Model-Based Decision Support Methodology with Environmental Applications

Andrzej P. Wierzbicki, Marek Makowski, and Jaap Wessels

Editors

International Institute for Applied Systems Analysis
Laxenburg, Austria

Contents

Contributors

Markus Amann has been leader of IIASA's Transboundary Air Pollution project since 1991. The project provided scientific support to the 1994 negotiations on the Second Sulfur Protocol of the Convention on Long-range Transboundary Air Pollution, and provided support for other negotiations as summarized in Chapter 13. His research interests include real-world applications of decision support and optimization techniques and the science-policy interaction.
Web page: http://www.iiasa.ac.at/Research/TAP/
E-mail: amann@iiasa.ac.at

Günther Fischer has been leader of the project on Modeling Land Use and Land Cover Changes in Europe and Northern Asia at IIASA since 1995. He participated in the formulation, implementation, and application of a global model of the world food systems, known as IIASA's Basic Linked System. Günther Fischer also collaborates with the United Nations Food and Agriculture Organization (FAO) on the development, implementation, and application of FAO's Agro-ecological Zones (AEZ) methodology to national and regional resource appraisals.
Web page: http://www.iiasa.ac.at/Research/LUC/
E-mail: fisher@iiasa.ac.at

Janusz Granat leads the Decision Support System Laboratory of the National Institute of Telecommunications, Warsaw, Poland and he is a lecturer at the Institute of Control and Computation Engineering, Warsaw University of Technology. His scientific interests include the theory of multicriteria optimization, modeling, and decision support systems, and applications of decisions support systems in the telecommunications industry. He contributed to the development of the decision support system of the DIDAS family, implemented part of the IAC-DIDAS-N as well as the IAC-DIDASN++. He also developed the ISAAP module used in several applications described in this book. He has been collaborating with IIASA since 1988.
Web page: http://www.ia.pw.edu.pl/∼janusz
E-mail: J.Granat@ia.pw.edu.pl

Marek Makowski is a senior research scholar with the Risk, Modeling and Policy (RMP) Project at IIASA, on leave from the Systems Research Institute of the Polish Academy of Sciences, Warsaw, Poland. His research interests include the development of methodology, algorithms, and software for model-based decision support, especially for the generation of large and complex models and their analysis using

multicriteria optimization techniques. He had been actively participating in various joint activities with IIASA projects since 1976 until he joined IIASA in 1987; he was acting leader (1989–1991) and later a member of the Methodology of Decision Analysis Project. He has been collaborating with several IIASA projects in the development of various applications – three of them presented in this book – and developed several modular software tools described in this book.
Web page: http://www.iiasa.ac.at/∼marek
E-mail: marek@iiasa.ac.at

Sabine Messner is a senior research scholar with the Environmentally Compatible Energy Strategies Project at IIASA, and a consultant and systems analyst for TEMAPLAN GMBH, Vienna, Austria. Her special interests are in the development of long-term energy scenarios, integration of demand- and supply-side measures in energy systems models, endogenization of technological experience curves in long-term global energy models, stochastic optimization approaches, and decision support for the deregulated energy markets.
Web page: http://www.temaplan.at
E-mail: messner@iiasa.ac.at, messner@temaplan.at

Hirotaka Nakayama is professor of Applied Mathematics, Konan University, Kobe, Japan. His research interests include multi-objective optimization, knowledge discovery techniques, applications of (multi-objective) optimization techniques, and applications of machine learning. He has been collaborating with various methodological projects at IIASA for more than 20 years.
E-mail: nakayama@konan-u.ac.jp

Jerzy Paczyński is deputy director of the Institute of Control and Information Technology, Warsaw University of Technology. His current research interests concentrate on different aspects of modeling languages and applications of different programming paradigms. He has cooperated with IIASA since 1985 and contributed to the development of the decision support system of the DIDAS family as well as designing the MDL language and its compiler used in the DIDAS-N++ system.
E-mail: J.Paczynski@ia.pw.edu.pl

László Somlyódy is professor and head of the Sanitary and Environmental Engineering Department, Budapest University of Technology, Budapest, Hungary. His scientific interests include mechanical engineering, water quality and environmental management, systems analysis and mathematical modeling of lakes, rivers, reservoirs, and wastewater treatment. He was the leader of IIASA's Water Resources Project (1992–1996) and head of the Global Environmental Change Program (1992–1994).
E-mail: somlyody@vcst.bme.hu

Manfred Strubegger is a senior research scholar with the Environmentally Compatible Energy Strategies Project at IIASA, and a consultant and systems analyst for TEMAPLAN GMBH, Vienna, Austria. He specializes in data bank development and energy-related modeling issues. His special focus is on decision support for regional energy planning including multi-objective optimization, energy demand modeling,

support for decision makers from restructuring economies or developing countries, and the design of technology-related data bank systems.
Web page: www.temaplan.at
E-mail: strub@iiasa.ac.at, strub@temaplan.at

Jaap Wessels is full professor of stochastic operations research at Eindhoven University of Technology (Faculty of Mathematics and Computing Science). He is a fellow of BETA (research institute for operations management) and coordinator of the research program stochastic networks of EURANDOM (European research institute for stochastic mathematics). His mathematical research focuses on queueing theory, Markov decision processes and neural nets. In addition he is interested in decision support methodology and he applies his favorite research topics in operations management, communication processes, manpower policy making and environmental problems. Jaap Wessels was leader of IIASA's Methodology of Decision Analysis Project from 1991 to 1996.
Web pages: win.tue.nl, tm.tue.nl/beta, eurandom.tue.nl
E-mail: wessels@win.tue.nl

Andrzej Wierzbicki is the director of the National Institute of Telecommunications, Warsaw, Poland, and professor of Optimization and Decision Theory Institute of Control and Computation Engineering, Warsaw University of Technology, Warsaw, Poland. His current interests include the development of information society and civilization with applications of telematics methods and various aspects of multiple criteria analysis and decision support such as parallelization of optimization algorithms using multiple criteria approaches, applications of fuzzy set theory for describing uncertainty in decision support models, convergence of interactive approaches to decision support, multiobjective modeling and simulation, etc. From 1979 to 1984 he was chairman of IIASA's Systems and Decision Sciences Program. Since 1985 he has been organizing and leading a group of scientists from several institutes in Poland who have participated in various collaborative activities with IIASA resulting in the development of methodologies and software, many of them described in this book.
Web-page: http://www.itl.waw.pl
E-mail: a.wierzbicki@itl.waw.pl

Acknowledgments

The editors of this book would like to express their thanks for the support received while writing this book. Much of this support came from the International Institute for Applied Systems Analysis (IIASA) and from many friends there, but in particular from two consecutive Directors: Peter de Jánosi and Gordon MacDonald, who encouraged the authors to proceed with this book.

Thanks are also due to all the co-authors who cooperated on many chapters of this book. These are (in alphabetical order): Markus Amann, Günther Fischer, Janusz Granat, Sabine Messner, Hirotaka Nakayama, Jerzy Paczyński, László Somlyódy, and Manfred Strubegger, whose expertise and collaboration helped the editors significantly.

Many other friends helped us to develop our ideas and concepts; we mention here Yuri Ermoliev, Robert Fourer, Arthur Geoffrion, Jacek Gondzio, Włodzimierz Ogryczak, Andrzej Ruszczyński, Jerzy Sobczyk, Janusz Sosnowski, Artur Świętanowski, Grzegorz Wójcik, and Krystian Zorychta.

Finally, IIASA's Publication Department was indispensible for giving this book its final form. In particular, we acknowledge the support of Ewa Delpos, Jane S. Marks, Eryl Maedel, and Lilo Roggenland.

Without the help of all these friends, we would not have succeeded.

Aim and Audience

This book is self-contained in issues related to model-based decision support and multi-objective optimization. Readers will also find in it a review of modeling techniques or tools, and single-objective optimization methods and tools – though both these subjects are worthy of separate volumes.

The book consists of three parts. The first part provides the methodological background and describes various features of the decision-making environment and the ways in which decision support methodology can cope with that environment. We concentrate mostly on the relation between basic processes and possible decisions.

The second part treats various types of decision support tools and is more technical in nature. Several of the tools that are described are available from IIASA free of charge for use in research and teaching.

The third part considers how the methodology is used for environmental applications. Each chapter has one or more co-author who is an expert in the research area and skilled in modeling for this area. Also, many of the co-authors are experts in decision support methodology.

The book will be of interest to a wide range of specialists in various fields such as ecology and environmental sciences, including environmental engineering and energy systems engineering. It will also be of interest to readers with a background in economics and business administration. We try to present the material as simply as possible, but this is not a textbook for undergraduates. It requires a sound background in mathematics and a reasonable level of expertise in using computers creatively. Thus, we aim this book at a graduate student who – while preparing for a masters or doctoral thesis – works on computerized models of any disciplinary background (mostly of an analytical or operation research type, though we depart from the classical OR tradition) and wonders how to analyze and use them most effectively. The book might also interest operations research or decision analysis specialists, who develop methods and techniques that are aimed at model development and analysis (especially simulation- and/or optimization-based) for decision support.

Introduction

Jaap Wessels and Andrzej P. Wierzbicki

Decision making is a major component of living and, therefore, a fascinating topic for discussion and investigation. Several fields in science also occupy themselves with the nature of different aspects of decision making: philosophy, psychology, sociology, economics, etc. There are many types of decision making and all types are equally intriguing, e.g., How do people react when they are threatened? How should one select a spouse? What is the most appropriate factory layout? Which markets would be most profitable? In this short list we already recognize the distinction between individual decision making and institutional decision making. In the last 50 to 60 years, institutional decision making has attracted much attention, particularly for decisions regarding the design and control of military operations. Later, similar ideas were used and extended for decisions in trade and industry. Policy making in the public and semi-public sectors of society also began to be based more on systematic analysis.

In most institutional decision-making problems, there are three main aspects of concern:

- The information about the current situation and, possibly, about the past.
- The processes that are to be influenced by the decisions.
- The actual decision-making process.

For instance, decision making concerning acid rain illustrates the above three aspects clearly, as follows:

- The information consists of huge amounts of data regarding industrial, agricultural, and automotive activities. It also consists of data about wind and sun activities and about the types of power stations used.
- The processes to be influenced by the decisions consist of the production and emission of polluting material, together with the atmospheric transformation (chemical reactions and transportation) and the deposition process. Therefore the processes are basically of a physical and economic nature.
- The decision-making process consists of several interacting subprocesses at the local, national, and international levels.

In the history of decision support and decision analysis, one sees that many tools and methods have been developed to help make decisions. We also see that most of these methods and tools concentrate on one of the three main aspects of institutional decision-making problems. For instance, in most college textbooks dealing with operations research, we find several generic models describing the relation of the basic processes with the possible decisions. Each model is usually presented together with at least one technique for constructing a most favorable decision. Linear programming models and the simplex method provide the most classical example.

The first aspect, information, has been the source of inspiration for particular types of information systems that consist of one or more data bases and special methods for arranging the information. Typical examples are geographic information systems (GIS) and management information systems (MIS), each with their own way of structuring and storing the information and with their own way of dealing with the information.

Finally, the decision-making process has inspired several approaches to the structuring of the process of reaching decisions and ways of comparing different alternatives. The process of reaching decisions may be complex because the decisions themselves are complex and consist of many subdecisions, which should be taken in the right order (decision trees can be an appropriate tool to model this type of situation). The process of reaching decisions may also be complex because of the number of persons, departments and other groups involved, each with their own interests, constraints, and ambitions. In such cases, group decision-making methods or even negotiation support can be helpful.

One of the most striking features of the existing decision support methodology is that most tools and methods concentrate entirely on one of the three aspects and often only on a part of that aspect. This is understandable, because often it is difficult to integrate the other aspects. For instance, GISs are able to handle a wealth of information in a very user-friendly and enlightening way. However, it is difficult to translate this information for the basic processes to models in order to use this information for deriving consequences of possible decisions. A direct linkage is possible only if the process models are closely related to the information structure, as in guiding car-systems, where an optimal route is indicated on a map. This decision selection can easily be adapted if information about traffic jams or blocked bridges becomes available.

For more strategic decisions, however, it is unlikely that the process models relate so closely to the information structure. Even if something can be done, the approach is probably specific to a particular problem and therefore not generally applicable. One reason for this situation is that there are accepted standards for keeping information in data bases, but there are no accepted standards for model building.

Quite often, methods are implemented as if the designers are not even aware of the other aspects. As a result, implementations have been developed that are practically infeasible. For example, approaches to modeling and optimization of the basic processes have been commonly advertised that do not fit with the way information becomes available, or, more seriously, with the way the decisions are chosen.

Many institutional decision processes relate to well-defined processes of a technical, physical, or economic nature. This is particularly true for environmental problems, like the acid rain problem. In such cases there is usually a lot of knowledge about the basic processes and about the way they are influenced by possible decisions. However, not all institutional decision processes possess this property. Consider, for instance, the structuring of the funding organization for research in Poland. In such a case there is certainly a basic process to be influenced by the choices to be made, namely, the research process in Poland including quality, quantity, and distribution of research topics. However, there is not much accepted hard knowledge about the basic process and the core of the decision problem consists of a careful structuring of the influences of relevant institutions and of the decision process.

On the other hand, if there is a well-defined basic process and if the core problem essentially consists of deciding how to influence that basic process, then it is obvious that the starting point for any reasonable approach should be the modeling of the basic process, including the changes caused by possible decisions. As stated above, environmental policy problems usually belong to this class. The aim of this book is to present a decision support methodology for decision making concentrated around the basic processes, taking into account the information aspect and, particularly, the process of reaching decisions.

The methodology presented in this book is largely inspired by the experience at IIASA, the International Institute for Applied Systems Analysis. IIASA provided several environmental decision problems that helped develop our concepts of decision support. IIASA also provided the expertise about the basic processes together with the knowledge about the "decision environment". Many of the tools for implementing our concepts have been developed by several Polish groups of colleagues in a long-lasting cooperation with IIASA.

Methodology

In the first part of this book we consider the circumstances under which decision support methods should fulfill their tasks and also what these tasks are. From these considerations, we derive rules for constructing decision support systems. We concentrate on model-based decision support, that is, the relation between basic processes and possible decisions. However, the circumstances under which such decision support methods should work are highly co-determined by the available information and by the decision-making process. Therefore, these aspects strongly determine the rules for constructing decision support systems.

In Chapter 1, the concept of model-based decision support is introduced. Chapter 2 discusses the concept of a modern decision maker, the role of intuition versus decision support in decision making, the dynamics and phases of decision processes, experiences with decision support systems, and the position and way-of-operating of potential users of such systems. In Chapter 3, some basic concepts related to the architecture and implementation of decision support systems are presented. Chapter 4 shows how to use decision support systems for learning and for supporting decision making, while stressing multi-objective optimization and the reference point methodology developed at IIASA. Finally, Chapter 5 concentrates on the various paradigms of model analysis made possible by this methodology and applicable for model-based decision support.

Decision Support Tools

The second part of the book concentrates on more technical aspects of the basic types of tools needed in model-based decision support. These include tools for modeling, simulation, optimization, tools supporting choice, and user interfaces.

Chapter 6 describes modeling tools, both standard, commercially available, and nonstandard, together with tools for model simulation – in the classical sense as well as in a more advanced sense of inverse and softly constrained simulation. Chapter 7 gives a review of optimization tools, not only in the sense of solving single-objective optimization problems, but also as tools of multi-objective model analysis, advanced simulation techniques, etc. Chapter 8 presents in more detail some mathematical background, concepts, and tools for multi-objective optimization and reference point methodology, more generally presented in Chapter 4.

Chapter 9 describes various tools for supporting the choice of a preferred decision either for automating choice, generating options for interactive choice, or using a convergent interactive decision-support procedure. Chapter 10, finally, describes some user interfaces of decision support systems.

Environmental Applications

All four environmental applications presented in the third part of the book rely on many years of cooperation between the Methodology of Decision Analysis project at IIASA and various other – mostly environmental – projects at IIASA. They are all characterized by an intensive use of model-based decision support.

Chapter 11 discusses issues of river basin water quality management based on a multi-objective model analysis, as developed at IIASA for use in the decision-making support for some Central European river basins. Chapter 12 presents model-based methods of land-use planning, developed at IIASA in collaboration with the Food and Agriculture Organization of the United Nations (FAO). Chapter 13 describes issues of assessment of air quality, transboundary air pollution abatement, and related problems; models developed at IIASA in this field were used as a basis for international negotiations on pollution control. Chapter 14 presents an overview of various applications of model-based decision support in energy planning with environmental implications, as developed at IIASA in cooperation with several national and international authorities. Finally, the Epilogue presents conclusions and the lessons learned from the practice of using model-based decision support.

Appendix

This contains a short description of some of the software tools described in the book that are available from IIASA, free of charge, for research and educational purposes.

Part I

Methodological Background

Chapter 1

Model-Based Decision Support

Jaap Wessels and Andrzej P. Wierzbicki

In the introduction we mentioned the following three dominating aspects of decision making: (i) information about the current situation and history; (ii) the relation between basic processes and actions or decisions; and (iii) the decision process.

For each of these aspects, it is easy to argue that one aspect is more important and that the other two have a supporting position. This type of discussion is redundant. For each of these aspects, there are cases where that aspect is the most complicated and, therefore, requires most attention. However, for many problems all three aspects are essential. The acid rain problem, mentioned in the introduction, is a good example. In such highly complex decision situations, one needs good tools for handling any of the three aspects, and these tools should facilitate an integrated treatment of all three aspects.

There are many definitions of decision support systems (see e.g., Andriole, 1989), but generally it is agreed that they differ from information processing systems in at least two ways: they should contain models of selected decision situations, and they should support several phases of the decision process. The first aspect is simple, but without the other aspects we could not speak about decision support.

In this chapter, we will introduce and compare approaches for each of the different aspects (Sections 1.1 and 1.2). In Sections 1.3 and 1.4, various types of model-based decision support will be treated. Section 1.5 will emphasize the integration of the second aspect with the other aspects. This topic will be elaborated in more detail in subsequent chapters.

1.1 Aspect-Oriented Approaches

It is not laziness that made people isolate one aspect or even one subaspect of decision making and develop a decision support method entirely for this one. On the contrary, it is a well-established research strategy to concentrate on a particular aspect at the microlevel in order to avoid studying a world where everything influences everything else. And indeed, in this case, the strategy has worked. By isolating one aspect or subaspect it has been possible to develop very elegant and powerful methods of analysis. Fortunately, the protagonists of such methods were even able to produce problems in which their aspect or subaspect was the most dominant one. As a result, their method was highly effective.

We still recognize this feature in college texts where the applications fit marvelously to the methods. Regrettably, however, reality has become more and more demanding over the course of time. As a consequence, in most institutional decision problems, all three aspects are complicated and somehow dominant. Therefore, we need methods that integrate all three aspects. However, in developing integrated methods, we should not lose sight of how useful the divide-and-conquer strategy has been, and how poor the results have been when using general approaches for every problem. Hence, the most sensible way of dealing with decision making consists of exploiting our knowledge about aspect-oriented approaches and of modifying those approaches in such a way that they can be fitted together somehow.

Therefore, it is appropriate to start this book with a short overview of aspect-oriented approaches in order to be able later to consider the actual problem.

1.1.1 Information about the current situation and history

The information aspect has always been intriguing for computer scientists, as this aspect is most directly related to the use of computers. From an early stage information processing systems developed strongly in conjunction with computing technology. A well-presented set of data can be very helpful for a better understanding of a decision problem, or the current difficulties of a decision maker. Quite often, such an understanding naturally leads to sensible decisions.

Any information-oriented approach to decision support essentially relies on two factors:

- Methods of storing relevant information.
- Methods of processing the stored data in order that the information can be presented in a meaningful way.

Of course there is a relation between the two. For instance, in several situations, a meaningful way of presentation requires that different views on the data can be presented simultaneously or quickly after each other. This requirement constrains the ways in which data can be stored and processed.

In the course of time, data bases and data base management systems including query languages have matured. Nevertheless, for many decision situations, these tools are not able to provide the right information in the right way. There are mainly two reasons for this phenomenon:

- For some types of information, the usual data base technology does not provide the appropriate way of storing; this is typically the case for geographic information and for information regarding process mechanisms, e.g., time-dependent data.
- For many decision support purposes, the data base management technology does not provide the required flexibility and efficiency; this is typically the case for supporting tactical or strategic management decisions, but also for geographic information and other similar cases.

As a result, there has been and still is considerable effort spent in designing dedicated or special information systems for particular situations. Examples of the latter are geographic information systems (GIS), management information systems (MIS), and executive information systems (EIS).

Apart from the information systems that are made for particular decision situations, one can discern two types of information systems that are relevant with respect to decision support.

There are systems made to be used in a particular type of decision situation. Management information systems and executive information systems are good examples. Such information systems usually bear heavily on standard data base management technology.

In contrast, some types of information systems have been developed to handle particular data, which cannot be handled efficiently and effectively by traditional data base technology. Here geographic information and time-dependent data provide good examples. In such cases, dedicated data-storing and data-management methods have to be developed.

1.1.2 Relation between basic processes and actions

The relation between basic processes and actions constitutes an aspect that has always been the most intriguing for “hard” scientists. This is the aspect that is most directly related to the well-established theories of the sciences.

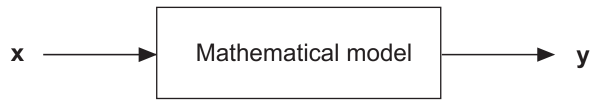


Figure 1.1. A mathematical model gives the resulting outcome $\boldsymbol{y}$ if it is fed with the decision parameter $\boldsymbol{x}$.

In most environmental decision problems, several of the basic processes are of a physical, chemical, or biological nature. For these processes there are usually extensive theories resulting in mathematical models for the processes. In such models we always find some parameters that are influenced, directly or indirectly, by decisions. For instance, in the river-basin water quality problem of Chapter 11, one may describe the water quality by equations that depend on process parameters and on emission parameters; the latter can be influenced by the decision makers. The emission parameters indicate where polluting materials are dropped in the river and in what quantities; the process parameters indicate how the water streams and the polluting materials change over time and place because of physical, chemical, and biological impulses. The process parameters might be influenced by changing the course of the river – by canalyzing it, by building dams, or by ensuring that it flows in a more natural bedding.

Consequently, a natural approach for describing the relation between basic processes and actions is to establish a mathematical model that uses some parameters that are related to the decisions as inputs and some characteristics of the resulting process as outputs. In our water example, the inputs could be the amounts of polluting material that are emitted at fixed places along the river, and the outputs could be the water quality and availability at inlets with drinking-water facilities.

This might be a natural approach, but it does not mean that the scientific knowledge provides such a model directly. What we see quite often is that science provides some sort of relation between the decisions and the interesting quantities, for instance, in the form of a set of equations or in the form of a differential equation. As a consequence, the mathematical model of *Figure 1.1* should not only contain the aforementioned relation, but also a numerical algorithm to compute, for any particular input $\boldsymbol{x}$, the output $\boldsymbol{y}$.

However, the situation may be more complicated. It quite often occurs that we do not have an explicit relationship between decision $\boldsymbol{x}$ and output $\boldsymbol{y}$, but that we only have a model of the way the process works. In that case, we can simulate the process for a given decision $\boldsymbol{x}$ and observe the related value of $\boldsymbol{y}$. In other cases, the available knowledge describing the relations between decisions and their outcomes is too detailed and complicated in most parts, with visible gaps in other parts. We must rely then on the art and methodology of model building – that is, on forming hypotheses to select model parts that approximate knowledge that is too complicated and how to extrapolate over gaps in knowledge.

Anyhow, let us suppose for a moment that we have a mathematical model of some sort, which is able to produce the output $\boldsymbol{y}$ related to any input $\boldsymbol{x}$. Throughout this book we shall use $\boldsymbol{x}$ to denote decisions or related decision variables or parameters, and $\boldsymbol{y}$ to denote decision outcomes or related outcome variables; some of the outcome variables can be later selected as objectives or criteria of decision choice.

The remaining problem is to find out how we can exploit these mathematical concepts. Note that $\boldsymbol{x}$ (and $\boldsymbol{y}$) might be a number, but, more likely, it is a vector or even a vector function of time. For our purposes it is most convenient to think of a vector $\boldsymbol{x} = (x_1, \dots x_n)$, with x_i indicating a real number, but more complex situations are treated similarly. Usually, there are natural restrictions to the decision parameters $\boldsymbol{x}$. For instance, if x_i is the emitted amount of pollution at site i, then x_i is naturally nonnegative; there may also be technical reasons for x_i not to exceed some upper limit. Therefore, we will be able to restrict our attention to vectors $\boldsymbol{x}$ in some subset $\mathcal{D}$ of $\mathbb{R}^n$.

Similarly, some values for the vector $\boldsymbol{y}$ will be totally unacceptable, e.g., because they may lead to a disaster. Therefore, we may also state that we are only interested in those $\boldsymbol{x} \in \mathcal{D}$ that lead to a vector $\boldsymbol{y} = (y_1, \dots, y_m)$ in a given set $\mathcal{O} \subset \mathbb{R}^m$ of acceptable outputs. Later, we shall denote such $\boldsymbol{x} \in \mathcal{D}$ that result in $\boldsymbol{y} \in \mathcal{O}$ by $\boldsymbol{x} \in X_0$. If we denote the vector $\boldsymbol{y}$ that is produced by the decision vector $\boldsymbol{x}$ by $\boldsymbol{y}(\boldsymbol{x})$, then we can describe our decision problem as:

(1) Find an attractive (or even the most attractive) vector $\boldsymbol{x} \in \mathcal{D}$ for which $\boldsymbol{y}(\boldsymbol{x}) \in \mathcal{O}$.

This is the essence of the model-based approach. However, the reality may be more complicated. The most important complication arises if there is a nonunique $\boldsymbol{y}(\boldsymbol{x})$ for any given decision vector $\boldsymbol{x}$. How can that feature arise?

The most common cause for this feature is that there are various other influences on the same process. These influences can be autonomous like the weather: if it rains a lot, the concentrations of polluting materials decrease in a river. These influences can also be intentional: in the case of a river on the border of two countries, one country might pollute the river more deliberately, if the other diminishes its emissions. The autonomous influences can be included in the model by introducing random elements. A way of avoiding this difficulty is to take yearly averages as outputs; however, this usually requires a reformulation of the model. Intentional uncontrolled influences can be included in the model in the same way as the decisions. However, then the real difficulty arises if one tries to find an attractive decision.

Let us avoid these difficulties for a moment and concentrate our attention on the selection problem as formulated in (1). If there is no $\boldsymbol{x} \in \mathcal{D}$ for which $\boldsymbol{y}(\boldsymbol{x}) \in \mathcal{O}$, then there is clearly no attractive decision vector and we might think of enlarging $\mathcal{D}$ and/or $\mathcal{O}$. So, let us suppose that for some vectors $\boldsymbol{x} \in \mathcal{D}$, the corresponding vectors $\boldsymbol{y}(\boldsymbol{x})$ belong to $\mathcal{O}$. Now we have a selection problem.

One may deal with this selection problem in various ways, which will be described in more detail in Sections 1.3 and 1.4. Here we restrict our attention to just one possibility, i.e., the case that all decision vectors $\boldsymbol{x}$ and their corresponding outcome $\boldsymbol{y}(\boldsymbol{x})$ can be evaluated on a one-dimensional scale, e.g., in terms of costs. This would imply that there is also some real-valued function $c(\boldsymbol{x})$ that indicates those costs and our problem (1) can be reformulated as:

(2) Find the vector $\boldsymbol{x} \in \mathcal{D}$, which satisfies $\boldsymbol{y}(\boldsymbol{x}) \in \mathcal{O}$, and for which a given cost function $c(\boldsymbol{x})$ is minimal.

If one can indeed formulate a practical decision problem in this way, then the problem has actually been solved, since (2) is a relatively standard mathematical formulation, which might confront us with some technical difficulties, but, essentially, the main work has been done. For many specific forms of (2), good solution procedures have been developed, e.g., linear programming, combinatorial optimization, mixed integer programming, and nonlinear programming. This formulation has resulted in an extensive research effort in the field of mathematical programming, which has its own research journals and conferences.

Regrettably, however, many problems cannot easily be reduced to the form of (2). This can occur, for example, because the decisions cannot be graded on a one-dimensional scale or, more seriously, because the possible decisions cannot be described in a natural way as the choice of some vector $\boldsymbol{x} \in \mathcal{D}$. We will come back to such cases in Sections 1.3 and 1.4. At the moment we present the formulation (2) as the kernel of an approach based on the relation between basic processes and actions in the ideal case. And indeed, the approach for cases that do not fit into the ideal formulation is also heavily based on this prototype. Actually, such approaches quite often use an optimization formulation like (2) as a tool by constructing appropriate artificial constraints and a more appropriate artificial criterion function.

1.1.3 Decision process

Finally, we come to the third aspect: the decision process. In this section, we indicate approaches based primarily on the decision process.

The characteristics of a decision process have always received most attention from social scientists and economists, as this aspect is most directly related to the way decision making is organized and to the process of choice. It is also surprising how decision situations can be cleared up by structuring the decision process and clarifying the preferences. In several cases such a clearing up does the job and is sufficient to understand which decisions should be taken.

Since the decision process has so many features, there are many ways of structuring the process and clarifying the preferences. Therefore, a whole phalanx of methods and approaches has been developed, mainly concentrating on one or two particular features. However, the integration of various approaches for different features, in a case when more features are relevant, is often quite straightforward. Therefore, it will be sufficient for the moment to mention some features, together with corresponding approaches.

A typical feature of complex decisions is that they consist of several related subdecisions. A typical relation is that one subdecision restricts the set of alternatives for another. For instance, if I am planning my holidays, then I have to decide on the destination, time, mode of travel, and type of accommodation. However, as soon as I decide to go to Paris, the possible set of travel arrangements is restricted considerably. For this feature the use of a decision tree can be very helpful. In such a tree, the subdecisions are represented in a sensible order and in a well-structured way. This relatively simple tool may help considerably; quite often, subdecisions become rather simple and apparent in this way.

Another important feature of decisions might be that they are not made by one person but by a group of persons who have to agree somehow on the ultimate choice. There are various ways of facilitating and supporting group decision making. Typically, they rest on two factors. The first involves extracting from the group their priorities and the criteria that are relevant for the decision. The second involves providing a means for discussing and exchanging evaluations of possible alternatives according to the chosen criteria.

A typical example is the choice by a board of a new chief executive officer from a set of applicants provided by a headhunter.

For several other examples, the group feature has to be treated in an integrated way with other features, for instance, with subdecisions. This occurs when I am planning holidays not just for myself, but also for my family.

Still another feature may be that there is some division of power, causing several parties to make their own decisions, but with all parties feeling the consequences of the decisions of the others. If one considers, for instance, the acid rain problem in Europe, then one recognizes easily that each country may decide autonomously on its emission for, say, the next five years, by issuing regulations on power generation and use, on manure production and handling, and car driving. However, it is also clear that several European countries suffer mostly from acid rain caused by the emissions of other countries. Therefore, the strongest stimulus for a country to decrease its emissions may well be the promise from neighboring countries that they will decrease their emissions as well. Clearly, in this situation, the decisions should be based on negotiations of some sort (cf. Chapter 13 for more details). Hence the question arises how one may facilitate or support negotiations. Negotiations support is even more difficult than group decision support. In the case of negotiation each party has to decide for itself upon the criteria and their relative relevance. There are many ways to achieve a conclusive decision. A way that is most appropriate in a particular situation depends very much on circumstances.

The last feature to mention here (although there are still many others) is the phase of choice in the decision process. This is usually based on complicated preferences for different alternatives. It has already been indicated (e.g., for the case of group decision making) that there might be several relevant criteria. In the case of selecting a chief executive officer, important criteria might be: (i) experience in similar functions; (ii) an innovation-oriented personality; (iii) management quality; and (iv) financial expertise.

