algorithm always adhered to the quantity boundaries perceived by the buyer. When these quantity boundaries are not coincided with suppliers' ranges of economic production quantity, such a method would result in inefficient quantity allocation. It is noted that suppliers have their ranges of economic production quantity which yield less production costs.

Cheng et al. (2011) attempted to improve the solution quality of the multi-sourcing problem by incorporating a stochastic search in the algorithm. In particular, the winner determination problem at the upper level is solved by a genetic algorithm which renders the opportunity to find quantities within a supplier's economic production range by a guided random search. At each iteration, the quantity allocation that returns a lower average bid price, which implies less production costs at the supplier side, would be retained and thus guides the quantity allocation approaching the economic quantities of suppliers. In addition, the genetic algorithm is more efficient in solving a larger scale problem, i.e. more suppliers involved in the auction. This genetic algorithm is formulated as follows, where the nomenclatures have been defined in Sect. 7.3 are used here:

BLDP-GA Algorithm

Step 0: Set $t = t'$

Step 1: Set generation counter $g = 0$
Randomly generate a pool of chromosomes, where the r-th chromosome represents a $q(r) = [q_1(r),\ldots, q_n(r)]$, denoting the quantities allocated to n suppliers.
Conduct feasibility checking of $q(r)$, $\forall r$.
Let G be the total number of chromosomes in the pool.

Step 2: Set chromosome counter $r = 1$

Step 3: Solve the bid determination problem with $q_i(r)$, $i = 1,\ldots, n$, and t for all n suppliers to obtain $p_i(r)$, $i = 1,\ldots, n$

Step 4: Compute the fitness $\alpha(r) = \min\{\mu_{\tilde{p}}(\bar{p}), \mu_d(d)\}$.

Step 5: Set $r = r + 1$
If $r > G$, then go to Step 6; otherwise go to Step 3.

Step 6: Let $h = \arg\max_r\{\alpha(r)\}$, and
$\alpha^*(g) = \alpha(h), q^*(g) = [q_1(h), \ldots, q_n(h)], p^*(g) = [p_1(h), \ldots, p_n(h)]$

Step 7: If stop criteria satisfied then go to Step 10, otherwise go to Step 8

Step 8: Chromosome operations.
Step 8.1: Reproduction
Step 8.2: Crossover
Step 8.3: Mutation
Step 8.4: Feasibility checking of $q(r)$, $\forall\ r$.

Step 9: Update G.
Set $g = g + 1$
Go to Step 2

1	2	3	4	…	n
01011010	00000000	01001001	00010100	…	1100100

Fig. 8.2 Solution represented by a chromosome in the genetic algorithm

Step 10: Let $l = \arg\max_g\{\alpha^*(g)\}$, and
$\alpha^{**}(t) = \alpha^*(l), q^{**}(t) = q^*(l), p^{**}(t) = p^*(l)$
Reset g
Set $t = t + 1$.
Step 11: If $\mu_d(d) \leq 0$ then go to Step 12; otherwise go to Step 1
Step 12: Let $s = \arg\max_t\{\alpha^{**}(t)\}$, and
$\alpha^{***} = \alpha^{**}(s), q^{***} = q^{**}(g), p^{***} = p^{**}(s)$
Stop.

As shown, the algorithm commences by setting the requested delivery time equal to the initial time demanded by the buyer. In Step 1, the initial solutions of quantity allocation are randomly generated in the form of chromosomes. The form of the chromosome is a string with n sections where the binary sub-string inside each section represents an allocated quantity as demonstrated by the example in Fig. 8.2. This exemplified chromosome indicates that 90 units are allocated to the first supplier, and the second supplier is not chosen in this demand allocation.

The feasibility of the solutions is examined according to Eqs. (8.8) and (8.9). The solutions are adjusted when violating the constraints. This adjustment is achieved in the following manner. If a $q_i(r) > Q_i^{\max}$ then it is reduced to $Q_i^{\max}$; and if a $q_i(r) < Q_i^{\min}$ and $q_i(r) > 0$ then it is increased to $Q_i^{\min}$. If $\sum_{i=1}^n q_i \neq Q$ then there are two possible cases:

Case 1, $\sum_{i=1}^n q_i(r) > Q$ then randomly decrease some $q_i(r)$ until $\sum_{i=1}^n q_i(r) = Q$; and

Case 2, $\sum_{i=1}^n q_i(r) < Q$ then randomly increase some $q_i(r)$ until $\sum_{i=1}^n q_i(r) > Q$.

The solution of quantity allocation is passed to suppliers and each supplier employs Step 3 to solve their production planning problem and submit an appropriate bidding price $p_i(r)$. The overall satisfactory degree serves as the fitness function of GA and is computed in Step 4. All chromosomes in the pool are evaluated one by one and then the best solution in the current generation is recorded as shown in Steps 5 and 6. Step 7 tests the stop criteria of the genetic algorithm. The stop criteria can be the maximum number of generation or a convergence assessment. If the stop criteria are not satisfied then the GA performs the operations of chromosomes, including reproduction, crossover, and mutation, to generate new chromosomes (Step 8). The reproduction operation is achieved by tournament selection (Goldberg 1989), where pairs of chromosomes in the pool are randomly picked and in each pair the one with greater fitness is reproduced. For crossover operation, the mask crossover method of Syswerda (1989) is used. A pair of chromosomes in the pool is randomly picked to act as the parents, and a binary string of the same length as the parent chromosomes is

A: 011001010100 → C: 011001110101
Mask: 000100100001
B: 101011110001 → D: 101011010000

Fig. 8.3 Example of mask crossover operation

Table 8.1 Performance comparison between Cheng's (2008) and BLDP-GA, Parameter settings: $Q_1^{\min} = Q_2^{\min} = Q_3^{\min} = 10,\ Q_1^{\max} = Q_2^{\max} = Q_3^{\max} = 250$

	IRA	BLDP-GA
q_1	150	192
q_2	150	10
q_3	150	248
t	7	7
$\bar{p}$	23.59	23.51
α	0.68	0.70

also randomly produced to serve as a mask. The parent chromosomes exchange their digits at positions where there is a 1 in the mask string, thus creating two offspring chromosomes. This procedure is illustrated in Fig. 8.3, where two parent chromosomes A and B exchange their digits at the 4th, 7th and 12th positions, producing offspring chromosomes C and D. The mutation operation is done by flipping a randomly chosen bit, i.e. randomly choosing a chromosome from the pool, randomly picking a bit inside this chromosome and then flipping the digit of this bit to create a new chromosome. The BFRA is then solved again with the newly generated chromosomes (Step 9). After the GA terminates, Step 10 records the best solution among all generations and uses it as the best (i.e. cheapest) allocation pattern under the current delivery time constraint. The delivery time is then increased by one unit of time to search for potentially improved solutions. Step 11 terminates the algorithm under the condition that $\mu_d(d) \leq 0$, i.e. the delivery time is absolutely unacceptable to the buyer, and thus no merit exists in further extending the delivery time to search for a cheaper solution. In the final step (Step 12), the optimum solution to the reverse auction problem is determined by comparing all the solutions found in Step 10 during the course of the iterations.

By using the same data of the multi-sourcing problem in Chap. 7, where the total demand is 450 units and three suppliers involved, the resulting performance comparison between the IRA algorithm in Chap. 7 and the BLDP-GA algorithm presented above is given in Table 8.1. As shown, the BLDP-GA is able to find a lower average unit price for the buyer, and the overall satisfactory degree α resulted from this solution is greater than that of IRA. It is noted that the quantity allocation obtained by IRA is 150 for each supplier. This solution is resulted from an initial setting of equal allocation in IRA, whereas BLDP-GA found a better solution by stochastic search.

To evaluate the performance of BLDP-GA under larger problem sizes, Cheng (2011) set five levels of problem sizes, namely 5, 10, 20, 40, and 80 suppliers, and compared the performances of BLDP-GA under different problem sizes. Relevant data were generated in the following manner. At the buyer side, $Q=450$, $\bar{p}^{\inf} = 22$, $\bar{p}^{\sup} = 27$, $t'=6$, $d^{\sup}=4$, $Q_i^{\min} = 10$, $\forall\, i$, $Q_i^{\max} = 250$, $\forall\, i$, and $S_i=5$, $\forall\, i$. At the supplier side, the bidding price preferences of each supplier are generated in a random fashion as: $p_i^{\inf} = rand\left(\bar{p}^{\inf} \times 90\%, \bar{p}^{\inf} \times 110\%\right)$, and $p_i^{\sup} = rand(\bar{p}^{\sup} \times 90\%, \bar{p}^{\sup} \times 110\%)$, where $rand(a, b)$ denotes a random number between a and b. The above data generation scheme is to enable an overlap of the price range between the suppliers and the buyer. The profit preference of Supplier 1 is fixed by $\pi_1^{\inf} = 3$ and $\pi_1^{\sup} = 6$, and the preferences of the remainder suppliers are generated based on Supplier 1's preference as $\pi_i^{\inf} = rand(\pi_1^{\inf} \times 80\%, \pi_1^{\inf} \times 120\%)$ and $\pi_i^{\sup} = rand(\pi_1^{\sup} \times 80\%, \pi_1^{\sup} \times 120\%)$. The cost data of Supplier are also fixed as $c_{11} = 17$, $c_{12} = 18.22$, $c_{13} = 22$, and $c_{14} = 24.5$, and the cost data of the remainder suppliers are also generated based on these data as $c_{ij} = rand(c_{1j} \times 85\%, c_{1j} \times 115\%)$, $i=2,\ldots, n$, $j=1,\ldots, 4$. The production status of each supplier includes its initial inventory, the MPS scheduled in each period, and the orders committed to customers in each period. In our experiments we considered a planning horizon of 12 periods. The initial inventory of each supplier is generated by $I_i^0 = rand(0, 100)$. The MPS is generated period by period; first, we randomly determined if an MPS is assigned to a period; if yes, then a production lot of 90 units is scheduled to that period, otherwise, it is set to 0. The committed customer order of each period is determined by taking a random integer between 0 and 50 units. The regular and overtime capacities of each supplier for the 12 periods are randomly generated as given in Table 8.2. The rationale behind this data generation is that the later periods have more abundant capacities since they are farther from the planning point and thus not yet fully utilized.

The experiment was run for 10 replications, and the average computational results are shown in Table 8.3. It is seen that the computational times did not increase dramatically when problem size grew as shown in Fig. 8.4a, which indicates that the proposed algorithm is efficient in solving this problem. An ANOVA analysis proves that there are significant differences between the average unit prices ($\bar{p}$) of different problem sizes (with p-value 0.00). In particular, the average unit price decreases when the number of suppliers increases, as shown in Fig. 8.4b. This result is reasonable for the buyer being able to find more beneficial suppliers when there are abundant suppliers in the pool.

8.3 Bi-level Optimization Problem Solved by Particle Swarm Optimization

Zhao and Gu (2006) used a modified PSO algorithm to solve the bi-level programming problem. Their approach is introduced next.

Table 8.2 Capacity data generation scheme

	Period											
	1	2	3	4	5	6	7	8	9	10	11	12
R	0	0	0	*rand* (0,30)	*rand* (0,50)	*rand* (10,50)	*rand* (20,70)	*rand* (30,70)	*rand* (40,70)	*rand* (0, 50)	*rand* (50,80)	*rand* (60,90)
O	*rand* (0,5)	*rand* (0,5)	*rand* (0,10)	*rand* (0,20)	*rand* (10,30)	*rand* (10,30)	*rand* (10,30)	*rand* (10,30)	*rand* (20,30)	*rand* (20,30)	*rand* (20,30)	*rand* (20,30)

R: regular capacity
O: overtime capacity

Table 8.3 Effects of problem sizes

Number of suppliers	$\bar{p}$	α	Computational time (s)[a]
5	23.76	0.65	45.71
10	22.98	0.80	68.05
20	22.56	0.89	88.27
40	22.26	0.95	107.69
80	22.10	0.97	134.83

[a]experiments was run on a PC with an Intel® Core(TM)2 Duo CPU E8400 @3.00 GHz and 1.96 GB RAM

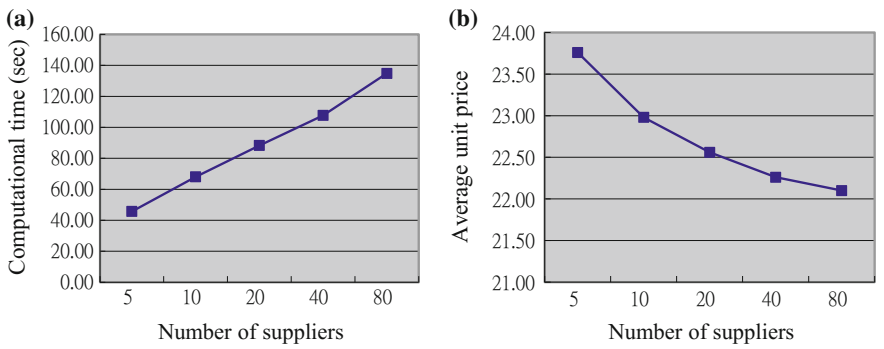

Fig. 8.4 **a** Computational time under different problem sizes, **b** Average unit prices under different number of suppliers

The bi-level programming model is given below.

$$\begin{aligned}
&\min_x F(x, y) \quad \text{(Upper level)}\\
&\text{Subject to:}\\
&\quad G(x, y) \leq 0\\
&\quad H(x, y) = 0\\
&\text{where } y \text{ solves:}\\
&\quad \min_y f(x, y) \quad \text{(Lower level)}\\
&\quad \text{Subject to:}\\
&\quad\quad g(x, y) \leq 0\\
&\quad\quad h(x, y) = 0
\end{aligned}$$

The upper-level objective function, $F(x, y)$, is constrained by inequality constraints $G(x, y)$ and equality constraints $H(x, y)$. $F(x, y)$ is also constrained by a lower-level decision y that minimizes $f(x, y)$, with inequality constraints $g(x, y)$ and equality constraints $h(x, y)$.

The population of PSO is called a *swarm* and each individual in the population of PSO is called a particle. The move of the individual particle at each generation follows rules similar to those presented in Eqs. (8.3) and (8.4), but with modification. The rules used by Zhao and Gu (2006) are as follows.