Unlike the single criterion in formulation (2) for the first aspect, alternatives are not necessarily evaluated with real numbers. Evaluation might equally be of a qualitative nature, such as very high (v), high (h), and moderate (m) for management quality. These evaluations may even be more complex if a criterion is subdivided into subcriteria; for instance, management quality might be subdivided into analytic power and executive power. In the case of two such subcriteria, one could have the following qualifications:

$$(v, v)\ (v, h)\ (v, m)\ (h, v)\ (h, h)\ (h, m)\ (m, v)\ (m, h)\ (m, m)$$

It is possible that the desirability of such qualifications cannot even be ordered in a simple way. In the above-quoted example of management quality, it is clear that a person with (v, v) is most desirable, but (m, m) is not necessarily worse than (m, v), as someone without good ideas might be dangerous if she/he scores high on executive power.

One of the tools for making preferences operable is the multi-attribute utility theory (MAUT; Rios, 1994). Another approach, based on slightly different assumptions, is the analytic hierarchy process (AHP; Saaty, 1980). There are also other approaches, such as outranking relations (Vincke, 1992). Indeed, clarifying the preference structure in one way or another might be very helpful (see Chapter 9).

We can see that when analyzing the decision process, we actually build another part and type of the model of the problem. We shall use the following basic distinction of model types:

- *Substantive models*, describing the relation between basic processes and actions and trying to represent a part of reality relevant for the decision situation being analyzed.

- *Preferential models*, representing the preferences of the decision maker and other aspects of the decision process.

1.2 Discussion of Aspect-Oriented Approaches

In the preceding section, some typical approaches were sketched for each of the three aspects of decision making. The present section is devoted to a discussion of those approaches. The essential issue is, of course, the usefulness of a given approach. However, we know already that each of the approaches can be very useful. Hence the remaining problem is the reverse question: If we have a specific decision problem, which one of these approaches is then most useful?

A satisfying answer to this question would require a classification of decision problems with an allocation of one or more approaches to each class. Without trying to make such a classification explicitly, one may already deduce some conclusions.

It has already been indicated that although the information aspect is quite basic, a concentration on information alone might be insufficient for decision support. For instance, a management information system may give all the information relevant to the production process of a firm, but it cannot predict the consequences of a decision to halve delivery times. However, it may well represent the financial and performance consequences of a structural change in the distribution network.

Several practical decision problems have certain features that make the approach based on modeling natural and relevant. In particular, formulation (2) or some of its extended, less strict variants – typically involving many objectives – frequently provide the natural setting. This is particularly the case if some relatively hard technical, physical, or economic processes play the central role in the sense that they have to be influenced somehow by the decision. However, in most cases that fit into this category, the decision process and the informational structure are rather complicated.

Consider, for instance, the acid rain problem in Europe. The core processes consist of emissions (caused by technical, economic, and social phenomena), and dispersion and deposition somewhere in Europe (here the reigning winds and other climate conditions are decisive). Essentially, the aim is to change the results of these core processes, and that can only be done by influencing the emissions, since changing the wind directions is not feasible.

However, the emissions themselves cannot be influenced directly, but the processes that generate them may. Processes to be influenced might include the mobility of the population, technical performance of cars, land use, ways of feeding cattle, use of electricity, and ways of generating electricity. Information regarding such processes can be well represented by data bases and other information systems.

This discussion reveals that it is a complicated decision process to achieve a decrease in emissions within any of the countries. It is already complicated to discover what measures might be taken to achieve a particular decrease, but to agree on such measures is a completely different issue. Thus, the core processes may be located easily in this case, but this does not imply that they represent the core of the decision problem. With this typical case in mind, we may conclude that a straightforward application of an approach based on the relation between basic processes and actions is useful only for a small percentage of decision problems.

There might also be problems that represent the third aspect – the specificity of the decision process – in almost pure form. For example, the partitioning of an inheritance among the children of a deceased person might be an example of a negotiation problem in pure form (Raiffa, 1980). Other problems might be, however, less purely negotiational. For example, the acid rain problem also has this negotiation aspect, but the complex relation between decisions and consequences plays a dominant role in this case.

Generally, only a small minority of decision problems can be treated well by selecting one aspect-oriented approach. We need approaches that integrate more than one aspect. In this book we will show that such an integration approach is possible for some institutional decision problems, for which the decision making ultimately relies on influencing relatively hard basic processes. In particular, the approach will be illustrated with environmental problems that belong to that category. This implies that models of the basic processes and decisions should play a core role in our approach (without denying the importance of the other aspects). Therefore, we will first comment on the relation between basic processes and actions.

1.3 Mathematical Programming for Decision Support

In this and the next section, various types of model-based decision support will be outlined by presenting different types of models. This section will be devoted to models that fit into formulation (2) of Section 1.1, with the exception that models with multiple criteria will also be treated. The next section will be devoted to models that emerge if not all assumptions leading to formulation (2) are satisfied.

In Section 1.1, it was shown that if the relation between basic processes and actions is important, then the essence of the decision problem might be represented by an elegant mathematical model fitting into the following formulation:

(2) Find the vector $\boldsymbol{x} \in \mathcal{D} \subset I\!\!R^n$, which satisfies $\boldsymbol{y}(\boldsymbol{x}) \in \mathcal{O} \subset I\!\!R^m$ and for which $c(\boldsymbol{x})$ is minimal.

Vectors $\boldsymbol{x} \in \mathcal{D}$, which satisfy $\boldsymbol{y}(\boldsymbol{x}) \in \mathcal{O}$, are called admissible or feasible solutions to the decision problem.

The arguments leading to this formulation include some assumptions that are clearly not always satisfied. Perhaps the most critical assumptions are that there is only one criterion and that this criterion can be represented by a real-valued function of the decisions. The first assumption will immediately be weakened by introducing k criteria that are represented by k real-valued functions $c_1(\boldsymbol{x})$, $c_2(\boldsymbol{x}), \ldots, c_k(\boldsymbol{x})$ on the decisions. Hence the aim will be to find a feasible $\boldsymbol{x}$ that makes all k criterion functions as low as possible in some sense. However, we first discuss the more classical case with $k = 1$.

1.3.1 Single criterion

If a decision problem has been reduced to the formulation (2), then the only remaining problem seems to be a matter of mathematical technique, i.e., how to find the minimizing $\boldsymbol{x}$ among the feasible ones (see e.g., Fletcher, 1987, or Chapter 7 for more details). Actually, there might still be two difficulties of a conceptual nature. First, the minimizing $\boldsymbol{x}$ might be nonexistent. And second, there might be more than one minimizing $\boldsymbol{x}$, leaving open the selection problem.

The first difficulty can be solved by accepting an $\boldsymbol{x}$ that is only approximately minimal or approximately admissible. The second difficulty may, for instance, be solved by applying a secondary criterion to the set of minimizing $\boldsymbol{x}$. This secondary criterion may be meaningful in the context of the problem or it may just be technical.

Hence, let us assume for the moment that those two difficulties will not bother us; what remains is a purely mathematical or computational problem. The complexity of this problem highly depends on the dimensions n, m, on the structure of the sets $\mathcal{D}, \mathcal{O}$, and on the structure of the functions $\boldsymbol{y}(\boldsymbol{x}), c(\boldsymbol{x})$.

In the simplest situation, we have just one decision variable $\boldsymbol{x} = x$, e.g., a dimensioning parameter that may be chosen from the positive real numbers, i.e., $\mathcal{D} = (0; \infty)$, whereas the consequences of the choice are all represented by the costs, i.e., $\boldsymbol{y}(x)$ and $\mathcal{O}$ are not needed. Let the costs consist of the construction cost, for instance $1 + 2x$, and also of long-range costs of dysfunctioning, for instance, $10e^{-x}$. In such a case the solution of (2) is a simple exercise (cf. *Figure 1.2*).

More interesting examples emerge from several planning problems with large numbers of decision variables (n) and large numbers of output functions (m), but luckily with simple forms for $\mathcal{D}$, $\mathcal{O}$, $\boldsymbol{y}(\boldsymbol{x})$, and $c(\boldsymbol{x})$. Typically, $\mathcal{D}$ and $\mathcal{O}$ consist of intervals for the individual independent variables x_i and dependent variables y_j (cf. *Figure 1.2*). At the same time, $\boldsymbol{y}(\boldsymbol{x})$ and $c(\boldsymbol{x})$ are often linear in $\boldsymbol{x}$. In summary, one may conclude that (2) becomes a linear programming model in this way. For such models, good tools are available even for very large problems (cf. Chapter 7).

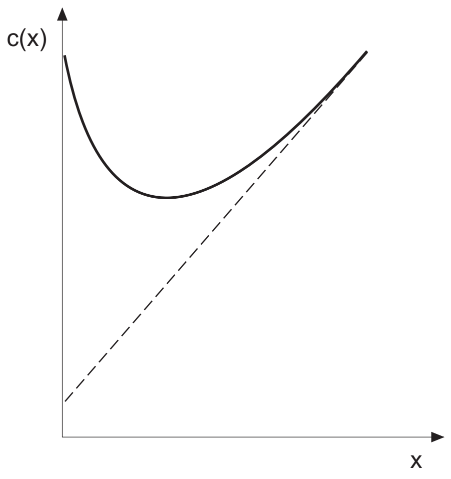


Figure 1.2. $c(x) = 1 + 2x + 10e^{-x}$ for $x > 0$ with minimum at $x = ln5$.

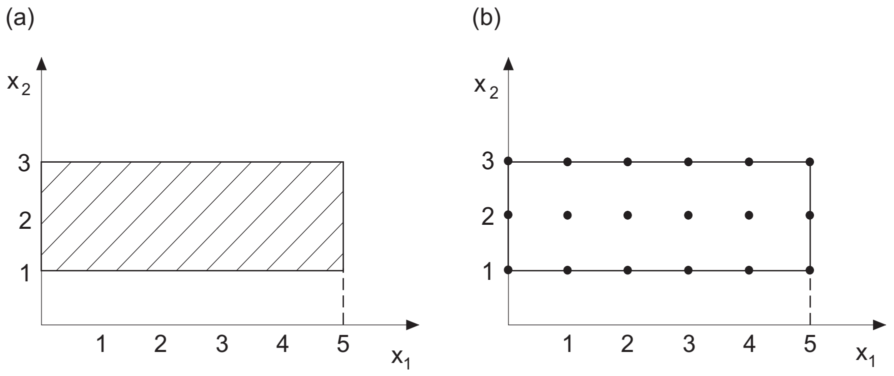


Figure 1.3. The set $\mathcal{D}$ for the case of interval bounds for each decision variable (a) and for the case when only integer values in these intervals are acceptable (b).

Other relevant models are obtained from this classical one by varying the structures of $\mathcal{D}$, $\mathcal{O}$, $\boldsymbol{y}(\boldsymbol{x})$, $c(\boldsymbol{x})$. If, for instance, $\boldsymbol{y}(\boldsymbol{x})$ and/or $c(\boldsymbol{x})$ are nonlinear, then we are in the field of nonlinear programming. If, on the other hand, $\boldsymbol{y}(\boldsymbol{x})$ and $c(\boldsymbol{x})$ are kept linear but $\mathcal{D}$ is restricted in such a way that all decision variables may take on only integer values in some interval (cf. *Figure 1.3*), then one obtains an integer linear programming model. By mixing the two concepts of linear and integer models, mixed integer linear programming models are obtained. Solving nonlinear or (mixed) integer linear programming models is definitely more complicated then solving linear programming models. Nevertheless, much progress in solving such models has been made (cf. Chapter 7). Many other variants are possible, but efficient solution techniques are available only for a limited number.

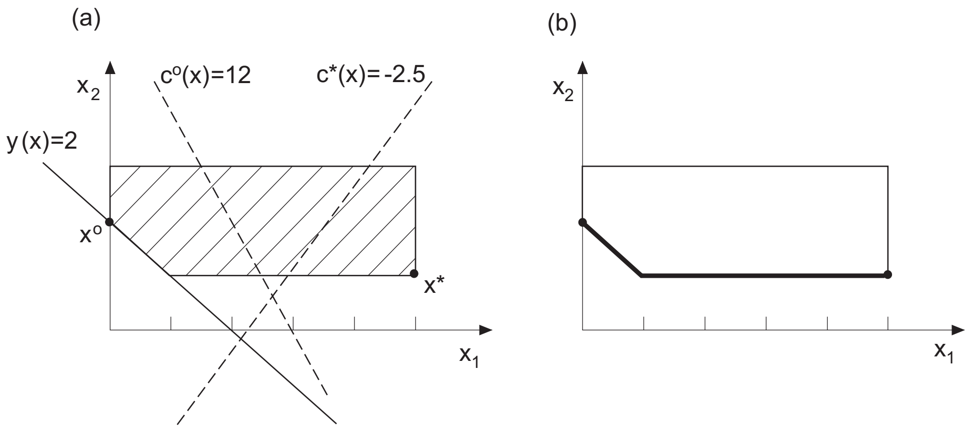


Figure 1.4. The shaded set in (a) indicates those vectors $\boldsymbol{x} \in \mathcal{D}$ that also satisfy $y(\boldsymbol{x}) \geq 2$; the dots indicate solutions $\boldsymbol{x}^0$ and $\boldsymbol{x}^*$, that minimize the functions indicated by the dotted lines. In (b), the bold line indicates the Pareto set (endpoints included).

1.3.2 Multiple criteria

Clearly, a weak point in formulation (2) is that it assumes that the quality of decisions $\boldsymbol{x}$ and $\boldsymbol{x}^*$ can be determined by comparing some real values $c(\boldsymbol{x})$ and $c(\boldsymbol{x}^*)$. Quite often there are several such criteria. In this section we will still suppose that each criterion may be represented by a real-valued function $c_i(\boldsymbol{x})$ on $\mathcal{D}$ or on $\mathbb{R}^n$, but extend the scope of the previous section by accepting that there are $k > 1$ of such functions.

It is unlikely that all such functions attain their minimum for the same $\boldsymbol{x}$. Anyhow, if that is the case, then we are essentially back where we were in the previous section. If that is not the case, then we have encountered an essential difficulty, which is not just a matter of mathematical technique, but which requires a new conceptual approach.

Let us suppose, for simplicity, that $k = 2$ and we have two real-valued criterion or objective functions $c^0(\boldsymbol{x})$ and $c^*(\boldsymbol{x})$, reaching their minimum at $\boldsymbol{x}^0$ and $\boldsymbol{x}^*$, respectively. In *Figure 1.4*(a), the example of *Figure 1.3*(a) is extended with the added output requirement that $\boldsymbol{y}(\boldsymbol{x}) = x_1 + x_2 \geq 2$ and with the criterion functions $c^0(\boldsymbol{x}) = 2x_1 + x_2$, $c^*(\boldsymbol{x}) = x_2 - x_1$ represented by dotted lines. In this case we clearly have $\boldsymbol{x}^0 = (0, 2),\ \boldsymbol{x}^* = (5, 1)$.

The basic problem becomes now: shall we simply choose between $\boldsymbol{x}^0$ and $\boldsymbol{x}^*$, or can we construct some compromise solutions between them? In the latter case, the remaining problem is to find a trade-off between the two criteria. This is not, strictly speaking, a mathematical problem, although it might be made into one with some extra assumptions on the relative importance of both subproblems.

In this example, one might transform c^* from a criterion into a constraint by adding $c^*(\boldsymbol{x}) \leq -1$ to the existing constraint $\boldsymbol{y}(\boldsymbol{x}) \geq 2$. In this way we get a single criterion optimization problem with a guaranteed value for $c^*(\boldsymbol{x})$. This would lead to the decision (2,1). It is also possible to construct a combined criterion out of c^0 and c^*. For example, if unit gains in both criteria would be equally valuable, we could construct a new criterion $c(\boldsymbol{x})$ by adding the old ones:

$$c(\boldsymbol{x}) = c^0(\boldsymbol{x}) + c^*(\boldsymbol{x}) = x_1 + 2x_2 \ .$$

Minimizing $c(\boldsymbol{x})$ would give $\boldsymbol{x} = (1, 1)$ straightaway. However, this is based on quite a strong assumption. Both of the above techniques of dealing with multi-objective optimization problems (introducing proxy constraints or summing up with importance weights), though simple and typically suggested by textbooks, have nevertheless some technical and, what is worse, practical drawbacks. These drawbacks are described in Chapter 4, where some other techniques are also presentedt – more complicated but giving better practical results.

Without introducing new assumptions that define compromise solutions, we can partition the set of feasible or admissible decisions (the shaded set in *Figure 1.4*(a), including the boundaries) and define a subset of such $\boldsymbol{x}$ for which there are surely no better decisions and another subset that can be improved upon. This can be defined as follows:

Definition 1: A decision $\boldsymbol{x}$ is called a Pareto solution if it is feasible and there is no such $\tilde{\boldsymbol{x}}$ that $c_i(\tilde{\boldsymbol{x}}) \leq c_i(\boldsymbol{x})$ for all $i = 1, \ldots, k$, with strict inequality for at least one i.

Figure 1.4(b) indicates Pareto solutions for the simple example presented here with $k = 2$. Alternative names are efficient or nondominated solutions, used particularly in cases when we do not minimize (or maximize) all criterion functions, but minimize some, maximize some and do other things (e.g., stabilize or treat as soft constraints; see Chapter 4 for more details).

The set of Pareto solutions is called the Pareto set; in the criteria or objective space, we call it the Pareto boundary or Pareto frontier. Clearly, it is reasonable to restrict attention to the Pareto set, but the problem remains how to select one solution from it (Sawaragi *et al.*, 1985). This selection is a conceptual problem, although, naturally, there is some relation between the chosen concept and the complexity of the resulting mathematical problem of finding the best decision within the chosen concept. This topic will be treated in more detail in Chapters 4 and 8.

1.4 Other Frameworks for Model-Based Decision Support

In this section, we shall outline how a modeling approach can be used for clarifying the relation between basic processes and decisions if the assumptions for a formulation like (2) do not hold. The presentation has no pretension of completeness; the aim is just to indicate which complications may occur and which concepts might be used to overcome them.

1.4.1 More complex decisions

The first assumption in developing formulation (2) in Section 1.1.2 was that possible decisions may be represented by vectors $\boldsymbol{x}$ in n-dimensional Euclidean space. This assumption is clearly quite strong, although there are many cases where it is reasonable. However, there are also many cases where decisions require another representation; for instance, if one of the decisions to be taken is the emission of a particular pollutant at a particular spot. This emission might be represented by the total emission per year (in tons for instance), but there might also be reasons to consider a much more detailed emission profile represented by the emission speed at each time instant within the year: $\boldsymbol{x}(t),\ 0 \leq t \leq T$.

In actual computations, such a representation might be replaced by a finite set of variables representing some time discretization – such as the emissions for each month in the year, or the emissions for each week or each day. With this type of discretization, we return to our original formulation, at the cost of a large number of decision variables. Even when maintaining the decision as a function of the continuous time indicator t, the idea of formulation (2) is not really spoiled, only the necessary mathematics are more complicated. In fact, such problems are the subject of variational calculus and the theory of optimal control.

This was only one way of complicating the representation of decisions. There are also decisions that are not naturally represented by real numbers. If a family has to choose between renting, buying, and building a house, where each of the three alternatives can either by realized in a particular town or a neighboring one, then we have six alternatives. One can number the alternatives from one to six. Or one can indicate each alternative with a pair (a, b), where a represents the type of house (rent, existing, new) and b represents the town (one town or the other). However, it is more difficult to relate such a numerical representation to the constraints and preferences.

Let us suppose that we have financial restrictions on the yearly costs for housing and on the initial costs. Alternatively, we want to minimize the total daily travel effort for all family members. In such a case we have in fact six scenarios and for each scenario we can check whether it fits the financial restrictions or not. From the remaining scenarios we may choose the one with the lowest travel effort.

In this way we remain completely within the optimization framework of Section 1.3.1; we only miss more elegant optimization techniques and have to consider all alternatives separately and explicitly. For this particular case with only six alternatives, such an approach may not be very cumbersome, but in many cases the number of possible decisions or scenarios may be very large.

A reasonable procedure for such a situation consists in first constructing a restricted set of representative scenarios and then, after their evaluation, constructing some more scenarios close to the preferred ones, etc. Also in this way we remain quite well within the optimization approach. In fact, we develop a kind of search procedure aiming at the optimum. Some of these search procedures are described in Chapter 9.

Another approach would be to apply a divide-and-conquer strategy by first deciding on the town and only afterwards on the type of house. However, this poses the difficulty of requiring a comparison of travel efforts without having full details of the location within each town, as these details are determined by the type of house that will be chosen.

One might agree that the framework of Section 1.3 is most elegant, because within that framework there is a formal link between decisions and consequences through the functions $\boldsymbol{y}(\boldsymbol{x})$ and $c_i(\boldsymbol{x})$. However, such an approach requires that the possible decisions can be represented by some mathematical entity where functions can be constructed in a natural way. The most suitable candidate for such an entity is the real valued vector, but some more general entities, like vector valued functions of a real valued parameter, are also possible. In many cases, regrettably, there is no natural representation for the decisions that allows the construction of functions $\boldsymbol{y}(\boldsymbol{x})$ and $c_i(\boldsymbol{x})$. In fact, this is true in most cases where alternative decisions represent structural choices rather than choices of dimensioning. In extreme cases, alternative structural choices may even require completely different techniques for evaluating the respective consequences. As stated before, this type of difficulty may destroy the elegance of the mathematical programming approach, but it does not destroy the underlying concept of relating decisions with consequences.

Many decision problems combine both structural and dimensioning decisions. It seems natural to use a hybrid approach in such cases, where the best dimensioning decisions are chosen for a structural choice by applying the mathematical programming approach, whereas the relevant structural choices are evaluated and compared by scenario analysis. For scenario analysis, many mathematical techniques are available. Usually, the scenario approach makes it possible to include more detailed simulation models. However, the evaluation of all possible scenarios is clearly an illusion, whereas in mathematical programming we can – at least in principle – evaluate and compare all feasible solutions.

1.4.2 Uncertain consequences

Another central assumption in Section 1.3 is that once a decision has been made, all relevant consequences are clear, at least after some computational processing. But this is surely not always the case. If consequences depend on the weather (rainfall or amount of sunshine), then those consequences become uncertain in the sense of depending on chance events. If consequences depend on processes that are not well understood (like adaptation of animals to changing circumstances), then those consequences become uncertain in the sense of lack of understanding. This division in two types of uncertainty is very helpful, but not very strict, as a big grey zone exists between these two types.

In the case of uncertainty depending on chance, it seems natural to extend the mathematical programming approach with chance elements. This can be done by adding chance elements to the functions $\boldsymbol{y}(\boldsymbol{x})$ and $c_i(\boldsymbol{x})$. For instance, in the case of linear programming, the constraints may be formulated as:

$$\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \tag{1.1}$$

where $\boldsymbol{A}$ is a $n \times m$ matrix, $\boldsymbol{x} \in \mathbb{R}^n, \ \boldsymbol{b} \in \mathbb{R}^m$. In this case, elements of the matrix $\boldsymbol{A}$ and/or the vector $\boldsymbol{b}$ may be considered as random variables. One approach would be to require that inequality (1.1) is satisfied with a certain probability only for the chosen decision $\boldsymbol{x}$. This is a relaxation of (1.1). In another approach, a secondary decision vector $\boldsymbol{z}$ is supposed to ensure that the consequences are not too bad if the realization of $\boldsymbol{A}$ and $\boldsymbol{b}$ appear to be unfavorable. For the one-criterion case we obtain the following minimization problem:

$$\text{minimize} \quad \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{d}^T\boldsymbol{z} \tag{1.2}$$

under constraints:

$$\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} \leq \boldsymbol{b}, \ \boldsymbol{x} \geq 0, \ \boldsymbol{z} \geq 0, \tag{1.3}$$

where $\boldsymbol{x}$ has to be chosen before the realizations of $\boldsymbol{A}$ and $\boldsymbol{b}$ are known, but $\boldsymbol{z}$ may be chosen afterwards; hence $\boldsymbol{z}$ is actually a function of $\boldsymbol{x}, \ \boldsymbol{A}, \ \boldsymbol{b}$. These approaches belong to the field of stochastic programming (see Ermoliev and Wets, 1988).

Naturally, it is much more difficult to include lack of knowledge. The most straightforward way of including this feature is through the concept of fuzziness and, indeed, fuzzy linear programming is a well-developed field (Zimmermann, 1987). Another approach uses scenario analysis and tries to identify the particular uncertainties related to each scenario. This identification is exploited next to indicate the possible ranges of consequences.

1.4.3 Complications with the preferential structure

In Section 1.3 it was assumed that the preferential structure could be modeled by putting bounds on some consequences and designating others as criteria. The strength of this choice is that different types of consequences can be treated separately. For each type of consequence it may be decided either to put bounds on it or to consider it as a criterion. In this way each type of consequence is measured on its own scale, which may be investment costs, levels of pollution, likelihood of reaching a threshold, etc. The main difficulty remains in finding a decision that strikes an acceptable balance between the different types of consequences, but this issue will be discussed in more detail in Chapters 4 and 8.

The basic assumption is that all types of consequences can somehow be measured on a scale; however, that is not necessarily true for all types of consequences. If some of the consequences cannot be measured on a clear scale, one may operate in either of two ways. First, one may choose not to accept this feature and try to develop an artificial scaling function. For such a procedure, multi-attribute utility theory provides support. Second, one may accept this feature and try to live with it. In that case one should accept that some types of consequences have hard information available while others do not. If one accepts this feature and acts accordingly, then the feature does not provide real complications. Indeed, it is possible to integrate hard and soft information on consequences into a well-structured decision process.

1.4.4 Complications with the decision-making process

It might appear that Section 1.3 assumes that a single person is responsible for making decisions, whereas in practice, many decisions are outcomes of complex explicit or implicit decision-making processes. This assumption arises because the approach of Section 1.3 is frequently presented as a complete approach, solving the entire decision problem. Such an approach, however, is not realistic. In fact, the approach can be embedded in a complex decision-making process, as will be illustrated in various chapters, in particular, the third part dealing with environmental applications. Of course, such an embedding has consequences for the implementation of the approach and, therefore, it is of utmost importance that the modeler knows the intricacies of the decision-making process and understands the role that will be played by the mathematical programming model. If this condition is not satisfied, the exercise is doomed to be a failure. Unfortunately, such failures are usually attributed to the approach adopted rather than to the lack of expertise of the modeler. Therefore, much emphasis will be given to this feature throughout the book.

1.4.5 Complications with modeling the basic processes

There are cases where the basic processes are already so complicated that one would prefer not to model them explicitly. In some cases the detailed modeling can be avoided by modeling on an aggregate level (the treatment of the acid rain problem in Chapter 13 is a good example of this). In other cases decomposition is helpful, but situations remain where there is no obvious way out within the mathematical programming approach. Even then, there are still several alternatives. Here we will mention three of them.

Scenario Analysis

The first alternative is scenario analysis. By formulating some scenarios, one obtains the possibility to evaluate realistic cases. Even when detailed modeling is too complicated to be feasible, it is often possible to evaluate particular scenarios by combining partial models and expert evaluation.

Rule-Based Systems

The second alternative is to use a rule-based system. This implies that the basic processes are not modeled analytically but evaluated logically by a system that imitates the argumentation of an expert; thus, a logical type of model is used instead of an analytical type.

Neural Nets

The third alternative is to use a neural net. This means that some black box is trained, with the use of examples, to give the evaluations.

While the latter two alternatives are more suited to the operational type of decision making, they can also be helpful as building blocks for parts of decision support systems aiming at more tactical or strategic decisions.

1.5 Conclusions

The first conclusion from this chapter is that complicated decision problems require a form of support that somehow integrates various aspects of decision problems: the relations between basic processes and decisions, the information, and the decision-making process itself. The second conclusion is that most general approaches to decision support are essentially based on only one of the main aspects. However, this does not imply that these approaches are useless. It has been stressed already that it is essential to be aware of the decision-making process when implementing an approach based on the relations between basic processes and decisions. When doing so, it is possible to develop models (including optimization models) that can play a useful role in the decision-making process. The same holds for the relation with information.

Many decision support systems are especially made for particular decision situations. In such systems, one often sees a well-balanced emphasis on all three aspects. One could say that the main thesis of this book is that it is possible to develop tools and techniques for constructing well-balanced decision support systems based on substantive models.

Chapter 2

The Modern Decision Maker

Andrzej P. Wierzbicki and Jaap Wessels

Chapter 1 presented in some detail what we mean by model-based decision support; this chapter is devoted to another basic concept – that of a decision maker. Because the world is changing rapidly – the information era has started and strongly influences our lives – we would like to reflect on the concept of a modern decision maker. This, however, is a difficult concept, and we shall discuss it starting from a more philosophical perspective.

By decision maker we could include anybody who makes decisions or takes part in a decision-making process; however, this describes practically anybody, as everyone makes many decisions daily. Thus, we limit the concept of a decision maker to those persons who are aware of the importance of decisions made by them or at least reflect on the way these decisions are made. Even such a limited concept is rather rich in scope. We can distinguish several types of decision makers, such as top-level, economic, or political decision makers; an analyst preparing the analysis of a given decision situation; or a user of a decision support system. We can also distinguish between various types of decisions: operational, strategic, creative, etc. (these are discussed in more detail later).

The definition of a modern decision maker is more difficult. Is she/he modern due to using computerized help, such as decision support systems or computer network communications? Or is she/he modern because of understanding computerized modeling and using its various means, such as logical, graph-theoretical, and operational research? Certainly, these are some dimensions of being modern. However, we propose a different and richer meaning of being modern: a person is modern because of using a set of basic concepts that are adequate for understanding the contemporary, complex world. This meaning is actually related to creativity and thus adaptability – two features that are especially important today, because the contemporary world is not only complex, but also changing rapidly. This topic is considered in Section 2.1 by discussing the relation between concepts and creative processes, as well as their role at the beginning of the information era. Section 2.2 reviews the concept of rationality, as we are interested in supporting rational rather than irrational decisions and decision makers. However, rational decisions might also include an element of intuition, which is not necessarily irrational; thus, Section 2.3 describes the role of intuition in decision making. Section 2.4 describes the dynamics of decision processes and their phases. Finally, Section 2.5 returns to the concept of the decision maker of the future.

2.1 Concepts, Creative Processes, and the Information Era

Language, logic, and mathematics are instruments for description, discussion, and formalization of our deeper thoughts in terms of images and basic concepts. It is obviously better for communication if basic concepts are well understood and have a well-defined meaning.

Alternatively, a concept is more valuable the richer it is, in a synergetic sense; the number and value of associated images related to a concept also depend on the level of expertise. Consider the concept of time: a specialist might ask you whether you prefer a definition of one sentence, but warn you that a one-hour lecture might be better, while the subject deserves a full course of lectures. Moreover, a physicist will talk about the relativity of time, an economist or system scientist about various aspects of time discretization and time discounting, and a mathematician about co-existing time-scales and boundary layer effects.

The same applies to many other concepts fundamental to our perception of the world, such as evolution or uncertainty. Especially important is the concept of cause and effect; however, it is still understood by most people – even philosophers – somewhat mechanistically and in terms of binary logic, while we know today that there are situations where fuzzy or multi-valued logic is appropriate; this concept is even more fundamentally changed if we include feedback mechanisms.

The concept of feedback is essential for a modern, systemic understanding of the world; however, it is perceived differently by different groups of people. An economist who has often been taught only the basic definition of this concept views it differently from a control theory specialist who investigates various feedback effects, from a specialist in the theory of chaos, who would say feedback with nonlinear dynamics might produce chaos in a deterministic system, which will change the way we think about the indeterminism of the universe.

Moreover, agreement on a single meaning of a concept does not necessarily guard us against hidden presuppositions in its accepted definition. The philosophy of science (see e.g., Kuhn, 1964; Vollmer, 1984) stresses that our mesocosmic experience results in the danger of such hidden presuppositions; special techniques are used (such as thought experiments) to clarify such presuppositions. While a clear meaning of a concept is needed for communication, there might be other aspects of cognitive processes where the richness of meaning is more valuable; one such aspect is creativity.

Any scientist with some experience in creative processes (i.e., those processes that require logical or experimental testing, and also result in new cognitive insights) would acknowledge the importance of the richness of concepts related to their field of study. We often perceive uneasiness about an aspect of a scientific problem, and try to look at it from various angles, through diverse images related to our basic concepts. After some time, a moment of cognitive enlightenment, a eureka effect comes. The mechanism of this effect relates to subconsciousness and intuition (there are many examples documented in the history of science), but sometimes we partly understand this mechanism. Often a vague association, an analogy helps; however, not all analogies are good and it is dangerous to equate creativity with analogies. It is more important to focus intuitively on a small part of an image that seems to deserve attention; if this part and the angle of focusing are chosen rightly, this part suddenly expands, rich in associated ideas.

Similar creative processes occur in other fields – in art, politics, even managerial decisions. It is helpful to study a subject thoroughly and to discuss it with others – this partly formulates our thoughts and might trigger associations; it also helps to relax and forget the problem for a while – an intuitive answer might then arise. It might also be helpful to extend interests, embark on interdisciplinary research, and even travel and be exposed to cross-cultural influences. How soon you perceive a eureka effect and how good your answer is depends on many aspects, such as level of expertise, thoroughness of study, cross-fertilization through discussions, interdisciplinary and cross-cultural contacts, and personal creativity or intuitive capabilities.

In summary, creative and cognitive processes rely on the richness of images and concepts, and also on the richness, even redundancy, of information; they frequently occur on a deep level of nonverbal associations, often subconsciously or intuitively.

This thesis is supported by the experience of many scientific disciplines. If a mathematician is asked how to choose between the infinite number of true mathematical theorems to be proven and how to choose the ones that are important, she/he would answer by saying with taste and intuition. Popper (1983) observes that scientific theories are neither deductions nor inductions from the facts; he admits that they might be created by intuition. However, he insists that a theory should be considered preferable to another if it has hitherto withstood criticism better than the other. His insistence on subjecting theories in empirical sciences to falsification tests is logically consistent with his admission that theories are often generated by intuition.

However, Popper does not include feedback understanding of cause and effect in his criticism of induction and paradoxes of prediction; they cease to be paradoxical with the feedback understanding. This example shows that basic concepts change and our understanding of the world depends on their adequacy.

We live in a period of fundamental technological and social change. The reasons for this change are variously described, but one of the most widely accepted is the hypothesis of the change of cultural eras – from the era of industrial civilization to the era of information civilization.[1]

A cultural era is defined by a system of concepts that are used for perceiving the world. Such a system must be widely accepted and embedded in the culture and education of the era; it can also be referred to as a cultural platform. The basic concepts that constitute the cultural platform of industrial civilization were developed before this era, and are usually dated from the eighteenth century, to coincide with the Enlightenment era. These concepts were based on the discoveries of Gutenberg, Copernicus, Galileo, Newton, Kepler, and others that helped to perceive the universe as a giant machine, to construct smaller machines and industrialize the world. Thus, the cultural platform of industrial civilization is mechanistic and reductionist: the analysis of the universe or a society is reduced to the analysis of their parts and relations between these parts. Most of our thinking about the world is still based on this platform.

The cultural platform of information civilization is being formed now, though it uses concepts developed over the past one hundred years. It started with the theories of Maxwell and Einstein, challenging the mechanistic perception of the universe. Later, developments in telecommunication and control science (automation) contributed to the understanding of the role of information in complex systems, including the fundamental concept of feedback, which changed the classical understanding of the relation of cause and effect. Finally, the development of computers

[1]This hypothesis was first formulated by Gross (1966), see also Toffler (1980) and Wierzbicki (1988a). The information society is often called a service society; however, the term information society seems to describe more accurately the changes that we experience today.

and computer science resulted in such diverse means of information processing that information became the basic productive input – along with capital, energy, raw materials, and human labor.

This change in proportions of productive inputs results in deep social changes and tensions, including structural unemployment in developed countries and difficulties in closing the gap between developed and developing countries. An average person attempts to understand the complexity of the contemporary world using the concepts of the previous cultural platform, which inevitably leads to frustration, a recourse to old ideologies, fundamentalism, and nationalism.

The new cultural platform is not fully formed yet; some essential concepts – such as the theory of chaos – are known only to a handful of specialists. The complete formation of the new cultural platform, and its acceptance and spread by our educational systems, might need several generations. Thus we cannot say that the era of information civilization has already arrived, though it has started,[2] and some reports (e.g., Bangemann, 1994) rightly stress that the information society will come soon, in the sense that it has started and is inevitable. Information technologies currently exist that will fundamentally change the society as we know it today. These are, for example, microprocessors with computing power comparable to that of supercomputers a few years ago, which will become cheap and in general use in the next few years; teleconferencing programs that make it possible to construct sophisticated videophones, available in every home; the expansive development of computer networking; computer integrated design, manufacturing and management; and many others.

Together with the increasing importance of global, municipal, local, or even home computer networks, such technologies will fundamentally change society, including the organization of everyday work, the way of living and, particularly, the way of making decisions. We will have much easier access to computerized information which, together with computerized models of selected decision situations and appropriate algorithms for processing these information and models, can be organized into decision support systems. However, in order to use such systems effectively, we should be aware of basic concepts concerning decision making, used when developing such systems. One of these is the concept of rational decisions.

2.2 Concepts of Rationality

During the last fifty years, there has been a remarkable development in the understanding of how economic markets function, related to the study of how economic

[2]The date of its beginning might be counted with the discovery of the TCP-IP protocol in computer networks just before 1980; however, full information civilization did not arrive with this date. Similarly, industrial civilization did not arrive until 1760 with the discovery of the steam engine.

decisions are made. This has included mathematical models of analyzing conflicts and cooperation in game theory, general models of decision preferences in value theory, graph theory and theory of relations as applied to decision making. By extending the results of such studies of economic behavior to other types of decision making, decision theory was developed.

In decision theory, a decision maker is considered rational if she/he chooses the most preferred decision, defined usually as maximizing her/his utility or value function. Actually, such a definition is slightly oversimplified. However, the assumption of choice based on preference – on the maximization of a utility or value function – has been very productive in the development of economic theory and mathematical game and decision theory. This has a long research tradition, starting with the old, intuitive perception of economic utility, formalized by von Neumann and Morgenstern (1944), through to the concept of a (cardinal) utility function, further developed by Debreu (1959), through to the study of preference orderings and the conditions of mathematical existence of an (ordinal) value function, and many other theoretical developments. This has led to such an expansion of mathematical economics, game and decision theory that no single book today would be able to even list or summarize the literature on the subject. Indeed, a paradigm in the sense of Kuhn (1970) has been formed around this assumption, with all its features (also including a psychological commitment to this assumption), reinforced by the sociological aspects of a field of science.

There have been many schools of criticism of this central assumption of decision theory, usually dependent on a certain disciplinary or cultural perspective; we consider such a school a framework for perceiving the rationality of decisions.

The most notable framework is the theory of satisficing behavior developed originally by Simon (1955, 1957), reflecting a more managerial and microeconomic perspective (consistent also with the behavior of an engineer). According to Simon, individual decisions are not optimized because they are too difficult (particularly in situations with uncertainty), the access to information is limited, and an individual optimization might lead to unnecessary conflicts in nonmarket situations with a limited number of actors (as in management). Instead of optimizing, an individual decision maker forms her/his own aspirations adaptively, through learning processes that play an important role in this theory, and is satisfied with the decisions if the aspiration levels are attained.

The satisficing framework of rationality was further developed through intensive studies, including empirical tests; finally, it was assimilated in the utility maximization paradigm under the name of bounded rationality.[3] Recently, the concept

[3]This choice of term illustrates a self-reinforcing aspect of the paradigm: while believing that a perfectly rational individual should optimize, it can be admitted that most individuals are less than perfect.

of adaptive aspiration formation was even shown to be a useful instrument for explaining market equilibration mechanisms, consistent with long-term utility maximization (Wall, 1993). Related to the satisficing framework but distinct by stressing a large organizational perspective (in corporate or military planning, also in centrally planned economies) is the framework of goal-oriented behavior (discussed, e.g., by Glushkov, 1972, and Pospelov and Irikov, 1976); this has been independently operationalized by Charnes and Cooper (1977) as the goal-programming method of multi-objective decision support.

From the perspective of decision support in multi-objective situations, a methodology of reference point optimization was later developed (Wierzbicki, 1980a, 1980b, 1986, 1992b; Lewandowski and Wierzbicki, 1989). This methodology uses both optimization and adaptive formation of aspirations, but as instruments of interaction between the decision maker and the computer, rather than as assumptions concerning human behavior. While this methodology generalizes the goal-programming method and can support decision makers who adhere to various rationality frameworks (e.g., who believe either in satisficing, goal-oriented, or utility-maximizing behavior), it also stresses the dynamics of a decision process and adaptive learning; we shall show later that it might also be useful in stimulating the intuition and creativity of a decision maker.

There are also many other schools of criticism regarding the assumption of utility maximization, e.g., from a psychological perspective, such as the theory of regret by Kahneman and Tversky (1982). A much older criticism comes from a purely mathematical perspective when examining extreme consequences or paradoxes related to this assumption. We mention here the paradox of Allais (1953), which motivated the development of the French school in decision analysis (see e.g., Roy, 1977). Much later, such paradoxes led to a postulate of nonlinear dependence of expected utility on probabilities (Machina, 1983). We also mention the impossibility theorem of Arrow (1951), which illustrates the difficulties of a social aggregation of individual utilities. Coming from outside the paradigm, but related to some of its central issues, was the development of the possibility calculus of multi-valued logic or fuzzy set theory introduced by Zadeh (1978). This was also developed by many other researchers as an alternative to probability calculus when describing subjective evaluations of uncertainty or lack of information in decision making.

Another important avenue of criticism of individual utility maximization comes from the evolutionary perspective, with the concept of evolution of cooperation (see Axelrod, 1984; and Rapoport, 1989). In a perfectly logical and mathematically valid way, this approach shows that the evolutionary interests of a population – and also the individual interests of members of this population – might be better served by adopting certain individual decision rules that take into account cooperation principles rather than by pursuing maximization of purely individual interests (whether short- or long-term). At first glance, this finding coincides with various empirical findings of cultural anthropology that stress the evolution of ethical norms as necessary cooperative principles. Nowadays, such phenomena are even observed in several animal populations. This finding might also be incorporated into the paradigm of individual utility maximization, as we could interpret it as an indication that individual utilities should consider social values more. Alternatively, the concept of evolution of cooperation does contradict more extreme interpretations of the paradigm of individual utility maximization (that it is better if everyone pursues their own individual interests) and contributes to a better understanding of emerging issues that require common action (e.g., the importance of preserving unpolluted environments or, more generally, preserving our planet for the survival of the human race).[4]

These are the schools of criticism of the term rationality related to decision analysis; however, this term is interpreted differently in various scientific disciplines. A possibly broader sense of rationality or validity of scientific statements is implied by the philosophy of science (see e.g., Popper, 1983). Even in this field, there are diverse interpretations of this concept: falsificationism by Popper (1959, 1975), historicism of changing paradigms by Kuhn, competition of scientific programs by Lakatos, etc. (see Hacking, 1981), versus evolutionary epistemology as initiated by Lorenz (1965; see Wuketits, 1984). From the more normative, Popperian perspective, rationality of a statement concerning the real world implies its empirical testability and falsifiability (at least in principle); if a statement relates to a method of communication rather than to the properties of the real world itself (e.g., a mathematical theorem) then it must be logical and consistent with the assumptions of the method.

When seen from this perspective, the term rationality in decision theory has been used rather unfortunately. Interpreted as a mathematical statement, the assumption of utility maximization is logical and consistent with many other developments of decision theory. However, interpreted as a description that real rational decisions are always made in the best perceived interests of the decision maker, this assumption is itself not rational in the normative, Popperian sense of the theory of science. Popper uses precisely this as an example of an unrefutable theory. If any other concerns (e.g., honor, altruism) can be subsumed into the individual utility, then the assumption of best perceived individual interests cannot be falsified in principle and expresses an ideology rather than a science. Kuhn argues, as opposed

[4]Moreover, this concept further contributes to the criticism of one of the basic postulates of logical positivism and empiricism of the early twentieth century: that the questions of ethics can be discussed only in emotive, and not cognitive terms.

to Popper, that certain components of ideology are always present in a paradigm of normal science (see Kuhn, 1970).

As described by Kuhn (1970), a paradigm is not necessarily formed or changed because of its empirical testability; its central statements are often chosen in order to simplify analysis, as in the case of Copernican theory. The testability of a paradigm depends on additional assumptions and interpretations; the same applies to the paradigm of utility maximization. If modestly interpreted in its descriptive power, this paradigm might be treated as empirically testable and thus rational in the sense of Popper. Indeed, a statement such as "an averaged behavior of certain aggregate classes of individuals acting on economic markets is usefully described by assuming that they maximize utility or value functions" is falsifiable and has been tested in several economic studies. Unfortunately, this aggregate statement has also often been applied to individual decision making, which has led to an unrefutable, absolute interpretation that every individual maximizes their aggregated, best perceived interests in every decision. Although such an interpretation has generated much criticism in the field of decision theory, it has also served to reinforce the paradigm.

We shall not elaborate further on this difficulty in decision theory. We shall rather use the term rational decision in its broadest sense; although the specific meaning of this term might depend on professional and cultural backgrounds (see e.g., Grauer *et al.*, 1985c; or Yu, 1990), we argue that a rational explanation of the way decisions are made should be – at least in principle – empirically refutable. In this sense, we can turn to the investigation of the role of intuition in decision making.

2.3 The Role of Intuition in Decision Making

The experiences of the authors in decision analysis and support, including various industrial or even political applications (e.g., trying to apply decision–analytical aids during the recent reform of the system of scientific institutions and their funding in Poland) indicate that there are additional dimensions to modern decision making.

The higher the level and experience of a decision maker, the less inclined she/he is to trust in various tools and methods of decision analysis and support.

In the case of repetitive decisions involving groups of experts, the high-level decision maker might agree to use some decision aid method purely for the aggregation of opinions. Even in this case, the method should be as simple as possible in order to be used by expert groups from various disciplines. Experts with a more technical or mathematical background might also be comfortable with multi-criteria evaluations and tools. If experts from various disciplines such as law, humanities, arts, and medicine are included, the decision aid will need to be uniform, thus, single-criteria evaluation and a simple, intuitively clear aggregation mechanism are the only acceptable option; any further complication will create difficulties.

In the case of strategic, higher-level decisions, any experienced decision maker will agree that they are of a multi-criteria type, but will still protest against using a formal aggregation procedure, particularly based on a disclosure and estimation of her/his preferences. Obviously, this attitude might result from the wish to have a better bargaining position in negotiations; however, when faced with such an argument, the experienced decision makers answer that they would rather preserve the right to change opinion and rely on their own intuition.

A clear conclusion is that decision scientists must make more effort to understand better the dichotomy between rational versus intuitive decisions and the role of intuition in decision-making processes.

In everyday language, we tend to associate the word intuitive with irrationality. This is probably due to Bergson (1903), who attached great importance to intuition but interpreted it as a somehow mystic force, which by definition could not be an object of research by rational means of inquiry. However, almost 100 years of scientific research in various fields has reversed this interpretation today. We shall show that intuitive decisions might be rational even in the narrowest sense of utility maximization and, moreover, that intuition, when appropriately defined, might be a subject of rational inquiry.

One criticism of the mainstream decision theory that went even further than the others claimed that decisions are made in a deliberative way, and that individual decisions are too complicated to be explained by analytical theory. This criticism was termed the soft systems approach and arose from general systems theory (Bertalanffy, 1968); it stressed the role of synergy (i.e., that the whole is bigger than the sum of its parts), as well as the fact that in a more complex cognitive or decision process, there is a eureka (after Archimedes) or aha effect, which can also be called a cognitive enlightenment effect. Thus, soft or deliberative decision making consists in trying to perceive the whole picture, observing it from various angles, and finding the right decision by expert intuition.

Various additional observations can be made to support such an approach. One of these concerns the issue of holism versus reductionism: an analytical approach involves a reductionist concentration on parts, while deliberation helps to concentrate on the whole (the gestalt perception of a problem). Another is the taxonomy of cognitive approaches by Harold Barnett (personal communication, IIASA), concerning the role of mathematical modeling at IIASA: hard models = soft thinking, soft models = hard thinking. This implies that if we concentrate too much on analytical approaches, we might miss insights that result from deliberation.

This avenue of criticism is presented in *Mind over Machine* by Dreyfus and Dreyfus (1986). Conceived as a criticism of concepts of artificial intelligence, this book convincingly presents the case for deliberative decision making. A major thesis of *Mind over Machine* suggests that the way decisions are made changes with the increasing level of expertise of the decision maker. Whereas a beginner, novice, or apprentice need analysis to support their decisions, specialists and experts either make decisions instantly, intuitively, or rely on deliberation.

The empirical evidence for operational decision making is abundant; to cite a simple example, when driving a car, a novice has to think before changing gears, whereas experts make decisions with their entire body. In the case of strategic decision making, when new types of decisions are often needed, a special experiment involving chess was developed (Dreyfus and Dreyfus, 1986): the analytical part of the brain of a chess player was saturated with tedious computations, while at the same time the player competed at chess; the results were analyzed depending on the level of experience of the player.

The results were predictable: apprentice players cannot play chess when the analytical parts of their minds are saturated, whereas it does not make any difference for chess masters. Dreyfus and Dreyfus (1986) state that these results are related to the way we perceive reality through images, gestalt, and pattern recognition; to the typical connections of roads of the brain established by training; to the functions of the right part of the brain; and to the concept of intuition. However, these authors refuse to analyze the latter concept, perhaps because the Bergsonian interpretation of intuition strongly influences philosophy (Bergson, 1903). Thus, while taking the results quoted above as a strong argument for the role of intuition in decision making, we must turn to other sources for a definition of intuition.

In order to define intuition, possibly the most useful information comes from the research on hemispheric asymmetry of the brain (see e.g., Springer and Deutsch, 1981). This research indicates that the typical left hemisphere is related to verbal, sequential, temporal, digital, logical and analytical, and rational thought activity, while the right hemisphere is related to nonverbal, visuo-spatial, simultaneous, analogue, and intuitive thought activity. Best substantiated by experimental results is the distinction between verbal and visuo-spatial processing in distinct hemispheres; further dichotomies are more speculative. However, there is also strong empirical evidence in split brain experiments for locating the verbally aware consciousness in the left hemisphere, while the right hemisphere has the more emotive and visuo-spatial consciousness, which we term subconsciousness.[5] Note that

[5]This refers to the experience of making a subconscious movement or subconsciously registering a visual image without immediately being aware of it at a conscious and verbal level; this concept is most akin to the everyday language interpretation of subconscious behavior, hence it is related to, but by no means identical with, Freudian subconsciousness.

in the dichotomies presented above after Springer and Deutsch (1981), rationality is still traditionally opposed to intuition.

When searching for a reasonable definition of intuition, we should take into account a theory of how people perceive the world. This has a long tradition in philosophy (see e.g., Wittgenstein, 1922; Ingarden, 1972; and Vollmer, 1984); the closest to our needs might be the approach of evolutionary epistemology (see e.g., Lorenz, 1965; and Wuketits, 1984). A modern experimental field concerned with similar questions is cognitive science with its theory of inner representation. This refers to what we might communicate by words and think abstractly in symbols, but we perceive the world by images and we have a mental image representation (see e.g., Kosslyn, 1980). Although this theory is strongly criticized (see Pylyshyn, 1984, and Gardner, 1984, for a broader view of this dispute), not only is it supported by much experimental evidence, but it can also be substantiated from the perspective of evolutionary epistemology by means of a thought experiment.[6]

This thought experiment considers a point in the development of an intelligent race, e.g., humans, when the language is still young and only a small number of words (say, a few thousand) is used. In such an experiment the following questions might then be asked: What would be the prevalent mode of thinking – would it be by words or by images (large informational objects)? Would they process emotions mostly in words, or in a manner similar to images? What would be the role of words and language at this stage of development? Suppose their world is similar to ours in relation to objects; what would be the logic used (perhaps implicitly) to describe these relations: binary or multi-valued? In this discussion we omit a full description of such a thought experiment, and present only its implications.

The development of language was a powerful evolutionary shortcut in the history of the human race. It was easier to think and communicate in words with other individuals; this not only helped evolutionarily in dealing with the environment, but also helped in the development of cultural achievements of the race. However, this shortcut left powerful parts of our brain only subconsciously utilized. If we acknowledge the essential difference of difficulty between processing images and processing words, we can postulate that our minds also work on a deeper layer of nonverbal image processing. This task employs a large part of the mind's processing potential, occurs sometimes consciously but often unconsciously, and uses multi-valued rather than binary logic. Our language can only try to approximate this deeper level by words and by a simpler, more decisive binary logic; but words will always have multiple meanings.

[6]See Wierzbicki (1992a). The technique of a thought experiment was used by Kuhn (1964), who showed that basic concepts applied in any scientific theory include very deep, often hidden assumptions about the real world; if these assumptions are too simplistic, the best way to examine their inconsistency is not necessarily through empirical experiments, but through thought experiments.

After reading all of the above, it would be tempting to define intuition as the nonverbal activity of the right hemisphere of the brain (Young, 1983); however, which activity? Such a definition would not be sufficiently operational; first we need to make some distinctions that would be testable, at least in principle.

The human mind undoubtedly works in a complicated, parallel, and distributed fashion. The potential of our mind is enormous and by far not efficiently utilized. Parts of the mind work consciously (i.e., we want to do something and are aware of this), often in the verbal form, but sometimes also in the older, nonverbal form. Parts of the mind work quasi-consciously (i.e., we want to do this but leave the details to other parts of the mind). Finally, parts of the mind work subconsciously (i.e., we do not know that we want to do this and are not aware that we do this, but we can become aware a posteriori of the results of the subconscious workings of our mind).

A decision is related to the preparation of an action. Actions are often consciously prepared and executed, but more usually involve a mix of conscious and quasi-conscious actions; moreover, many of us could quote examples of subconscious actions. Finally, training and experience influence the purposefulness and other qualities of subconscious actions, as in the distinction made by Dreyfus and Dreyfus (1986) between a novice and a master expert.

Thus, we can define intuitive decisions as quasi-conscious and subconscious information processing, leading to an action, utilizing aggregated experience and training and performed (most probably) by a specialized part of the human mind.

Together with this definition, it is necessary to distinguish between at least two types of intuitive decisions: operational, such as when performing functions for which we are well trained; and strategic,[7] for solving a novel problem (including creative activity). Note that our definition is rather broad: it includes intuitive decisions that have conscious components but that also take advantage of the parallel processing by the parts of our mind specialized in subconscious or quasi-conscious operations. Thus, every day our mind makes many intuitive operational decisions (e.g., when we think on other matters while taking a walk). The role of training in making good decisions is fully understood and institutionalized in human society through our education systems. All training in repetitive activities is aimed at delegating them to the quasi-conscious level, i.e., automating them, by establishing shortened or more easily used connections between synapses – specialized roads in the brain.

One could object that the inclusion of these trained activities into the concept of intuitive decisions is unnecessary. However, such an inclusion corresponds well with the everyday understanding of intuition; for example, many persons would say that they intuitively or instinctively turn off an alarm clock upon awakening. The

[7]Sometimes a third, tactical type of decision can also be distinguished.

use of the word instinct is less substantiated in this case, since it has the connotation of an intuition that is acquired genetically, and not through learning.

Moreover, such an inclusion broadens the possibilities of empirical studies of intuitive decisions. The definition of such decisions given above makes empirical studies possible: we can ask subjects of experiments not to use the conscious part of their mind; we can make sure that they do not use it by concentrating their attention on other matters or, as done by Dreyfus and Dreyfus (1986), by saturating the conscious part of the mind; finally, we can use split-mind experiments in clinical research. We can also distinguish between the cases when the subjects are well trained in certain operational decisions. The inclusion of intuition acquired by repetitive training is reasonable because it might have many aspects in common with the truly creative intuition.

Such research would also be valuable for synthesizing the classical, analytic decision theory with some of its novel and creative approaches, such as the concept of habitual domains (see Yu 1990, 1995). It would be desirable to devote more time and research to such investigations of various aspects of intuitive decision making. Such research might also help to distinguish cases where rational decisions are better than intuitive ones. For example, ad hoc intuitive decisions might be worse than well prepared analytical decisions. This circumstance was often stressed as the reason for the development of classical decision theory. However, if the definition of an intuitive decision is not precise and can imply any rash decision, then the statement that any well-prepared decision is better than a rash one is rather self-evident, except in critical cases when there is not enough time (and then it pays off to be well trained and to have well-developed intuition!). Thus, further studies in comparing intuitive and analytical decisions would be very valuable, but must be based on well understood definitions.

2.4 Dynamics and Phases of Decision Processes

The above discussion leads to the following questions: If most decisions are made intuitively, how can we support them using computerized information and models? In particular, how can we better understand and support creative or strategic intuitive decision making?

The complexity and the speed of change of information in the modern world make the use of computerized decision support self-evident. However, the reflection on the role of intuition in decision making changes the focus of decision support. Here we can use the analogy of training in repetitive decision activities and automating them – decision support is aimed at the automation of decisions. However, the issue is to what extent, at which stages, and which parts of a decision process will be reserved for human decision makers.

Classical decision theory concentrated almost exclusively on automating the actual choice. In his criticism of the mainstream theory, Simon (1957) defined the following essential phases of an analytical decision process:

- Intelligence: information and data gathering.
- Design: selecting or constructing a model of the decision situation.
- Choice: examining decision options and choosing between them.

Later, another essential phase of implementation was added (Lewandowski and Wierzbicki, 1989). These phases could be subdivided into further phases. However, such a subdivision of a decision process places the emphasis on analytical decision making. We shall show later that there are various possible subdivisions of a decision process, depending on the particular nature of the process and on a selected emphasis of its special features. For creative or strategic, intuitive decision processes, the following subdivision of their phases was proposed by Wierzbicki (1992a, 1992b):

- Recognition: this starts with a subconscious feeling of uneasiness (well known, for example, by top-level decision makers: if they are not satisfied by some information in a briefing, they ask for further amplification). This feeling is sometimes followed by a conscious identification of the type of problem.
- Deliberation or analysis: if we feel confident as experts, a deep thought deliberation suffices (as also suggested by Dreyfus and Dreyfus, 1986). Otherwise an analytical decision process is useful, but with the final elements of choice delayed.
- Gestation and enlightenment: this is an extremely important phase – we must have time to forget the problem in order to let our subconsciousness work on it. The expected eureka effect might come but not be consciously noticed; for example, after a night's sleep it is easier to generate new ideas (which is one of the reasons why group decision sessions are more effective if they last at least two days).
- Rationalization: this phase can be sometimes omitted if we implement the decision ourselves; however, in order to communicate our decision to others we must formulate our reasons. The word rationalization is used here in a neutral sense, without necessarily implying self-justification or advertisement, though often including the latter.[8] Note the similarity of this phase to the classical phase of choice.

[8]For example, when writing a paper, we obviously rationalize our deeper thoughts – and sometimes also advertise them. Moreover, abstract reasoning helps us to transcend our mesocosm experiences – hence rationalization in abstract terms is evolutionarily valuable.

- Implementation: this might be conscious, after rationalization, or immediate and even subconscious.

Thus, there are five phases of this intuitive decision process, four in an analytical process as opposed to one (and the second phase might actually include two or three previously known phases). An essential issue involves how to support various phases of an intuitive decision process, as with analytical decision support. Alternatively, if we cannot support them directly, can we support them somehow indirectly during a more analytically supported decision process?

Regardless of whether the support is direct or indirect, it is clear that we must concentrate more on the phases of intelligence and of design and analysis, preceding the phase of choice. During a decision support process, there must be enough time and opportunity for learning and thus enriching intuition, helping in gestation and possible enlightenment – all this occurring before the phase of choice or rationalization. We shall show in later chapters of this book how to use model-based decision support and reference point methodology for this purpose. Here we discuss the possibilities of directly supporting the phases of an intuitive decision process.

It is not easy to support the phase of recognition – we must first learn more about how it works – but there might be a special training in perceptiveness. To support deliberation, we might use techniques such as for indirect support of intuition, as described in subsequent chapters.

Especially important is the phase of gestation. Its possible mechanism consists in trying to utilize the enormous processing potential of our mind at the nonverbal level: if not bothered by conscious processing, it might turn to a task specified before as the most important but forgotten by the conscious ego. An interesting fact is that cultural institutions exist for supporting gestation and enlightenment. The advice to empty one's mind, concentrate on void or beauty, and forget the prejudices of an expert from Japanese Zen meditation or tea ceremony practice are useful devices for letting our subconsciousness work (Japanese scientists also attach much importance to the role of intuition in decisions, as in the Shinayakana Systems Approach, see Sawaragi and Nakamori, 1992). The diplomatic custom of receptions to relieve the tension of international negotiations, sometimes accompanied by the unspoken rule of not "talking shop", might serve similar purposes.

As to the phase of rationalization, we have enough support, which includes all computerized means of supporting science and management, with graphic presentation tools, text editors, etc. The issue of supporting the phase of implementation has not been sufficiently studied even for analytical decision processes and deserves more attention.

Can we draw analogies between the support of strategic intuitive decision processes and the training for operational intuitive decisions? There is at least one analogy: letting your subconsciousness work certainly helps, if you are well trained. Korean archers won at the 1992 Olympics using Zen meditation techniques just before competition: is this irrational? One could answer that it is not, as every competing athlete knows the importance of concentration before competition; if we understand how our subconsciousness works, it might help in concentration.

How should the conclusions from the above discussion influence the way of constructing or using decision support systems? Any decision process is multiobjective, particularly if it includes elements of intuition; it should also be creative and aim for novel solutions. If the process is supported analytically by computerized information processing and models, the information not only should be presented to the decision maker in graphic terms, but it should also be rich and multidimensional; we do not know a priori on which piece of information their subconsciousness will focus. In the organization of decision support, we should avoid an early aggregation of objectives into a utility or value function, avoid even pair-wise comparison and other known techniques related to the paradigm of utility maximization. We should also not require consistency from the decision maker; if they know from the beginning what they want precisely, the solution will not be creative. The support should be concentrated rather on helping to generate new options, even to reformulating or formulating new problems, and organizing the dynamics of the decision process in such a way that it leaves enough time for gestation.

Recall that the term decision maker has a very wide interpretation: it might mean either any human involved in everyday activities or only top-level politicians. However, computerized decision support is used now mostly by analysts or by modelers. Analysts can be defined as specialists in various substantive fields that use computers, data and various types of mathematical (logical, analytical, etc.) models for their work, and modelers are a type of analyst specializing in model building. Obviously, the quality of an analyst's work relies on combining his/her expertise and intuition with computerized modeling; thus, good analysts are often adverse to using decision analysis and optimization in their work. The reason for this attitude is the unfortunate concentration of decision analysis on the phase of choice, with the aim of automating decision processes through trying to elicit preference information and then suggesting the optimal decision.

However, as illustrated in subsequent chapters, if we concentrate on multiobjective decision theory and optimization, we can use the tools of this field in order to support the intuition of an analyst rather than to replace his/her intuitive power of choice by a computerized system. This is achieved by applying such tools to the earlier phases of problem formulation and analysis and by treating optimization as a flexible tool of analysis, and not as the goal. In particular, optimization can be used as a tool of multi-objective model simulation that takes into account hard and soft constraints, includes inverse simulation problems (in which model output, not model input, is specified by the analyst), and various aspects of multi-objective model analysis. This provides a sufficiently flexible toolbox to help in treating computerized models as a kind of virtual reality, in which the attainability of various aspirations and the consequences of various approaches to the decision problem are analyzed.

2.5 The Decision Maker of the Future

We observed that decision support systems are used today mostly by analysts or modelers. Although this might change in generations to come, we should not expect this change too soon. Note that high-level political decision makers are elected because of their ability and quality of intuitive generalizations in decision making – thus, they will not use computerized decision support unless they grow up in a culture where such tools are used everyday and by everybody.

The volume and the speed of change of information processed by computerized means will grow together with the maturing of the information society. The education system already takes these facts into account today. With the development of an information society, the role of education will also substantially change. As information processing becomes one of the essential productive factors, the access to information and the ability to process it with modern means will become one of the main social issues. Thus, access to good education, adequate for the challenges of an information society, will be essential for success in life. However, education will at the same time concentrate less on preparing for a designated profession or position in life, but more on providing general abilities and teaching a system of concepts adequate for understanding the new era of the information society. All this indicates that computerized decision support will be used in a much broader sense in the future than it is today.