Assume the current position in an N-dimensional search space $X_i^k = \left(x_1^k, \ldots, x_n^k, \ldots, x_N^k\right)$, where $x_n^k \in [l_n, u_n]$, $1 \leq n \leq N$, l_n and u_n is lower and upper bound for the n-th dimension, respectively, and the current velocity V_i^k, $V_i^k = \left(v_1^k, \ldots, v_n^k, \ldots, v_N^k\right)$, which is bounded by a maximum velocity $V_{\max}^k = \left(v_{\max,1}^k, \ldots, v_{\max,n}^k, \ldots, v_{\max,N}^k\right)$ and a minimum $V_{\min}^k = \left(v_{\min,1}^k, \ldots, v_{\min,n}^k, \ldots, v_{\min,N}^k\right)$. In each iteration of PSO, the swarm is updated by the following equations (Kennedy and Eberhart 1995):

$$V_i^{k+1} = \omega V_i^k + c_1 r_1 \left(P_i^k - X_i^k\right) + c_2 r_2 \left(P_g^k - X_i^k\right) \tag{8.5}$$

$$X_i^{k+1} = X_i^k + V_i^{k+1} \tag{8.6}$$

where P_i is the best previous position of the i-th particle (i.e. *pbest*) and P_g is the best position among a set of particles. According to different definitions of P_g, it can be the best position among all the particles in the swarm (i.e. *gbest*) as defined in Sect. 8.1.2, or it can be taken from some smaller number of adjacent particles of the population (also known as *lbest*). r_1 and r_2 are elements from two uniform random sequences: $r_1 \sim U(0, 1)$ and $r_2 \sim U(0, 1)$. The variables c_1 and c_2 are acceleration constants, which control how fast a particle will move in a single iteration. ω is a linearly varying inertia weight over the generations that was proposed by Shi and Eberhart (1997) and was reported to significantly improve the performance of PSO. The inertia weight factor ω is defined as follows:

$$\omega = \omega^{\max} - iter \times \frac{\omega^{\max} - \omega^{\min}}{iter^{\max}} \tag{8.7}$$

where $\omega^{\max}$ and $\omega^{\min}$ are the maximum and minimum of ω, and $iter$ is the current iteration with $iter^{\max}$ as the maximal number of iterations. P_i and P_g can be updated continually during the course of iterations and the final P_g is regarded as the best solution provided by PSO.

To avoid the premature convergence of the algorithm, the convergence status of the swarm is measured by the variance of the population's fitness (Zhensu and Zhirong 2004). This measure $\left(\sigma^2\right)$ is defined as follows.

$$\sigma^2 = \frac{1}{m} \sum_{i=1}^{m} \left[\frac{f(X_i) - f_{\text{avg}}}{f_d} \right]^2 \tag{8.8}$$

where m is the population size, X_i is the position of the i-th particle, $f(X_i)$ is the fitness of X_i, and f_{avg} is the average fitness of all particles. The parameter f_d is used to constrain the value of σ^2, and is determined by:

$$f_d = \begin{cases} \max_i \left\{ \left| f(X_i) - f_{\text{avg}} \right| \right\}, if \max_i \left\{ \left| f(X_i) - f_{\text{avg}} \right| \right\} > 1 \\ 1, \text{otherwise} \end{cases} \tag{8.9}$$

When the algorithm is premature convergence and thus the P_g is at a local optimum, an artificial adjustment on P_g may push the particles shifting their directions and entering into a new searching space. This can be achieved by introducing a random mutation operator on P_g with a certain probability (*prob*) that is determined by:

$$prob = \begin{cases} \lambda, if \sigma^2 < \sigma_d^2 \text{ and } \left| f\left(P_g\right) - f^* \right| > 1 \\ 0, \text{otherwise} \end{cases} \tag{8.10}$$

where λ is a constant in the range of [0, 1], σ^2 is the variance defined by Eq. (8.8) with σ_d^2 as its threshold, $f(P_g)$ is the fitness of P_g, f^* is the theoretical optimum of the problem, and $\delta > 0$ is a convergence precision. The activation of the mutation on P_g is based on the probability computed by Eq. (8.10), and when it is activated, P_g is disturbed by the following rule:

$$P_g \leftarrow P_g \times (1 + \eta) \tag{8.11}$$

where η is a normal random number, i.e. $\eta \sim N(0, 1)$.

The steps of the modified PSO algorithm by Zhao and Gu (2006) is as follows.

Step 0. Randomly initialize the position (X_i) and velocity (V_i) of particle i, $\forall\, i$.
Step 1. For each particle i, set $P_i = X_i$, and set $P_g = X_i^*$ (the current best position among the particles in the swarm).
Step 2. For each particle i in the swarm,
  Step 2.1 Update velocities V_i and positions X_i by Eqs. (8.5)–(8.7)
  Step 2.2 Calculate the fitness of the current particle, $f(X_i)$
  Step 2.3 If $f(X_i) > P_i$, then set $P_i = X_i$
  Step 2.4 If $f(X_i) > P_g$, then $P_g = X_i$.
Step 3. Calculate σ^2 according to Eqs. (8.8) and (8.9)
Step 4. Calculate the mutation probability *prob* based on Eq. (8.10)
Step 5. Generate a random number $r \in [0, 1]$, if $r \leq prob$, update P_g by Eq. (8.11).
Step 6. If the stop criterion is satisfied, go to Step 7; otherwise go to Step 2.
Step 7. Stop the algorithm, and output the best solution, P_g.

Zhao and Gu (2006) solved the BLP problem by an interactive approach called Bilevel Iterative Algorithm based on PSO (PSO-BLIA), where the upper-level model is solved by using the modified PSO algorithm while the lower-level model is solved by conventional optimization methods. The steps of PSO-BLIA are described as follows.

Step 0. Initialization.
Randomly generate feasible solutions of the lower-level model.
Determine the swarm size m for the upper-level model. Randomly initialize the position (X_i) and velocity (V_i) of particle i, $\forall\ i$.
Step 1. For each particle i, set P_i=X_i, and set $P_g = X_i^*$ (the current best position among the particles in the swarm).
Step 2. For each particle i in the swarm,
Step 2.1 Update velocities V_i and positions X_i by Eqs. (8.5)–(8.7)
Step 2.2 Solving the lower-level model.

Given X_i, solve the lower-level model by conventional optimization methods, and obtain the solution y_i^*, $\forall\ i$.

Step 2.3 Compute the fitness for the upper-level mode, $F(X_i, y_i^*)$, $\forall\ i$.
Step 2.4 Update P_i.

If $F(X_i, y_i^*) < F(P_i, y_{P_i})$, then $P_i \leftarrow X_i$ and $y_{P_i} \leftarrow y_i^*$, where y_{P_i} is the best solution for the lower-level model given P_i.

Step 2.5 Update P_g.
If $F(X_i, y_i^*) < F(P_i, y_{P_g})$, then $P_i \leftarrow X_i$ and $y_{P_g} \leftarrow y_i^*$, where y_{P_g} is the best solution for the lower-level model given P_g.
Step 3. If the stop criterion is satisfied, go to Step 5; otherwise go to Step 4.
Step 4. Mutation operation.
Perform disturbance to P_g based on Eqs. (8.10) and (8.11). Solve the lower-level model with P_g and obtain the solution y_{P_g}. Go to Step 2.
Step 5. Stop the algorithm, and output the best solution, P_g and y_{P_g}.

8.4 Bi-level Optimization Problem Solved by Tabu Search

Wen and Huang (1996) employed a simple tabu search to solve a mixed-integer linear bi-level programming problem. Their approach is introduced in this section.

Let $x = (y, z)$ be the vector of decision variables, where y is the subvector of integer variables determined by the higher-level, and z is the subvector of continuous variables determined by the lower-level problem. Then the mixed-integer linear bi-level programming problem is presented below:

MIBLP:

$$\max f(x) = cy + dz \tag{8.12}$$

s.t. y is a zero-one vector

where z solves: (8.13)

$$\max g(z) = hz \tag{8.14}$$

$$\text{s.t.} \quad Ay + Bz \leq b, \tag{8.15}$$

$$z \geq 0, \tag{8.16}$$

in which $c \in \mathbb{R}^{n_1}, d, h \in \mathbb{R}^{n_2}, A \in \mathbb{R}^{m \times n_1}, B \in \mathbb{R}^{m \times n_2}$ and $b \in \mathbb{R}^m$. Let $X = \{(y, z) : Ay + Bz \leq b, y \in \{0, 1\}, z \geq 0\}$ denote the problem constraint region, and $W_g(X) = \{(\bar{y}, \bar{z}) \in X : (\bar{y}, \bar{z})\}$ denote the feasible region of the higher-level DM, where $(\bar{y}, \bar{z})$ is the optimal solution to the lower-level problem when y is fixed at $\bar{y}$.

The process of the tabu search algorithm is a succession of moves that transform a given feasible solution into an optimal or a near optimal one. A move s is first defined as:

$$s(y', z') = (y'', z''),$$

where y' is a vector of zero-one decision variables of the higher-level DM and is randomly generated, and z' is the resulting optimum of the lower-level problem given y', i.e. the optimal solution to the following linear program:

LPZ:

$$\begin{aligned} \max \quad & hz \\ \text{s.t.} \quad & Bz \leq b - Ay', \\ & z \geq 0. \end{aligned}$$

Let y'' be a vector obtained by flipping an element of y', e.g. $y'_j \leftarrow 1 - y'_j$. z'' is then obtained by solving the above linear programming with the higher-level decision fixed at y''. In this setting, $x'' = (y'', z'')$ is an adjacent point of x', and thus the neighborhood of x' is defined as:

$$N(x') = \{x'' \in X : s(x') = x''\}.$$

To avoid reversal moves, two Tabu list T_0 and T_1 are constructed, each with a fixed length k, and both composed of the indices of the higher-level decision variables. T_0 contains the indices of variables that were changed from 1 to 0 in the past k iterations, while T1 contains the indices of variables changed from 0 to 1 in the past k iterations. A move becomes tabu if the altering variable belongs to T_0 or T_1. A variable that has been set to 0 is not allowed to become 1 until k other variables have been set to 0; similarly, a variable set to 1 is not allowed to be set to 0 until k other variables have been set to 1.

During the neighborhood search, if s is not a tabu move, then for any feasible solution x' in X, x'' is selected as an adjacent point of x' such that

$$cy'' + dz'' = max\{f(x'') : s(x') = x'', x'' \in N(x')\}.$$

The aspiration level $A(f(x))$ is set as the value of the higher-level objective by the current solution x'. Cancelation of the tabu status of a move becomes acceptable when the move satisfies the aspiration condition. It is also allowed the move to a worse solution, call a downhill move, in order not to trap in a local optimum. Let $NS = n_1$ denote the total number of generated starting points, and k represent their

counter. Furthermore, let $ND = n_1/2$ be the allowable number of downhill moves during a search path. Therefore, $y'_{[K]}$ indicates the higher-level decision by the k-th generated starting point, and $y''_{[i]}$ is the i-th point in the neighborhood of $x'_{[k]}$.

The simple tabu search algorithm developed by Wen and Huang (1996) to solve the MIBLP problem is presented below.

Step 0. (Initialization)
Set $NS = n_1$, $ND = n_1/2$ and set $k \leftarrow 1$.

Step 1. Set $i \leftarrow 1$ and $j \leftarrow 0$. If $k > NS$, go to Step 4; otherwise, go to Step 2.

Step 2. (Choice)
Step 2.1 Randomly generate $y'_{[k]}$ and solve LPZ to obtain $\left(y'_{[k]}, z'_{[k]}\right)$. If $k = 1$, compute $f^* = cy'_{[k]} + dz'_{[k]}$.
Step 2.2 If $i \leq n_1$, find $y''_{[i]}$ and solve LPZ to obtain $\left(y''_{[i]}, z''_{[i]}\right)$ and go to Step 2.3; otherwise, go to Step 2.4.
Step 2.3 If $x''_{[i]}$ is tabu and does not satisfy the aspiration condition, set $i \leftarrow i + 1$ and go to Step 2.2; otherwise, record $\bar{f} = cy''_{[i]} + dz''_{[i]}$, set $i \leftarrow i + 1$, and go to Step 2.2.
Step 2.4 Select $\left(y'_{[K]}, z'_{[K]}\right)$ with its high-level objective value being $\bar{\bar{f}} = max\left\{\bar{f}_{[I]} : i = 1, 2, \ldots, n_1\right\}$.

Step 3. (Update)
Step 3.1 Update tabu list T_0 and T_1, and the aspiration level $A(f(x))$.

If $\bar{\bar{f}} \leq f^*$, set $j \leftarrow j + 1$, go to Step 3.2; otherwise, set $f^* = \bar{\bar{f}}$, $i = 1$ and $i = 0$, and go to Step 2.2.

Step 3.2 If $j > ND$, set $k \leftarrow k + 1$, go to Step 1;. otherwise, set $i = 1$ and go to Step 2.2.

Step 4. (Termination)
The near-optimum is obtained with f^*; stop the algorithm.

8.5 Discussions

This chapter presents a taxonomy of the metaheuristic algorithm applied to the solving of multi-level programming problems. Three metaheuristic algorithms introduced are genetic algorithm, particle swarm optimization and tabu search. The applications demonstrated in this chapter generally adopt an interactive strategy and can be classified in the nested sequential approach.

The use of metaheuristic algorithms in solving the multi-level programming problems alleviates the difficulty of nonlinearity resulting from KKT transformation of the original problem, and also frees the solution procedure from the strict assumptions of optimality that troubles the KKT transformation approach.

The applications in this chapter all limit to bi-level programming problems. The application to solve problems with more than two levels is challenging. The nested structure of the solution procedure when using interactive approaches would make the implementation and computation of the algorithms much more complicated.

Chapter 9
Neural Networks for Solving Multi-level Programming

Artificial neural networks or, simply, neural networks (NNs) have been widely developed in solving mathematical programming (MP) or optimization problems. Among various NNs, a couple of networks have been utilized for optimization (Looi 1992; Kumar 2004), such as Hopfield neural networks (HNNs) (Hopfield 1982), self-organizing feature maps (Kohonen 1982), and Boltzmann machines (Ackley et al. 1985). Research studies on neurons and their networks for information processing have drawn much attention by scholars in the fields of physics, electronics, and electricity. They have tried to formulate the problems mimicking the behavior of biological NNs to abstract principles of computation employed by the brain. Hopfield (1982, 1984) made a momentous contribution for neural computation by solving a traveling salesman problem (TSP) and a linear programming (LP) problem (Hopfield and Tank 1985; Tank and Hopfield 1986). Although Shirazi and Yih (1989) pointed out some drawbacks on the approach, further improvements and verifications have been made by later investigators (Ricanek et al. 1999).