What characteristics, then, will or should a future decision maker have? Evidently, she/he must be able to use various computerized information and model processing tools. However, more important is being informed of the basic concepts that allow an understanding of the coming cultural era. Finally, the decision maker should know in what proportions to blend computerized, analytical decision support and intuition, i.e., how to use decision support in order to enlarge her/his habitual domain (see Yu, 1995).

Chapter 3

Architecture of Decision Support Systems

Marek Makowski and Andrzej P. Wierzbicki

After explaining model-based decision support and whom it might support, we turn now to the question of what decision support systems are. In order to discuss this question, this chapter is organized along the theme of architecture of a decision support system (DSS); however, we also present other related issues. In Section 3.1, we describe the basic components of a DSS. The impact of classification of stages of a decision process on the architecture of a DSS is described in Section 3.2. Interfaces and other modules of a DSS are described in Section 3.3. Model types and classification of a DSS are presented in Section 3.4. Section 3.5 deals with the design and implementation of a DSS. The final section outlines the structure for two examples of a DSS, which are presented in Chapters 11 and 12, respectively.

A computerized DSS is in fact a software product, often with a complicated structure, sometimes equivalent to a customized software environment; however, usually it can be run on diversified hardware configurations. By its architecture we thus understand not the hardware, but the software composition and organization of the system.

Historically, the architecture of a DSS evolved together with the definitions of a DSS. The concept of a DSS was first used in the late 1970s in opposition to earlier data processing systems: a DSS was supposed to have more functions than simple data processing or a management information system (MIS). This in turn was prompted by the development of interactive computing (see Sprague, 1983; Andriole, 1989). A corresponding early definition of a DSS given by Sprague stressed that it is an interactive computer-based system designed to help decision makers use data and models to solve unstructured problems. However, interactive computing is today taken for granted, not only in DSSs; the discussion of the role of intuition in the previous chapter shows that even for problems that seem to be well structured, an approach encouraging learning about the problem rather than solving it might be preferable. Thus, only the issue of helping decision makers maintained its importance, although both the interpretation of what a decision maker is and what it means to help her/him also evolved over time.

3.1 Basic Components of a DSS

Early DSSs were primarily data oriented, but soon it was acknowledged that there should be more possibilities of evaluating alternatives or decision options and even suggesting "best" decisions. Therefore, model-based decision support was introduced (see e.g., Bonczek *et al.*, 1981).

The most straightforward way of model-based decision support provides the opportunity to evaluate scenarios through some form of simulation. A more sophisticated form suggests decision options or supports a search process for the most attractive option. This can be based on using various analytical decision support tools, including optimization algorithms or solvers, together with models in analytical form (sometimes also called operation research form).

However, there are also other ways of obtaining decision options or alternative solutions. For instance, we can use knowledge bases and rule-base reasoning, or models in logical form together with algorithms of an inference engine type (i.e., software to check the validity of a logical statement). However, in environmental applications we usually encounter some physical or biological basic processes that have reasonable analytical models. In theory, we could convert any analytical model into a logical one by defining sufficiently basic concepts or facts and specifying the necessary logical relations between them; however, the growth of computational complexity related to such a conversion might be prohibitive. Therefore, we restrict our attention to models of an analytical type. Of course, rule-based reasoning can sometimes also be connected to analytical models, for drawing conclusions from analytical results, or for constructing scenarios, etc.

Whatever the model type used in a DSS, the importance of a model base was recognized early; a widely accepted view is that a DSS has at least three main components: a data base, a model base, and a user interface (Sprague and Watson, 1993). The user interface is an obvious main component of a modern DSS. As in all modern software products, the interactive character of a DSS implies that the interface should be intelligent and user friendly. For example, even data base software is equipped with a user-friendly and diversified interface; this has a decisive impact on the usefulness of the software package. If augmented with the definitions of accounting variables and accounting formulae (which are in fact models), even spreadsheet software can be interpreted as a simple DSS implementation.

However, there are two additional but essential components of a modern DSS. One of them, essential when using analytical models, is a library of algorithms or an algorithmic base.[1] It is obvious that when we use models we must also have algorithms to evaluate and analyze them. We should stress here, however, the diversity of algorithms used for analytical models, which makes it necessary to distinguish an algorithmic base or library as a separate component of – or addition to – a modern DSS.

Another essential component of a modern DSS holds responsibility for the overall organization and the decision process; we shall call it the organizing module. Often, this component is implemented as part of the user interface and, therefore, usually it is not mentioned separately. However, there are two reasons to consider it separately, even if it might not be a separate module in implementation.

First, the organizing module links together all other parts of the DSS, is responsible for data handling (including access to data and archiving any information that a user may want to store), and the order of using various other parts of the DSS. Second, this order usually reflects a philosophy of, and a specific approach to, the organization of the decision process. In short, the organizing module has to check whether a particular module can be used at a given stage of the decision process (e.g., if all steps that provide a consistent set of data have been executed).

In this book we follow a practically tested philosophy of this organization: we allow the user as much flexibility as possible in organizing the decision process. The role of the organizing module is therefore to suggest to the user a sensible organization. As has been stressed in the DSS literature, decision processes have an essentially dynamic character. Thus, other components of a DSS might have various functions during consecutive phases of a decision process. Moreover, as already stressed in Chapter 2, a unique way of describing a decision process and organizing it does not exist; therefore, the more flexibility left to the user, the better.

Some specific approaches to decision support, even when concentrating on the dynamics of decision processes, unfortunately presuppose an organization of the decision process and impose it on the user. This is, however, usually not acceptable to the user. A good implementation of such an approach must be at least flexible and allow the user to start from any chosen phase of the decision process.

[1]When using logical type models, one type of inference engine might suffice, while algorithmic solvers for analytical models are more diversified.

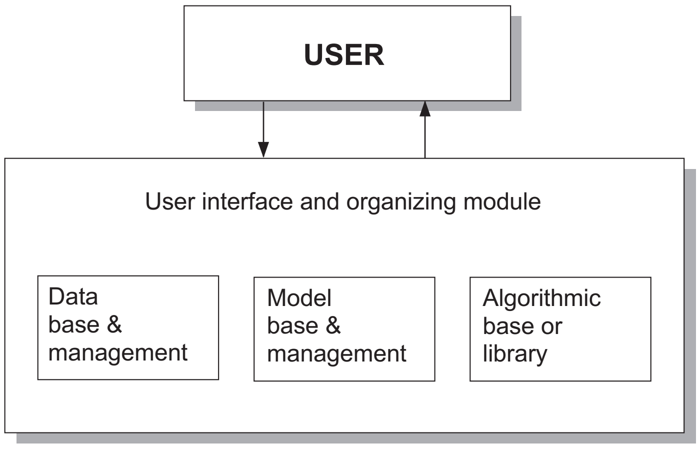

USER
User interface and organizing module
Data base & management
Model base & management
Algorithmic base or library

Figure 3.1. Major components of a model-based DSS.

Alternatively, decision processes do follow some stages and it is useful for the user to have – at least in a software tutorial – some suggestions about possible phases of such processes. Moreover, the functions of a DSS in various stages of the decision process are diversified to such a degree that a clear distinction of possible stages is necessary. We conclude that a DSS should also help the user in a rational organization of the decision process; this is, in fact, the main purpose of the organizing module. This help, however, must be flexible enough and consist, for example, in defining various typical functions of the DSS that the user might select.

After this preliminary discussion, we recall (see Wessels and Wierzbicki, 1993) one of the more current definitions of a DSS: a DSS is a computerized system that supports its users in a rational organization and conduct of a decision process (or its selected phases) and, besides a data base, also contains a pertinent knowledge representation in the form of models of decision situations as well as appropriate algorithms for using these models. From this definition, a general scheme of a DSS is shown in *Figure 3.1*.

The scheme in *Figure 3.1* stresses an important aspect of the definition of a DSS: the sovereign position of the user. In any phase of a decision process, a DSS must not replace the user in their sovereign decision making; it only helps them in various tasks. Operational decisions of a repetitive nature might be automated to some degree; but even then the human decision maker has usually the authority to override the automatic equipment. In all other cases, the final responsibility for the outcomes of the decision process rests with the users of a DSS, especially when novel, strategic decisions are made and the main aim of a DSS is to enhance the creativity and intuition of the user rather than to automate the decision making.

The sovereign position of the user is also decisive for the success of a DSS owing to two main reasons: (i) market – the user will not buy an unfriendly or domineering system; and (ii) psychology – the user values her/his expertise, intuition, and right to change opinions.

3.2 Stages of a Decision Process and their Impact on the Architecture of a DSS

The functions of the basic components of a DSS shown in *Figure 3.1* might strongly depend on the specific purpose of the system. The organization of the decision process, even if flexibly interpreted as a list of possible functions of the system, depends on the adopted definition of the stages of the decision process.

In this book, we concentrate on the support of the analytical type of decision processes as discussed in Chapter 2, although we stress the value of learning and we recognize the importance of the intuition of the user of the DSS. Moreover, we assume that the user might be, in most cases, not the final decision maker, but an analyst or a modeler. Because we concentrate on model-based decision processes, we shall use the essential distinction between two types of models used in the decision process, indicated already in Chapter 1:

- Substantive models: these aim at representing a part of outside reality relevant for the decision situation being analyzed.
- Preferential models: these are supposed to represent the preferences of the user, either only partially or more completely (in the sense of modeling a value or utility function).

Note that substantive models (sometimes also called core models, see later chapters) should be as precise as possible and might be developed either as part of the decision process or independently. Alternatively, preferential models might change during the decision process or be imprecise. In Chapter 2 we argued that precise preferential models should not be developed too early, as this might be detrimental to the learning capabilities and the intuition of the user. Thus we would either build the substantive models independently, or in the early stages of the decision process; we should introduce the preferential models gradually toward the end of the process.

In practice, the early stages of the decision process devoted to model building might often be treated separately and not considered as part of the decision process. However, we shall include them here for the sake of completeness. A possible specification of the phases of the decision process might then be the following:

Early Stages

- Problem recognition: intelligence and preliminary data gathering, alternative hypotheses about problem and model type, and structure.
- Model design: creating one or several alternative models, including the integration of typical model blocks, data acquisition, and estimation of substantive model parameters.

Middle Stages

- Model analysis: simple or multi-objective substantive model simulation (the last term includes inverse simulation or simulation with soft constraints and will be discussed in more detail in Chapter 5), optimization with various objectives, sensitivity, and post-optimal analysis.
- Option generation and preselection: using the substantive model(s) for generating a number of decision options or decision scenarios, possibly with the help of a partial preferential model; if the number of decision options is large, assisting the user in their preselection.

Late stages

- Choice: estimating the preferential model through interaction with the user; suggesting a solution to the decision problem.
- Implementation: applying the decision and monitoring its effects, with possible recourse to any earlier phase.

A recourse to any earlier phase should be possible after any phase of the decision process. Note that each phase of the above process requires a different type of support for the user, different focus of attention, and often different algorithms. Various approaches to decision support concentrate on some phases of the process while often neglecting other phases; very often, the attention is concentrated either on data acquisition and processing or on the phase of choice.

In this book, we concentrate mostly on the middle phases of model analysis and option generation, though we also discuss some issues related to early stages, such as model building, and to late phases, such as choice. The earlier phases were often neglected in the more classical approaches to decision analysis; at the same time, they are usually of essential importance for analysts using model-based decision support. This is also the reason why we situate such functions as sensitivity and post-optimal model analysis at the middle phases of the decision process, while more traditionally they are situated after the phase of choice. Sensitivity of solutions is an aspect that should be investigated for each relevant decision option, and a comparison of sensitivity of various options should also influence the phase of choice.

All these phases of a decision process might involve various users of a DSS – modelers and analysts in earlier phases, actual decision makers or their advisors in later phases, etc. The concentration on selected stages of the decision process also has a clear impact on the architecture of a DSS. Systems that concentrate mostly on the phase of choice usually have a slightly different architecture from DSSs presented in this book; these concentrate mainly on model analysis, option generation, and preselection. In both cases, the main elements of a DSS as shown in *Figure 3.1* will be present, but with different proportions and emphases.

We should also note that the first phase of problem recognition is especially important and should not be neglected; however, until now there have been no general and reliable approaches to decision support in this phase for a single decision maker or user. If the problem recognition relies on teamwork, however, reliable techniques of groupware exist (see Johansen, 1988; Bostrom; 1992). These techniques support group brainstorming via a computer network, thus saving much time and stimulating more complete participation than face-to-face meetings where participants have to speak sequentially. Another approach is that of a decision room as suggested by Nunamaker (1990). It is still an open question whether such groupware techniques would be equally useful for problem recognition when applied by an individual user. In this phase, we must rely more explicitly on the knowledge, expertise, and intuition of the decision maker or analyst; the groupware techniques facilitate the use and enhance these qualities through teamwork.

3.3 Interfaces and Other Modules of a DSS

A very important component of a DSS is the user interface. Obviously, this should be consistent with the organizing module, should stress the changing focus of attention during the consecutive phases of a decision process, and allow easy switching between these phases; however, there are also other requirements. Ideally, there should be as many interfaces as there are users, and a good user interface has options allowing it to be personalized. The importance of using graphical symbols and advanced graphic representations of both data and various results of analysis is widely recognized. However, it should be noted that each profession (e.g., accountants, control engineers, manufacturing engineers) has its own language of graphical symbols corresponding to basic concepts for this profession and acquired during a professional education. Thus, a good user interface in a DSS should use not only universally accepted symbols (such as icons in a windowing system) but should be customized for the needs of a specific profession. We are still far from the development of good user interfaces for a DSS, despite many creative attempts to include new concepts in these (see Chapter 10).

As data acquisition and processing functions of a DSS are well described elsewhere (e.g., Bonczek *et al*., 1981), we shall not comment in detail on the functions of the data base management component of a DSS. It should be noted only that data formats in a typical data base might not necessarily match the needs of algorithms, e.g., for a statistical estimation of model parameters. Generally, data formats often need to be converted when switching between various modules of a DSS. Similarly, we shall not comment in detail on various approaches to the estimation of model parameters. There are several professional software packages for statistical analysis, including model parameter estimation; however, they are typically not integrated with the software for a more advanced model analysis.

The last two components of a DSS – a model base and an algorithmic base – determine the usefulness of a model-based DSS. They are strongly interlinked, because algorithms must match not only the types of models used, but also the particular functions of a DSS at a given phase of the decision process. Thus, a certain compromise between the generality of an algorithm and its usefulness for particular functions must be made. This even holds for models of logical type: in that case one inference engine might be sufficient for most types of tasks; however, if we go into the detailed functions of a DSS, it turns out that specific variants of logical knowledge processing are needed.

When using analytical models, the situation is more complex. Development of a mathematical model that meets the requirements of a DSS is a complex problem in itself. Currently, there are no universally accepted standards for the development and maintenance of mathematical models; however, there is hope that structured modeling (see Section 6.4.7) as proposed by Geoffrion (1987a) will provide such standards.

Most model analysis tasks – starting with parameter estimation in the early phases, but also including advanced simulation, multi-objective simulation, and optimization in the middle and late phases – can be reduced to single-objective optimization. However, specific optimization algorithms are needed for each of these tasks and various analytical model types require further diversification of algorithms. An algorithm to be included in a DSS – such an algorithm is often called a solver – must be robust: it should run effectively for a wide variety of model instances and should not fail for more difficult computational tasks without explanation and advice to the user. Thus, optimization should be treated as a reliable tool, not as an abstract goal in a DSS. We comment in more detail on various algorithms for model-based DSSs in Chapters 6 and 7; here we restrict our attention to the functions of a model base.

3.4 Model Types and DSS Classifications

Experience in building and using a model-based DSS indicates that the model base should support the following requirements in the early (which might not be counted as part of the decision process), middle, and late stages:

Early Stages of Model Building

- Creation of new models either from scratch, or through modification of an existing model, or through integration and possibly modification of some existing or standard model blocks.
- Model parameter estimation using a reliable statistical analysis and parametric optimization procedure.
- Cataloging and accessing models in the model base, typically in a hierarchical structure corresponding to various model instances and diversified formulations of model-related problems.

Middle Stages of Model Analysis

- Simple model simulation including an interactive definition of model inputs, visual representation of simulation results, and comparison with experimental data.
- Advanced model simulation and analysis, including multi-objective analysis and optimization, sensitivity, and post-optimal analysis.

Late Stages of Choice and Implementation

- Using models for option generation and preselection, with partial preferential models and special tools designed for this purpose.
- Using models for the final decision choice, including the estimation of complete preferential models and various techniques supporting choice.
- Using models for monitoring actual decision results with the purpose of checking model adequacy and viability.

Unfortunately, these requirements are somewhat idealistic; in most cases, DSSs concentrate on only a part of these tasks. Some of these tasks will be discussed in more detail in the following chapters; here we comment only on issues concerning creating models and using various model types.

Creating models is an art that requires knowledge and an understanding of various model types. We should stress, therefore, basic distinctions between the classes of models used in decision support. The first of these concerns a distinction already mentioned: preferential versus substantive models. The former are intended to represent the preferences of the user and are important in the phase of choice (but we should not rely too much on explicit preferential models, because this might automate the decision making too much and violate the principle of the sovereignty of the user by replacing her/him in this crucial phase). The latter models represent substantive knowledge and expertise about objective aspects of a decision situation. They are formed during the earlier phases of a decision process, and clearly influence the phase of choice. It is good practice to keep these two kinds of models separated in a DSS. Substantive models constitute the kernel of model-based decision support; they are formulated by modelers – expert specialists and analysts – and form the basis of the knowledge encoded in a DSS.

Another basic distinction already mentioned between the two classes of substantive models concerns logical models and analytical models. In expert systems, artificial intelligence, etc., logical models are used; the methodologies of knowledge bases (which might be considered as a special case of model bases) and inference engines (a special case of algorithmic bases) provide many powerful tools for decision support. But there are also inherent limitations to knowledge representation by logical models. The second class of analytical models can better represent more complicated, non-binary cause and effect relations, e.g., in terms of feedback loops; such models are also more diversified than logical models.

Most books on DSSs stress the distinction between an operational research type, control theory type, and other types of analytical models; however, these types are not generic. A more generic taxonomy of analytical model types can be obtained by subdividing them into linear and nonlinear, static and dynamic. However, we must stress several more basic aspects of such a taxonomy.

It is important to distinguish between models with continuous and discrete decision variables. For very simple cases of decision situations it is also simpler to assume discrete values of decision variables (such as enumerating decision options). However, for more complex situations the reverse is likely to happen: analyzing models with discrete (binary or integer-valued) decision variables leads to greater computational complexity than those with continuous-valued ones. For example, although a continuous-valued nonlinear model might be solved only approximately, its approximate solution can be obtained much faster than a precise solution for a more computationally demanding discrete-valued model. Therefore, models with mixed (i.e., continuous and discrete) variables are typically considered in association with the discrete, not the continuous case.

We must also distinguish between models with various types of representation of uncertainty; uncertainty is an essential element of decision analysis and its representation should be adequate for the specific decision situation.

Implicit uncertainty representation (sometimes called deterministic or averaged representation) assumes that it is sufficient to construct a substantive model for either a representative or averaged case. Explicit uncertainty representation has various more specific types, such as the following: probabilistic, dynamic probabilistic, or stochastic; interval or set-valued; fuzzy set; and others, including, for example, rough set (see Pawlak, 1991).

Sometimes the term uncertainty is reserved for its probabilistic or stochastic type while other types are referred to as expressing imprecision. This distinction is appropriate when counterposing probabilistic and fuzzy set approaches, but less so for other approaches.

We must also remember that the assumption of the relevance of the average case in implicit uncertainty representation is only an assumption. This assumption should be tested later by analyzing the sensitivity of the model to various parameters and by post-optimal analysis, as in practice this assumption is very often made but seldom tested. Experience in modeling shows that the average case assumption, when tested, works well for models with mild nonlinearity and without strong influence of constraints; if the nonlinearity is strong or if the solutions of the model are strongly influenced by constraints (as in the case of linear programming models), then the average case assumption might give conclusions qualitatively different from a more detailed explicit representation of uncertainty.

Various types of explicit uncertainty representation correspond to different aspects of uncertainty, and are adequate under diverse assumptions. The probabilistic or stochastic representation assumes that there is a frequency-type representation (distribution) of outside events influencing the decision situation. For most cases of such decision situations, a complete (both substantive and preferential) stochastic model is necessary. Only under very restrictive, additional assumptions (Gaussian distributions, linear types of models with a quadratic form of the optimized outcome), stochastic models may be replaced by equivalent average case models (i.e., with implicit uncertainty representation). If there is no frequency-type representation of uncertain events, it is doubtful whether probabilistic or stochastic models are the most suitable (although they have often been applied to such cases, e.g., with subjective probability estimations).

The interval or set-valued approach to uncertainty representation concentrates on the estimation of worst cases. In some applications, this provides very useful information; however, it must be stressed that the worst case estimates might explode (i.e., tend to infinity) and thus not be useful for more complex models.

The fuzzy set theory concerns not the probability, but the possibility of uncertain events that are treated by multi-valued logic with a continuous change of the value of truth between 0 and 1. This theory is best suited to express imprecise (e.g., verbal and natural language) statements or the possible degrees of satisfaction of a decision maker. It is based on a different set of basic axioms from the probability theory; both of them can be used together to deal with outside events, e.g., those that have a given frequency-type representation together with some verbal evaluations. Other possibilities, such as rough set theory, are related to multi-valued logic with discrete truth evaluations (e.g., unknown or not decidable, and true or false).

A third distinction of model type, equally important as the two already mentioned, involves static and dynamic models. In a simplified definition, a static model describes reality in a given time period, while a dynamic model takes explicitly into account many time periods and the relations between the states of modeled reality in consecutive time periods. A more precise definition requires specification of the basic axioms of the concept of a dynamic system or model. This concept is particularly important for decision analysis because it alludes to the definition of a dynamic recourse of decisions: we can make part of the decisions in the first time period and postpone the other decisions until later periods after we have observed the effects of uncertain events. We can also attribute a priori certain decisions to each time period in a manner typical for control theory and engineering. Dynamic models can be discrete time (i.e., defined on a discrete set of time periods) or continuous time (this is later discretized for computations). In physics, chemistry, and engineering another type of dynamic model with distributed parameters is used. The state of the model is a distribution of states over a spatial domain. Originally described by partial differential equations in continuous time, models with distributed parameters are discretized both in space and time for effective computations.

Bearing in mind these basic distinctions of model types, we can list the most frequently used types:

- Linear models with continuous variables of the linear programming type, further subdivided into static and dynamic, deterministic or stochastic, or with fuzzy coefficients.
- Linear models with discrete or mixed variables and similar further subdivisions.
- Nonlinear models with continuous variables and similar further subdivisions.

Nonlinear models with discrete or mixed variables are obviously also possible but less frequently applied. Dynamic models with distributed parameters, valuable for simulations in many research endeavors, are also more difficult to be used together in DSS because of computational complexity.

Finally, what might be most important for decision support is that all analytical models can be treated either as single-objective, or multi-objective. If we want to enhance the creativity of various users, then substantive models should be formulated multi-objectively. This does not mean that several objective functions will be specified a priori, but that various model outcomes can be selected as objectives and optimized. This selection of objectives might change during model analysis.

We should also note that all analytical models can be represented in various forms. The most important are the form of a set of equations (which can have various formats, more or less convenient for the model builder; see following chapters) and the form of a graph – or, equivalently, of a block diagram, often more convenient to the model builder than the form of a set of equations. The form of a graph is also convenient for representing models of queuing theory, network-type models, etc., which often are very useful for DSSs, but not easily classifiable in the above taxonomy.

When creating a model we must remember therefore that a model is defined not only by its subject, but also by its purpose, that is, the same subject might be modeled by different model types depending on the purpose of modeling (see Wierzbicki, 1984). Moreover, the concerns of future users must be taken into account when constructing a model. All this explains our initial statement: creating models is an art, and requires much knowledge (after all, models encode knowledge).

The various model types discussed above influence possible classifications of DSSs. In the following list we show criteria for possible classifications of model-based DSSs.[2]

- *Application area*. A dedicated DSS should be user-oriented. The functions and detailed specification of a DSS should be determined with the participation of future users, and its user interface should rely on symbols and graphic representations typical for a given application area and thus well understood by the user.
- *Application type*. This applies to strategic, tactical, and operational decisions. A DSS for strategic decisions should support learning about the problem and innovative ways of solving it, whereas a DSS for operational decisions might concentrate on information processing and the optimization of typical solutions.
- *Substantive model type*. This refers to expert systems with logical models and analytical systems with analytical models (often and not quite precisely called operations research models). Further subdivision occurs for linear, nonlinear, dynamic continuous models, and discrete models of various classes, etc.
- *Preferential model type*. This corresponds to a selected rationality framework together with selected algorithms of choice.

[2]Various books propose diverse approaches to such classifications; for a more detailed discussion of the approach followed here (see Lewandowski and Wierzbicki, 1989; Wessels and Wierzbicki, 1993; Makowski, 1994a).

- *Representations of uncertainty*. For example, with probabilistic or stochastic models, set-valued models, fuzzy set or even rough set models (these various ways can be used to construct substantive models to be included in preferential models).
- *Type and variety of algorithmic tools*. This refers to the tools contained in the algorithmic base of a DSS that should not be limited to a single algorithm but must contain a variety of them.
- *Principles of interaction with the user*. Very often, a crucial aspect of a DSS is how the user can influence a decision suggested by the DSS and how this decision is explained. This has an obvious impact on the construction of the user interface, which is also influenced by the application area and other aspects, including the number of users and the type of their cooperation. For example, a DSS might be designed to serve a single user or a team of users with the same interests and goals. Quite different types of DSSs could be designed also to serve a group of users that might have different interests but must achieve a joint decision. Other types of DSSs are needed to support bargaining and negotiations in game-like situations, when each user can implement her/his own decisions.

A good DSS should take into account the various criteria discussed above.

3.5 Design and Implementation of a DSS

The issue of design and implementation of a DSS is discussed in many books and from various angles. The fast development of computer hardware and software has facilitated this, but has also meant that certain subjects might retain actuality only for a few years; however, here we shall present some general comments.

The process of designing and implementing a DSS is case specific for each decision problem, and can be subdivided into phases. As software issues are decisive for the design and implementation of a DSS, the phases of this process are often specified similarly to other software engineering projects. However, the methodology of software engineering develops almost as fast as computer hardware and software.

Andriole (1989) suggests the following phases in the design and development of a DSS:

- Requirements analysis and specification, including profiling the future DSS tasks and users.
- System modeling, i.e., developing a functional model of the DSS meeting the requirements.

- Software selection and/or design, including all software modules of the entire DSS – in particular the user interface – as implied by the functional model.
- Hardware selection, including processing, memory, and input–output elements.
- System packaging – putting all parts together, and testing and documenting them.
- System transfer – training the user in system functions and testing the system in its actual functions.
- Evaluation, including the determination of necessary revisions in any of the above stages when repeated the next time.
- Feedback, i.e., actual recourse to any of the above stages.

As in any other decision process, there is also the possibility of a recourse to earlier stages after any of the above stages – not necessarily all of them – are completed. Because of the time and other resources needed to develop and/or revise software, it is advisable to perform all these stages as carefully as possible, in order to minimize future revisions. In particular, the time spent on a precise and comprehensive requirement specification is always valuable; the system prototype must be good enough to interest the future user in investing enough time and effort for the necessary testing and evaluation.

We shall not comment in detail on all these stages of design and development of a DSS, but shall stress only some aspects of them, not necessarily in the same order as listed above.

3.5.1 Requirement analysis

Requirement analysis is equivalent to the phase of problem recognition in any decision process. Therefore, requirement analysis might be classified as one of the most creative phases of DSS development. Fortunately, requirement analysis is almost never done by a single person; thus, it is advisable to apply all known techniques to enhance the creativity of teamwork in this phase, e.g., by applying groupware and brainstorming to stimulate more comprehensive requirement specifications. A specially important role of the future user or users must be, however, preserved in this phase.

There are many lists of typical issues or questionnaires that might help in requirement analysis; a review of them is given by Andriole (1989).

Before the era of interactive computing and object-oriented, modular programming, a prototyping of a software product was considered a bad habit: it was assumed that the requirements for the product should be precise enough to develop the product without any repetitions of the development process. However, interactive computing increased the necessity to involve future users in software design and development, particularly in the case of a DSS. Future users should participate

Table 3.1. Forecast (made in 1991) of capabilities of a typical workstation.

	1991	1996	2001
Memory (MB)	16–64	64–256	256–1,024
CPU speed (MIPS)	20	200	1,000
Disk space (GB)	0.2	1	10
Price ($)	10,000	3,000	Below 1,000
Network throuput (Mb/sec)	10	100	1,000

Abbreviations: MB, megabytes; MIPS, millions of integer operations per second; GB, gigabytes; Mb/sec, megabits per second.

in the development process starting with requirement specifications; however, no matter how carefully done, no requirement specification is ever complete without some experience of the full software product. Therefore, today prototyping is considered the only way to produce a good DSS. The future user is asked not only to provide input for the initial stage of the DSS design but, in particular, to experiment with and to evaluate the prototype system, which leads to a revision of requirement specifications. The design and development process is then repeated, hopefully only for the second time. Such a recourse to the software development process is today much easier than in the past, because object-oriented programming facilitates the development of modular software – and without a modular software structure, all software revisions are extremely costly.

3.5.2 Hardware considerations

An appropriate selection of computer hardware should take into consideration the hardware currently being used by the decision maker and her/his future needs. Future developments in the requirements for DSS functions are inevitable – and it is better to make sure that the requirements are not underestimated. Another aspect concerns software tools provided by a specific computer hardware manufacturer and the related issues of software portability.

Prices for computer hardware are rapidly decreasing, and computer hardware capabilities are rapidly increasing. This trend is even faster than the most optimistic forecasts made several years ago. In order to justify this statement we present in *Table 3.1* a summary of a panel discussion held in 1991 concerning the future of computer hardware (see Makowski, 1994a).

Data given in this table characterize the computing capabilities corresponding to a typical workstation in 1991. The prices in *Table 3.1* do not refer to a typical workstation, but to a configuration with computing power equivalent to the 1991 configuration. Many researchers did not consider this forecast realistic when confronted with it in 1991. Today, the development of hardware seems to be even faster than forecast some years ago: the computing power of the 1996 forecast workstation corresponded to high end PC, which costs about the same as the forecast price for a workstation with the computing power of a typical 1991 workstation. Moreover, the network throughput in 1999 is already much larger than that forecast for 2001.

The decision about the selection of hardware – although crucial in itself – can be, for most applications, easily reached with cooperation between future users and experienced software developers. Usually, the hardware-related costs are just a small fraction of all costs related to the development of a DSS. Therefore, hardware is nowadays no longer a substantial limitation in the design and development of a DSS, although for a DSS that requires the solution of computationally demanding tasks, the hardware costs might still be remarkable.

3.5.3 Software considerations

Software-related problems are key components in any decision related to the design and implementation of a DSS. This is due to three mutually related factors of costs, implementation, and credibility, discussed as follows.

Costs. Software development and maintenance is a major cost component of most DSSs. Direct costs of development of any dedicated piece of software are higher than those of purchasing a ready-from-the-shelf package, whereas the relation of their functionality is just the opposite. All this refers to situations when a ready software package can be purchased, provided that this package suits the problem – which means that the costs of learning, customizing and upgrading the package and the indirect costs of using it are negligible. Indirect costs might be due to the difference of time that is required from a user for learning a ready-from-the-shelf package and that required by using a dedicated DSS. Direct costs of dedicated software development may be greatly reduced by two factors:

- By using appropriate, modular software tools (both as parts of a DSS and for software development).
- By increasing reusability of the developed software, which again depends on a modular structure of the software and on its level of testing, as well as on its portability between various hardware architectures.[3]

[3]This implies that the popular quick and dirty programming technique should never be used in DSS development – even if the development project is far behind schedule.

Implementation. The general belief is that software is never ready on time. This belief is expressed both by users (see Emery, 1987), and by software developers. The latter formulated a so-called 5% principle which says that the last 5% of the scheduled software development time usually takes about 50% of the real development time. Very often, the testing phase of software development is under-evaluated despite the commonly known fact that there is probably never enough software testing. Therefore, quite often either the software is not ready on time or an inadequately tested version of the software is released.

Credibility. There is probably no existing example of a bug-free software package.[4] Thus, it is important to rely on software engineering techniques that not only make the software development easier but also result in better end products. Such techniques include a modular structure of each DSS component, the application of object-oriented programming languages, as well as utilization of well-tested software tools. Reliability of a DSS is obviously a key condition for its usability.

The above arguments can be summarized with the general advice that one should prefer programming languages and tools that make it possible to speedily develop software that is robust (reliable), reusable, and portable. Owing to the reasons presented in previous sections, these tools should also make it possible to perform fast prototyping, modifications, testing, etc.

With such diversified objectives for software development, no single set of general rules exists that should be followed. However, it is sensible to make a list of issues that should be considered. The following software trends belong to these issues:

- A growing importance of networking and data sharing. This will have many implications; e.g., on collaborative computing, computing on the move, and the use of cellular communications will become popular. Another example is the use of network-oriented programming languages, such as JAVA, which is a successful attempt to provide a portable tool that integrates network and object-oriented paradigms.
- Significant challenges for the integration of symbolic and numerical, especially parallel, computation and visualization. This will include tools that will ease development of software in different computing environments and for specific problems.

[4]Therefore, almost all software packages sold on the market contain various disclaimer statements within the licence agreement.

- A gradually increasing share of object-oriented programming languages. This includes a rapid increase of usage of the C++ language.[5] Another trend that might influence future DSS implementation involves fourth generation, non-procedural languages.
- The increasing power of programming tools, e.g., for symbolic computations. However, one should be aware that the more powerful a tool, the less probable is its full control, and tracing bugs is becoming more difficult.
- Increasing demand for guaranteed software, i.e., software that is guaranteed to function according to a given specification. The majority of general purpose software that is widely used today is accompanied by a licence agreement that typically disclaims any responsibility of the software manufacturer for proper functioning of the software. However, the requirements for the software components that are used for the development of complex software systems will have to become much more rigorous than those of today.

Each of the above issues will have a substantial influence on the software industry in coming years. However, an analysis of these problems is beyond the scope of this book. We would like to mention here only one powerful new technology called DecisionNet (cf. Bhargava *et al*., 1996). The software industry and organizations that will use this technology will be able to provide services via the internet (as opposed to the current practice of distributing copies of software.) This will solve a number of problems that the software industry is facing today (distribution of updates of software, illegal copies, demonstration/trial versions, etc.).

DSS software development can be considered an example of a structure with three layers. On the first layer there are fundamental software tools that can again be subdivided into three types: programming languages; general purpose tools (such as utilities for the development of a user interface or for handling typical data structures); and specialized tools specific for a given category of problems (such as for a given type of mathematical programming problem). On the second layer a DSS builder uses these tools and combines them with other modules into a customized, problem-specific DSS. On the third layer the DSS is used by its user (analyst, decision maker).

It is usually difficult and impractical to separate the first two layers, as the generation of a DSS often requires the development of new tools, which, however, should be made reusable whenever possible. Generally, when developing software for a DSS, the following three issues should be taken into account:

Software robustness and reusability. Robustness is understood to mean the ability of software to function according to the specification and the needs of the user

[5]This forecast was made in 1991; it is not only confirmed, but remains topical now.

under wide differences in data and parameter values. This includes appropriate behavior for possibly any data supplied for the program, any action of the user and for typical hardware problems. Software for DSSs should obviously have a high level of robustness. Software robustness is also a necessary condition for software reuse, and justifies careful design and testing of all software components. A large part of DSS software should be reusable both in different applications and by different teams without the necessity of large modifications. This requirement necessitates some additional programming effort, but the effort is well spent when we consider the time necessary for debugging and testing entirely new software. Such reusable software should be carefully designed, tested, and converted into libraries that are well documented. The reuse of some specific software modules, such as algorithmic solvers, requires additionally an agreement on the common structure of data (e.g., when we want to reuse a solver for several instances of a specific class of mathematical programming problem).

Software portability. The portability of the developed software can be relatively easily achieved, if several basic principles are observed during its development. Conforming to these principles pays back, when the developed software can be reused on another hardware or with another operating system. We mention here two basic rules.

The first rule states that software should be developed in a standard programming language (i.e., any compiler-specific language extensions should be avoided; in those rare instances where the use of such extensions is rational, they should be isolated in separate classes). The current standard of C++ contains a powerful library, which also includes STL (Standard Template Library). Use of the new standard C++ library has two advantages. It enables the programmer to work much easier, because the library handles many typical data structures and operations on them. It also increases the performance of the resulting software, as these operations will now be tuned for a specific hardware or operating system platform by a vendor of a compiler.

The second rule states that the user interface should be isolated as a group of classes, and implemented with the use of a commercial library that is portable between various platforms of operating systems. Today, many portable tools exist – especially for the development of a graphical user interface, which before was one of the main reasons for the limited portability of the software developed under DOS. Portability is now easy to achieve and remains mainly a question of careful software design and proper selection of software tools.

Software efficiency. This is especially important for algorithmic solvers in a DSS: the choice of algorithm for solving a problem is critical for DSS software efficiency.

A classical example of basic improvements of a simple algorithm (for selecting prime numbers) that finally result in cutting the execution time by a factor of several thousands is provided in Bentley (1988). However, this applies not only to algorithmic solvers, but also to all DSS software. A well-known rule of thumb states that 20% code accounts for 80% execution time. This critical part of the code, however, might be easily identified with the help of a profiler. Therefore, it is important to know where to spend programming efforts for increasing software efficiency, and where the use of various programming tricks merely for the amusement of the programmer must be severely limited (an indiscriminate use of such tricks often strongly decreases the maintainability of software).

3.5.4 Other issues in DSS development

Future trends in DSS development might be determined by new approaches, such as combining logical knowledge-based and analytical model-based approaches, or using artificial neural nets, or genetic algorithms. All these trends, however, might be dominated by the following methodological issues that most probably will determine the future of DSS development.

The issue of decision automation versus decision support. It is tempting to think that the increasing power of computers might be used for decision automation; however, for reasons described in this and previous chapters, one should be very careful in automating decisions. It can be done only with the full agreement and understanding of the decision maker (the user) and limited to decisions that are repetitive and tedious for her/him. Even in this case, the decision maker should have the possibility to modify the automated decisions. In the case of innovative or strategic decisions, an essential function of future DSSs should be to support the intuition of the decision maker, rather than replace it by automated decisions.

The issue of a user's control of decisions suggested by a DSS. While the user should have the possibility of overriding decisions suggested by a DSS, the system should provide more functions. The essential question is what are the premises (assumptions, model features, model parameter values) that result in the difference between the decision suggested by the DSS and intuitively preferred by the user? And how far should the user modify the premises (how far is it necessary to change assumptions, model features, and parameters) in order that the DSS suggests a decision coinciding with the one preferred intuitively by the user? Only after understanding the answers to these questions, can the user obtain full support from the DSS and enjoy full controllability of it. Some methodologies of decision support – in particular, those based on reference- or aspiration-led multi-objective optimization – provide full controllability for the user; others do not. In particular, another question is how to provide full controllability in knowledge-based expert systems.

The issue of computational versus cognitive complexity of models. As indicated above, we shall expect in the future more and cheaper computing power, e.g., one thousand times more computing power at the same price in ten years. This does not necessarily mean, however, that we could use one thousand times more complicated models in a DSS. If the computational complexity of solving a problem grows exponentially, the increased computing power might be sufficient only for a model that is twice as complicated. On the other hand, if we do not increase the complexity of the model but require that it should be more thoroughly analyzed, then the increased computing power might be sufficient for an essential improvement in the cognitive aspects of decision support. Thus, we should not seek new dimensions in a DSS by increasing the complexity of models, but should rather think about putting the increase of computing power to a better use and understanding of models without increasing their computing demand.

3.6 Examples of DSS Architecture

The general structure of a DSS discussed earlier in this chapter is only an outline. In practical applications, specific structures might differ in many details. In order to illustrate the advantages of a modular structure of a DSS, we outline in this section the architecture of two DSSs that are presented in more detail in Chapters 11 and 12, respectively.

The structure of the DSS used for the Nitra River basin case study presented in Chapter 11 is illustrated in *Figure 3.2*. The implementation of this DSS is based on modular and portable software tools that support the following functions of a DSS:

- A graphical user interface (GUI) for handling all the interaction with the user. It is linked with the MCMA package which handles the definitions of criteria, aspiration, and reservation levels, as well as the generation of a multi-criteria problem. The MCMA includes ISAAP (Interactive Specification and Analysis of Aspiration-based Preferences, see Chapter 10 for details), that is directly linked with LP-MULTI (see Makowski, 1994c), which in turn is a modular tool for handling multi-criteria problems using the approach presented in Chapters 8 and 10. ISAAP allows for interactive specification of aspiration and reservation

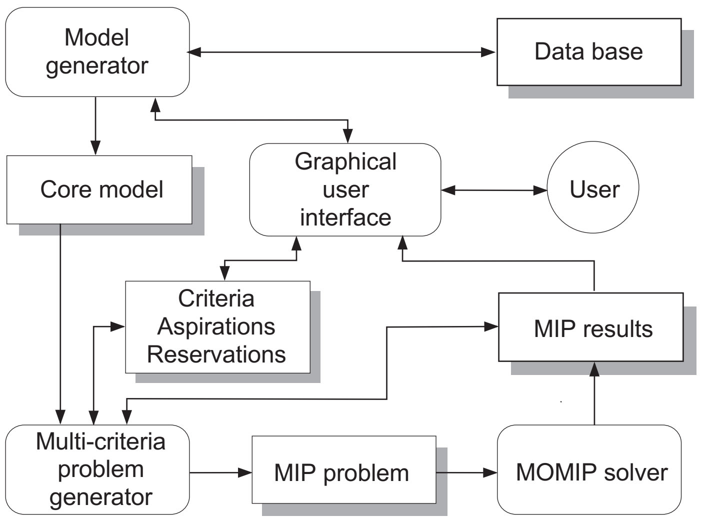


Figure 3.2. Structure of a DSS for water quality management in the Nitra River basin.

levels (and, optionally, for using piece-wise linear membership functions for criteria values between those levels), as well as the changing of a criterion's status. The resulting parametric mixed-integer programming (MIP) problem is based on the core (substantive) model and the aspiration and reservation levels (or on a more precise specification of preferences supported by ISAAP) that represent the current preferential structure of a DM.

- A problem-specific model generator for generating the core model that relates wastewater emissions, treatment decisions, and the resulting ambient water quality. It is important to stress again that the core model includes only physical and logical relations, and not the preferential structure of the DM.
- A modular solver for mixed integer programming problems, MOMIP, developed by Ogryczak and Zorychta (1996).
- A data interchange tool LP-DIT described by Makowski (1994b). This tool provides an easy and efficient way for defining and modifying MIP problems, as well as interchanging of data between a problem generator, a solver, and software modules that serve for problem modification and solution analysis.

The structure of the DSS for land-use planning presented in Chapter 12 is the same as that presented above. Only one substantial module, namely, another problem-specific model generator for generating the land-use core model, had to be developed. Another difference is of a technical nature. The resulting optimization model is of the LP type, therefore the HOPDM solver by Gondzio and Makowski (1995a) is automatically selected by the DSS instead of the MOMIP solver.

Each of these two DSSs has problem-specific utilities for analysis of the results, but the kernel of the software for both DSSs is the same. Additionally, both DSSs can be used on two hardware and operating system platforms, namely, Sun Solaris and MS-Windows (versions for other platforms can be easily developed, because the software is developed with standard C++ and uses a portable library for the GUI).

Chapter 4

Reference Point Methodology

Andrzej P. Wierzbicki

As indicated by the title of this book and the content of previous chapters, we are mainly concerned with using models for clarifying complex decision situations, particularly in environmental applications. This has been one of the major areas of research at the International Institute for Applied Systems Analysis (IIASA), Laxenburg, Austria, and the third part of this book presents several of the studies from there. However, before describing specific cases of model-based decision analysis for environmental applications, it is important to outline the methodological approach that is frequently used in these studies. This approach was started and developed at IIASA in 1980 (see Kallio *et al.*, 1980; Wierzbicki, 1980b), specifically as a tool for environmental model analysis, although since then it has also found applications in engineering design and other fields of decision support. This approach is termed multi-objective reference point methodology.

The main assumption of this approach is the use of multi-criteria optimization as a tool supporting actual decision selection, and facilitating learning about various possible outcomes of decisions as predicted by relevant models. Multi-criteria optimization is also used for generating scenarios[1] of possible development patterns

[1]We should stress here that the concept of a scenario is usually understood in terms of scenario simulation and analysis, i.e., specifying parameters and decisions as a scenario for a given model, then simulating the model for this scenario, finally analyzing the outcomes. However, in this book we shall also use the concept in a broader sense, including scenario generation, or inverse scenario analysis. This can be done by specifying parameters and desired model outcomes (but not decisions) and using multi-objective optimization tools, in particular the reference point methodology described here, to generate the decisions that correspond to such inverse scenarios.

using the accumulated expertise of the analyst. This approach is thus devised for a specific type of decision process that typically arises when using environmental or economic models for generating future development scenarios, or when using engineering models for computer-aided design.

While the reference point methodology has been presented in detail elsewhere (Lewandowski and Wierzbicki, 1989), basic concepts related to this methodology and the decision process are presented. Thereafter we turn to various concepts of multi-objective analysis and reference point methodology. Section 4.1 presents the concepts of Pareto-optimality and efficiency and their role in decision support, and the relation of reference point methodology to efficiency. Section 4.2 discusses multi-objective decision models, objective ranges, and objective aggregation. The concepts of reference points and achievement functions in reference point methodology are presented in Section 4.3. Finally, Section 4.4 presents a broad discussion of the underlying methodological assumptions, the type of decision process considered, and the ways of using reference point methodology for supporting learning and choice. All material in this chapter has a rather conceptual, methodological character, while more detailed or technical aspects of the issues discussed in this chapter will be presented in Chapters 8 and 9.

4.1 Pareto-Optimality, Efficiency, and Reference Point Approaches

We have mentioned before the concept of Pareto-optimality (see Chapter 1), also called vector-optimality or multi-objective optimality (many other names, such as polyoptimality, have also been used). Its essence is the evaluation of a decision or its outcome by several criteria (selected between decision outcomes): the decision is Pareto-optimal if we cannot improve (find another decision that improves) one of the criteria outcomes without deteriorating other criteria outcomes. Strictly speaking, the term Pareto-optimal is usually reserved for when we either maximize or minimize all criteria. In more general cases, when we minimize some criteria, maximize others, or even do different things with other criteria (e.g., keep them close to prescribed values), we speak about efficient decisions and outcomes; thus, the concept of efficiency just generalizes the concept of Pareto-optimality.

The concept of efficiency is essential for model-based decision support, because it helps to select such decisions and outcomes that might be interesting for the decision maker or decision support system (DSS) user. If we define only one criterion, represented by a so-called objective function in a mathematical programming model, then its optimization usually defines only one decision and its outcomes. If we define several objectives, it is easier to help a decision maker or a user of a DSS who is interested in many options of decisions and their outcomes. Evaluating all possible (or admissible in a model) decisions and their outcomes might be too much work. It is therefore often useful to assume that the user of a DSS is interested only in efficient decisions and outcomes,[2] which still might be many, but usually much less than all possible or admissible decisions. This assumption is reasonable because it implies that decisions that might be improved in one objective or criterion without deteriorating any of the other objectives or criteria should not be considered as viable decision options.

Because there are usually still many efficient decisions, their evaluation and overview must be organized somehow. For this, various concepts of vector-optimality might be helpful (see Chapter 8 for details). We present here only some basic concepts. Suppose that the general form of a (substantive) model for decision support is:

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}); \quad \boldsymbol{x} \in X_0. \tag{4.1}$$

In this model $\boldsymbol{x} \in \mathbb{R}^n$ denotes a vector of decision variables (we suppressed a possible dependence of the model on model parameters) and X_0 is a set of admissible decisions that is usually defined by a set of additional inequalities or equations called constraints. $\boldsymbol{y} \in \mathbb{R}^m$ is a vector of model outputs or decision outcomes. Later we discuss more complicated substantive model forms; here we assume the simple form for the simplicity of presentation of basic concepts.

$Y_0 = \boldsymbol{f}(X_0)$ is called the set of attainable outcomes. The modeler, when analyzing the substantive model, might specify several model outputs as especially interesting; we call these objectives or criteria, and denote them by $q_i = y_j$, forming an objective vector $\boldsymbol{q} \in \mathbb{R}^k$, a vector in the objective space. While this vector and space might change during the decision process (according to specific tasks and changes of preferences specified by the modeler), we denote the relation between decisions and their outcomes by $\boldsymbol{q} = \boldsymbol{F}(\boldsymbol{x})$. $Q_0 = \boldsymbol{F}(X_0)$ is called the set of attainable objectives.

Let us assume that all objectives are, e.g., maximized. A Pareto-optimal decision and its outcome are such that there are no other admissible decisions and thus attainable objectives that would improve any objective component without deteriorating other components. A decision and its outcome that are not Pareto-optimal are called Pareto-dominated. In this case another admissible decision exists with an outcome that is not worse in all components and better at least in one, hence

[2]Naturally, we must then assume – in order to preserve the sovereignty of the decision maker – that she/he has the right to change the objectives or the sense of efficiency whenever it is needed (e.g., if other than efficient – in the current sense of efficiency and with currently defined objectives – decisions are interesting to the decision maker, they might be generated by changing the objectives or definition of efficiency and evaluating decisions and outcomes that are efficient in a new sense).

it dominates the former decision outcome. Equivalently, Pareto-optimal decisions and outcomes can also be called Pareto-nondominated.

There are also other types of Pareto-optimality or efficiency discussed in more detail in Chapter 8. However, for all such types, the sets of Pareto-optimal decisions and outcomes typically contain many elements, not just a single decision and its outcome. Thus, Pareto-optimality is an essentially weaker concept than single-criterion optimality. Pareto-optimality does not tell us which decisions to choose, it tells us only which decisions to avoid. This nonuniqueness of Pareto-optimal decisions has been considered a drawback in classical decision analysis; thus, in addition to a substantive model, a complete preferential model was usually assumed that specified a definite utility or value function whose maximum defined – hopefully, uniquely – "the optimal" decision and outcome.

Such an approach, however, is not necessarily the best for interactive decision support, where the user of the DSS (or the modeler in our case) is supposed to exchange information with the DSS, experiment with the model and modify possible solutions. The nonuniqueness of Pareto-optimal decisions is then an advantage, not a drawback. To use this advantage, we need only an additional way of controlling the selection of Pareto-optimal decisions by parameters specified by the user.

There are essentially two main methods of parameterizing Pareto-optimal decisions:

- By using weighting coefficients, i.e., specifying how much relative importance we assign to various objectives. Mathematically, the method corresponds to, e.g., maximizing the weighted sum of all objective functions over the set of admissible decisions. When the weighting coefficients are all positive (if we maximize all criteria), the maximization of the weighted sum results in Pareto-optimal decisions. However, more important is the issue of whether we could produce all Pareto-optimal decisions (which is called a complete parametric characterization of the Pareto frontier). When using the maximization of a weighted sum, we can sometimes produce all Pareto-optimal decisions and outcomes by changing weighting coefficients, but only under restrictive assumptions – e.g., the set of attainable objectives must be convex (or even strictly convex in order to get unique answers).
- By using goals or reference objectives in decision space, i.e., specifying what objective outcomes we would like to achieve. This method might work in a much more general setting than the method of using weighting coefficients, but it is more complicated mathematically. At first glance, an appealing mathematical method would be to minimize a distance measure or simply a norm of the difference between the goal and the attainable objective vector. Such techniques of norm minimization were first used historically, either in the displaced

> ideal method of Zeleny (1974) or in the broad family of goal programming techniques starting with the work of Charnes and Cooper (1977). However, simple examples (refer below) show that norm minimization might produce decisions that are not Pareto-optimal, thus additional assumptions are necessary. They amount, generally, to limiting the use of goals to objective values that are highly unrealistic. This motivated the development of a different approach (the reference point approach) that uses reference objectives that can be realistic, but avoids norm minimization and instead uses more complicated functions to be optimized (usually, maximized), called order-consistent achievement functions.

Thus, reference point methodology could be considered as a generalization of goal programming, aiming at using arbitrary (not only unrealistic) goals or reference objectives and obtaining only efficient outcomes, at the cost of avoiding norm minimization and replacing it by optimization of a more complicated function. We shall discuss below the relations between these methods in more detail.

The main advantages of goal programming are related to the psychologically appealing idea that we can set a goal in objective space and try to come close to it. Coming close to a goal suggests minimizing a distance measure (usually a norm of the difference) between an attainable objective vector (decision outcome) and the goal vector.

The basic disadvantage relates to the fact that this idea is mathematically inconsistent with the concept of Pareto-optimality or efficiency. One of the basic requirements – a general sufficient condition for efficiency – for a function to produce a Pareto-optimal or vector-optimal outcome (when minimized or maximized) is an appropriate monotonicity of this function. However, any distance measure is obviously not monotone when its argument crosses zero. Therefore, distance minimization cannot, without additional assumptions, result in Pareto-optimal solutions.

Consider, for example, the simplest case when the goal vector is in itself an attainable decision outcome but not an efficient objective vector; norm minimization then leads to the obvious solution with objectives equal to the goals. Even for convex outcome sets, either special tricks or restrictive assumptions are needed in goal programming to provide for efficiency of obtained decision outcomes. If, however, the set of attainable objectives is not convex (e.g., discrete, as in *Figure 4.1*), then norm minimization generally cannot result in efficient outcomes. Both objectives q_1 and q_2 in this figure are to be maximized and the Pareto-optimal outcomes, denoted by circles, are to the "North–East" of the set of attainable objectives. There are many intuitively reasonable vectors of goals, such as $\bar{\boldsymbol{q}}^1$, which would produce inefficient outcomes, such as $\boldsymbol{q}^1$, if a norm as a measure of the distance is minimized.

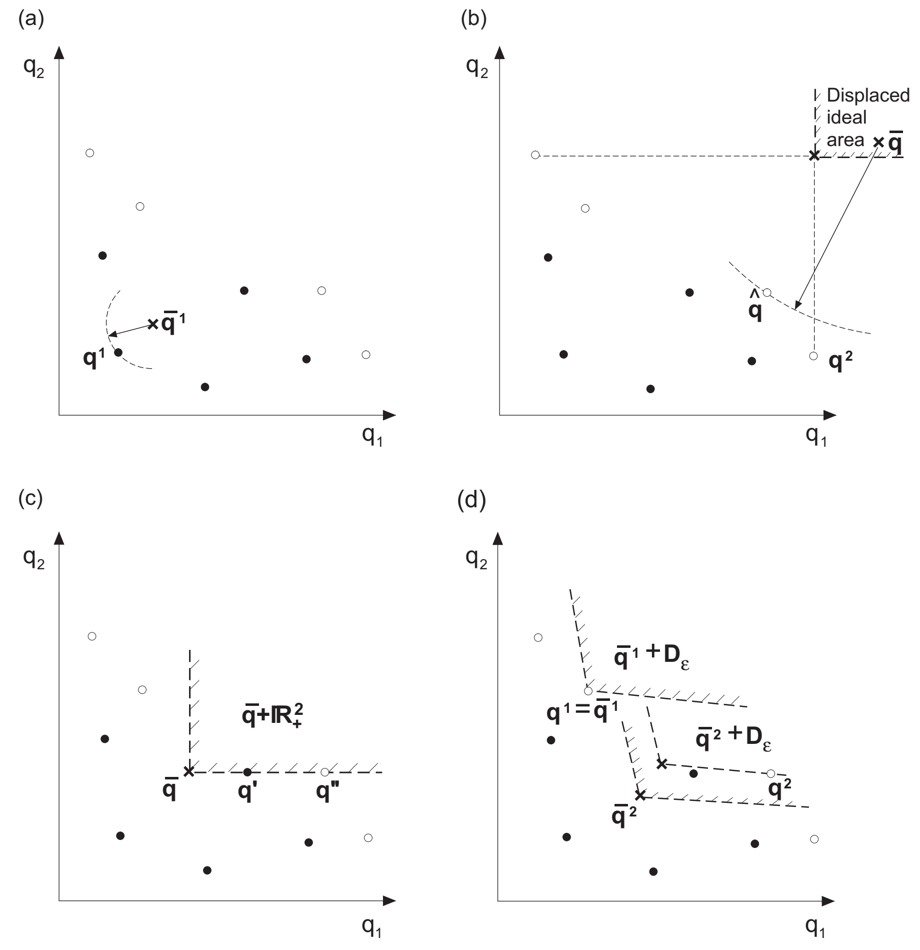


Figure 4.1. Examples of discrete outcomes using various methods: (a) goal programming or norm minimization; (b) displaced ideal; (c) maxmin method; and (d) reference point method. Circles indicate nondominated points, crosses indicate goals or reference points.

The problem with setting a goal and trying to come close to it is how to provide for efficiency of resulting outcomes. There are two ways to do this: either to limit the goals or to change the sense of coming close to the goal.

Trying to limit the set of goals is the essence of the displaced ideal method of Zeleny (1974): if we select goals that are sufficiently distant from the set of attainable objectives, then we can prove that norm minimization will result only in efficient outcomes, regardless of what norm we use or what properties the set of attainable objectives have. This is illustrated in *Figure 4.1(b)*, where the goal $\bar{q}$ is in the displaced ideal area and the outcomes resulting from norm minimization are efficient. However, such limitation of admissible goals means losing the intuitive appeal of the goal programming approach: if unrealistic goals have to be set, the approach loses its basic advantages.

Trying to change the sense of coming close to the goal changes the nature of the goal. Reference points are goals interpreted consistently with basic concepts of vector optimality; the sense of "coming close" to it is special and certainly does not mean norm minimization. If we accept the logic of Pareto-optimality, then coming close to a given reference point should mean:

- Objective outcomes are in some sense uniformly close to the given reference point, if the latter is not attainable (while the precise sense of uniform closeness might be modified by demanding that the resulting decisions and their outcomes remain Pareto-optimal).
- Objective outcomes are precisely equal to the given reference point, if the latter is Pareto-optimal, which, somewhat simplified, means attainable without any surplus.
- Objective outcomes are in some sense uniformly better than the given reference point, if the latter is attainable with some surplus. Therefore, the reference point is inefficient, not Pareto-optimal (where the sense of uniform improvement can be variously interpreted).

The first two cases coincide (almost) with goal programming; the third case is, however, essentially different: it means not coming close in any traditional sense, but "coming close or better".

This change of the sense of coming close is deeply related to how people make decisions in reality and how computers should support decisions. In turn, this is related to the concept of satisficing decisions of Simon (1957), which was used as a description of how people make actual decisions (particularly in large organizations) and the concept of quasi-satisficing decisions of Wierzbicki (1982a) which describes how a computerized DSS should help a human decision maker.

According to Simon (1957), real decision makers do not optimize their utility when making decisions, for many reasons. Simon postulated that actual decision makers, through learning, adaptively develop aspiration levels for various important outcomes of their decisions. They then seek decisions that would result in one of two outcomes, that is:

- Objective outcomes as close as possible to the aspiration levels, if the latter are not attainable (which corresponds to an optimization of decisions, but in the sense of the distance from aspiration levels).

- Objective outcomes equal to aspiration levels, if the latter are attainable (which corresponds to stopping improvements).

Satisficing decision making can be mathematically represented by goal programming. In the case of attainable aspiration levels, the decision maker might learn to increase them, but usually not for current decisions, only for future ones. One can ask why; the most probable answer is that decision-making processes are difficult and this assumption reflect inherent human characteristics such as being careful or lazy. Many studies have shown that such satisficing behavior of a decision maker, though it might seem peculiar, is often observed in practice. In particular, the use of various reference levels by decision makers (such as aspiration levels, but also including reservation levels, of which the latter are very important in the theory of negotiations) has been repeatedly confirmed in practice.

Independent of whether a real, human decision maker would (or could, or should) optimize in all cases, we can require that a good computer program supporting decisions through model analysis should behave like a hypothetical, perfectly rational decision maker. An important exception is that the program should not "outguess" its user, the real decision maker, by trying to construct a model of her/his preferences or utility function, but should instead accept simple instructions that characterize such preferences.

Thus, the methodology of reference point approaches assumes two things: first, that the instructions from a user to the computerized DSS have the convenient form of reference points, including aspiration levels and possibly reservation levels, and second, that the user is not asked how she/he determines the reference points. An essential departure from Simon's assumptions and from goal programming techniques, however, is the following feature: the methodology of reference point approaches assumes that the computerized DSS tries to improve a given reference point, if this point is attainable.

Therefore, the behavior of the DSS – not that of its user – is similar to perfect rationality. It does not minimize a norm, but optimizes a special achievement function which is a kind of a proxy utility or value function (of the DSS) such that the decisions proposed by the DSS satisfy the three cases of "coming close or better" described previously. Because of the difference – in the last case of "coming better" – with the satisficing behavior, we call such behavior quasi-satisficing. The latter can be compared to the behavior of a perfect staff (either a single staff member or a team of them) that supports a manager or boss, who gives instructions to the staff in the form of reference levels. The staff works out detailed decisions that are guided by the given reference point.

However, being perfect, the staff does not correct attainability estimates (real, human staff might behave otherwise) and does not report to the boss that the reference point is attainable when it really is not. Instead, the staff proposes decisions that result in outcomes as close as possible to the desired reference point and reports these decisions together with their not quite satisfactory outcomes to the boss. If the reference point is attainable without any surplus, the perfect staff works out the decisions needed to reach this point and does not argue that a different point and different decisions might be better. If the reference point is attainable with surplus, the perfect staff does not stop working and start gossiping over drinks – as Simon's model of satisficing behavior would suggest. Instead, the staff works out decisions that would result in a uniform improvement of outcomes as compared to reference levels, and proposes decisions together with improved outcomes to the boss. Obviously, only a computer program could behave in this perfect, quasi-satisficing manner at all times.

However, goal programming corresponds precisely to satisficing behavior: if the aspiration levels are attainable, then attainable outcomes precisely equal to them exist, thus the corresponding distance is zero. Since we cannot get a distance less than zero, the optimization is stopped (the staff prepares drinks for relaxation).

Thus, reference point optimization is a generalization of the goal programming approach to such cases when we can and want to improve (minimize or maximize) certain outcomes beyond their reference points. For this purpose, a special class of order-consistent achievement functions, sometimes similar but always different from distance functions, was developed, investigated in detail, and applied in many examples and DSSs. We shall discuss the details of these functions in Chapter 8; here we shall present only some basic concepts related to them.

4.2 Multi-Objective Models, Objective Ranges, and Objective Aggregation

While the complexity of various types of multi-objective models will be discussed in more detail in the next two chapters, here we stress the characteristics of some models, already described in the simple form of equation (4.1). Such a model describes a mapping from the set of admissible decisions X_0 into the set of attainable objectives Q_0. The latter set is not given explicitly – we usually only know how to compute its elements – and the issue is how to analyze the elements and properties of the Pareto ("North–East") frontier of this set. This is usually called a multi-objective analysis of a model.

After specifying any set of variables in a model as objectives q_i, the modeler should first know – at least approximately – the ranges in which these variables might vary. This is also important because we often aggregate objectives, i.e., combine them into one function (not necessarily by summation). Many objectives might also have various units of measurement and must be rescaled to dimension-free units before aggregation. Thus, any system supporting multi-objective optimization and model analysis must include a function for estimating these ranges.

The usual way of such an estimation is to compute a so-called ideal or utopia point by optimizing separately each objective and to estimate its counterpart – the nadir point (a lower bound on objectives that are maximized, or upper bound on those minimized). A simple way of approximating nadir components (although certainly not the best) is to take the worst values of objective components that occur when we are computing the best values of other components during the calculations of the utopia point. The exact computations of the nadir point might be much more difficult than the computations of the ideal point. This issue is discussed further in Chapter 8.

In any case, we can assume that the modeler has defined (either arbitrarily or by computing the utopia point and estimating the nadir point) some estimates of ranges of each objective value:

$$q_{i,lo} \leq q_i \leq q_{i,up} \quad i = 1, \ldots, k \tag{4.2}$$

where $q_{i,up}$ (for maximized objectives, and $q_{i,lo}$ for minimized ones) is at least as high or low as the corresponding utopia point component. The range $q_{i,up} - q_{i,lo}$ is approximately as large as the range utopia–nadir. After specifying such ranges, we can reduce objectives to dimension-free scales (e.g., percentage) and then discuss the relative importance of criteria, their weights, and interpret further concepts such as the trade-off coefficients (see Chapter 8). Note that without such a reduction (therefore without estimating objective ranges) any specification of the relative importance of objectives is pointless.