Technically, the HNN approach to optimization is to handle a dynamic system in which its energy function, or a Lyapunov function, characterizes the behavior of the networks and represents the problems to be solved. Its architecture can be realized by the use of an electronic circuit, possibly on a VLSI circuit, as an online solution with a parallel distributed process (Cichocki and Unbehauen 1993). These special characteristics are beneficial for real-time optimization and have been applied in many areas such as pattern recognition, scheduling, manufacturing, and numerous business applications (Looi 1992; Sharda 1994; Smith and Gupta 2000; Wong et al. 2000; Zhang and Huang 1995).

Due to renewed interests in NNs by extending HNNs, various NNs have been proposed. For instance, Kennedy and Chua (1988) extended the Tank and Hopfield's work to solve a nonlinear programming (NLP) problem. Rodríguez-Vázquez et al.

Part of the content of this chapter is reproduced from our previous works (Wen et al. 2009; Shih et al. 2004).

C.-B. Cheng et al., *Fuzzy and Multi-Level Decision Making: Soft Computing Approaches*, Studies in Fuzziness and Soft Computing 368,
https://doi.org/10.1007/978-3-319-92525-7_9

(1988, 1990) presented switched-capacitor NNs for solving LP and NLP problems. Later, Zhang and Constantinides (1992) introduced a Lagrange NNs for solving general NLP problems, and Cichocki and Unbehauen (1993) proposed the Lagrange multiplier method-based NN in solving the NLP problem. Xia et al. (2005) offered a primal and dual method for solving linear and quadric programming problems.

Shih et al. (2004) applied an HNN to sole the multi-level programming problems. The original multi-level problem is first transformed to a single-level problem by introducing the KKT optimality conditions as presented in Chap. 2. As mentioned in Chap. 2, the auxiliary problem becomes nonlinear due to the introduction of the complementary slackness conditions. Shih et al. (2004) then used a HNN to solve the transformed problem. The remainder of this chapter starts with the discussion on the application of HNNs to optimization problem, and then introduces the approach of Shih et al. (2004) in solving the multi-level problem with the HNN.

9.1 Definition of Lyapunov Function

Functions which make use of a continuous scalar function of the state vector are called a Lyapunov function. The Lyapunov function proves the stability of a certain fixed point in a dynamical system or autonomous differential equation. One must be aware that the basic Lyapunov Theorems for autonomous systems are a sufficient, but not necessary tool to prove the stability of an equilibrium system. Finding a Lyapunov Function in a certain equilibrium situation might be a matter of luck. Trial and error is the common method to apply when testing Lyapunov functions on some equilibrium systems.

Haykin (1999) provided a serious of theorems to define and supports us general formulation to illustrate the Lyapunov function as follows.

Theorem 9.1 *The equilibrium state* $\bar{\mathbf{x}}$ *is stable if in a small neighborhood of* $\bar{\mathbf{x}}$ *there exists a positive definite function V(x) such that its derivative with respect to time is negative semi-definite in that region.*

Theorem 9.2 *The equilibrium state* $\bar{\mathbf{x}}$ *is asymptotically stable if in a small neighborhood of* $\bar{\mathbf{x}}$ *there exists a positive definite function V(x) such that its derivative with respect to time is negative definite in that region. A scalar function V(x) that satisfies these requirements is called a Lyapunov function for the equilibrium state* $\bar{\mathbf{x}}$.

The function $V(\mathrm{x})$ is positive definite in the state space S, and for all x in S, it satisfies the following requirements:

(i) The function $V(\mathrm{x})$ has continuous partial derivatives with respect to the elements of the state vector x.
(ii) $V(\bar{\mathbf{x}}) = 0$.
(iii) $V(\mathrm{x}) > 0$ if $\mathrm{x} \neq \bar{\mathbf{x}}$.

Let $\bar{\mathbf{x}} = \mathbf{0}$ be an equilibrium state of the autonomous system $\dot{x} = f(x)$ and let $\dot{V}(x) = (\partial V/\mathrm{d}x)(\mathrm{d}x/\mathrm{d}t) = \nabla f(x)$ be the time derivative of the Lyapunov function V. There exist three types of stable equilibrium, such as:

(1) Stable equilibrium
If the Lyapunov function V is locally positive definite and the time derivative of the Lyapunov function is locally negative semi-definite, where $\dot{V}(x) \leq 0, \forall\, x \in S$ for some neighborhood space S, then the equilibrium is proven to be stable.

(2) Locally asymptotically stable equilibrium
If the Lyapunov function V is locally positive definite and the time derivative of the Lyapunov function is locally negative definite, where $\dot{V}(x) \leq 0,\ \forall x \in S \backslash \{0\}$ for some neighborhood space S, then the equilibrium is proven to be locally asymptotically stable.

(3) Globally asymptotically stable equilibrium
If the Lyapunov function V is globally positive definite and the time derivative of the Lyapunov function is globally negative definite, where $\dot{V}(x) \leq 0, \forall x \in \Re^n \backslash \{0\}$, then the equilibrium is proven to be globally asymptotically stable.

In many problems of the HNN, the energy function can be considered as a Lyapunov function. However, the existence of a Lyapunov function is sufficient but not necessary for stability. The Lyapunov function $V(\mathrm{x})$ provides the mathematical basis for the global stability analysis of the non-linear dynamical system. The global stability analysis is much more powerful in its conclusion than local stability analysis—that is, every globally stable system is also locally stable, but not vice versa.

9.2 Hopfield Neural Networks for Optimization

HNNs are considered as feedback (or recurrent) without learning from a pattern set (Zurada 1992). The HNNs consist of a set of neurons and a corresponding set of unit delays, forming a multiple loop feedback system. Each feedback loops is equal to a neuron. Basically, the output of each neuron is feedback, via a unit delay element, to each of the other neurons in the system.

HNNs are not affected by additional inputs, and each neuron is fully connected with the remaining neurons by weights and always updates the connection weights. The associative memory or content-addressable memory is a memory organization that accesses memory by its contents and it always searches for the closest matching from all stored prototypes (Du and Swamy 2006). This is analogous to the storage of information in an associative memory as biological neurons (Kohonen 1988). In other words, the process of association and information retrieval is simulated by the dynamical behavior of a highly interconnected system of nonlinear circuit elements with collective computational abilities (Hopfield 1982).

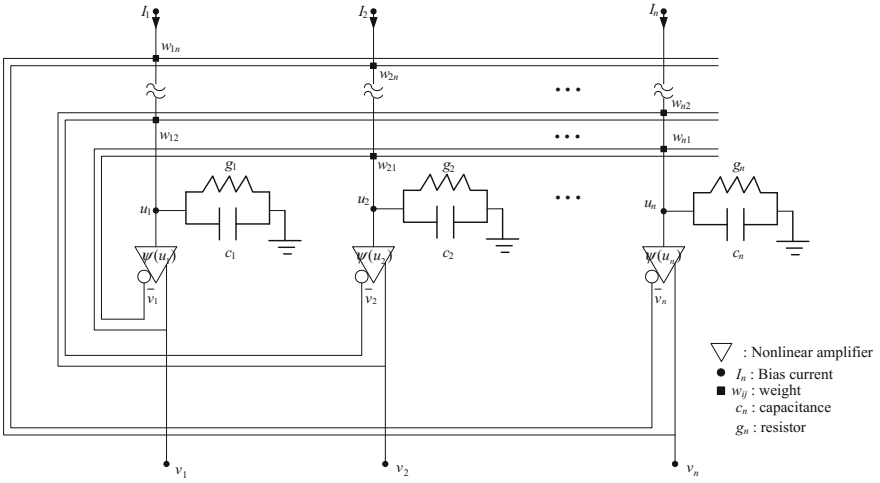

Fig. 9.1 Basic structure of HNNs

Due to the above characteristics, HNNs are easily realized for their capability of parallel computation as a result of being a fully connected property. The networks use electric circuits to simulate the actions of biological neurons, and the basic model can be implemented by interconnecting an array or resistors, non-linear amplifiers with symmetrical output, and external bias current as shown in Fig. 9.1. There are n neurons, and each neuron is represented by a resistor g_i, a capacitance c_i, and a non-linear amplifier with an activation function $\psi\ (u_i)$, $i=1, 2,\ldots, n$, respectively. The resistance-capacitance charging equation determines the rate change of u_i, $i=1, 2,\ldots, n$, where u_i is the input voltage of the amplifier. The input is mapped to the output voltage v_i or $\bar{v}_i$ (normal or invert) through the function $\psi\ (u_i)$, $i=1, 2,\ldots,n$, respectively. The choice of connecting the normal or invert output is dependent on the positive or negative value of the conductance or weight w_{ij}, $i=1, 2,\ldots, n$, $j=1, 2,\ldots, n$. The input of each neuron comes from two sources, external inputs I_i and inputs from other neurons with the interconnection strength or weights w_{ij} from neuron j to neuron i (Hopfield 1982). In addition, Hopfield has shown that the sufficient condition for a stable network is that its synaptic weights are symmetric, i.e., $w_{ji} = w_{ij}$ with $w_{jj} = 0$, and i.e., the system has a Lyapunov function (Hopfield 1982). Here, the weights are derived from the function, and the neurons stay transit according to the local information (i.e., feedback) until a steady state is reached (Burke and Ignizio 1992).

After a NN model is formulated, Hopfield and Tank (1985) tried to solve travelling salesman problems (TSPs) on a hardware circuit as illustrated above, and later they (Tank and Hopfield 1986) extended the concept to deal with LP problems. However, the model lacks the material technologies for further development (Hopfield 1988). Most of the later works are simulated on digital computers, not on the hardware implementation. From a practical viewpoint, HNNs have been applied to many areas

of mathematical programming problems, and we organize them in a systematic way. Interested readers can refer to Burke and Ignizio (1992), Cichocki and Unbehauen (1993), Looi (1992), Sharda (1994), Smith (1999), Widrow et al. (1994), and Zhang (2000) for details.

9.2.1 Discrete and Continuous Hopfield Networks

HNN is a recurrent NN, synaptic connection pattern, whereby there is a Lyapunov function for the activity dynamics (Hopfield 1982, 1984). In the original HNN model the state of the system evolves from any initial state to a final state where it is a (local) minimum of the Lyapunov function. Based on the output functions, HNNs can be classified into two popular forms: discrete and continuous time models.

9.2.1.1 Discrete Hopfield Networks

The actions of the neurons in discrete/discrete-time HNN can be illustrated as $y_j(k+1) = \psi\left[\sum_{j=1}^{n} w_{ij}x_j(k) - \theta_i\right]$, $i=1,\ldots, n$, $j=1,\ldots,n$, as in Fig. 9.2, where the unit delay z^{-1} is $y_j(k+1) = \psi\left[v_j(k+1)\right]$, $v_j(k+1) = \sum_{j=1}^{n} w_{ij}x_j(k) - \theta_j$, θ_i is an externally applied threshold, x_j is input neurons, and $v_j(k) = v_j(kT_s)$ ($k = 0, 1,\ldots,k$ denote the discrete time and T_s is the sampling period). In general, we always assume the sample period is normalized to unity ($T_s = 1$).

In the discrete HNN, the units use a bipolar output function where the states of the units or zero, i.e., the output of the units remain the same if the current state is equal to some threshold value: $x_i = 0$, if $\sum_j w_{ij}x_j < \theta_i$; otherwise, $x_i = 1$, if $\sum_j w_{ij}x_j > \theta_i$, and no change otherwise. Here, w_{ij} is the connection weight between units i and j, and

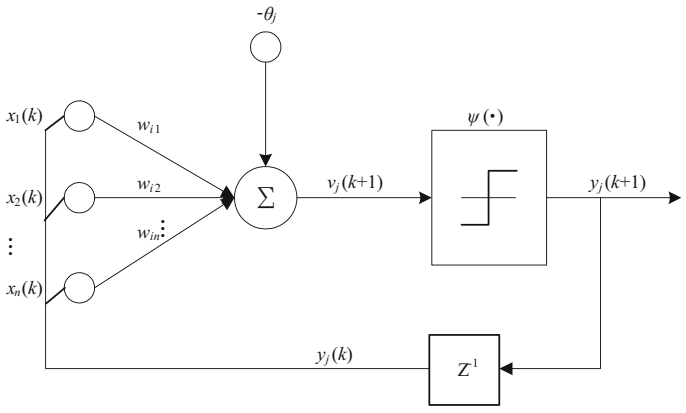

Fig. 9.2 Discrete-time Hopfield structure

θ_i is the threshold of unit i. It is obvious that the energy function is non-increasing with the updating of the neuron's state (Hopfield 1982). In general, the property of a recurrent network can be described as an energy function, which is used to improve the stability of recurrent network.

Hopfield (1982) presented an energy function to demonstrate its stability as:

$$E_{\text{DHNN}} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} x_i w_{ij} x_j - \sum_{i=1}^{n} x_i I_i \tag{9.1}$$

and showed that the energy function is a Lyapunov function, which can lead the final state into a stable state, if the neurons are repetitively updated one at a time in any order. On the other hand, the local minimum of the energy function corresponds to the energy of the stored patterns. According to the work of Bruck and Sanz (1988), the network can converge to a minimum from any initial state. Thus, the characteristic is helpful for applications. In addition, one famous NN for optimization, the Boltzmann machine, is an extension of discrete HNN. It only replaces the deterministic local search dynamics of the HNN by randomized local search dynamics (Kumar 2005; Kurita and Funahashi 1996), but it has only been applied to a small number of problems.

9.2.1.2 Continuous Hopfield Networks

The continuous or continuous-time HNN is a generalization of the discrete case. The common output functions utilized in the networks are sigmoid and hyperbolic tangent functions. Figure 9.3 shows the common circuit of the continuous-time HNN, where $T_{cj} = R_j C_j, j = 1,\ldots,n$, is the integration time constant of the jth neuron, and θ_j is an externally applied threshold. The integrator can be realized by an operational amplifier, capacitor C_j, and resistor R_j. Here, $\gamma_j > 0$ is called the forgetting factor of the integrator, which forces the internal signal v_j to zero for a zero input. To formulate the network, we establish a differential equation for the activation potential $v_j(t)$ as $T_{cj}\left(\mathrm{d}v_j/\mathrm{d}t\right) = -\gamma_j v_j + \left(\sum_{i=1}^{n} w_{ji} x_i - \theta_j\right)$. The output of the neuron is given by $y_j = \psi(\cdot)$, which is a continuous output function such as a sigmoid function.

Hopfield (1984) proposed an energy function for the continuous HNN as:

$$E_{\text{CHNN}} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} x_i w_{ij} x_j - \sum_{i=1}^{m} x_i I_i + \sum_{i=1}^{n}\left(\frac{1}{R_i}\right)\int_0^{x_i} g_i^{-1}(x_i)\mathrm{d}x \tag{9.2}$$

where the function $g_i^{-1}(x_i)$ is a monotone increasing function. Hopfield and Tank (1985, 1986) introduced the continuous HNN to solve the TSP and LP problems. Afterwards, many researchers implemented HNN to solve the optimization problem, especially in MP problems. Hence, the continuous model is our major concern. In general, the continuous model is superior to the discrete one in terms of the local

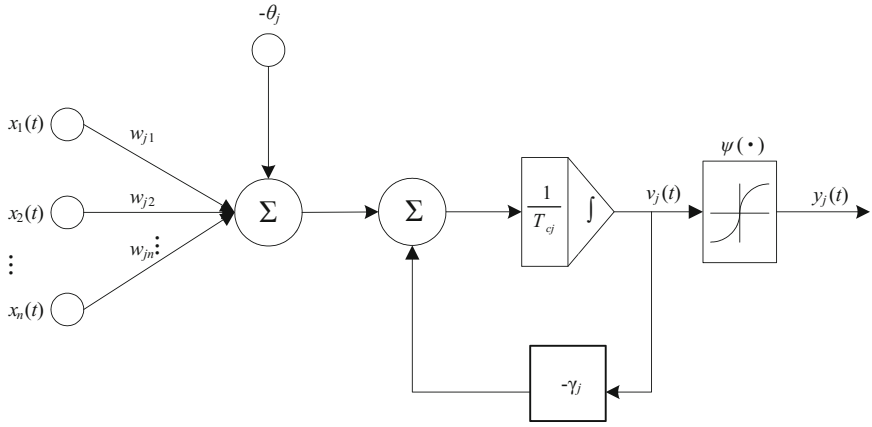

Fig. 9.3 Continuous-time Hopfield structure

minimum problem, because of its smoother energy surface. Hence, the continuous HNN has dominated the solving techniques for optimization problems, especially for combinatorial problems.

9.2.2 Methods of Hopfield Networks

From the computational aspect, the operation of HNN for an optimization problem manages a dynamic system characterized by an energy function, which is the combination of the objective function and constraints of the original problem. Three commonly used techniques to solve optimization problems with constraints are: penalty functions, Lagrange functions (or augmented Lagrange functions), and primal and dual functions. The technique of penalty functions uses penalty terms to combine the constraints and the objective function to yield an unconstrained optimization problem, and then construct and minimize an energy function based on the combined objective function. Similarly, Lagrange functions (or augmented Lagrange functions) relax the constraints and take advantage of Lagrange multiples to construct an energy function to be operated. The technique of primal and dual functions is made up of a primal function and a dual function and try to reach the minimum gap between the functions.

9.2.2.1 Penalty Function Method

The penalty function method is a popular technique for optimization in which it is used to construct a single unconstrained problem. A searching method can be adopted to search for the solution in the decision space, and the steepest descent

approach is the popular technique for obtaining the searching direction (Cichocki and Unbehauen 1993).

Consider an LP problem:

$$\min f(\boldsymbol{x}) = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x},\ \text{s.t.}\ \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b} (= g_j(\boldsymbol{x}) \geq b_j,\ j = 1, 2, \ldots, m) \text{ and } \boldsymbol{x} \geq 0$$

where $\boldsymbol{x}$, $\boldsymbol{c} \in \Re^{n\times 1}$, $\boldsymbol{A} \in \Re^{m\times n}$, and $\mathbf{b} \in \Re^{m\times 1}$. Tank and Hopfield (1986) first proposed the NN structure to solve the LP problem, and its energy function can be defined as:

$$E_{\mathrm{TH}}(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{j=1}^{m} \left(g_j^{+}(\boldsymbol{x})\right)^2 + \sum_{i=1}^{n} x_i^2 \Big/ 2sR_{ii} \tag{9.3}$$

where R is an $n \times n$ diagonal matrix, $s>0$ is a penalty parameter, and $g^{+} = [g_1^{+}, g_2^{+}, \ldots, g_m^{+}]^{\mathrm{T}}$ is a penalty vector when $g_j(\boldsymbol{x}) < b_j$. Here, the penalty parameter s must be sufficiently large to lead the last term to be neglected. However, this assumption might make their model unreliable for solving the LP problem (Kennedy and Chua 1988).

To improve Tank and Hopfield's NN, Kennedy and Chua (1988) proposed a kind of NN structure with an inexact penalty function, where the energy function is

$$E_{\mathrm{KC}}(\boldsymbol{x}) = f(\boldsymbol{x}) + \frac{s}{2} \sum_{j=1}^{m} \left(g_j^{+}(\boldsymbol{x})\right)^2 \tag{9.4}$$

Here, $s>0$ is a penalty parameter, and for an inexact penalty rule, the solution converges to LP as $s \to \infty$. However, there is difficulty in choosing a huge number of parameters (Cichocki and Unbehauen 1993). Rodríguez-Vázquez et al. (1988) directly use another penalty method to transform the LP problem into an unconstraint optimization problem. Their energy function is illustrated as:

$$E_{\mathrm{RV}}(\boldsymbol{x}, \alpha) = f(\boldsymbol{x}) + \alpha \left| \sum_{j=1}^{m} \min\left\{0, g_j(\boldsymbol{x}) - b_j\right\} \right| \tag{9.5}$$

where $\alpha > 0$. A non-negative function will satisfy it. In addition, for each discrete-time step k, we have $x_i(k) = \max\{x_i(k), 0\}$. Once the feasible region is reached, the trajectory moves toward the minimal direction of the objective function. Although Rodríguez-Vázquez et al. (1990) pointed out that their network has no equilibrium point in the classical sense, however, according to the investigation of Lan et al. (2007), their network can converge to an optimal solution of the corresponding convex programming problem from any initial state.

Maa and Shanblatt (1992) used a two-phase NN structure for solving the problem. In the first phase, $t < t_1$, t_1 is randomly selected, and the structure is the same as in Kennedy and Chua. The stability of the network is dependent on how to choose the penalty parameter s and time parameter t_1, but it is not easy to choose t_1. If the initial

solution does not fall into a feasible region, then the solution does not converge to the stable state in the final. In addition, Chong et al. (1999) analyzed a class of NN models for solving LP problems by dynamic gradient approaches based on exact non-differentiable penalty functions. They developed an analytical tool helping the systems converge to a solution within a finite time.

9.2.2.2 Lagrange Function Method

Similar to the penalty function methods, Lagrange multiplier and augmented Lagrange multiplier methods merge an objective function and constraints into an energy function of the target networks. Aside from some earlier works, Zhang and Constantinides (1992) proposed a Lagrange method and an augmented Lagrange method for solving LP problems through HNNs. Their energy functions by Lagrange and augment Lagrange multipliers are:

$$E_{\mathrm{L}}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{j=1}^{m} \boldsymbol{\lambda}\big(g_j(\boldsymbol{x})\big), \tag{9.6}$$

and

$$E_{\mathrm{aL}}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{K}) = f(\boldsymbol{x}) + \sum_{j=1}^{m} \boldsymbol{\lambda}\big(g_j(x)\big) + \frac{1}{2}\sum_{j=1}^{m} \boldsymbol{K}\big|g_j(x)\big|^2 \tag{9.7}$$

where constraints $g_j(\boldsymbol{x})$ should be modified as $g_j(\boldsymbol{x}) = 0$, and $\boldsymbol{\lambda}$ and K are Lagrange multiplier vectors and positive penalty parameters, respectively. Both energy functions utilize two types of terms to fit the real and dual variables, resulting in a longer computational time. It can be proven that the addition of the quadratic penalty term in Eq. (9.7) increases the positive definiteness of Lagrange's Hessian matrix (Gill et al. 1981). Furthermore, if the coefficients in K are sufficiently large, then Hessian matrix of the Lagrange can force to all eigenvalues of its elements to be greater than zero. From the standpoint of implementation, this characteristic is of great importance since the solution can be sought in iterations.

Gill et al. (1981) mentioned that if a function is convex, then gradient-based searching methods are guaranteed to converge to a local minimum. In addition, Zhang and Constantinides (1992) also suggested that the penalty parameter K is no more than 5 for convergence. According to Shih et al. (2004), K should be a very small number, e.g., 0.01, in order to obtain an optimal solution. Shih et al. (2004) introduced an augmented Lagrange multiplier method to solve multi-objective and multi-level problems based on the LP approach. Its energy function $E_{\mathrm{S}}(\boldsymbol{x}, \boldsymbol{\lambda})$ is:

$$\begin{aligned} E_S(\boldsymbol{x},\ \boldsymbol{\lambda}) = {} & \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{\lambda}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) - \frac{\alpha}{2}\boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{\lambda} \\ & + \frac{\boldsymbol{K}}{2}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \end{aligned} \tag{9.8}$$

The function includes penalty parameters, a Lagrange multiplier and a regularization term, $\alpha\ \boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{\lambda}$ is for improving the stability of the method (Ham and Kostanic 2001), and α is a small positive number, e.g., 0.001. Each iteration requires the two steepest gradients with $\boldsymbol{x}$ and $\boldsymbol{\lambda}$ for the direction of searching for the stable point, thus obtaining a rather close solution.

9.2.2.3 Primal and Dual Method

The duality theorem provides that the primal solution equals to the dual solution when reaching an optimal solution in LP (Bazaraa et al. 2006). The energy function can be constructed based on the primal and dual rules. However, such an energy function is generally complicated since both primal and dual variables are involved. Thus, the literature often limits the discusses to small-sized problems (Wang 1997, 1998; Xia and Wang 1995; Xia 1996).

Xia and Wang (1995) first used bounded variables to construct a new NN approach to solve the LP problem with no penalty parameters. They suggested that the equilibrium point is the same as the exact solution when simultaneously solving the primal problem and its dual problem. Hence, they defined an energy function for solving LP problems as $E_{\mathrm{xw}}(\boldsymbol{x}, \boldsymbol{y})$:

$$E_{\mathrm{XW}}(\boldsymbol{x},\ \boldsymbol{y}) = \frac{1}{2}(c^T\boldsymbol{x} - b^T\boldsymbol{y})^2 + \frac{1}{2}\boldsymbol{x}^T(\boldsymbol{x} - |\boldsymbol{x}|) + \frac{1}{2}\boldsymbol{y}^T(\boldsymbol{y} - |\boldsymbol{y}|) + \frac{1}{2}\left\|A^T\boldsymbol{x} - b\right\|_2^2 + \frac{1}{2}\left[A^T\boldsymbol{y} - c\right]\left[(A^T\boldsymbol{y} - c) - \left|A^T\boldsymbol{y} - c\right|\right] \quad (9.9)$$

Their NN approach has three advantages: (i) avoiding a significant amount of parameters, (ii) escaping from an infeasible solution, and (iii) being able to estimate the duality gap indirectly. Afterwards, Xia and Wang (1998) showed many NNs structures which include penalty and primal-dual forms for solving optimization problems. They proved that the approach improved the Maa and Shanblatt's parameters selection (Maa and Shanblatt 1992), and demonstrated that it can be used in general mathematical programming problems.

Malek and Tari (2005) presented two methods to solve the primal LP problem and found their optimal solution for both primal and dual LP problems. They claimed that their methods give better solutions for dual variables and better optimal solutions. In addition, they realized that the new network can solve LP problems with efficient convergence within a finite time.

9.3 Hopfield Neural Networks for Multi-level Programming

Recall the bi-level programming problem discussed in earlier chapters:

$$\max_{x_1} f_1(x_1, x_2) = c_{11}^{\mathrm{T}} x_1 + c_{12}^{\mathrm{T}} x_2 \quad \text{(Upper level)} \tag{9.10}$$

where x_2 solves:

$$\max_{x_2} f_2(x_1, x_2) = c_{21}^{\mathrm{T}} x_1 + c_{22}^{\mathrm{T}} x_2 \quad \text{(Lower level)} \tag{9.11}$$

subject to:

$$(x_1, x_2) \in X = \{(x_1, x_2) \mid A_1 x_1 + A_2 x_2 \leq b \text{ and } x_1, x_2 \geq 0\}$$

where A_1 and A_2 are $m \times n_1$- and $m \times n_2$-dimensional matrices, respectively; c_{11}, c_{21} and x_1 are n_1-dimensional vectors; c_{12}, c_{22} and x_2 are n_2-dimensional vectors; and b is an m-dimensional vector. As presented in Chap. 2, the KKT conditions are applied to transform the original problem to its first level auxiliary problem, and then the mixed-integer approach by Fortuny-Amat and McCarl (1981) is used to linearize the first level auxiliary problem. The resulting mixed-integer problem is reproduced as follows:

$$\begin{aligned} &\max\ f_1(x_1, x_2) = c_{11}^{\mathrm{T}} x_1 + c_{12}^{\mathrm{T}} x_2 \\ &\text{subject to:} \\ &\quad A_1 x_1 + A_2 x_2 \leq b, \\ &\quad w \leq (1 - \eta) M, \\ &\quad A_1 x_1 + A_2 x_2 - b \leq M\eta, \\ &\quad w^{\mathrm{T}} A_2 = c_{22}, \\ &\quad \eta \in \{0, 1\}, \\ &\quad x_1, x_2, w \geq 0, \end{aligned} \tag{9.12}$$

To solve (9.12), the Lagrange function method discussed in Sect. 9.2.2.2 is used to construct the energy function. To minimize the energy function of the form in Eq. (9.8), find partial derivatives of the equation with respect to $\boldsymbol{x}$ and $\boldsymbol{\lambda}$, and obtain the following differential equations:

$$\frac{\partial E_s(\boldsymbol{x}, \boldsymbol{\lambda})}{\partial \boldsymbol{x}} = \boldsymbol{c} + K\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) + A^T\boldsymbol{\lambda} \tag{9.13}$$

$$\frac{\partial E_s(\boldsymbol{x}, \boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}} = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} - \alpha\boldsymbol{\lambda} \tag{9.14}$$

To use a discrete Hopfield neural network, the above differential equations are rewritten to the following system of difference equations:

$$x_j(k+1) = \begin{cases} x_j(k) - \mu_j(k)\left\{c_j + \sum_{i=1}^{m} a_{ij}[K r_i(k) + \lambda_i(k)]\right\}, & \text{if } x_j(k+1) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{9.15}$$

$$\lambda_i(k+1) = \lambda_i(k) + \nu_i(k)\,[r_i(k) - a\,\lambda_i(k)] \tag{9.16}$$

where $\boldsymbol{x} = \left[x_1, \ldots, x_j, \ldots, x_n\right]^T$, $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_i, \ldots, \lambda_m]^T$, $r_i(k) = \sum_{j=1}^{n} a_{ij}x_j - b_i$, $K \geq 0$, $\alpha \geq 0$, and $\mu_j(k)$ and $\nu_i(k)$ are learning rates. Equations (9.15) and (9.16) are implemented by a discrete Hopfield neural network, and x_j and λ_i are updated repeatedly until their steady states are reached, at which the optimum is considered to be obtained.