For dimension-free objectives or criteria, a classical way to compute an efficient point (e.g., a Pareto point for all objectives maximized) is to maximize a weighted linear scalarizing function:

$$s_1(\boldsymbol{q}, \alpha) = \sum_{i=1}^{k} \alpha_i \frac{q_i - q_{i,lo}}{q_{i,up} - q_{i,lo}}, \;\; \alpha_i \geq 0, \;\; \sum_{i=1}^{k} \alpha_i = 1 \tag{4.3}$$

where α_i are weighting coefficients that might be interpreted as weights of importance of objectives. If these coefficients are positive, then each maximal point of the above function is Pareto-optimal.

Conversely, if the set Q_0 is convex, for each Pareto-optimal decision $\hat{\boldsymbol{x}}$ there exist weighting coefficients $\hat{\alpha}_i \geq 0, \forall i = 1, \ldots k$, such that the maximum of the function $s_1(\boldsymbol{F}(\boldsymbol{x}), \hat{\alpha})$ with respect to $\boldsymbol{x} \in X_0$ is attained at $\hat{\boldsymbol{x}}$.

Thus, it might seem that the weighted linear scalarizing function covers all that is needed for multi-objective optimization. However, there are at least two important reasons to consider such functions as highly inadequate. First, the necessary condition of Pareto-optimality, stated above with the help of the linear scalarizing function, does not account for nonconvex (particularly discrete) sets Q_0. The second reason for the inadequacy of a weighted sum for multi-objective analysis is the fact that maximal points of linear scalarizing functions might depend (in important cases) discontinuously on the weighting coefficients. Both reasons are discussed in more detail in Chapter 8.

For these reasons, it is better to use other functions rather than the weighted sum in order to aggregate various objectives. Historically, and particularly in goal programming approaches (see Charnes and Cooper, 1977), much attention has been given to the use of a norm that characterizes a distance of an attainable outcome $\boldsymbol{q} \in Q_0$ from a reference point $\bar{\boldsymbol{q}} \in \mathbb{R}^k$:

$$s_2(\boldsymbol{q}, \bar{\boldsymbol{q}}) = \| \boldsymbol{q} - \bar{\boldsymbol{q}} \| \, . \tag{4.4}$$

We assume that all objectives are maximized, but the distance function should be minimized. Unfortunately, such a distance function is inadequate if we would like to use the reference point as the main parameter to change or control the selection of Pareto-optimal solutions. We would like to admit the use of any reasonable reference point, e.g., at least in the range $\boldsymbol{q}_{lo}$, $\boldsymbol{q}_{up}$, including both $\bar{\boldsymbol{q}} \notin Q_0$ and $\bar{\boldsymbol{q}} \in Q_0$. However, if $\bar{\boldsymbol{q}} \in Q_0$, then the minimal points of such a distance function are very rarely[3] Pareto-optimal.

For a general, nonconvex case, Pareto-optimality of the minimal points of a distance function can be secured only if we assume that the goals are in the displaced ideal set, as indicated in *Figure 4.1(b)* (see Zeleny, 1974). However, such unrealistic reference points, very distant from the set of attainable objectives, cannot easily be used as the main parameters for changing or controlling the selection of Pareto-optimal outcomes. Thus, in the displaced ideal and other similar techniques, weighting coefficients were additionally used in a distance function in order to control the selection of its minima (see Chapter 8). However, the dependence of Pareto-optimal or efficient points on weighting coefficients is not transparent enough. Therefore, the use of reference points as the main parameters controlling the selection of Pareto-optimal points would be much more attractive; reference points usually provide better interpretations for the modeler than weighting coefficients. For this reason, instead of a distance function we must use a different, albeit related, function. This function is called an achievement scalarizing function or an order-consistent achievement scalarizing function which, as opposed to a distance function, preserves the monotonicity even if the point $\boldsymbol{q} = \bar{\boldsymbol{q}}$ is crossed.

[3]Only if $\bar{\boldsymbol{q}}$ happens to be Pareto-optimal.

4.3 Simple Forms of Achievement Functions

We shall discuss here only the simplest forms of achievement functions; a more detailed discussion is provided in Chapter 8. Suppose all objectives are maximized and are already rescaled to be dimension-free and aspiration levels $\bar{q}_i$ are used for these objectives. The simplest form of an order-consistent achievement function is then as follows:

$$\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}) = \min_{1 \le i \le k} (q_i - \bar{q}_i) + \varepsilon \sum_{i=1}^{k} (q_i - \bar{q}_i). \tag{4.5}$$

where $\varepsilon > 0$ is a small positive parameter. This function, a prototype of all order-consistent achievement scalarizing functions, is monotone with respect to all q_i, thus all its maxima are Pareto-optimal. The opposite is shown in Chapter 8: for all Pareto-optimal decisions defined in a specific way (with some additional requirements which, however, are reasonable in applications) reference points $\bar{\boldsymbol{q}}$ exist where the maxima of the above function correspond to these Pareto-optimal decisions.

Other order-consistent achievement functions similar to (4.5) were also used in reference point methodology or other similar approaches to multi-objective optimization (see e.g., Sawaragi *et al.*, 1985; Steuer, 1986; Wierzbicki, 1986, 1992b). One further example is given of a general form of an order-consistent achievement function that will be used frequently in this book. Note that the terms $(q_i - \bar{q}_i)$ in (4.5) can be considered as the simplest example of more general functions $\sigma_i(q_i, \bar{q}_i)$ of the objective variable q_i and the reference or aspiration level $\bar{q}_i$. Such function σ_i must have special monotonicity properties and is often defined as a piece-wise linear function (refer to Chapters 8 and 10); this function is called the partial achievement function. When substituting the terms $(q_i - \bar{q}_i)$ in (4.5) by $\sigma_i(q_i, \bar{q}_i)$, we obtain the following more general form of the order-consistent achievement function:

$$\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}) = (\min_{1 \le i \le k} \sigma_i(q_i, \bar{q}_i) + \varepsilon \sum_{i=1}^{k} \sigma_i(q_i, \bar{q}_i))/(1 + k\varepsilon). \tag{4.6}$$

The achievement function $\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}})$ – in either form presented above, as well as other similar functions – is nondifferentiable. In the case of linear models, the nondifferentiability of the achievement function is unimportant, because the function is concave and its maximization can be equivalently expressed as a linear programming problem by introducing additional constraints and dummy variables (see e.g., Chapter 5). However, in the case of nonlinear models, optimization algorithms for smooth functions are more robust (i.e., work more reliably without the necessity of adjusting their specific parameters to obtain results) than algorithms for nonsmooth functions. Therefore, there are two approaches to the maximization of such achievement functions. The first approach is to introduce additional constraints and dummy variables similarly as for linear models. The second approach is a useful modification of the achievement function by its smooth approximation (see Chapter 8).

The reference point approach is related not only to the method of displaced ideal or goal programming, but also to many other techniques of multi-objective optimization. A popular technique (Polak, 1976) is to maximize the minimal of maximized objectives or of their increases above postulated reservation levels. Its use was indicated in *Figure 4.1(c)*. It corresponds to the maximization of the achievement function (4.5) with $\varepsilon = 0$ which, however, has some disadvantages. This minmax method guarantees only weak Pareto-optimality (see Chapter 8 for a more detailed discussion of this definition). In *Figure 4.1(c)*, weakly Pareto-optimal is the point $\bar{\boldsymbol{q}}'$, which is clearly dominated by the Pareto-optimal point $\bar{\boldsymbol{q}}''$. Note that the use of the achievement function (4.5) with $\varepsilon \neq 0$, as indicated in *Figure 4.1(d)*, does not produce weakly Pareto-optimal points that could be improved in one component of the objective vector.[4]

Finally, we briefly consider the concept of a neutral or compromise solution. Zeleny (1974) introduced the concept of a compromise solution (e.g., a Pareto-optimal decision) where its objective outcome is situated "somewhere in the middle" of the Pareto set in objective space. The precise meaning of somewhere in the middle is specified either by defining the displaced ideal point or by using various weighting coefficients. When using the basic achievement function (4.5) and assuming that all objectives are already rescaled to be dimension-free, we might, however, define a neutral compromise solution more precisely: it corresponds to the result of maximizing function (4.5) while putting the reference point precisely at the ideal point and using weighting coefficients equal to each other (i.e., not using them at all, as in the function (4.5)) for the dimension-free objectives with equal ranges. Such a neutral compromise solution is a good starting point for analyzing an interactive process of a multi-objective model.

When using the reference point methodology, the process of analyzing a multi-objective model is simply constructed: first we let the decision maker or the modeler play freely with reference points and ask the DSS to respond to each reference point by maximizing the achievement function, e.g., of the form (4.5). This produces the Pareto-optimal/efficient decisions and their objective outcomes associated with this reference point. By doing this, the decision maker or modeler gains insight into the behavior of the multi-objective, substantive model of the decision situation. This initial phase might be supplemented, if necessary, with various other

[4]This is because the level sets of this function, denoted $\bar{\boldsymbol{q}} + D_\varepsilon$ in *Figure 4.1(d)* (the notation D_ε is used and explained in more detail in Chapter 8) are "broader" than the level sets $\bar{\boldsymbol{q}} + \mathbb{R}^2_+$ in *Figure 4.1(c)*.

techniques of using reference points when supporting choice (see Chapter 9). Experience in multi-objective model analysis shows that the initial phase of learning by freely changing reference points might be most important, particularly for supporting the development of intuition of the decision maker or the modeler.

4.4 How to Support Learning and Choice

We now turn back to a broader discussion and interpretation of the underlying methodological assumptions, theoretical results, and the decision process considered in the reference point methodology.

We assume in this methodology that the decision maker (in this case, a scientist analyzing environmental models, an analyst, or an engineering designer) develops, modifies, and uses substantive models that are specific to her/his profession and that express essential aspects of the decision situation as perceived by her/him. We therefore call the decision maker or the user of a DSS a modeler. In the decision process, the modeler might have to specify at least partly her/his preferences and thus define a preferential model. However, we assume that the modeler preserves the right to change these preferences, thus the form of the preferential model is rather general, e.g., restricted to specifying only which decision outcomes should be maximized or minimized. We call such forms of preferential models partial, because they relate to a specification of a partial order in the objective space.

Such a decision process might be subdivided into various phases. As discussed in Chapter 3, we might either include the early phases concerned with problem recognition and model building, or consider them as lying outside the decision process. We include them for the sake of completeness and consider the following phases:

- *Phase 1.* Problem recognition and formulation, data gathering, and substantive model selection.
- *Phase 2.* Formulation of a substantive model; initial analysis, including model validation.
- *Phase 3.* Selection of a partial preferential model, detailed analysis of the substantive model, generation of scenarios or design options.
- *Phase 4.* Final selection of a scenario or a design, implementation, feedback from practice.

Phase 1, though extremely important, will not be considered in detail. Many known methods of decision analysis and support can be applied for phase 4; however, they require more detailed specification of preferential models and shall be discussed in Chapter 9. The reference point methodology concentrates on methods and techniques that might be used to support phases 2 and 3, although, as shown in Chapter 9, it is also very useful when supporting the final choice.

While it is well known that multi-objective optimization provides various techniques for supporting phase 3, we stress that such techniques, in particular reference point methodology, can be usefully extended to also support phase 2 – often very important and time consuming for the modeler. The application of these techniques in phase 2 might be called multi-objective modeling and simulation, which is understood here mainly as a tool of learning by the modeler of various possibilities and outcomes predicted by her/his models.

The emphasis on learning by investigating possible model outcomes and scenarios requires some more detailed comments. There are many approaches to the concept of learning in systems theory, computer sciences, control sciences, etc. An important observation, related to the concept of dual control of Feldbaum (1966), stresses the issue of the costs of learning. If we want to make optimal decisions, we must have precise models and draw optimal conclusions from them; if we do not have such models, we will make suboptimal decisions.

However, we usually only approximately know what optimality means. Even if we can define a sense of optimality, a concentration on decisions close to optimality might (and usually does) prevent us from gaining additional information. Often the most relevant information can be obtained by implementing decisions that are far from being optimal. Thus, in any realistic decision optimization, there is a trade-off between decisions that are good for learning but otherwise costly and decisions that might be optimal after we have already learned. This trade-off has a dynamic character: it is better to concentrate on learning at the beginning of a control (or decision) process and then to shift the emphasis gradually toward obtaining optimal decisions.

Feldbaum (1966) proposed certain mathematical models of dynamic learning and decision processes and analyzed the problem of dual optimality (in the dual sense of learning and getting optimal decisions) of such a process. However, he was mostly concerned with the issue of learning by a machine – a robot or a computer – and in such cases where the costs of learning can be well specified. Since Feldbaum's work, many other issues of learning by machines have been studied extensively, generally in relation to pattern recognition, expert systems and knowledge engineering, or artificial intelligence. While some tools of expert systems are useful in decision support, we are however more interested in a different issue: how to support learning of a human analyst or decision maker, not of a computer.

In such a case, Feldbaum's observations also apply, but only qualitatively. We certainly learn by mistakes and changing opinions; such learning must have its cost – at least in terms of cost measured in time, if we can use computerized models for experimenting with proxy reality in order to avoid the costs of real-life decision experiments.

As indicated in previous chapters, learning should enhance the intuitive capabilities of an analyst or decision maker as an expert in the field of her/his specialization. If we aim to support such learning by optimization and decision–analytical tools, we must essentially depart from classical approaches of decision analysis. In the early stages of a decision process, we cannot concentrate on modeling explicit preference or utility representation. We cannot even require that the decision maker should be consistent: the inconsistency of the decision maker is valuable in learning. We should rather concentrate on supporting various experiments performed with the help of the substantive model,[5] testing various "What if?" questions.

Diverse experiments with models must in principle rely more on model simulation than on model optimization, at least if the latter is narrowly interpreted as finding the best decisions when we know what to optimize. However, the experience of working with complicated computerized models shows that the classical model simulation tools are not sufficient to address a range of pertinent modeling questions. One such question is: Given certain (model) outcomes, which decisions would produce these outcomes? Answering such questions – typical, for example, for goal programming – involves a process generally called inverse simulation. Even more complicated is the issue of soft constraints in simulation: if we simulate a model and want to constrain some outcome variable, but softly, not with the help of a hard mathematical inequality that cannot be violated, how do we proceed with this model simulation? These and other similar questions of generalized model simulation can be, as shown in Chapter 5, resolved by applying specially modified techniques of reference point multi-objective optimization, treated not as a goal (of finding best decisions) but as a tool for supporting generalized simulation and multi-objective model analysis. During these analyses, the final choice of decisions is not explicitly supported but even postponed.

We suggest the use of "hard" optimization tools to support "soft" learning, deliberation, and intuition formation. We reflect on the framework of satisficing decision making proposed by Simon (1957). We add some necessary modifications, because Simon used it as a description of how people make actual decisions (particularly in large organizations), whereas we would like to use it for organizing the interaction between a human decision maker or analyst and computerized models and programs supporting model analysis.

According to Simon (1957), real decision makers do not optimize their utility when making decisions. He discerns three reasons for this phenomenon:

[5]These experiments might be more valuable the less standard the assumptions are used in them; see the comments in Chapter 2 on the role of thought experiments in clarifying the consistency of assumptions.

- The optimization of decisions, particularly in the face of uncertainty, is too difficult.
- Information and knowledge necessary for optimization is usually incomplete.
- In organizations with more than one decision maker, an unrestricted individual optimization might lead to unnecessary conflicts.

These reasons have been confirmed by much research since their original formulation by Simon (1957), although some of this research has also resulted in new interpretations. While uncertainty might be represented in various ways, let us consider the example of stochastic optimization. Forty years ago it might have seemed impossible to solve stochastic optimization problems more complicated than purely academic exercises; however, the advances both of computer technology and of optimization techniques in this period have been considerable.[6] It is now possible to solve complicated stochastic optimization problems, including, e.g., applications to international financial markets. The arguments about the power of human intuitive information processing, presented in Chapter 2, indicate that people could also solve such problems, possibly even better than modern computers.

However, because intuitive information processing is essentially different from analytical information processing, we could say that people do not usually solve optimization problems in their minds, as such problems are too difficult in their analytical versions. Also, people do not often optimize decisions in the classical sense, e.g., the "economic man" optimizing his utility as a consumer or producer is only a convenient mental model for an aggregated behavior of people on economic markets. With all these reservations, however, there are no reasons preventing us from using modern computer technology and the sophistication of modern optimization techniques to support the analysis of knowledge represented in computerized models.

The second point of Simon (1957) – about the incompleteness of information and knowledge – is crucial. It implies the importance of learning, stressed also by Simon[7] and underlined since then by many other researchers. There are "no limits to learning" (see Botkin *et al.*, 1979), not only in the sense of educational efforts of societies, but also in the sense that all our knowledge is only a model of reality, only to some extent tested, validated, or having withstood repeated falsification attempts by applications. We can and certainly will learn more, thus modify our knowledge and models. This is also, according to Popper (1945), one of the basic premises of an open society: nobody can say that they know the ultimate truth. One of the

[6]Some advances of stochastic optimization are related particularly to the research done at IIASA (see e.g., Ermoliev and Wets, 1988; or Ruszczyński, 1993).

[7]The concepts of Feldbaum (1966) concerning the cost of learning and dual control went further than those of Simon (1957), but Simon was the first to stress the importance of learning in decision processes.

crucial aspects of learning is checking whether the basic concepts we use to understand reality (such as time, space, matter, cause and effect, feedback, complexity, uncertainty and chaos; see e.g., Wierzbicki, 1992a) are adequate for describing the increasingly complex world. Essential for such learning is also the analysis of various parts of existing knowledge, with the use of computerized models and software tools helping in the analysis.

The third point of Simon – concerning conflict escalation in game-like situations with limited numbers of participants – is also very important; extensive research in this field has produced a better understanding of this issue. The noncooperative equilibria of games with a finite number of players are known to be inefficient, i.e., worse for all participants than a cooperative equilibrium. Worse cases can occur if the noncooperative equilibria are nonunique and symmetric to an extent that makes it impossible to rationally predict a choice between them (as illustrated by the game of chicken, and by most cases of multiple criteria games). Players can then insist on decisions or moves related to such equilibria that are more favorable to them, but the joint result is a disequilibrium much worse than that predicted by any equilibrium; this is the prototype case of conflict escalation.

The results of Axelrod (1984) and Rapoport (1989) on the concept of evolution of cooperation and on tit-for-tat-like strategies suggest that human societies learn – though slowly and with much cost – to distinguish the possible causes of various conflict escalation cases and to establish norms of behavior that prevent such cases. Thus, the issue of possible conflict escalation can be taken into account as an additional constraint on individual decisions, no matter how they are prepared – by utility optimization or by other mechanisms.

Earlier in this chapter we described the mechanisms of satisficing decision making as proposed by Simon (1957) and the quasi-satisficing decision making as used in reference point methodology. Satisficing decision making, including its mathematical model (expressed by the goal programming approach), is based on aspiration levels or goals but also represents inherent human caution or laziness. Quasi-satisficing decision making also uses the concept of aspiration or reference levels but describes the behavior of a tireless computer trying to help a human decision maker. Because of this difference, the corresponding mathematical tools are also different: they correspond to the minimization of a norm in the case of satisficing and goal programming and to a maximization of a more complicated achievement function in the cases of quasi-satisficing and the reference point method.

Thus, reference point optimization is a generalization of the goal programming approach to such cases when we can and want to improve certain outcomes beyond their reference points. In this approach, we can use various achievement functions (i.e., the proxy value functions of the computerized DSS, not necessarily that of its user) that were developed, investigated in detail, and applied in many examples and DSSs (refer to Parts II and III of this book, particularly Chapters 8 and 10).

The main conclusion that can be drawn from both theory and practice, and that is integrated in the reference point methodology, is that if we want to learn, we must postpone choice; if we postpone choice long enough, it might become self-evident. In this sense, optimization is used not necessarily to mean the goal of choice, but rather a tool of learning. That does not imply that the decisions obtained by applying reference point methodology are arbitrary. If the decision maker has learned enough, her/his value function has stabilized, and she/he would like to have support in the final stage of actual decision choice, such support can also be provided by the reference point methodology, including interactive procedures of choosing best decisions with proven convergence. In such cases, optimization again becomes the tool (if not necessarily the goal) of choice. This subject will be discussed in detail in Chapter 9.

Chapter 5

Multi-Objective Modeling

Andrzej P. Wierzbicki

Standard textbooks on operational research or optimization techniques often distinguish between model simulation and model optimization. The same distinction is often made by practical modelers. We shall show here, however, that advanced model analysis might combine both these functions, together with multi-objective model optimization, or special functions of inverse model simulation, or softly constrained simulation. In this sense, we can speak of multi-objective model analysis or even multi-objective modeling. Such an approach uses various optimization tools and other algorithms, but not as a goal, only as a tool of multi-objective analysis. This approach differs essentially from the classic one, where optimization is viewed as a goal, a tool of ultimate, optimal choice. At the end of this chapter we comment also on the role of final choice in modeling for decision support; however, here we concentrate more on supporting learning than on supporting choice.

Section 5.1 discusses the basic functions of a model-based DSS and their relations to multi-objective modeling. Section 5.2 presents some general principles of building multi-objective models. Section 5.3 presents some typical and less typical – but more practical – formats of linear models and introduces the concepts of inverse and softly constrained multi-objective simulation. Section 5.4 discusses basic ways of formulating and analyzing nonlinear models. Section 5.5 presents three examples of constructing and analyzing multi-objective models: a nonlinear one, a simple dynamic one, and a more complicated nonlinear and dynamic one with delays. Finally, Section 5.6 indicates how a modeler might be supported in a final choice by multi-objective model analysis.

5.1 Basic Functions of a DSS

In various applications of reference point methodology to support multi-objective analysis of diverse models, the general decision process is further specified, together with various functions of software tools supporting such decision processes and forming together a specific DSS. Typically, such functions and tools include:

- Model generation.
- Model compilation and parameter setting.
- Optimization problem solving.
- Graphical interaction with the user and data interchange.
- Organization of the learning and decision process.

We shall comment on these functions and tools, with some of them described in more detail in subsequent chapters.

5.1.1 Model generation

Model generation corresponds to transforming the mathematical form of a model into a computerized form, according to specified standards. This might be performed when using either a general modeling language or a specifically developed software tool called a model generator. The standards might in turn depend on other software tools used as part of the DSS. We should stress that there are many modeling languages (see Chapter 6) but until now there have been no universally accepted, good standards for model generation, although a candidate for such standards exists – called structured modeling (see Geoffrion, 1987b).

5.1.2 Model compilation and parameter setting

A generated model is typically run many times on a computer with various parameter settings. Thus, it is advantageous to compile the model, i.e., to prepare its binary form of the shortest possible execution time, taking into account the structural form of the model. This is important even for relatively simple, linear models for which we do not need special computation of derivatives. For nonlinear models that will be subject to (single- or multi-objective) optimization, derivatives of various model relations are necessary in order to compute gradients of optimized functions[1] and/or constraints. Older software tools required that the modeler took

[1]Extensive experience with nonlinear optimization software shows that using numerical approximations of derivatives or gradient-free optimization methods should be considered only as a measure of last resort, applicable to smaller models when the analytical form of the gradients – or even higher

the responsibility for needed derivatives, e.g., used separate symbolic differentiation software and then imported the analytical forms of derivatives into the file describing the model. A more contemporary approach is to use specialized model compilers, which include symbolic differentiation of the model.

The originally generated and compiled substantive model can be called a core model; its various instances and related problems investigated with the help of this model can be specified by setting its parameters.

5.1.3 Optimization solvers

When performing either a single-objective or a multi-objective optimization, or a more sophisticated simulation variant (inverse, softly constrained – discussed later in more detail), we need an optimization solver. This is a software tool that repeatedly uses the compiled model – or certain parts of it (objective function values, gradients, constraints) – and finds model inputs or decisions that optimize a given function ("the" objective function in single-objective optimization, an achievement scalarizing function in multi-objective optimizations, etc.). There are many commercial or public domain optimization tools and systems, most of which can be used as optimization solvers in a multi-objective model analysis and DSS. They might require, however, special model generation, compilation, and data interchange tools; they usually have such tools associated with them, but are often adapted only to single-objective optimization.

Optimization solvers might also have other associated modules, such as presolvers, postsolvers, or postoptimal analysis modules. Presolvers convert the original optimization problem to another problem (having the same value of the goal function) that can be solved easier; postsolvers convert the data produced by a solver back to the original format; and postoptimal analysis modules perform an additional sensitivity analysis of optimal solutions. We must be aware, however, that there are at least three classes of optimization problems and that optimization solvers are typically specialized for one or two such classes. These classes are:

- Linear programming solvers.
- Integer and mixed programming solvers.
- Nonlinear programming solvers.

Important issues of dynamic and stochastic optimization are often reformulated and redefined in such a way that a solver from the three types listed can be applied to them; in other cases a dedicated solver has to be developed. Many solvers are

derivatives – is truly not accessible. Current symbolic software can help to prepare analytical expressions even for subgradients of nondifferentiable functions.

designed to handle problems of at least two classes: for example, MINOS[2] solves nonlinear programming problems, but exploits for this purpose modules developed for solving linear problems. In a software system for multi-objective model analysis and decision support, several alternative optimization solvers might be used.

5.1.4 Graphical interface for the user and data interchange

This interface combines two functions: the organization of the learning and decision process (which will be commented upon below), and the data interchange with the user (the modeler). The modeler's data interchange with the system during the learning and decision process is best supported graphically. For example, the specification of reference levels (reservation and/or aspiration) by the user might be supported graphically in terms of membership functions (of the fuzzy sets of satisfaction of the user). The results of an experiment with the model – its simulation or optimization – might be expressed numerically, but they are better represented graphically. Some interface functions are general, thus prototypes exist of these interfaces, such as ISAAP.[3] Such prototypes, however, may need to be customized for specific applications, which have specific requirements for graphic data representation. For example, a dynamic model might require a specific type of representation of the dynamic trajectories of model outcomes – as well as of associated reference trajectories.

When the parameters and reference point given by the user are determined and the model is run together with other software tools, it is necessary to exchange data (on model inputs, outputs, derivatives, etc.) between various software modules. All this requires certain standards of data interchange and might be performed by a specialized software – a data interchange tool. This tool simply represents and operationalizes the knowledge where and how various data are represented and/or needed in other software modules (Makowski, 1994b).

5.1.5 Organization of the learning and decision process

Part of the interface functions is a menu (e.g., organized in appropriate windows) of various functions of the system. As noted before, it is not advisable to go further and impose a certain sequence of tasks and phases of the decision process upon

[2]One of the oldest and best tested optimization systems, developed in the Stanford Optimization Laboratory, see e.g., Gill *et al.* (1981); see also Chapter 7.

[3]ISAAP (a tool for interactive specification and analysis of aspiration-based preferences) is a graphical interface supporting an interactive specification of user preferences in terms of membership functions (together with reservation and aspiration levels) of fuzzy sets of user satisfaction with model outcomes. ISAAP is a public domain software developed at the Institute of Automatic Control and Computation Engineering, Technical University of Warsaw, in cooperation with IIASA. See Granat and Makowski (1998), also Chapter 10.

the user; a flexible approach is much more preferred. The hierarchy of a menu might represent a suggested organization of the learning and decision process (see Chapter 12 for an example). A natural order exists for performing various functions of the system and it might be suggested to the user, but rather in the form of a tutorial. Naturally, the sovereign user has the right to override any such suggested order.

The generation and compilation of a substantive model, also often called a core model, is a preliminary function and might or might not be included in the menu. Once a core model is compiled, the user might perform various model experiments. These experiments include a single model run in a simple model simulation, or multiple model runs in parametric simulation, or multiple model runs as required by various optimization or advanced simulation experiments. Each model experiment might be defined either by selecting it as a separate system function from the menu, or individually by the user when selecting a combination of existing system functions. For each model experiment, the parameters of the model and the experiment are specified to define a model analysis problem. For example, a simple simulation problem is defined by specifying model parameters and model inputs (decisions), or a parametric simulation problem is defined by specifying several such scenarios of model parameters and inputs. After solving each model analysis problem, the problem results might be attached to it; some model analysis problems might be subdivided into smaller subproblems or problem instances, with corresponding problem instance results.

An example of a suggested order of model analysis experiments is as follows:

- Simple simulation.
- Parametric simulation.
- Single-objective optimization.
- Objective range estimation.
- Inverse and/or softly constrained simulation.
- Multi-objective optimization and analysis.
- Post-optimal parametric experiments.
- User-defined model experiments.
- Multi-objective choice.

It is assumed that the modeler, after performing various simple and parametric simulation runs, knows enough to propose certain optimization objectives. The modeler might learn more by first optimizing them separately (single-objectively). Then she/he might choose a set of objectives to be taken into account, define their character and perform the function of objective range estimation (in which all necessary single-objective optimizations might be done automatically by the system).

Before making an interactive analysis of the efficient outcome set in multi-objective analysis, the modeler might first learn more by using simulation experiments of a more sophisticated inverse and softly constrained nature. The multi-objective analysis might consist of looking at several instances of efficient responses to reference points selected by the user, e.g., starting with a neutral efficient solution that corresponds to a reference point in the middle of the respective estimated range of objectives. The neutral efficient solution is usually not satisfactory for the user, hence the interaction using reference point changes might be extensive. Often, such an interaction results in a selection of one or several solutions – decisions with their outcomes – that are in some sense satisfactory to the user. Sometimes, she/he might also need support in the multi-objective choice of such solutions (see the end of this chapter and Chapter 10).

When one or several solutions are selected, they should be additionally tested by post-optimal sensitivity or parametric experiments. The post-optimal parametric experiments correspond to some defined parametric changes – either of model parameters, or of reference points – and repeated optimization runs showing how the results of an optimization experiment might change. The user-defined experiments are actually a generalization of such post-optimal parametric experiments: they are simply sequences, defined by the user, of any of the other model experiments discussed previously.

However, not all of the experiments listed above need to be included in the menu. For example, in most applications to complicated environmental models, an explicit support of the stage of multi-objective choice was not needed – the intuition of the modeler was sufficient to transform the results of entire learning processes and to suggest conclusions. Although a specific procedure of multi-objective choice might be included, it is not necessarily a part of a system of multi-objective model analysis. However, if the system includes a function "user-defined model experiments", the modeler might individually organize such additional support.

In the following sections, we illustrate some of the issues already discussed in a more detailed framework, and consider the principles of model building and some basic types of large-scale models.

5.2 Principles of Building Multi-Objective Models

This section and the ones following show that textbook formats of constructing mathematical programming models are not necessarily the best for practical applications. This is because they correspond to specific tasks of optimization, while a model should be analyzed flexibly. Constructing a computerized mathematical model that can be later flexibly analyzed is partly based on knowledge, but mostly constitutes an art based on experience and intuition.

Thus, when building a mathematical, computerized model, it is important to follow some general rules that are the culmination of substantial experience in this field.

- First, clarify the purpose of the modeling work. If the purpose is simply to "build a model of something", rethink the purpose. Modeling even a small part of reality – e.g., hot water in a cup – and using all of one's accumulated knowledge, will inevitably result in models too complicated even for the biggest computers. Thus, a more limited purpose must be defined for a given modeling project – whether it involves a desire to know how sugar is dissolved in this hot water, or in what conditions the cup might break from thermal stresses, or simply how much more hot water must be added in order to bring the water in the cup to a desired temperature.
- Second, remember that the model is only an approximation and parts of it that should be more precise need to be chosen carefully. Both the purpose and the accuracy of modeling must be limited in order to obtain a tractable model. For example, the water flow in a river can be described by a partial differential equation (PDE). In a computerized model, such an equation would be discretized and approximated by a set of difference equations; however, there are two ways of doing this. One way is to use a PDE solver that automatically adjusts the discretization grain to obtain a given accuracy; this way is correct, for example, when studying flow dynamics, but expensive in terms of computational power. The other alternative is to assume a specific, coarse grain discretization and to deal with a difference equation model; this way saves much computational time and is correct, for example, when studying the impacts of many possible waste treatment technologies in an environmental economy study.
- Third, describe the model in common terms, and in terms related to the modeler's field of expertise. Also attempt to make the model as transparent as possible. Clarify for which audience the model is addressed, and make the model transparent for this audience (usually for colleagues who would continue the modeling efforts). Each discipline and field of expertise has its own language, symbols, and basic concepts. Do not hesitate to use them even if they are inconsistent with a theoretical model format. For example, we shall use (in one of the next sections) the concept of the pitch of gear teeth, basic for a number of disciplines in mechanical engineering. This concept corresponds to a ratio of some other variables, hence requires nonlinear modeling. If we could linearize other parts of a model involving the pitch of gear teeth, should we avoid using this concept? The answer would be no, because we can always add a nonlinear outcome variable to a linear model. As long as this outcome is not optimized, linear solvers might be sufficient for model analysis. However,

mechanical engineering expertise often results in using the pitch of gear teeth as a parameter or an input variable, which requires fully nonlinear modeling. Even if a linear model is simpler, our colleagues – mechanical engineering experts – should not be forced to use it against their will.

- Fourth, trust in the knowledge of good experts, distrust assumptions, language, and model formats that they use. For example, a researcher in a field of environmental economics might have read an elementary book of operations research in which only linear, integer, and dynamic programming (in the sense of the specific dynamic programming algorithm, not the dynamic optimization problem with many possible solution algorithms) were stressed for the sake of compactness of description. Other forms of optimization (e.g., nonlinear, nondifferentiable, and multi-objective) were only mentioned as too complicated to be included in detail. The researcher is then relieved to find that it is not very complicated, and decides to build a linear programming (or dynamic programming) solver for his/her model of environmental economics. This could be a mistake in terms of research time wasted. The correct way would be to concentrate on model building and analysis while first trying various available solvers. Also helpful is to test what formulation of the model is most useful for a given problem (as opposed to testing what formulation of the model is theoretically interesting or complies with a given class of optimization tools).

  Experts in optimization tend to specialize and thus often view optimization algorithms more as goals than tools. For example, an expert in mixed integer optimization might not address the inverse model simulation problem (described in Section 5.3). This is because even if the original model format were to be linear with integer variables, inverse model simulation with most norms (but not with an augmented Chebyshev norm, see Section 5.3) would result in a nonlinear optimization problem; this would be too difficult to solve when combined with integer variables. Thus, experts make assumptions that do not necessarily coincide with those of a modeler; one needs to be aware under what assumptions they formulate a given opinion. The same concerns influence model formats favored by optimization experts; most model formats found in textbooks on operations research are chosen for their theoretical elegance and the convenience of optimization specialists, rather than for the convenience of the modelers.
- Fifth, distrust "fashionable" analytical approaches. Many analytical approaches have favorable descriptions in textbooks because they are theoretically simple and general, elegant, and powerful. However, most general analytical approaches tend to be computationally expensive when increasing model dimensions. A classic example is the dynamic programming algorithm: it is conceptually simple and very general, thus has attracted many researchers and modelers. However, while the dynamic programming algorithm works effectively

for smaller size test problems, a small increase of model dimensions toward a medium size results in an exponential explosion of computing times.[4] Similarly, when using neural net models (which theoretically are very general and powerful) one must be sure of wanting only to produce automatized expertise. If the aim of using models is to gain a deeper understanding of underlying phenomena, neural net models might not provide sufficient insight into these; however, if we want to automatize, for example, pattern recognition in environmental data obtained from satellite images, neural net models might be an excellent tool.

- Sixth, do not take shortcuts through analytical concepts and stages of analysis. Modeling is a decision process in itself, with a complexity and stages similar to various types of decision processes. If shortcuts are taken through any of these stages, there could be potentially adverse effects. For example, modeling in terms of a (single-objective) optimization problem – that is, specifying an objective function and constraints – is a shortcut in analysis. A modeler usually thinks first in terms of model inputs – parameters and decision variables – and outputs – either final variables of substantive interest, called decision or model outcomes, or intermediate variables needed for model construction. These variables might be constrained by specifying their ranges, but usually only part of the constraints is hard; many constraints might be soft (i.e., not describe hard physical or economic reality, but other assumptions of the modeler). In the modeling process, the modeler should go first through model formulation, validation, etc.; optimization can be used, but as a tool for modeling experiments, changing objective functions and parameters, and surveying the efficient frontier in a multi-objective formulation, etc.
- Seventh, look out for surprises in the model; do not stop model analysis until a large number of experiments have been performed with it. A model is more valuable the more lessons that can be learned from it. Therefore, unexpected behavior should be observed, e.g., high sensitivity to some parameters or decision variables when simulating; practical nonuniqueness of solutions when optimizing; and cusp (sudden change of behavior) or chaotic (practical unpredictability in deterministic models) effects in highly nonlinear models. Therefore, the size and complication of the model should be built up consecutively, starting with its simpler variants and analyzing them until they hold no more surprises for the modeler; once this has been done, more complicated or larger variants of the model can be explored.
- Eighth, make the model reusable. Good modeling is a large investment in terms of the modeler's time and intellectual effort; make sure that the model has the

[4]If an algorithm has an exponential complexity, it might produce results in minutes for a model with n variables and require days or months for a model with $n + 1$ variables.

possibility of being used again either by yourself or by others. Therefore, make it as simple and transparent as possible, and include all documentation of experiments performed with the model.

Beside these very general principles, several approaches on the methodology of model building also exist (see e.g., Wierzbicki, 1984). We stress again that the simulation–optimization dualism, emphasized in some textbooks, applies only to single-objective optimization, treated as a goal and not as an instrument. A modeler often finds it necessary to use additional algorithms, including optimization algorithms, which are treated as varied instruments of more flexible and diversified simulation. In order to illustrate this, we consider the typical components of a simulation model. Such a model might contain the following:

- Actions or decisions represented by decision variables. Although decision variables tend to be treated jointly with other model parameters or exogenous variables in simulation, it is good practice to distinguish the variables that might represent possible actions (sometimes called control variables).
- Potential objectives represented by outcome variables. Although in single-objective optimization there is only one variable, and in simulation we treat most model variables as possible outcomes, it is good practice to distinguish between variables that might be of significance to future users and those that are introduced only to help in modeling.
- Various intermediate variables: state variables, balance variables, etc. These are useful for a flexible model formulation; even in single-objective optimization, so-called proxy variables are often used though seldom stressed; in simulation, a good choice of intermediate variables is essential for the flexibility of modeling.
- Parametric variables or parameters. These might remain constant during model simulations but are essential for model validation and alternative model variants.
- Constraining relations: inequalities, equations, etc. These determine the set of admissible decisions and might be divided into direct decision constraints that involve only decision variables and indirect constraints that involve both outcome and intermediate variables. Although often in simulation only direct constraints are used and tend to be treated rather flexibly, there are cases when indirect constraints are necessary and result in an increase in the difficulty of simulation.
- Outcome relations that determine the extent to which the outcome variables depend on the decision variables and parameters. These are often indirect, specified with the help of intermediate variables and equations (such as state

equations in dynamic models, sometimes with recursive or implicit formulae). In optimization, outcome relations are often treated simply as a part of the constraints; while this is correct mathematically, it is bad practice in modeling and even in more sophisticated approaches to optimization.

- A representation of model uncertainty. Various representations of uncertainty might require the use of special, often complicated optimization algorithms.

We might have various objectives when analyzing such a model, depending also on the phase of model analysis; thus modeling is essentially multi-objective.

Several methodological issues are related to the above classification; one of these involves the distinction between hard and soft constraints. It is well known that constraints that are usually represented with a standard form, e.g., an inequality, tend to model two quite different classes of phenomena of the real world. One of these classes contains balances that must be satisfied (such as the balance of energy in a physical model, or domains of model validity); these are called hard constraints. The other class contains balances that we would like to satisfy (such as the balance in a budget sheet); these constraints can be violated (at an appropriate cost) and are called soft constraints.

Soft constraints can be modeled in single-objective optimization by appropriate penalty terms in the objective function; however, the question then arises as to what constitutes their permissible violations. Soft constraints are actually proxies for additional objectives. Good practice in modeling is to consider the representation of every constraint as either hard or soft. In simple simulations, soft constraints are recognized in a natural way: the modeler is interested in how much they are violated, but does not think about them as real constraints. In more complicated simulations involving additional algorithms, the distinction between hard and soft constraints is useful. For example, if the model involves some indirect constraints (on outcome variables), in simple simulations we naturally assume that they are all soft; if some of them are hard, it is a sign that the model is not well formulated and these constraints should be treated as part of an implicit outcome relation that should be resolved with the help of additional algorithms.

This leads to another methodological issue: the distinction between an explicit and implicit model formulation. Usually, simple simulation models are formulated in an explicit way: each outcome relation can be computed by an explicit formula and multiple outcome relations do not contain loops in the definition of subsequent variables. If a system of equations or inequalities has to be resolved in order to obtain a model's solution (which might happen if hard indirect constraints are included in outcome relations, or if multiple outcome relations contain loops), then the model has an implicit formulation and a special resolving operation must be defined together with the model. Such a resolving operation might be a fixed-point or other iterative algorithm, as for economic equilibrium models, or an optimization algorithm. All systems of equations or inequalities might be resolved by optimization algorithms and there is (theoretically and in simplified analyses) no need to distinguish implicit model formulation in optimization problems. However, more sophisticated optimization algorithms also take into account the structure and the implicit formulation of a model.

A resolving operation is iterative; additional time is needed to perform the iterations until an accuracy criterion is satisfied. However, there is a more important issue involved than computational time. If the model represents some phenomena from real life, then the resolving operation should also have a real-life interpretation – such as an equilibration mechanism on an economic market.

Suppose, for example, that the model is of a discrete-time dynamic type, but we assume – for computational convenience – that the resolving operation is finished separately for each discrete time period. A good modeler should be aware that the dynamics of a resolving operation introduce a second, faster time scale in the model;[5] however, the modeler should not necessarily assume that this mathematical trick correctly represents reality. If the modeler assumes that it does not, the model should not be simplified. An explicit mechanism of the resolving operation should be included in the formulation of model outcome relations, hopefully simulating the actual behavior of the process.

There are many other methodological issues in model building related to model validation, etc.; however, they are not considered in detail here. Some of these shall be illustrated in the examples of specific classes of models.

5.3 Large-Scale Linear Models

Linear models provide a good starting point in modeling. In the case of large-scale models, a practical way to develop a model is to prepare first a linear version and then augment it by necessary nonlinear parts.

In a textbook, the standard form of a linear programming problem is usually presented as:

$$\text{“}\underset{\boldsymbol{x}\in X_0}{\text{maximize}}\text{”}\ \boldsymbol{q} = \boldsymbol{C}\boldsymbol{x} \in \mathbb{R}^k; \tag{5.1}$$

$$X_0 = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \in \mathbb{R}^m,\ \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u}\}, \tag{5.2}$$

where “maximize” might mean single-objective optimization if $\boldsymbol{q}$ is a scalar, or be understood in the Pareto sense, or in the sense of another predefined partial order implied by a positive cone, as explained in Chapter 4. Much research has been

[5]Related to phenomena called boundary layer effects in mathematical modeling.

done on the specification of Pareto-optimal or efficient decisions and objectives for linear models (see e.g., Gal, 1977; Steuer, 1986), but even more attention should be paid to the practical aspects of using multi-objective linear models. Note that the standard form uses the equality form of constraints $\boldsymbol{Ax} = \boldsymbol{b}$ in order to define X_0. Other forms of linear constraints can be converted to the equality form by introducing dummy variables as additional components of the vector $\boldsymbol{x}$, but the reason for doing so is actually theoretical elegance. In the practice of linear programming it is known that the standard form is unfriendly to the modeler (Gill *et al.*, 1981). Thus, specific formats of writing linear models have been proposed, such as MPS or LP-DIT format (see e.g., Makowski, 1994b). Without going into the details of such formats, it can be noted that they correspond to writing the set X_0 in the form:

$$X_0 = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{b} \leq \boldsymbol{y} = \boldsymbol{Ax} + \boldsymbol{Wy} \leq \boldsymbol{b} + \boldsymbol{r} \in \mathbb{R}^m,\ \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}, \tag{5.3}$$

where the vector $\boldsymbol{x}$ denotes actual decisions rather than dummy variables, thus $\boldsymbol{x}, m, n$ denote different variables from those in standard textbook form. The model output $\boldsymbol{y}$ is composed of various intermediary variables (hence it depends implicitly on itself, though often in a directly computable way related to a lower triangular form of matrix $\boldsymbol{W}$: outputs defined later depend on outputs defined initially, but not vice versa). It is essential for the modeler to have freedom to choose any of the outputs y_j, including the actual decisions x_j, as an objective variable q_i (j and i are indices of elements of corresponding vectors). In some formats, however, it was assumed that the single objective function must be written in the first row.

Even more complicated formats of linear models are required if we allow for the repetition of some basic model blocks indexed by additional indices. This is the case in linear dynamic models, where

$$\begin{aligned} X_0 \;=\; & \{\boldsymbol{x} \in \mathbb{R}^n :\ \boldsymbol{w}_{t+1} = \boldsymbol{A}_t\boldsymbol{w}_t + \boldsymbol{B}_t\boldsymbol{x}_t; \\ & \boldsymbol{b}_T \leq \boldsymbol{y}_t = \boldsymbol{C}_t\boldsymbol{w}_t + \boldsymbol{D}_t\boldsymbol{x}_t \leq \boldsymbol{b}_t + \boldsymbol{r}_t \in \mathbb{R}^m, \\ & \boldsymbol{l}_t \leq \boldsymbol{x}_t \leq \boldsymbol{u}_t;\ \ t = 1, \ldots T\}, \end{aligned} \tag{5.4}$$

and where $\boldsymbol{w}_t$ is the dynamic state of the model (the initial condition $\boldsymbol{w}_1$ must be given), the index t is usually interpreted as (discrete) time, and $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots \boldsymbol{x}_T)$ is a decision trajectory (also called control trajectory). Similarly, $\boldsymbol{w} = (\boldsymbol{w}_1, \ldots \boldsymbol{w}_{T+1})$ is a state trajectory while $\boldsymbol{y} = (\boldsymbol{y}_1, \ldots \boldsymbol{y}_T)$ is the output trajectory. The vector of variable $\boldsymbol{w}$ should be considered a part of the vector $\boldsymbol{y}$ (it is an intermediary variable, always accessible to the modeler); however, it is denoted separately. This distinction is made because of its special importance, for example, when differentiating the model, we must account for the state variables in a special way; in control theory, the observability issue concerns the distinction between accessible and inaccessible components of the dynamic state of a real plant (Wierzbicki, 1984). Other similarly complicated forms of linear models result, for example, from stochastic optimization.

A modeler who has developed or modified a complicated (e.g., dynamic) large-scale linear model should first validate it by simple simulation, i.e., assume some common-sense decisions and check whether the outputs of the model also make sense to her/him. However, because of the multiplicity of constraints in large-scale models, it might so happen that the common-sense decisions are not admissible (in the model); thus, even simple simulation of large-scale linear models might be difficult.

Inverse simulation can be important for the modeler, whereby she/he assumes desired model outcomes $\bar{\boldsymbol{y}}$ and checks (as for classical goal programming) whether admissible decisions exist that result in these outcomes. Generalized inverse simulation consists in specifying a reference decision $\bar{\boldsymbol{x}}$ and testing whether this reference decision could result in the desired outcomes $\bar{\boldsymbol{y}}$. This can be written in the goal programming format of norm minimization or in the format of maximizing an achievement function. It is also useful to apply the augmented Chebyshev norm (with a changed sign, as we maintain the convention that achievement functions are usually maximized while norms are minimized), i.e.,

$$\begin{aligned}\sigma(\boldsymbol{y},\bar{\boldsymbol{y}},\boldsymbol{x},\bar{\boldsymbol{x}}) \;=\;& -(1-\rho)(\max_{1\le i\le n}|x_i-\bar{x}_i| + \varepsilon\sum_{i=1}^{n}|x_i-\bar{x}_i|) \\ & -\rho(\max_{1\le j\le m}|y_j-\bar{y}_j| + \varepsilon\sum_{j=1}^{m}|y_j-\bar{y}_j|). \qquad (5.5)\end{aligned}$$

The coefficient $\rho \in [0;1]$ indicates the weight given to achieving the desired output versus keeping close to the reference decision. It is assumed for simplicity's sake that all variables are already rescaled to be dimension-free.

A multi-objective optimization system based on reference point methodology can clearly help in such instances of inverse simulation. In such cases, we stabilize all outcomes and decisions of interest and apply partial achievement functions of the form $\sigma_i(y_i,\bar{y}_i)$ to them, similar to those outlined in Chapter 4 for objectives q_i. An overall achievement function then takes the form:

$$\begin{aligned}\sigma(\boldsymbol{y},\bar{\boldsymbol{y}},\boldsymbol{x},\bar{\boldsymbol{x}}) \;=\;& (1-\rho)(\min_{1\le i\le n}\sigma_i(x_i,\bar{x}_i) + \varepsilon\sum_{i=1}^{n}\sigma_i(x_i,\bar{x}_i)) \\ & +\rho(\min_{1\le j\le m}\sigma_j(y_j,\bar{y}_j) + \varepsilon\sum_{j=1}^{m}\sigma_j(y_j,\bar{y}_j)). \qquad (5.6)\end{aligned}$$

It is more convenient for the modeler if such functions are defined inside the DSS, which also has a special inverse simulation function; this prompts the modeler to define which (if not all) decisions and model outputs should be stabilized and at which reference levels.

Even more important for the modeler might be another generalization of the above function, termed simulation with elastic constraints or softly constrained simulation. Common-sense decisions might appear inadmissible for the model, if all constraints are interpreted as hard mathematical inequalities or equations. However, we have already stressed that it is good modeling practice to distinguish between hard constraints that can never be violated and soft constraints, which in fact represent desired relations and are better represented as additional objectives with given aspiration levels. Thus, in order to check the actual admissibility of a common-sense decision $\bar{\boldsymbol{x}}$, the modeler should first consider which constraints in the model are hard and which might be softened and included in the objective vector $\boldsymbol{q}$. Thereafter, simulation with elastic constraints might be performed by maximizing an overall achievement function similar to the above, but defined with respect to objectives q_i. This can be expressed as:

$$\begin{aligned}\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}, \boldsymbol{x}, \bar{\boldsymbol{x}}) \quad = \quad & (1-\rho)(\min_{1\le i\le n} \sigma_i(x_i, \bar{x}_i) + \varepsilon \sum_{i=1}^{n} \sigma_i(x_i, \bar{x}_i)) \\ & +\rho(\min_{1\le j\le m} \sigma_j(q_j, \bar{q}_j) + \varepsilon \sum_{j=1}^{m} \sigma_j(q_j, \bar{q}_j)), \end{aligned} \tag{5.7}$$

where we do not assume that all objectives are stabilized, but include also maximized or minimized objectives corresponding to inequalities in soft constraints. Again, it is better if the multi-objective optimization system has a special function (called elastic simulation), which invokes the specific form of achievement function and prompts the modeler to specify necessary data, in particular, to indicate the constraints that might be softened.

It should be stressed here that if, for a linear model, either (5.6) or (5.7) is maximized with concave piece-wise linear partial achievement functions σ_i, then the underlying optimization problem can be converted to a linear programming problem. If a partial achievement function (e.g., $\sigma_i(x_i, \bar{x}_i)$) is piece-wise linear but concave, then it can be expressed as the minimum of a number of linear functions:

$$\sigma_i(x_i, \bar{x}_i) = \min_{l\in L_i} \sigma_{il}(x_i, \bar{x}_i), \tag{5.8}$$

where $\sigma_{il}(x_i, \bar{x}_i)$ are linear functions. Assume that a similar expression is valid for $\sigma_j(q_j, \bar{q}_j)$. The maximization of the function (5.7) can then be equivalently expressed as the maximization of the following function of additional variables z, z_i, w, w_j:

$$(1-\rho)(z + \varepsilon \sum_{i=1}^{n} z_i) + \rho(w + \varepsilon \sum_{j=1}^{m} w_j) \tag{5.9}$$

with additional constraints:

$$\begin{aligned}\sigma_{il}(x_i, \bar{x}_i) &\geq z_i,\ \forall l \in L_i \\ z_i &\geq z,\ \forall i = 1, \ldots n \\ \sigma_{jl}(q_j, \bar{q}_j) &\geq w_j,\ \forall l \in L_j \\ w_j &\geq w,\ \forall j = 1, \ldots m. \end{aligned} \tag{5.10}$$

Similar conversion principles apply if we have a mixed integer linear programming model; these can even express piece-wise linear models that are not concave (or not convex in the case of function minimization). Thus, we can use inverse simulation or even softly constrained simulation for mixed integer programming models (however, not all heuristic algorithms related to specific forms of objective functions in mixed integer optimization would work for such optimization problems). Inverse simulation or softly constrained simulation can also be applied to nonlinear models. In these cases we can either approximate the achievement functions with their smooth variants, use nondifferentiable solvers, or use an analogous reformulation as in the linear case. However, the latter is at the risk that some nonlinear programming solvers might not work very well in such a case.

Another possibility is to test scenarios related to various values of parameters (generally, all coefficients even of a linear model might depend nonlinearly on additional parameters $\boldsymbol{z}$), and thus test elements of model uncertainty. Similar techniques of multi-objective optimization can be used for this purpose (see Zakrzewski, 1989), where a description and applications (for agricultural economics) of a multi-objective linear optimization system MENTAT is presented with emphasis on supporting the initial stages of model testing and analysis. Even with the applications limited to linear models, a general conclusion can be drawn. This conclusion is further supported by specific examples for nonlinear models: we can define and usefully apply several functions of multi-objective modeling and simulation that consist in utilizing multi-objective optimization techniques for the early stages of model analysis.

5.4 Nonlinear Models

Standards for nonlinear models are less developed than user-friendly standards for defining linear models. The classic textbook format for (multi-objective optimization of) such models is simply:

$$\text{“}\underset{\boldsymbol{x} \in X_0}{\text{maximize}}\text{”}(\boldsymbol{q} = \boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^k); \tag{5.11}$$

$$X_0 = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{g}(\boldsymbol{x}) \leq 0 \in \mathbb{R}^m\}, \tag{5.12}$$

where $\boldsymbol{f}$ is a vector composed of a model's consecutive objective functions $f_i(\boldsymbol{x})$ and $\boldsymbol{g}$ is a vector composed of a model's consecutive constraints of the set of admissible decisions $g_j(\boldsymbol{x})$. However, such a format is seldom convenient for more complicated models. In these cases it is useful to consider various model outputs $\boldsymbol{y}$ and define both the objectives and constraints in terms of such model outputs.

While some standards for specific nonlinear optimization systems exist – such as used in MINOS, GAMS, AIMMS, and AMPL (Brooke *et al.*, 1988; Bisschop and Entriken, 1993; Fourer *et al.*, 1993) – they are devised more for single-objective optimization purposes than for practical multi-objective modeling and analysis.[6] A useful standard was developed in the multi-objective nonlinear optimization system DIDAS-N (Kręglewski *at al.*, 1988).[7] Briefly, it consists in defining subsequent nonlinear model output relations:

$$\begin{aligned} y_1 &= f_1(\boldsymbol{x}, \boldsymbol{z}); \\ \ldots &= \ldots \\ y_{j+1} &= f_{j+1}(\boldsymbol{x}, \boldsymbol{z}, y_1, \ldots y_j), \;\; j = 1, \ldots m-1; \\ \ldots &= \ldots \\ y_m &= f_m(\boldsymbol{x}, \boldsymbol{z}, y_1, \ldots y_{m-1}), \end{aligned} \tag{5.13}$$

together with bounds for decision variables and outputs:

$$x_{ilo} \le x_i \le x_{iup}, \;\; i = 1, \ldots n; \;\; y_{jlo} \le y_j \le y_{jup}, \;\; j = 1, \ldots m \tag{5.14}$$

(bounds for model parameters $\boldsymbol{z}$ are less essential). This way, a directly computable (explicit, except for bounds) nonlinear model is defined. Implicit models can be defined by specifying $y_{jlo} = y_{jup}$ for some j, which is then taken into account and resolved during optimization. Any variable y_j (and x_i, if needed) can be specified as a maximized, minimized, or stabilized objective.

The model equations and bounds are specified using a computer spreadsheet format. The DIDAS-N system includes advanced automatic (algebraic) functions of model differentiation: it presents to the modeler all required partial and full derivatives and prepares an economical way of computing numerically the derivatives of the overall achievement function in a smooth form similar to equation (8.37). A specific robust optimization solver, described in Chapter 7, was developed and included in the system.

[6]We maintain that multi-objective analysis is more practical, as our experience shows that a model should be analyzed multi-objectively even if it is used later for single-objective optimization only.

[7]Developed at the Institute of Automatic Control and Computation Engineering, Technical University of Warsaw, in cooperation with IIASA. Available as public domain software from IIASA, see Appendix.

However, DIDAS-N is a closed, nonmodular system written in PASCAL, difficult for using with larger models, particularly when large-scale linear model parts are included. Therefore, a new system called DIDAS-N++ was later developed (Granat *et al.*, 1994). This system was written in C++ with a modular structure. It includes the possibility of selecting optimization solvers and a choice and customization of a graphical user interface, together with a preferred option for specifying user preferences in terms of fuzzy membership functions controlled by aspiration and reservation levels.

The format of the nonlinear model definition in DIDAS-N++ is similar to that of DIDAS-N (while equation (5.13) does not need to be written consecutively, the system checks its sequence and directs computability while warning the modeler about any loops in model definition). However, the nonlinear part can be linked to a linear part, which is indicated by the general format:

$$\begin{aligned} \boldsymbol{y}_1 &= \boldsymbol{A}_1\boldsymbol{x}_1 + \boldsymbol{A}_c\boldsymbol{x}_c, \\ \boldsymbol{y}_2 &= \boldsymbol{f}(\boldsymbol{x}_2, \boldsymbol{x}_c, \boldsymbol{z}, \boldsymbol{y}_1, \boldsymbol{y}_2), \end{aligned} \tag{5.15}$$

where $\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{x}_1, \boldsymbol{x}_2$ denote the vectors of model outputs and decision variables specific for the linear and nonlinear parts, while $\boldsymbol{x}_c$ is the vector of decision variables common to both parts.

The model is initially analyzed by an algebraic processing and compiling module. This produces an executable file easily linked with other modules of the system (i.e., the organizing module, a graphic interface, a selected solver) and contains all information on how to compute model outputs and their derivatives, together with the possibility of modifying the values of decision variables, bounds and parameters. Thus the compiling process might take a long time for complicated models, but the repetitive runs of the compiled model (needed for its simulation and optimization) are relatively short.

Especially difficult for such compilations are dynamic nonlinear models, which take the general form indicated by the following equations:

$$\begin{aligned} y_{1,t+1} &= h_1(\boldsymbol{w}_t, \boldsymbol{x}_t, \boldsymbol{z}_t, t), \\ \ldots &= \ldots \\ y_{j,t+1} &= h_j(\boldsymbol{w}_t, \boldsymbol{x}_t, \boldsymbol{z}_t, y_{1,t}, \ldots y_{j-1,t}, t), \\ \ldots &= \ldots \\ y_{m,t+1} &= h_m(\boldsymbol{w}_t, \boldsymbol{x}_t, \boldsymbol{z}_t, y_{1,t}, \ldots y_{m-1,t}, t), \end{aligned} \tag{5.16}$$

where the dynamic state $\boldsymbol{w}_{t+1}$ to be used in the next time instant $t = 1, \ldots T$ is defined as part of the model outputs selected by the modeler, $\boldsymbol{w}_{t+1} = \{y_{j,t}\}_{j \in J_{st}}$ or, $\boldsymbol{w}_{t+1} = \boldsymbol{I}_s\boldsymbol{y}_t$, where $\boldsymbol{I}_s$ denotes a selection matrix and $\boldsymbol{w}_1$ is a given parameter vector. The next chapter provides an illustrative example of such a model, specified either in DIDAS-N format or in other formats.

Note that if we wanted to compute the derivatives of model outputs with respect to given parameters z_j,[8] then we would have to solve equation (5.16) together with the corresponding linearized equations defined for each such parameter and for $t = 1, \dots T$. If we express equation (5.16) as:

$$\boldsymbol{y}_t = \boldsymbol{h}(\boldsymbol{w}_t, \boldsymbol{x}_t, \boldsymbol{z}_t, \boldsymbol{y}_t, t); \quad \boldsymbol{w}_{t+1} = \boldsymbol{I}_s \boldsymbol{y}_t; \quad t = 1, \dots T; \quad \boldsymbol{w}_1 \text{ given,} \tag{5.17}$$

then the linearized equations take the form:

$$\begin{aligned} \frac{\partial \boldsymbol{y}_t}{\partial z_j} &= \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{w}}_{(t)} \Delta \boldsymbol{w}_t + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}_{(t)} \Delta \boldsymbol{x}_t + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{y}}_{(t)} \frac{\partial \boldsymbol{y}_t}{\partial z_j} + \frac{\partial \boldsymbol{h}}{\partial z_j}_{(t)} \\ \Delta \boldsymbol{w}_{t+1} &= \boldsymbol{I}_s \frac{\partial \boldsymbol{y}_t}{\partial z_j}; \quad t = 1, \dots T; \quad \boldsymbol{w}_1 \text{ given.} \end{aligned} \tag{5.18}$$

The derivatives in equation (5.18) are evaluated from the solutions of the model (5.17):

$$\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{w}}_{(t)} = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{w}} (\boldsymbol{w}_t, \boldsymbol{x}_t, \boldsymbol{z}_t, \boldsymbol{y}_t, t), \tag{5.19}$$

etc., while $\Delta \boldsymbol{x}_t = 0$ if we do not assume an explicit dependence of $\boldsymbol{x}_t$ on z_j in the model. The necessary computations of derivatives can be rather voluminous. Fortunately, special techniques exist of backward sweep or for utilizing adjoint equations that considerably shorten both the algebraic and numerical determination of derivatives in such complicated models (refer to Wierzbicki, 1984; Griewank, 1994; and Chapter 6).

A further generalization of the dynamic model (5.16) is the inclusion of the influence of multiple delays in state and decision variables:

$$\begin{aligned} y_{1,t+1} &= h_1(\boldsymbol{w}_{\vartheta,t}, \boldsymbol{x}_{\vartheta,t}, \boldsymbol{z}_t, t), \\ \dots &= \dots \\ y_{j,t+1} &= h_j(\boldsymbol{w}_{\vartheta,t}, \boldsymbol{x}_{\vartheta,t}, \boldsymbol{z}_t, y_{1,t}, \dots y_{j-1,t}, t), \\ \dots &= \dots \\ y_{m,t+1} &= h_m(\boldsymbol{w}_{\vartheta,t}, \boldsymbol{x}_{\vartheta,t}, \boldsymbol{z}_t, y_{1,t}, \dots y_{m-1,t}, t), \end{aligned} \tag{5.20}$$

where:

$$\boldsymbol{x}_{\vartheta,t} = (\boldsymbol{x}_t, \boldsymbol{x}_{t-1}, \dots \boldsymbol{x}_{t-\vartheta}); \quad \boldsymbol{w}_{\vartheta,t} = (\boldsymbol{w}_t, \boldsymbol{w}_{t-1}, \dots \boldsymbol{x}_{t-\vartheta}), \tag{5.21}$$

[8]The parameters are assumed here, for simplicity, to be constant in time; if they change in time or if we compute derivatives with respect to decisions $x_{i,t}$, we must increase their number.

while ϑ denotes the maximal delay and the future current state $\boldsymbol{w}_{t+1}$ is defined as $\boldsymbol{w}_{t+1} = \boldsymbol{I}_{st}\boldsymbol{y}_t$. The complete state $\tilde{\boldsymbol{w}}_{\vartheta,t}$, which contains all the initial conditions necessary to start a simulation of the model, also includes the past states $\boldsymbol{w}_{t-1}, \ldots \boldsymbol{x}_{t-\vartheta}$ and decisions $\boldsymbol{x}_{t-1}, \ldots \boldsymbol{x}_{t-\vartheta}$, but not the current decision $\boldsymbol{x}_t$, thus $\tilde{\boldsymbol{w}}_{\vartheta,t} = (\boldsymbol{w}_t, \boldsymbol{w}_{t-1}, \ldots \boldsymbol{x}_{t-\vartheta}, \boldsymbol{x}_{t-1}, \ldots \boldsymbol{x}_{t-\vartheta})$. The inclusion of delays additionally complicates the determination of the necessary derivatives (imagine having to program by hand all the derivatives for a model such as the one above). Hence, special versions of algebraic processing software and model compilers are necessary when taking into account such models.