Example 9.1 Consider a bi-level programming problem (Wen and Hsu 1991):

$$\begin{aligned} &\text{Max}_{x_1}\ f_1 = -2x_1 + 11x_2 \\ &\text{where } x_2 \text{ solves} \\ &\quad \text{Max}_{x_2}\ f_2 = -x_1 - 3x_2 \\ &\quad \text{Subject to:} \\ &\qquad x_1 - 2x_2 \leq 4 \\ &\qquad 2x_1 - x_2 \leq 24 \\ &\qquad 3x_1 + 4x_2 \leq 96 \\ &\qquad x_1 + 7x_2 \leq 126 \\ &\qquad -4x_1 + 5x_2 \leq 65 \\ &\qquad x_1 + 4x_2 \leq 8 \\ &\qquad x_1, x_2 \geq 0 \end{aligned}$$

Denote the constraint set of the above problem by Y. Applying the KKT transformation, the above problem becomes:

$$\begin{aligned} &\text{Max } f_1 = -2x_1 + 11x_2 \\ &\text{Subject to :} \\ &\quad x \in Y \\ &\quad x_1 - 2x_2 - 4 \leq M\eta_1, \quad w_1 \leq (1-\eta_1)M \\ &\quad 2x_1 - x_2 - 24 \leq M\eta_2, \quad w_2 \leq (1-\eta_2)M \\ &\quad 3x_1 + 4x_2 - 96 \leq M\eta_3, \quad w_3 \leq (1-\eta_3)M \\ &\quad x_1 + 7x_2 - 126 \leq M\eta_4, \quad w_4 \leq (1-\eta_4)M \\ &\quad -4x_1 + 5x_2 - 65 \leq M\eta_5, \quad w_5 \leq (1-\eta_5)M, \\ &\quad -x_1 - 4x_2 + 8 \leq M\eta_6, \quad w_6 \leq (1-\eta_6)M, \\ &\quad -2w_1 - w_2 + 4w_3 + 7w_4 + 5w_5 + 4w_6 = -3, \end{aligned}$$

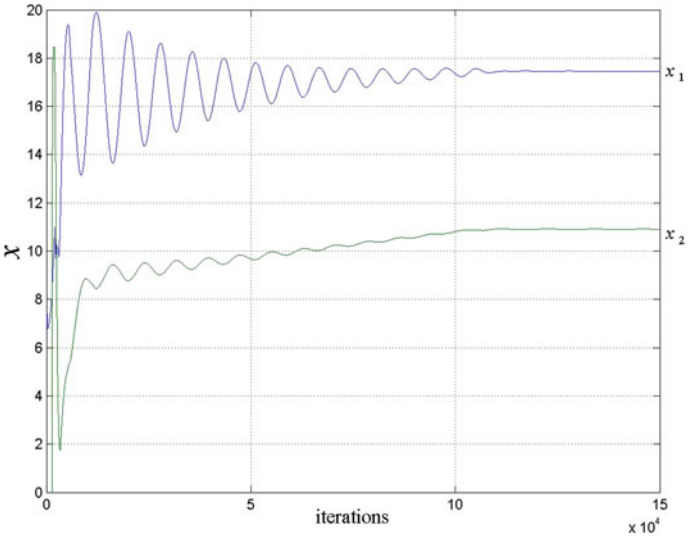

Fig. 9.4 Trajectory of $\boldsymbol{x}$ versus the number of iterations (Example 9.1)

Table 9.1 The solutions with different values of parameter K (Example 9.1)

	$K=0.1$	$K=0.5$	$K=1.0$
x_1	17.4546	17.5000	17.5000
x_2	10.9089	10.9000	10.9000
f_1	85.0889	85.0909	85.0909
f_2	−50.1813	−50.2000	−50.2000

$$\eta_1, \eta_2, \eta_3, \eta_4, \eta_5, \eta_6 \in \{0,\ 1\},$$
$$w_1, w_2, w_3, w_4, w_5, w_6 \geq 0.$$

The above formulation is reformulated as an unconstrained optimization problem by applying Lagrange relaxation and constructed as an energy function as the form in Eq. (9.8) with branch-and-bound tree for 0–1 variables $\eta_n, n = 1, 2, \ldots, 6$. Using the MATLAB software with $M = 1000$, $K = 0.5$ and $\mu = 0.005$, the problem was solved and the solutions are $(x_1^*, x_2^*) = (17.5000,\ 10.9000)$ with $(f_1^*, f_2^*) = (85.0909, -50.2000)$. The solution trajectory of $\boldsymbol{x}$ versus the number of iterations is shown in Fig. 9.4, and the effects of the value of K $(0.1 \leq K \leq 1.0)$ on solutions are shown in Table 9.1.

Example 9.2 Consider a bi-level decentralized programming problem with a central decision-maker at the upper level and three independent division units at the lower level (Anandalingam 1988):

$$
\begin{aligned}
&\text{Max}_{x_1}\ f_1 = x_1 + x_2 + 2x_3 + x_4\\
&\text{where } x_2, x_3 \text{ and } x_4 \text{ solve,}\\
&\quad \text{Max}_{x_2}\ f_{21} = x_1 + 3x_2 + x_3 + x_4\\
&\quad \text{Max}_{x_3}\ f_{22} = x_1 + x_2 + 3x_3 + x_4\\
&\quad \text{Max}_{x_4}\ f_{23} = x_1 + x_2 + x_3 + 3x_4\\
&\quad \text{Subject to:}\\
&\qquad 3x_1 + 3x_2 \le 30\\
&\qquad 2x_1 + x_2 \le 20\\
&\qquad x_3 \le 10\\
&\qquad x_2 + x_4 \le 15\\
&\qquad x_4 \le 10\\
&\qquad x_1 + 2x_2 + 2x_3 + x_4 \le 40\\
&\qquad x_1, x_2, x_3, \text{ and } x_4 \ge 0
\end{aligned}
$$

We denote the above constraint set as Z. After applying the KKT conditions to the original problem, a new auxiliary problem with single level is obtained. By exploiting Fortuny-Amat and McCarls' suggestion (Fortuny-Amat and McCarl 1981), we obtain a the formulation below:

$$
\begin{aligned}
&\text{Max } f_1 = x_1 + x_2 + 2x_3 + x_4\\
&\text{Subject to:}\\
&\quad x \in Z\\
&w_1 \le (1-\eta_1)M, \quad 3x_1 + 3x_2 - 30 \le M\eta_1\\
&w_2 \le (1-\eta_2)M, \quad 2x_1 + x_2 - 20 \le M\eta_2\\
&w_3 \le (1-\eta_3)M, \quad x_3 - 10 \le M\eta_3\\
&w_4 \le (1-\eta_4)M, \quad x_2 + x_4 - 15 \le M\eta_4,\\
&w_5 \le (1-\eta_5)M, \quad x_4 - 10 \le M\eta_5,\\
&w_6 \le (1-\eta_6)M, \quad x_1 + 2x_2 + 2x_3 + x_4 - 40 \le M\eta_6,\\
&3w_1 + w_2 + 2w_6 = 3\\
&w_3 + w_4 + 2w_6 = 3\\
&w_4 + w_5 + w_6 = 3\\
&\eta_1, \eta_2, \eta_3, \eta_4, \eta_5, \text{ and } \eta_6 \in \{0, 1\}\\
&w_1, w_2, w_3, w_4, w_5, \text{ and } w_6 \ge 0.
\end{aligned}
$$

Following the same procedure in Example 9.1, the above formulation is first reformulated as an unconstrained optimization problem, and then solved by using

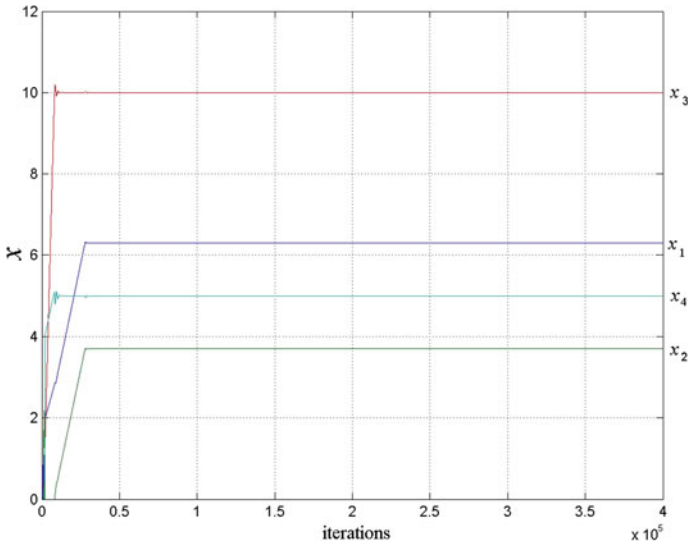

Fig. 9.5 Trajectory of $\boldsymbol{x}$ versus the number of iterations (Example 9.2)

Table 9.2 The solutions with different values of parameter K (Example 9.2)

	$K=0.1$	$K=0.5$	$K=1.0$	$K=2.0$
x_1	9	7.7	6.3	6.1
x_2	1	2.3	3.7	3.9
x_3	10	10	10	10
x_4	5	5	5	5
f_1	35	35	35	35
f_{21}	27	29.6	32.4	32.8
f_{22}	45	45	45	45
f_{23}	35	35	35	35

MATLAB. With $M = 1000$, $K = 0.5$ and $\mu = 0.01$, the solution obtained is $(x_1{*}, x_2{*}, x_3{*}, x_4{*}) = (7.7, 2.3, 10, 5)$ with objectives $(f_1{*}, f_{21}{*}, f_{22}{*}, f_{23}{*}) = (35, 29.6, 45, 35)$. We can find that alternative optimal solutions exist, with respect to different K values, in this example. The trajectory of $\boldsymbol{x}$ versus the number of iterations is shown in Fig. 9.5, and the effects of different K value ($0.1 \leq K \leq 2.0$) are presented in Table 9.2.

9.4 Hybrid Neural Network Approach for Multi-level Programming

Neural network approaches combining with other heuristics, referred to as hybrid neural network approaches, have be proposed to solve the bi-level problems. Lan et al. (2007) combined the neural network of Rodríguez-Vázquez et al. (1990) and a tabu search to solve linear bi-level problems. Yaakob and Watada (2011) Combined a Boltzmann machine (BM) and a GA to solve mixed integer quadratic bi-level programming problems. These studies are introduced next.

9.4.1 Neural Network with Tabu Search

Same as the transformation procedure of the bi-level problem to a single objective problem presented in Sect. 9.3, the hybrid approach of Lan et al. (2007) first transforms the original bi-level problem to a single-objective mixed-integer and linear problem of the form of Eq. (9.12).

The suggested core solver by Lan et al. (2007) relies on the network by Rodriquez-Vazquez et al. (1990), and the 0–1 variables (η) are selected by a tabu strategy. The binary variables selected by tabu strategy form the input for the neural network optimization. If the solution is infeasible, the binary variables are marked as forbidden; otherwise, further binary variables are generated for the next iteration. The process continues until either the tabu maximum terminal size is reached or there is no better solution in the tabu list. The procedure can be summarized in the following steps.

Step 1. (Initialization)

- Step 1.1 Set the maximum tabu search iteration N, the tabu list TL, inventory list IL, and NA (No admittance) $= 1$.
- Step 1.2 Set a tabu index $I = 1$ for tabu iteration.

Step 2. (The Selection of Tabu Strategy)

- Step 2.1 Randomly generate the binary variables set $\eta_{[I]}$ at iteration I from IL' (the complementary infeasible variables set of IL).
  Go to *Step 2.2*.
- Step 2.2 if $\eta_{[I]}$ is not in TL, go to Step 3; otherwise, go to *Step 2.1*.

Step 3. (Neural Network Computation)
Solve the mixed-integer linearized problem with $\eta_{[I]}$ by the neural network of Rodriquez-Vazquez et al. (1990), and obtain the primal and dual solutions.
Go to Step 4.

Step 4. (Decision Rule)

Step 4.1 If the solution is satisfactory and all constraints are satisfied, then compute the upper-level objective $F_{[I]}$, and go to *Step 4.2*; otherwise go to *Step 4.3*.

Step 4.2 If $I \leq N$ and $\eta_{[I]}$ is not in *TL*, record the current $\eta_{[I]}$ and the $F_{[I]}$ in the *TL*. Set $I \leftarrow I + 1$ and go to *Step 2.1*; otherwise, go to *Step 5*.

Step 4.3 Record $\eta_{[I]}$ in *IL*, and set $I \leftarrow I + 1$. Go to *Step 2.1*.

Step 5. If $I \leq N$ and $NA \leq TL$, let $NA \leftarrow NA + 1$ and $I \leftarrow I + 1$, go to *Step 2.1*; otherwise, go to *Step 6*.

Step 6. (Termination)
The algorithm terminates, and the near-optimum solution is reached with the objective value $F_{[I]}$.

Example 9.3 Recall Example 9.1.

The selection of the parameters of the neural network and the tabu list are important factors, which influence the performance of the algorithm. The penalty parameter is chosen as 1 for simplification, and the learning rate is set as 0.001 or 0.0001. The stopping condition is set as $|\text{norm}(\boldsymbol{x}(\text{k}+1))—\text{norm}(\boldsymbol{x}(\text{k}))| < 10^{-6}$. The maximum number of iterations is 50,000 and the length of the tabu list is 7. The best solution obtained is $(x_1{*}, x_2{*}) = (17.4500, 10.9080)$ with $(f_1{*}, f_2{*}) = (85.085500, -50.1740)$. The results are very close to those obtained in Example 9.1. Solutions based on different penalty values are listed in Table 9.3.

9.4.2 Boltzmann Machine with GA

Yaakob and Watada (2011) presented a GA-based hybrid meta-heuristic algorithm to solve mixed integer quadratic bi-level programming problems, containing both Hopfield and Boltzmann machine neural networks which are called meta-controlled BM. Their solution framework consists of three phases, the first phase generates a partial solution of the upper level problem, the second phase employs the meta-controlled BM to solve the lower level problem with the partial solution from the first phase, and in the third phase, the solution of the lower level is sent to the upper level for evaluating the upper-level objective. The above three phases are executed iteratively until the whole problem is solved with satisfaction.

Consider a mixed integer quadratic bi-level programming problem below.

Upper-level:

$$\text{Max } T = R + r \tag{9.17}$$

$$\text{where } R = \min \sum_{a=1}^{n} \sum_{b=1}^{n} \sigma_{ab} m_a \alpha_a m_b \alpha_b \tag{9.18}$$

Subject to:

$$\sum_{a=1}^{n} \mu_a m_a \alpha_a \geq P \tag{9.19}$$

$$\sum_{a=1}^{n} m_a \alpha_a = 1 \tag{9.20}$$

$$\sum_{a=1}^{n} m_a = S \tag{9.21}$$

$$m_a \in \{0, 1\}, a = 1, 2, \ldots, n$$
$$\alpha_a \geq 0, \ a = 1, 2, \ldots, n$$

In the above formulation, T denote a total cost of the bi-level problem, R is the cost at the upper-level while r is that of the lower-level; σ_{ab} is the covariance between

Table 9.3 Computational results with various penalty values, K, for Example 9.3

K	Tabu list	Binary variable set	Solution
1	–	–	No feasible solution
2	List 1	[0 0 0 1 1 0 0]	$x_1 = 0.0000$, $x_2 = 2.0000$, $f_1 = 22.0000$
2	List 2	[0 1 1 1 1 1 0]	$x_1 = 5.3028$, $x_2 = 17.2410$, $f_1 = 179.057900$
2	List 3	[0 1 1 0 0 1 1]	$x_1 = 9.8780$, $x_2 = 16.5900$, $f_1 = 162.709500$
2	List 4	[1 0 1 1 1 1 0]	$x_1 = 17.4500$, $x_2 = 10.9080$, $f_1 = 85.085500$
5	List 1	[0 0 0 1 1 0 0]	$x_1 = 0$, $x_2 = 2.0000$, $f_1 = 22.0000$
5	List 2	[1 0 1 1 1 1 0]	$x_1 = 17.4500$, $x_2 = 10.9080$, $f_1 = 85.085500$
10	List 1	[0 0 0 1 1 0 0]	$x_1 = 0$, $x_2 = 2.0000$, $f_1 = 22.0000$
10	List 2	[1 0 1 1 1 1 0]	$x_1 = 17.4500$, $x_2 = 10.9080$, $f_1 = 85.085500$
20	–	–	No feasible solution

a and b, and μ_a is the technical coefficients of the constraints; m_a is 0–1 decision variable for a and α_a is the real-valued decision variable of a. The lower-level problem is solved by the meta-controlled BM and thus is formulated with energy functions of two components, a Hopfield layer and a BM layer.

Lower-level:

Hopfield layer

$$E_h = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sigma_{ij}m_i s_i m_j s_j + \frac{1}{\beta}\sum_{i=1}^{n}\mu_i m_i s_i \tag{9.22}$$

BM layer

$$E_{bm} = -\frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sigma_{ij}m_i x_i m_j x_j + 2\sum_{i=1}^{n}\sum_{j=1}^{n}m_i x_i m_j x_j\right) + 2\sum_{i=1}^{n}m_i x_i + \frac{1}{\beta}\sum_{i=1}^{n}\mu_i m_i x_i \tag{9.23}$$

in which, $\beta \geq 0$, σ_{ij} is the covariance between i and j, and μ_i is the technical coefficients of the constraints; m_i is 0–1 decision variable for i and x_i is the real-valued decision variable of i. β is the output of the i-th unit of the Hopfield layer and determined based on α from the upper-level and used to weight the expected profits.

The generation of the partial solution of the upper level is carried out by a GA. Solutions are encoded as chromosomes by setting an allele to the number of binary variables and the string length to the number of choices.

The meta-controlled BM is a double-layered neural network consisting of a Hopfield and a Boltzmann neural networks. The functions of the layers are as follows:

- The Hopfield layer serves as a "supervising layer" that selects a number of units from the initial architecture.
- The Boltzmann layer acts as an "executing layer" that determines the optimal units from the units selected by the Hopfield layer.

A Boltzmann Machine is a network of symmetrically connected, neuron like units that make stochastic decisions about whether to be on or off. When unit i is chosen to update its binary state, it is turned on with a probability given by the logistic function:

$$prob(s_i = 1) = \frac{1}{1 + e^{-z_i}} \tag{9.24}$$

where z_i is the input of the unit and is the sum of its own bias, b_i, and the weights w_{ij} on connections coming from other active units:

$$z_i = b_i + \sum_j s_j w_{ij} \tag{9.25}$$

The probability of a state vector $\boldsymbol{v}$ is determined by the *energy* of the state vector over the energies of all possible binary state vectors:

$$P(\boldsymbol{v}) = \frac{\mathrm{e}^{-E(\boldsymbol{v})}}{\sum\limits_{\boldsymbol{u}} \mathrm{e}^{-E(\boldsymbol{u})}} \tag{9.26}$$

Same as the Hopfield network, the energy of state vector $\boldsymbol{v}$ is defined as:

$$E(\boldsymbol{v}) = -\sum_{i} s_i^{\boldsymbol{v}} b_i - \sum_{i<j} s_i^{\boldsymbol{v}} s_j^{\boldsymbol{v}} w_{ij} \tag{9.27}$$

where $s_i^{\boldsymbol{v}}$ is the binary state assigned to unit i by state vector $\boldsymbol{v}$. The learning of BM is to find weights and biases that define a Boltzmann distribution in which the training vectors have high probability. The learning can be improved by using simulated annealing (Kirkpatrick et al. 1983), which scales down all of the weights and energies by a factor, T, analogous to the temperature of a physical system.

The algorithm by Yaakob and Watada (2011) is summarized as follows.

Step 1. Randomly generate a series of solution combinations. Employ GA to select a set of potential solution combination and pass the combination to the lower level.
Step 2. Initialize the meta-controlled BM.
Step 3. Give the weight $1/\beta$ to the Hopfield layer and the BM layer.
Step 4. Execute the Hopfield layer.
Step 5. For units in the Hopfield layer, if the output value of a unit in is 1, then add some amount of this value to the corresponding unit in the BM layer. Execute the BM layer.
Step 6. After executing the BM layer for a predefined iterations, decrease the temperature.
Step 7. If the output value of the BM layer is sufficiently large, add a certain amount of the value to the corresponding unit in the Hopfield layer.
Step 8. Repeat Step 4 to Step 7 until the temperature reaches the restructuring temperature.
Step 9. Restructure the BM layer using selected units in the Hopfield layer.
Step 10. Execute the BM layer until it reaches the stop criterion.
Send the obtained solution to the upper-level.
Step 11. Evaluate the upper level objective.
Step 12. The processes proceed interactively until the problem is optimally resolved.

9.5 Discussions

This chapter investigated the solution procedure of MLP problems by Hopfield neural networks. In contrast to other approaches that are generally implemented by programming software, the architecture of Hopfield neural networks easily enable their implementation on an electronic circuit for real-time problem-solving or large-size problems. Though the examples illustrated in this chapter were still carried out by computer simulation on a MATLAB platform, the networks can be realized on a VLSI which houses a friendly environment for the hardware to manage the problems more efficiently. Hybrid approaches that combine neural networks with other heuristics demonstrate the possibility of improving the performances of neural networks by.

The KKT conditions transformation of the original MLP problem can result in a highly nonlinear energy function, and thus creates difficulty for the Hopfield neural network to solve the problem, especially when the MLP problem has more than three levels. The choice of an appropriate value of the parameters in energy function, i.e., K, also hinders the application of Hopfield neural networks to solving MLP problems. Though several pioneers suggested various guidelines to select the desirable parameters, more rigorous investigations are expected.

References

Ackley, D.H., Hinton, G.E., Sejnowski, T.J.: A learning algorithm for Boltzmann machines. Cogn. Sci. **9**(1), 147–169 (1985)

Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows. Prentice-Hall, Englewood, New Jersey (1993)

Aiyoshi, E., Shimizu, K.: Hierarchical decentralized systems and its new solution by a barrier method. IEEE Trans. Syst. Man Cybern. **11**(6), 444–449 (1981a)

Aiyoshi, E., Shimizu, K.: A new computational method for Stackelberg and min-max problems by use of a penalty method. IEEE Trans. Autom. Control **26**(2), 460–466 (1981b)

Aiyoshi, E., Shimizu, K.: A solution method for the static constrained Stackelberg problem via penalty method. IEEE Trans. Autom. Control **29**(12), 1111–1114 (1984)

Al-Khayyal, F.A.: An implicit enumeration procedure for the general linear complementarity problem. Math. Progr. Stud. **31**, 1–20 (1987)

American Production and Inventory Control Society: APICS Dictionary, APICS Educational Society for Resource Manage (2004)

Anandalingam, G.: A mathematical programming model of decentralized multi-level systems. J. Oper. Res. Soc. **39**, 1021–1033 (1988)

Anandalingam, G., Apprey, V.: Multi-level programming and conflict resolution. Eur. J. Oper. Res. **51**, 233–247 (1991)

Anandalingam, G., Friesz, T.L.: Hierarchical optimization: an introduction. In: Anandalingam, G., Friesz, T.L. (eds.) Annals of Operations Research, vol. 34, pp. 1–11 (1992)

Anandalingam, G., White, D.J.: A solution for the linear static Stackelberg problem using penalty functions. IEEE Trans. Autom. Control **35**, 1170–1173 (1990)

Anderson, C.A., Fraughnaugh, K.M., Parker, M., Ryan, J.: Path assignment for call routing: an application of tabu search. Ann. Oper. Res. **41**, 299–312 (1993)

Atanassov, K.T.: Intuitionistic fuzzy sets. Fuzzy Sets Syst. **20**(1), 87–96 (1986)

Bard, J.F.: A grid search algorithm for the linear bilevel programming problem. In: 14th Annual Meeting of the American Institute for Decision Sciences, San Francisco, CA, 22–24 Nov, vol. 2, pp. 256–258 (1982)

Bard, J.F.: Coordination of a multidivisional organization through two levels of management. Omega **11**, 457–468 (1983a)

Bard, J.F.: An efficient point algorithm for a linear two-stage optimization problem. Oper. Res. **31**, 670–684 (1983b)

Bard, J.F.: An algorithm for solving the general bilevel programming problems. Math. Oper. Res. **8**, 260–272 (1983c)

C.-B. Cheng et al., *Fuzzy and Multi-Level Decision Making: Soft Computing Approaches*, Studies in Fuzziness and Soft Computing368,
https://doi.org/10.1007/978-3-319-92525-7

Bard, J.F.: An investigation of the linear three level programming problem. IEEE Trans. Syst. Man Cybern. **14**(5), 711–717 (1984)
Bard, J.F., Falk, J.E.: An explicit solution to the multi-level programming problem. Comput. Oper. Res. **9**, 77–100 (1982)
Bard, J.F., Moore, J.T.: A branch and bound algorithm for the bilevel programming problem. SIAM J. Sci. Stat. Comput. **11**, 281–292 (1990)
Bard, J.F., Moore, J.T.: An algorithm for the discrete bilevel programming problem. Naval Res. Logistics **39**, 419–435 (1992)
Barnes, J.W., Laguna, M.: Solving the multiple machine weighted flow time problem using tabu search. IIE Trans. **25**(2), 121–128 (1993)
Bazaraa, M.S., Sherali, H.D., Shetty, C.M.: Nonlinear Programming Theory and Algorithms, 3rd edn. Wiley, New York (2006)
Beil, D.R., Wein, L.M.: An inverse-optimization-based auction mechanism to support a multi-attribute RFQ process. Manage. Sci. **49**(11), 1529–1545 (2003)
Bellman, R.E.: Dynamic Programming. Princeton University Press, Princeton, New Jersey (1957)
Bellman, R.E., Giertz, M.: On the analytic formalism of the theory of fuzzy sets. Inf. Sci. **5**, 149–157 (1973)
Bellman, R.E., Lee, E.S.: Functional equations in dynamic programming. Aequationes Math. **17** (1), 1–18 (1978)
Bellman, R.E., Zadeh, L.A.: Decision making in a fuzzy environment. Manage. Sci. **17**, B141–B164 (1970)
Ben-Ayed, O.: Bilevel linear programming. Comput. Oper. Res. **20**, 485–501 (1993)
Ben-Ayed, O., Blair, C.E.: Computational difficulties of bilevel linear programming. Oper. Res. **38**, 556–560 (1990)
Bialas, W.F., Karwan, M.H.: On two-level optimization. IEEE Trans. Autom. Control AC-27 211–214 (1982)
Bialas, W.F., Karwan, M.H.: Two-level linear programming. Manage. Sci. **30**, 1004–1020 (1984)
Bialas, W.F., Karwan, M.H., & Shaw J.: A parametric complementary pivot approach for two-level linear programming. Technical Report 80-2, State University of New York at Buffalo, Operations Research Program (1980)
Bichler, M.: An experimental analysis of multi-attribute auctions. Decision Support Syst. **29**(3), 249–268 (2000)
Bichler, M.: The Future of e-Markets: Multidimensional Market Mechanisms. Cambridge University Press, Cambridge (2001)
Bichler, M., Kalagnanam, J.: Configurable offers and winner determination in multi-attribute auctions. Eur. J. Oper. Res. **160**(2), 380–394 (2005)
Bichler, M., Kaukal, M., Segev, A.: Multi-attribute auctions for electronic procurement. In: Proceedings of the First IBM IAC Workshop on Internet Based Negotiation Technologies, vol. 3, pp. 18–19, Yorktown Heights, NY (1999)
Bracken, J., McGill, J.M.: Mathematical programs with optimization problems in the constraints. Oper. Res. **21**, 37–44 (1973)
Branco, F.: The design of multidimensional auctions. RAND J. Econ. **28**, 63–81 (1997)
Bruck, J., Sanz, J.: A study on neural networks. Int. J. Intell. Syst. **3**(1), 59–75 (1988)
Burke, L.I., Ignizio, J.P.: Neural network and operations research: an overview. Comput. Oper. Res. **19**, 179–189 (1992)
Burton, R.M.: The multilevel approach to organizational issues of the firm—a critical review. Omega **5**, 395–414 (1977)
Candler, W., Townsley, R.: A linear two-level programming problem. Comput. Oper. Res. **9**, 59–76 (1982)
Candler, W., Fortuny-Amat, J., McCarl, B.: The potential role multilevel programming in agricultural economics. Am. J. Agric. Econ. **63**(3), 521–531 (1981)
Cassidy, R., Kirby, M., Raike, W.: Efficient distribution ofresoumces through three levels of govemment. Manage. Sci. **17**, B462–B473 (1971)

Chang, P.T., Lee, E.S.: Fuzzy decision making: a survey, in between mind and computer: Fuzzy science and engineering. In: Wang, P.Z., Floe, K. (eds.), World Scientific Publishing Co., Singapore, pp. 139–182 (1993)
Che, Y.K.: Design competition through multidimensional auctions. RAND J. Econ. **24**, 668–680 (1993)
Chen, C.L., Chang, C.Y., Sun, D.Y.: Solving multi-objective dynamic optimization problems with fuzzy satisfying method. Optimal Control Appl. Methods **24**(5), 279–296 (2003)
Cheng, C.B.: Solving a sealed-bid reverse auction problem by multiple-criterion decision-making methods. Comput. Math. Appl. **56**(12), 3261–3274 (2008)
Cheng, C.B.: Reverse auction with buyer-supplier negotiation using bi-level distributed programming. Eur. J. Oper. Res. **211**(3), 601–611 (2011)
Cheng, C.B., Lo, C.Y.: Multi-project scheduling by fuzzy combinatorial auction. In: Proceedings of the 3rd IEEE International Conference on Cybernetics, pp. 1–6, Exeter, England, UK, 21–23 June (2017)
Cheng, C.B., Lai, Y.J., Chan, K.: Solving a reverse auction problem by bi-level distributed programming and genetic algorithm. Int. J. Revenue Manage. **5**(2–3), 234–260 (2011)
Cheng, C.B., Wu, H.C., Chan, C.H.: Design of a combinational auction mechanism for television advertising market in Taiwan. Res. J. Appl. Sci. Eng. Technol. **4**(20), 4105–4111 (2012)
Cho, K.I., Kim, S.H.: An improved interactive hybrid method for the linear multi-objective knapsack problem. Comput. Oper. Res. **24**(11), 991–1003 (1997)
Chong, E.K., Hui, S., Zak, S.H.: An analysis of a class of neural networks for solving linear programming problems. IEEE Trans. Autom. Control **44**(11), 1995–2006 (1999)
Cichocki, A., Unbehauen, R.: Neural Networks for Optimization and Signal Processing. Wiley, New York (1993)
de Vries, S., Vohra, R.: Combinatorial auctions: a survey. INFORMS J. Comput. **15**(3), 284–309 (2003)
de Werra, D., Hertz, A.: Tabu search techniques: a tutorial and an application to neural networks. Oper. Res. Spectr. **11**(3), 131–141 (1989)
Dell'Amico, M., Trubian, M.: Applying tabu search to the job-shop scheduling problem. Ann. Oper. Res. **41**(3), 231–252 (1993)
Dormady, N.: Carbon auction revenue and market power: an experimental analysis. Energies **9** (11), 897 (2016)
Du, K.L., Swamy, M.N.: Neural Networks in a Softcomputing Framework. Springer, London, UK (2006)
Easton, F.F., Moodie, D.R.: Pricing and lead time decision for make-to-order firms with contingent orders. Eur. J. Oper. Res. **116**(2), 305–318 (1999)
Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, pp. 39–43, Nagoya, Japan (1995)
El Khatib, S., Galiana, F.D.: Negotiating bilateral contracts in electricity markets. IEEE Trans. Power Syst. **22**(2), 553–562 (2007)
Emiliani, M.L.: Regulating B2B online reverse auctions through voluntary codes of conduct. Ind. Mark. Manage. **34**, 526–534 (2005)
Epstein, R., Henríquez, L., Catalán, J., Weintraub, G.Y., Martínez, C.: A combinational auction improves school meals in Chile. Interfaces **32**(6), 1–14 (2002)
Erlebach, T., Kellerer, H., Pferschy, U.: Approximating multiobjective knapsack problems. Manage. Sci. **48**(12), 1603–1612 (2002)
Esogbue, A.O., Bellman, R.E.: Fuzzy dynamic programming and its extensions. TIMS Stud. Manage. Sci. **20**, 147–167 (1984)
Fliege, J., Vicente, L.N.: Multicriteria approach to bilevel optimization. J. Optim. Theory Appl. **131**(2), 209–225 (2006)
Fortuny-Amat, J., McCarl, B.: A representation and economic interpretation of a two-level programming problem. J. Oper. Res. Soc. **32**, 783–792 (1981)

Fung, L.W., Fu, K.S.: An axiomatic approach to rational decision making in a fuzzy environment. In: Zadeh, L.A., Fu, K.S., Tanaka, K., Shimura, M. (eds.) Fuzzy Sets and their Applications to Cognitive and Decision Processes, pp. 227–256. Academic Press, NY (1975)
Gendreau, M., Marcotte, P., Savard, G.: A hybrid tabu-ascent algorithm for the linear bilevel programming problem. J. Global Optim. **8**(3), 217–233 (1996)
Gill, P.E., Murray, W., Wright, M.H.: Practical Optimization. Academic, London (1981)
Glover, F.: Future paths for integer programming and links to artificial intelligence. Comput. Oper. Res. **13**, 533–549 (1986)
Glover, F.: Tabu search—part I. ORSA J. Comput. **1**, 190–206 (1989)
Glover, F., Laguna, M., Marti, R.: Principles of tabu search (2008). https://www.researchgate.net/publication/228346477_Tabu_Search
Goicoechea, A., Hansen, D.R., Duckstein, L.: Multiobjective Decision Analysis with Engineering and Business Applications. Wiley, New York (1982)
Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley, Reading, MA (1989)
Green, R.J., Newbery, D.M.: Competition in the British electricity spot market. J. Polit. Econ. **100** (5), 929–953 (1992)
Ham, F.M., Kostanic, I.: Principles of Neurocomputing for Science and Engineering. McGraw-Hill, NY (2001)
Hansen, P., Jaumard, B., Savard, G.: New branch-and bound rules for linear bilevel programming. SIAM J Sci. Stat. Comput. **13**(5), 1194–1217 (1992)
Haurie, A., Savard, G., White, D.L.: A note on: an efficient point algorithm for a linear two-stage optimization problem. Oper. Res. **38**, 553–555 (1990)
Haykin, S.: Neural Networks: A Comprehensive Foundation, 2nd edn. Prentice-Hall, New Jersey (1999)
Hendry, L.C., Kingsman, B.G.: Customer enquiry management: part of a hierarchical system to control lead time in make-to-order companies. J. Oper. Res. Soc. **44**(1), 61–70 (1994)
Hertz, A., de Werra, D.: Using tabu search techniques for graph coloring. Computing **39**, 345–351 (1987)
Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan, Michigan, Ann Arbor, MI (1975)
Hopfield, J.J.: Neural networks and physical systems with emergent collective computational abilities. Proc. Nat. Acad. Sci. **79**(8), 2554–2558 (1982)
Hopfield, J.J.: Neurons with graded response have collective computational properties like those of two-state neurons. Proc. Nat. Acad. Sci. **81**(10), 3088–3092 (1984)
Hopfield, J.J.: Artificial neural networks. IEEE Circ. Devices Mag. **4**(5), 3–10 (1988)
Hopfield, J.J., Tank, D.W.: Neural computation of decisions in optimization problems. Biol. Cybern. **52**, 141–152 (1985)
Hwang, C.L., Masud, A.S.M.: Multiple Objectives Decision Making: Methods and Applications. Springer, Berlin (1979)
Hwang, C.L., Yoon, K.: Multiple Attribute Decision Making: Methods and Applications. Springer, Berlin (1981)
Jackson, C.: Technology for spectrum markets. Ph.D. thesis, Department of Electronical Engineering, Massachusetts Institute of Technology, Cambridge, MA (1976)
Judice, J.J., Faustino, A.M.: An experimental investigation of enumerative methods for the linear complementarity problem. Comput. Oper. Res. **15**(5), 417–426 (1988)
Judice, J.J., Faustino, A.M.: A sequential1cp method for bilevel linear programmig. Ann. Oper. Res. **34**, 89–106 (1992)
Kacprzyk, J.: Multistage Decision-making Under Fuzziness Theory and Applications. Verlag TUV Rheinland, Koln (1983)
Kacprzyk, J., Esogbue, A.O.: Fuzzy dynamic programming: main development and applications. Fuzzy Sets Syst. **81**(1), 31–45 (1996)

Kennedy, M.P., Chua, L.O.: Neural networks for nonlinear programming. IEEE Transp. Circ. Syst. **35**(5), 554–562 (1988)
Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
Kim, T.J., Suh, S.: Toward development a national transportation planning model: a bilevel programming approach for Korea. Ann. Reg. Sci. **12**, 65–80 (1988)
Kingsman, B., Hendry, L., Mercer, A., de Souza, A.: Responding to customer enquiries in make-to-order companies: problems and solutions. Int. J. Prod. Econ. **46–47**, 219–231 (1996)
Klamroth, K., Wieck, M.M.: Dynamic programming approaches to the multiple criteria knapsack problem. Naval Res. Logistics **47**(1), 57–76 (2000)
Klir, G.I., Folger, T.A.: Fuzzy Sets, Uncertainty, and Information. Prentice-Hall, Englewood, NJ (1988)
Kohonen, T.: Self-organized formation of topologically correct feature maps. Biol. Cybern. **43**(1), 59–69 (1982)
Kohonen, T.: An introduction to neural computing. Neural Netw. **1**(1), 3–16 (1988)
Kolstad, C.D.: A review of the literature on bi-level mathematical programming. Report no. LA-I0234-MS, Los Alamos National Laboratory, Los Alamos, New Mexico (1985)
Kolstad, C.D., Lasdon, L.S.: Derivative evaluation and computational experience with large bilevel mathematical programs. J. Optim. Theory Appl. **65**(3), 485–499 (1990)
Kumar, S.: Neural Networks: A Classroom Approach. McGraw-Hill (2004)
Kumar, S.: Neural Networks: A Classroom Approach. McGraw-Hill, Singapore (2005)
Kuo, R.J., Han, Y.S.: A hybrid of genetic algorithm and particle swarm optimization for solving bi-level linear programming problem-a case study on supply chain model. Appl. Math. Model. **35**(8), 3905–3917 (2011)
Kurita, N., Funahashi, K.I.: On the Hopfield neural networks and mean field theory. Neural Netw. **9**(9), 1531–1540 (1996)
Kutanoglu, E., David Wu, S.: On combinatorial auction and Lagrangean relaxation for distributed resource scheduling. IIE Trans. **31**(9), 813–826 (1999)
Lai, Y.J.: Hierarchical optimization: a satisfactory solution. Fuzzy Sets Syst. **77**(3), 321–335 (1996)
Lai, Y.J., Hwang, C.L.: Possibilistic linear programming for managing rate risk. Fuzzy Sets Syst. **54**(2), 135–146 (1993)
Lai, K.K., Li, L.: A dynamic approach to multi-objective resource allocation. Eur. J. Oper. Res. **117**(2), 293–309 (1999)
Lan, K.M., Wen, U.P., Shih, H.S., Lee, E.S.: A hybrid neural network approach to bilevel programming problems. Appl. Math. Lett. **20**(8), 880–884 (2007)
Lee, E.S., Li, R.J.: Fuzzy multiple objective programming and compromise programming with Pareto optimum. Fuzzy Sets Syst. **53**, 275–288 (1993)
Lee, E.S., Zhu, Q.: Fuzzy and Evidence Reasoning. Physica-Verlag, Heidelberg, Germany (1995)
Legillon, F., Liefooghe, A., Talbi, E.G.: Cobra: a cooperative coevolutionary algorithm for bi-level optimization. In: 2012 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8, Brisbane, Australia (2012)
Li, R.J.: Multiple objective decision making in a fuzzy environment. Ph.D. dissertation, Department of Industrial Engineering, Kansas State University, Manhattan, Kansas (1990)
Li, H.: A genetic algorithm using a finite search space for solving nonlinear/linear fractional bilevel programming problems. Ann. Oper. Res. **235**, 543–558 (2015)
Li, D., Haimes, Y.Y.: Multiobjective dynamic programming: the state of the art. Control Theory Adv. Technol. **5**(4), 471–483 (1989)
Li, R.J., Lee, E.S.: Fuzzy approaches to multicriteria de Novo programs. J Math. Anal. Appl. **153** (1), 97–111 (1990)
Li, X., Tian, P., Min, X.: A hierarchical particle swarm optimization for solving bilevel programming problems. In International Conference on Artificial Intelligence and Soft Computing, pp. 1169–1178. Springer, Berlin, Heidelberg (2006)

Lin, D.Y., Unnikrishnan, A., Waller, S.T.: A genetic algorithm for bi-level linear programming dynamic network design problem. Transp. Lett. **1**(4), 281–294 (2009)
LINGO, Lindo System Inc., Chicago (1992)
Liu, B.: Stackelberg-Nash equilibrium for multilevel programming with multiple followers using genetic algorithms. Comput. Math. Appl. **36**(7), 79–89 (1998)
Looi, C.K.: Neural network methods in combinatorial optimization. Comput. Oper. Res. **19**(3/4), 191–208 (1992)
Maa, C.Y., Shanblatt, M.A.: Linear and quadratic programming neural network analysis. IEEE Trans. Neural Netw. **3**, 380–394 (1992)
Malek, A., Tari, A.: Primal-dual solution for the linear programming problems using neural networks. Appl. Math. Comput. **167**(1), 198–211 (2005)
Malek, M., Guruswamy, M., Pandya, M.: Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem. Ann. Oper. Res. **21**(1), 59–84 (1989)
Marcotte, P.: Network design problem with congestion effects: a case of bilevel programming. Math. Program. **34**, 142–162 (1986)
Martello, S., Toth, P.: Knapsack Problems: Algorithms and Computer Implementations. Wiley, Chichester, West Sussex (1990)
Mathieu, R., Pittard, L., Anandalingam, G.: Genetic algorithm based approach to bi-level linear programming. Rech. Operationnelle **28**, 1–21 (1994)
McAfee, R.P., McMillian, J.: Auctions and bidding. J. Econ. Lit. **25**(2), 699–738 (1987)
McMillian, J.: Selling spectrum rights. J. Econ. Perspect. **8**(3), 145–162 (1994)
Migdalas, A.: Bilevel programming in traffic planning: models, methods and challenge. J. Global Optim. **7**, 381–405 (1995)
Milgrom, P.: Putting Auction Theory to Work. Cambridge University Press, Cambridge, UK (2004)
Oduguwa, V., Roy, R.: Bi-level optimisation using genetic algorithm. In: Proceedings of the 2002 IEEE International Conference on Artificial Intelligence Systems (ICAIS '02), pp. 322–327, Divnomorskoe, Russia (2002)
Onal, H.: A modified simplex approach for solving bilevel linear programming problems. Eur. J. Oper. Res. **67**(1), 126–135 (1993)
Park, S., Rothkopf, M.H.: Auctions with endogenously determined allowable combinations. RUTCOR Research Report 3-2001, Rutgers University, New Brunswick, NJ (2001)
Pramanik, S., Roy, T.K.: Fuzzy goal programming approach to multilevel programming problems. Eur. J. Oper. Res. **176**(2), 1151–1166 (2007)
Rassenti, S.J., Smith, V.L., Bulfin, R.L.: A combinatorial auction mechanism for airport time slot allocation. Bell J. Econ. **13**(2), 402–417 (1982)
Ricanek, K., Lebby, G.L, Haywood, K.: Hopfield like networks for pattern recognition with application to face recognition. In: IEEE International Joint Conference on Neural Network, vol. 5, pp. 3265–3269 (1999)
Rodríguez-Vázquez, A., Dominguez-Castro, R., Rueda, A., Huertas, J.L., Sánchez-Sinencio, E.: Switched-capacitor neural networks for linear programming. Electron. Lett. **24**(8), 496–498 (1988)
Rodríguez-Vázquez, A., Domínguez-Castro, R., Rueda, A., Huertas, J.L., Sánchez-Sinencio, E.: Nonlinear switched-capacitor 'neural networks' for optimization problems. IEEE Trans. Circ. Syst. **37**, 384–397 (1990)
Rothkopf, M.H., Pekeč, A., Harstad, R.M.: Computationally manageable combinatorial auctions. Manage. Sci. **44**(8), 1131–1147 (1998)
Saaty, T.L.: The Analytic Hierarchical Process, 2nd edn. RWS Publications, Pittsburgh, PA (1990)
Salkin, H.M., De Kluyver, C.A.: The knapsack problem: a survey. Naval Res. Logistics **22**(1), 127–144 (1975)
Salman, F.S., Kalagnanam, J.R., Murthy, S., Davenport, A.: Cooperative strategies for solving the bicriteria sparse multiple knapsack problem. J. Heuristics **8**(2), 215–239 (2002)

Salukvadze, M.: An approach to the solution of the vector optimization problem of dynamic systems. J. Optim. Theory Appl. **38**(3), 409–422 (1982)
Sandholm, T.: Approaches to winner determination in combinatorial auctions. Decis. Support Syst. **28**(1–2), 165–176 (2000)
Savard, G., Gauvin, J.: The steepest descent direction for the nonlinear bilevel programming problem. Oper. Res. Lett. **15**(5), 265–272 (1994)
Shafer, G.: A Mathematical Reasoning of Evidence. Princeton University Press, Princeton, NJ (1976)
Sharda, R.: Neural networks for the MS/OR analyst: an application bibliography. Interfaces **24**(2), 116–130 (1994)
Shi, Y., Eberhart, R.C.: A modified particle swarm optimizer. In: Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 303–308 (1997)
Shih, H.S.: Fuzzy approach to multilevel knapsack problems. Comput. Math. Appl. **49**(7–8), 1157–1176 (2005)
Shih, H.-S., Lee, E.S.: Fuzzy multi-level minimum-cost flow problems. Fuzzy Sets Syst. **107**(2), 159–176 (1999)
Shih, H.S., Lee, E.S.: Discrete multi-level programming in a dynamic environment. In: Yoshida, Y. (ed.) Dynamic Aspects in Fuzzy Decision Making, Studies in Fuzziness and Soft Computing, vol. 73, pp. 79–98. Physica-Verlag, Heidelberg (2001)
Shih, H.S., Lai, Y.J., Lee, E.S.: Fuzzy approach for multi-level mathematical programming problems. Comput. Oper. Res. **23**(1), 73–91 (1996)
Shih, H.S., Wen, U.P., Lee, E.S., Lan, K.M., Hsiao, H.C.: A neural network approach to multiobjective and multilevel programming problems. Comput. Math. Appl. **48**(1–2), 95–108 (2004)
Shimizu, K., Ishizuka, Y., Bard, J.F.: Nondifferentiable and Two-level Mathematical Programming. Kluwer Academic Publishers, Boston (1997)
Shirazi, B., Yih, S.: Critical analysis of applying Hopfield neural net model to optimization problems. In: IEEE International Conference on Systems, Man and Cybernetics, vol. 1, pp. 210–215, 14–17 Nov (1989)
Smart, A., Harrison, A.: Reverse auctions as a support mechanism in flexible supply chains. Int. J. Logistics Res. Appl. **5**(3), 275–284 (2002)
Smith, K.A.: Neural networks for combinatorial optimization: a review on more than a decade of research. INFORMS J. Comput. **11**(1), 15–34 (1999)
Smith, K.A., Gupta, J.N.D.: Neural networks in business: techniques and applications for the operations research. Comput. Oper. Res. **27**(11/12), 1023–1044 (2000)
Sugeno, M., Takagi, T.: Multi-dimensional fuzzy reasoning. Fuzzy Sets Syst. **9**, 313–325 (1983)
Syswerda, G.: Uniform crossover in genetic algorithms. In: Proceedings of 3rd International Conference on Genetic Algorithms, pp. 2–9 (1989)
Taha, H.A.: Operations Research: An Introduction, 10th edn. Pearson Education, Upper Saddle River, New Jersey (2016)
Talbi, E.G.: A taxonomy of metaheuristics for bi-level optimization. In: Talbi, E.-G. (ed.) Metaheuristics for Bi-Level Optimization. Springer, Berlin (2013)
Tank, D., Hopfield, J.J.: Simple 'neural' optimization networks: an A/D converter, signal decision network and a linear programming circuit. IEEE Trans. Circuits Syst. **33**(5), 533–541 (1986)
Teich, J.E., Wallenius, H., Wallenius, J., Koppius, O.R.: Emerging multiple issue e-auctions. Eur. J. Oper. Res. **159**(1), 1–16 (2004)
Tully, S.: The B2B tool that really is changing the world. Fortune **141**(6), 132–137 (2000)
Vicente, L.N., Calarmai, P.H.: Bilevel and multilevel programming: a bibliography review. J. Global Optim. **5**(3), 291–306 (1994)
Vickrey, W.: Counterspeculation, auctions and competitive sealed tenders. J. Finance **16**(1), 8–37 (1961)

Visee, M., Teghem, J., Pirlot, M., Ulungu, E.L.: Two-phrases method and branch and bound procedures to solve the bi-objective knapsack problem. J. Global Optim. **12**(2), 139–155 (1998)
Wang, J.: Primal and dual assignment networks. IEEE Trans. Neural Netw. **8**(3), 784–790 (1997)
Wang, J.: Primal and dual neural networks for shortest-path routing. IEEE Trans. Syst. Man Cybern. Part A Syst. Humans **28**(6), 864–869 (1998)
Wang, Z.-W., Nagasawa, H., Nishiyama, N.: A method for generating nondominated solutions to a multiobjective-headquarters, three-level decentralized system. Comput. Ind. Eng. **27**(1–4), 405–408 (1994)
Wemers, B.M.: Aggregation models in mathematical programming. In: Mitra, G. (ed.) Mathematical Models for Decision Support, pp. 295–305. Springer, Berlin (1988)
Wen, U.P.: Mathematical methods for multilevel linear programming. Ph.D. thesis, State University of New York, Buffalo, NY (1981)
Wen, U.P., Bialas, W.F.: The hybrid algorithm for solving the three-level linear programming problem. Comput. Oper. Res. **13**(4), 367–377 (1986)
Wen, U.P., Hsu, S.T.: Linear bi-level programming problems—a review. J. Oper. Res. Soc. **42**, 125–133 (1991)
Wen, U.P., Huang, A.D.: A simple tabu search method to solve mixed-integer linear bilevel programming problem. Eur. J. Oper. Res. **88**, 563–571 (1996)
Wen, U.P., Lan, K.-M., Shih, H.-S.: A review of Hopfield neural networks for solving mathematical programming problems. Eur. J. Oper. Res. **198**, 675–687 (2009)
White, D.J.: A penalty function approach for solving bi-level linear programs. J. Global Optim. **3**, 1397–1419 (1993)
White, D.J., Anandalingam, G.: A penalty function approach for solving bi-leve1linear programs. J. Global Optim. **3**(4), 394–419 (1993)
Widmer, M.: Job shop scheduling with tooling constraints: a tabu search approach. J. Oper. Res. Soc. **42**(1), 75–82 (1991)
Widmer, M., Hertz, A.: A new heuristic method for the flow shop sequencing problem. Eur. J. Oper. Res. **41**(2), 186–193 (1989)
Widrow, B., Rumelhart, D.E., Lehr, M.A.: Neural networks: applications in industry, business and science. Commun. ACM **37**(3), 93–106 (1994)
Wolfe, P.: Simplex method for quadratic programming. Econometrica **27**, 382–398 (1959)
Wong, B.K., Lai, V.S., Lam, J.: A bibliography of neural networks in business: techniques and applications research: 1994-1998. Comput. Oper. Res. **27**(11–12), 1045–1076 (2000)
Xia, Y.: A new neural network for solving linear programming and its application. IEEE Trans. Neural Netw. **7**(2), 525–529 (1996)
Xia, Y., Wang, J.: Neural network for solving linear programming problem with bound variables. IEEE Trans. Neural Netw. **6**(2), 515–519 (1995)
Xia, Y., Wang, J.: A general methodology for designing globally convergent optimization neural networks. IEEE Trans. Neural Netw. **9**(6), 1331–1343 (1998)
Xia, Y.S., Feng, G., Wang, J.: A primal–dual neural network for online resolving constrained kinematic redundancy in robot motion control. IEEE Trans. Syst. Man Cybern. Part B Cybern. **35**(1), 54–64 (2005)
Yaakob, S.B., Watada, J.: Solving bilevel programming problems using a neural network approach and its application to power system environment. SICE J. Control Meas. Syst. Integr. **4**(6), 387–393 (2011)
Yen, J., Langari, R.: Fuzzy Logic: Intelligence, Control, and Information. Prentice Hall, Upper Saddle River, NJ (1999)
Zadeh, L.A.: Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets Syst. **1**(1), 328 (1978)

Zadeh, L.A.: In: Zadeh, L.A., Yager, R.R., Ovchinnikov, S., Tong, R.M., Nguyen, H.T. (eds.) Fuzzy Sets and Applications: Selected Papers. Wiley, New York (1987)
Zadeh, L.A.: Fuzzy logic, neural networks, and soft computing. Commun. ACM **37**(3), 77–84 (1994)
Zhang, X.S.: Neural Networks in Optimization. Kluwer Academic, London (2000)
Zhang, S., Constantinides, A.G.: Lagrange programming neural networks. IEEE Trans. Circ. Syst. II Analog Digital Signal Proc. **39**(7), 441–452 (1992)
Zhang, H.C., Huang, S.H.: Applications of neural networks in manufacturing: a state-of-the-art survey. Int. J. Prod. Res. **33**(3), 705–728 (1995)
Zhao, Z., Gu, X.: Particle swarm optimization based algorithm for bilevel programming problems. In: Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications (ISDA'06), pp. 951–956, Jinan, China (2006)
Zhensu, L.V., Zhirong, H.: Particle swarm optimization with adaptive mutation. Acta Electronica Sinica. **32**(3), 416–420 (2004)
Zhu, Q.: Evidential reasoning in logic systems. Ph. D. dissertation, Kansas State University, Manhattan, Kansas (1992)
Zimmermann, H.-J.: Fuzzy programming and linear programming with several objective functions. Fuzzy Sets Syst. **1**, 45–55 (1978)
Zimmermann, H.-J.: Fuzzy Set Theory and Its Applications, 3rd edn. Kluwer, Boston (1996)
Zimmennann, H.-J., Zysno, P.: Latent connectives in human decision making. Fuzzy Sets Syst. **4**(1), 37–51 (1980)
Zurada, J.M.: Introduction to Artificial Neural Systems. West, NY (1992)