5.5 Some Illustrative Engineering Applications

The discussion presented in previous sections might seem abstract, therefore, before applying the described methodology to the analysis of environmental models, the applicability of the methodology will be illustrated with some engineering examples.

5.5.1 The design of a spur gear transmission unit

The first example concerns a classic problem in mechanical design – the design of a spur gear transmission unit (Osyczka, 1994). The mechanical outlay of this unit is shown in *Figure 5.1.* The design problem consists of choosing mechanical dimensions (e.g., the width of the rim of the toothed wheel b; the diameters of the input and output shafts d_1 and d_2; the number of teeth of the pinion wheel z_1) in order to obtain a best design. However, there is no single measure of the quality of design of such a gear transmission. Even when attempting only to make the unit as compact as possible (which can be done by minimizing the volume of the unit while satisfying various constraints related to mechanical stresses, and to an expected lifetime of efficient work of the gear unit), we should take into account other objectives, such as the distance between the axes or the width of the rim of the toothed wheel (which is also a decision variable).

The specification of a mathematical model that expresses the available knowledge on designing such gear units is a matter for expert opinion. After all, the modeler is a specialist in her/his specific field and knows best how to choose substantive models for a given problem. This is also the reason why methods for supporting the modeler in model analysis are presented, which do not supplement her/him in the final decisions. Therefore, in the example of gear unit design, a specific model selected by a specialist (Osyczka, 1994) is presented.

The equations of the corresponding model summarize the experience in designing such units and contain tables of coefficients obtained by empirical, mechanical

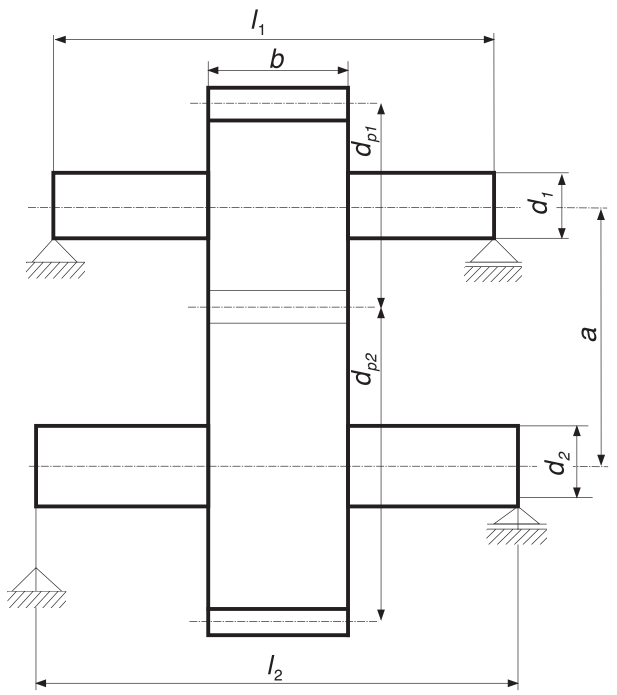


Figure 5.1. A diagram of a spur gear transmission unit with main dimensions indicated: b, the width of the toothed wheel rim; d_1, d_2, the diameters of the input and output shafts; d_{p1}, d_{p2}, the diameters of the toothed wheels.

studies. While such original data are very valuable, an analytic approximation of them might be more useful for model analysis. Thus, these tables were approximated by exponential functions. The problem might be then specified in a classic textbook format in equations such as (5.11, 5.12), where three objective functions $f_i(\boldsymbol{x})$ and 14 constraints $g_j(\boldsymbol{x})$, some nonlinear and some expressing simple bounds are defined. Here it is presented in a form similar to the textbook format (although the model was rewritten in the DIDAS-N++ format, because this system was used for further analysis).

Model Parameters

The parameters and basic data for the model are:

$$N = 12.0 \quad \tilde{n} = 280.0; \quad i = 0.317; \quad \Delta i = 0.01; \quad \tilde{z}_1 = 20,$$

where N is the input power (kW); $\tilde{n}$ is the rotational input speed (rev/min); i is the velocity ratio; Δi is the allowable deviation of velocity ratio; and $\tilde{z}_1$ is the number of teeth of the pinion.

The geometric data are:

$$b\tilde{m}_{min} = 5.0; \quad b\tilde{m}_{max} = 10.0; \quad a_{max} = 293.8,$$

where $b\tilde{m}_{min}$ is the minimum $b/\tilde{m}$ coefficient ($\tilde{m} = d_{pi}/z_i$, $i = 1, 2$, is the pitch of the gear teeth, while d_{pi} are the standard diameters of the gear wheels and b is the teeth width); $b\tilde{m}_{max}$ is the maximum $b/\tilde{m}$ coefficient; and a_{max} is the maximum distance between the axes (mm).

The material data are:

$$k_{g1} = 105; \quad k_{g2} = 105; \quad k_{o1} = 62; \quad k_{o2} = 62; \quad k_s = 70,$$

where k_{g1} is the allowable bending stress for the pinion (MPa); k_{g2} is the allowable bending stress for the gear (MPa); k_{o1} is the allowable surface pressure for the pinion (MPa); k_{o2} is the allowable surface pressure for the gear (MPa); and k_s is the allowable torsional stress of the shaft (MPa).

Other data are:

$$K_b = 1.12; \quad K_{z1} = 1.87; \quad K_{z2} = 1.3; \quad K_p = 1.25,$$

where K_b is the coefficient of the concentrated load; K_{z1} is the coefficient of the equivalent load for the pinion; K_{z2} is the coefficient of the equivalent load for the gear; and K_p is an overload factor.

Calculated data are:

$$T = 8000; \quad y_c = 3.11,$$

where T is the time of efficient work of the gear, and y_c is a coefficient for the assumed pressure angle.

Decision Variables

The decision variables are: the width of the toothed wheel rim b (which is also an objective); the diameters d_1 and d_2 of the input and output shafts; the number of teeth of the pinion wheel $\tilde{z}_1$; and the pitch of gear teeth $\tilde{m}$ (the last two decision variables are actually discrete).

Model Objectives

The objectives are: the volume of the gear unit $q_1 = f_1$ (mm^3); the distance between the axes $q_2 = f_2$ (mm); the width of the toothed wheel rim $q_3 = f_3$ (mm):

$$\begin{aligned} q_1 &= ((\frac{\pi}{4}\tilde{m}^2(\tilde{z}_1^2+\tilde{z}_2^2)b)+\frac{\pi}{2}d_1^3+\frac{\pi}{2}d_2^3)*10^{-5} \\ q_2 &= \frac{(\tilde{z}_1+\tilde{z}_2)}{2}\tilde{m} \\ q_3 &= b. \end{aligned} \tag{5.22}$$

The constraints on the decisions concern various geometric relations and mechanical stresses, such as:

- g_1 expresses the bending stress of the pinion:

  $$g_1 = k_{g1} - P_{og} * w_1/(b*\tilde{m}) \tag{5.23}$$

  where:

  $V = \pi * \tilde{m} * \tilde{z}_1 * \tilde{n}/60000; \quad K_d = (14.5+V)/14.5;$

  $P_{max} = 102 * N * 9.81/V; \quad P_{og} = P_{max} * K_p * K_b * K_d;$

  $w_1 = 4.7607 * exp(-0.104531 * (\tilde{z}_1 + 1.28627)) + 1.67421.$

- g_2 expresses the bending stress of the gear:

  $$g_2 = k_{g2} - P_{og} * w_2/(b*\tilde{m}) \tag{5.24}$$

  where:

  $w_2 = 4.7607 * exp(-0.104531 * (\tilde{z}_2 + 1.28627)) + 1.67421.$

- g_3 expresses the surface pressure of the smaller wheel:

  $$g_3 = k_{o1} - P_{o1}/(b*\tilde{m}*\tilde{z}_1)*(1+\tilde{z}_1/\tilde{z}_2)*y_1 \tag{5.25}$$

  where:

  $P_{o1} = P_{max} * K_p * K_b * K_d * K_{\tilde{z}_1};$

  $y_1 = 28.4869 * exp(-0.290085 * (\tilde{z}_1 - 1.78811)) + 3.31178.$

- g_4 expresses the surface pressure of the greater wheel:

  $$g_4 = k_{o2} - P_{o2}/(b*\tilde{m}*\tilde{z}_2)*(1+\tilde{z}_2/\tilde{z}_1)*y_c \tag{5.26}$$

  where:

  $P_{o2} = P_{max} * K_p * K_b * K_d * K_{z2}.$

- g_5, g_6 express the torsional stresses of input and output shafts:

  $$g_5 = k_s - M_{s1}/W_{01}; \quad g_6 = k_s - M_{s2}/W_{02} \tag{5.27}$$

  where:

$$\begin{array}{rclrcl} M_{s1} & = & 9549296 * N/n; & W_{01} & = & (\pi * d_1^3)/16; \\ M_{s2} & = & M_{s1}/(z_1/z_2); & W_{02} & = & (\pi * d_2^3)/16. \end{array}$$

- g_7, g_8, g_9 express the deviations of the velocity ratio and the relation between $\tilde{m}$ and d_1:

$$\begin{aligned} g_7 &= i - \tilde{z}_1/\tilde{z}_2 + \Delta i;\ g_8 = \tilde{z}_1/\tilde{z}_2 - i + \Delta i; \\ &\quad g_9 = \tilde{m} * (\tilde{z}_1 - 2.4) - d_1. \end{aligned} \tag{5.28}$$

- Other constraints include:

$$\begin{aligned} g_{10} &= \tilde{m} * (\tilde{z}_2 - 2.4) - d_2; g_{11} = b/\tilde{m} - b\tilde{m}_{min}; \\ &\quad g_{12} = b\tilde{m}_{max} - b/\tilde{m} \\ g_{13} &= a_{max} - (\tilde{z}_1 + \tilde{z}_2)/2 * \tilde{m}; \quad g_{14} = \tilde{z}_2 - \tilde{z}_1/i. \end{aligned} \tag{5.29}$$

In the model defined above, the exponential approximations of empirical data tables are expressed by the functions w_1, w_2, y_1. All these equations have been presented purposely: to stress that a computerized mathematical model can be complicated. The model presented above is small – because it is static, not dynamic – compared to other models used in applications. However, the model represents advanced knowledge in mechanical engineering and the selection of its various details relies on expert intuition: good modeling is an art. Even for such a small model, the reader should imagine programming the model, supplying it with all necessary derivatives, and selecting by hand values of decision variables that would satisfy the required constraints. Imagine that all this should be done without specialized software supporting the model analysis.

When using specialized software, the modeler should use first a model generator, then a model compiler; a good model compiler will automatically determine all needed derivatives. Even when a quickly executable, compiled core model is available, the modeler might have trouble with simple model simulation. The form of the model is complicated (not convex) and without good experience in mechanical design it is difficult to select values of decision variables that are acceptable.

Figure 5.2 shows the results of an inverse simulation of the model with two model outcomes – objectives q_1 and q_3 denoted by f_1 and f_3, respectively – and two decision variables denoted by d_1 and d_2, all stabilized.[9] However, as the aspiration and reservation levels were arbitrarily selected, even the inverse simulation cannot give satisfactory results. The optimization of a corresponding achievement function indicates that arbitrary reference levels cannot be realized in this model. The contours indicated in *Figure 5.2* represent the values of membership functions

[9]In *Figures 5.2*, *5.3*, and *5.4*, we use actual interaction screens of an earlier version of ISAAP in DIDAS-N++.

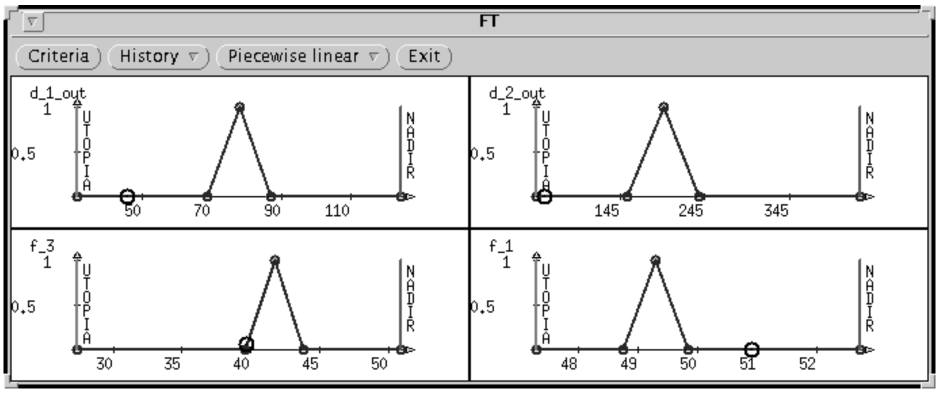


Figure 5.2. Interaction screen of DIDAS-N++ in the inverse simulation case, arbitrary aspiration levels.

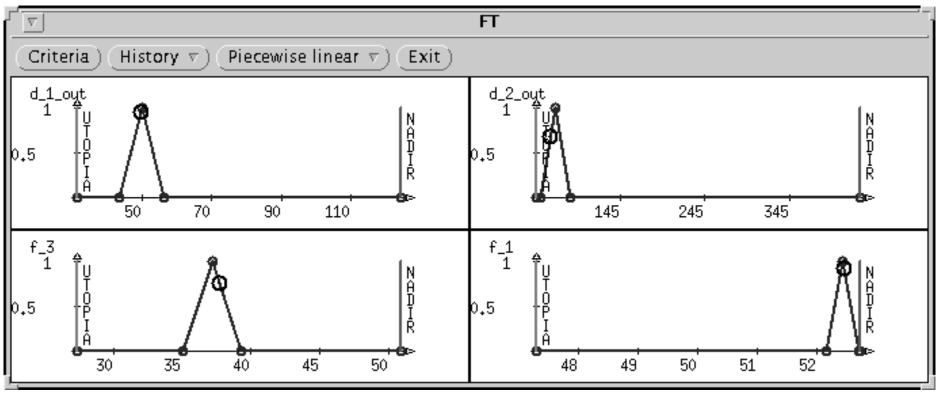


Figure 5.3. Interaction screen of DIDAS-N++ in the inverse simulation case, aspiration levels based on mechanical experience.

$\mu_i(q_i, \bar{q}_i, \bar{\bar{q}}_i)$ and the circles on these contours indicate the attained levels of objectives. Values 0 of these membership functions at circled points indicate that the requirements of the modeler cannot be satisfied.

In order to find results that are admissible for the model, other aspiration levels must be selected using the experience of a designer. In *Figure 5.3* the aspirations were set according to data given by Osyczka (1994). Since the model was changed by using the exponential approximation of data tables, the results of the inverse simulation with membership values close to 1 indicate a positive validity test of the model. However, the inverse simulation results are not efficient in the sense of minimization of objectives (the results given by Osyczka might be efficient for his model, but our model was changed by including approximating functions).

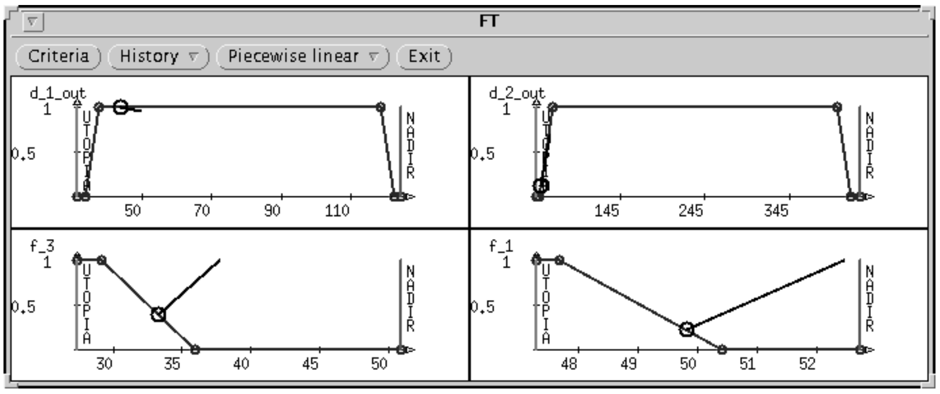


Figure 5.4. Interaction screen of DIDAS-N++ in the softly constrained simulation case, improvements of both objectives.

Improvement of both (or even all three) objectives considered can be obtained by switching to softly constrained simulation, as shown in *Figure 5.4*. In this first example, the soft constraints on decision variables were relaxed in such a way as to obtain efficient results for the problem of minimizing both selected objectives. In *Figure 5.4*, the improvement of objective values is shown by line segments leading to circles that indicate the attained values. A serious model analysis would clearly not stop at the results of such an experiment; many other experiments, including post-optimal parametric analysis, might be necessary. However, the above example is presented only as an illustration of some basic functions of a system of computerized tools for multi-objective model analysis and decision support.

5.5.2 A model of ship navigation support

A second application example shows the usefulness of including dynamic formats of models in multi-objective optimization systems. This example concerns ship navigation support (see Śmierzchalski and Lisowski, 1994): the aim is to control the course of a ship in such a way as to maximize the minimal distance from possible collision objects while minimizing the deviations from the initial course of the ship (see *Figure 5.5*).

This is a dynamic problem, with the equations of the model described initially by a set of differential equations for $t \in [0; T]$:

$$\begin{aligned}
\dot{w}_1(t) &= v_1 \sin x(t) \\
\dot{w}_2(t) &= v_1 \cos x(t) \\
\dot{w}_{1j}(t) &= v_j \sin \psi_j, \quad j = 2, \ldots \breve{n} \\
\dot{w}_{2j}(t) &= v_j \cos \psi_j, \quad j = 2, \ldots \breve{n},
\end{aligned} \tag{5.30}$$

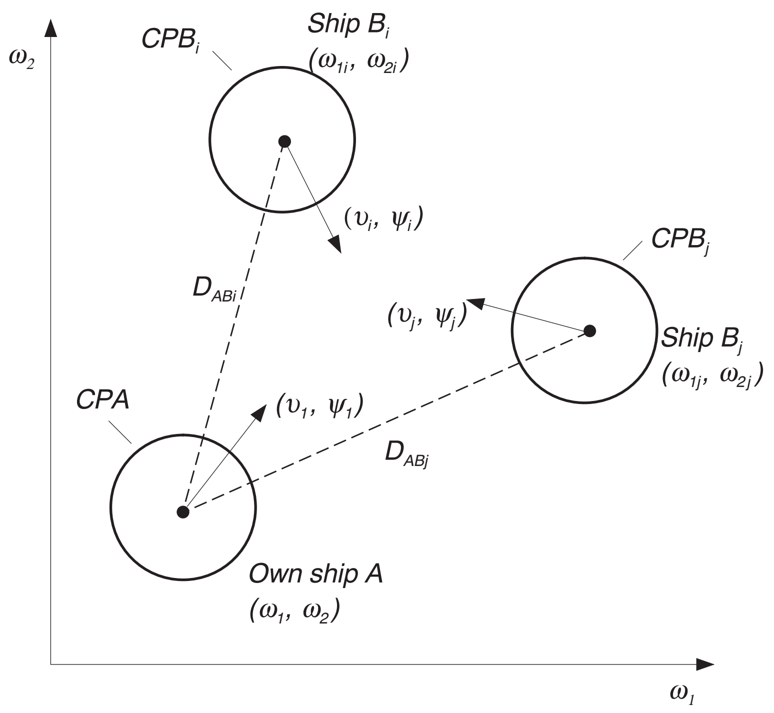


Figure 5.5. A diagram of a ship collision control situation (CPA: safe zone for ship A): own ship A has to evade other ships B_i, B_j while maintaining safe distances indicated by safe zones.

with initial values of ship positions given as the vector $\boldsymbol{w}(0)$. Between other model outcomes, the objectives can be modeled as:

$$\begin{aligned} q_1 &= \min_{t\in[0;T]} \min_{j=2,\dots\breve{n}} ((w_1(t) - w_{1j}(t))^2 + (w_2(t) - w_{2j}(t))^2) \\ q_2 &= \int_0^T (x(t) - \psi_1)^2 dt, \end{aligned} \tag{5.31}$$

where q_1 represents the (squared) minimal distance that should be maximized and q_2 represents the (squared) average deviation from the initial course, which should be minimized.

In order to be applied to a DIDAS-N system, this model was discretized in time, with the resulting model form similar to equation (5.16). This model is not analyzed in more detail here; this example was given only to show the practical sense of using dynamic models with multi-objective analysis and optimization. Experiments with this type of model support the conclusions about the usefulness of algebraic model differentiation and model compiling, of multi-objective modeling, and inverse or softly constrained simulation for the modeler.

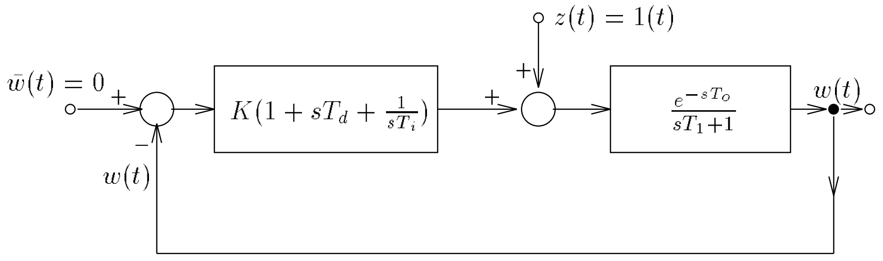


Figure 5.6. A block diagram of a control system for an inertial plant with delay and a PID controller, with indications of the transfer functions (the ratios of Laplace transforms of input and output signals) of system elements.

5.5.3 The design of a PID controller for a control system

The third example illustrates the usefulness of introducing specific formats for dynamic models with delays. It concerns a classic problem in automatic control: the adjustments of parameters of a PID (proportional–integral–differential) controller to obtain good properties of the control system. The model can be described in various ways; for a control engineer, a useful model description is a block diagram of the control system (see *Figure 5.6*), where the transfer functions of the assumed model of the control plant (inertial with delay) and of the PID controller are indicated.

The decision variables are the parameters K, T_i, and T_d of the controller; various model outputs can be defined as possible objectives, e.g., the overshoot κ and the control time ω:

$$\begin{aligned} \kappa &= \mid \min_{t\in[0;T]} w(t) \mid / w_{max}, \quad w_{max} = \max_{t\in[0;T]} w(t) \\ \omega &= \min\{t : \mid w(\vartheta) \mid \leq 0.05 w_{max} \ \ \forall \vartheta \geq t\}, \end{aligned} \tag{5.32}$$

or various integrals of control error over time:

$$I_1 = \textstyle\int_0^T \mid w(t) \mid dt; \quad I_2 = \int_0^T \mid w(t) \mid^2 dt; \quad I_4 = \int_0^T \mid w(t) \mid^4 dt. \tag{5.33}$$

In order to show the model in a form similar to equation (5.20), it must be converted first from the form of transfer functions (ratios of Laplace transforms) to the form of a difference–differential equation, then discretized over time. This transformation is not as simple as for the ship navigation model, thus the discretization details are briefly presented here. A discrete time variable is introduced:

$$\tau = \tfrac{t}{\Delta t}; \quad \tau_0 = \tfrac{T_0}{\Delta t}; \quad \tau \in \{-\tau_0, -\tau_0 + 1, \ldots, 0, 1, \ldots \Theta = \tfrac{T}{\Delta t}\}, \tag{5.34}$$

while rather small values of Δt and large values of Θ must be used in order to adequately model the problem; therefore, the resulting models are rather large. During the discretization, adequate care must be taken to realistically model the differential part of the controller. A resulting set of equations might be as follows:

$$\begin{aligned}
w_{\tau+1} &= e^{-\frac{\Delta t}{T_1}} w_\tau + (1 - e^{-\frac{\Delta t}{T_1}}) v_{\tau-\tau_0};\ w_\tau = 0\ \forall\tau = -\tau_0, \ldots 0 \\
v_\tau &= u_\tau + z_\tau;\ \ z_\tau = 0\ \ \forall\tau = -\tau_0, \ldots –1,\ \ z_\tau = 1\ \ \forall\tau = 0, \ldots \Theta \\
u_\tau &= u_{1,\tau} + u_{2,\tau};\ \ u_{1,\tau} = u_{2,\tau} = 0,\ \ \forall\tau = -\tau_0, \ldots –1 \\
u_{1,\tau} &= u_{1,\tau-1} - K\frac{\Delta t}{T_i} w_\tau - K(1 - \frac{\Delta t}{T_i})(w_\tau - w_{\tau-1}) \\
u_{2,\tau} &= e^{-F\frac{\Delta t}{T_d}} u_{2,\tau-1} - K\frac{T_d}{\Delta t}(1 - e^{-F\frac{\Delta t}{T_d}})(w_\tau - w_{\tau-1}), \qquad (5.35)
\end{aligned}$$

where $u_{1,\tau}$ represents the output signal of the PI part of the controller and $u_{2,\tau}$ is the output signal of the D part, while F is a filtering coefficient. The outputs of the model must include possible objective functions:

$$\begin{aligned}
\kappa &= |\min_{0\le\tau\le\Theta} w_\tau| / w_{max}, \quad w_{max} = \max_{0\le\tau\le\Theta} w_\tau \\
\omega &= \Delta t \min_{0\le\tau\le\Theta} \{\tau :\ |w_\vartheta| \le 0.05 w_{max}\ \ \forall\vartheta \ge t\} \\
I_1 &= \sum_{\tau=0}^{\Theta} |w(t)|\,\Delta t; I_2 = \sum_{\tau=0}^{\Theta} |w(t)|^2\,\Delta t; I_4 = \sum_{\tau=0}^{\Theta} |w(t)|^4\,\Delta t. \qquad (5.36)
\end{aligned}$$

The output ω might be actually a discontinuous function of decision variables K, T_i, T_d; therefore, we should use it only as an additional model outcome, not as an objective. The outputs κ, I_1 are nondifferentiable functions of the decision variables, but their nondifferentiability does not influence optimization. The model requires major computational effort to be algebraically preprocessed and compiled; however, the computational effort required for optimization runs with such a compiled model is moderate. The entire model, although complicated, is directly computable: it can be simply simulated (i.e., run for given values of K, T_i, T_d without violating any constraints), inverse or softly constrained simulated, or optimized multi-objectively for variously chosen objectives. Various conclusions can be drawn from such experiments; for example, if we start optimization from such parameter values that the behavior of the control system is aperiodic (nonoscillatory), and try to optimize κ and other objectives, then the optimization algorithm is stuck. This is because for such parameter values, the value of κ is identically 0. Such experiments reinforce the usefulness of multi-objective modeling tools in model development and analysis.

5.6 Supporting a Final Choice

The user of a multi-objective model analysis and DSS should now be in a position of understanding inverse and softly constrained simulation and multi-objective optimization, and how these can help in model analysis. However, some confusion may still exist as to what scenario or decision should be finally chosen. This choice is hypothetical to some extent, and usually concerns decision analysis specialists rather than modelers. However, after investigating the models in various ways with the tools outlined above, the modeler might require some help in the final selection of decisions and model outcomes (scenarios, designs, etc.). This selection must represent the modeler's choice of results of modeling, which might depend on the purpose of the entire model analysis. The modeler might choose differently when, for example, asked to present three possible scenarios of future developments as an expert, or when preparing a learned paper validating the model.

For these reasons, help in selecting scenarios, decisions, and model outcomes must be specific. The problem of choice has an extensive literature; theoretically, we could apply various results from the broad field of decision theory and analysis (see e.g., Rios, 1994). In practical terms, however, we are greatly limited. We cannot assume much about the preferences of the modeler. We know that they might change during the decision process, hence such help must occur after extensive model analysis. The modeler often intuitively evaluates objectives and their use on a ratio scale, thus we cannot apply arguments only valid for ordinal scale. Therefore, if the modeler has formed a value function, this function is nonlinear and nonseparable.[10] The modeler, if faced with uncertainty about the model, would prefer to perform additional experiments with it rather than answer numerous questions necessary for identifying her/his expected utility function.

Nevertheless, a DSS module representing a tool for supporting choice might be needed. It is possible to reduce the problem of selecting scenarios, designs, or model outcomes to an application of some known techniques of multiple attribute choice within a discrete set of options. The aspiration-based or reference point methodology might be a good tool for generating a given number of decision scenarios or options, together with corresponding outcomes. The modeler should first select such a set. If the modeler requires support in generating the set of options, various simple techniques can be suggested, such as the following:

- To start with the options generated when computing utopia components.
- To add the neutral solution with the reference point located in the middle of nadir–utopia ranges.

[10]If we draw conclusions about ordinal value functions, the assumptions needed for their separability and linearity are weak; however, this does not apply to value functions measured on more specific scales.

- To then add several solutions with reference points in the middle of the interval between the neutral solution and the solutions corresponding to various utopia components.
- To then add several solutions with reference points in the middle of intervals between the solutions corresponding to various reference components; etc.

Note that the reference point methodology offers the possibility of looking for a compromise between any two options. This can be done by taking the reference point in the middle of the interval joining the (outcomes of the) two options and letting the system find a corresponding efficient outcome and decision (either the closest or the uniformly most improved, as compared to the reference point) by optimizing the appropriate achievement function.

If the set of options is defined, the modeler can choose a method of multi-attribute decision selection, e.g., the ELECTRE, the PROMETHEE, and the analytical hierarchy process (AHP); see Saaty, 1980; Roy and Vincke, 1981; and Brans and Vincke, 1985). For other known techniques, see Vincke (1992) for a survey and Chapter 9 of this book.

Part II

Decision Support Tools

Chapter 6

Modeling Tools

Jerzy Paczyński, Marek Makowski, and Andrzej P. Wierzbicki

This chapter gives a general overview of some modeling tools that support the specification and verification of a mathematical model, as well as its analysis, maintenance, and documentation. While computerized modeling is an art, it is an art based on knowledge – both in the disciplinary field relevant to the phenomena represented by a model and in the interdisciplinary field of computational techniques of modeling. This chapter attempts to give a basic overview of the second interdisciplinary field.

The chapter starts with a brief section on the art of modeling and proceeds to describe the taxonomy of analytical models – linear versus nonlinear; dense, sparse, and other structures; continuous versus discrete domains of variables; static versus dynamic models; lumped versus distributed parameters; and deterministic versus nondeterministic models. The next section comments on basic assumptions of modeling languages (i.e., those concerning the relation of languages and cognitive processes), as well as those related to imperative, object-oriented, and other programming paradigms. With this general background, so-called algebraic modeling languages and related conventions are described in more detail. A simple example of a nonlinear dynamic model illustrates the use (and possible traps) of various implementations of selected modeling languages. The following section describes some issues of model analysis and transformation, in particular, issues of automatic (computational) model differentiation, once again illustrated by the simple example considered before. The final section summarizes guidelines for specifying a model aimed at supporting decision making.

6.1 The Art of Modeling

Each model represents a partial truth about some selected part or phenomena of the surrounding world. It must be confined to a well-defined area of interest, it can only be valid for a specific purpose, and real phenomena will always be only partially represented by the model. A modeler must find a way of avoiding too much detail while preserving the essential features of the specific situation; in doing so, she/he is guided by intuition and taste. Therefore, modeling remains and will remain an art.

The choice of mathematical and software tools and methods used for the implementation of computerized mathematical models depends strongly on the relevant domain of science or engineering – on its peculiarities, traditions, software market, and sometimes legal regulations. Thus, any attempt to formulate a unified and all-embracing modeling methodology and tools will remain difficult. Nevertheless, some unification and standardization work is urgently needed; however, this must be based on a clear definition of the term modeling. Modeling is an area of intensive growth in many different areas. The wealth of related results and invested resources will always work against any potential unification, although standardization might help to save invested resources and to obtain the results more efficiently. One form of standardization called structured modeling is presented in Section 6.4.7. Another contemporary form of standardization proposed by the Federation of European Simulation Societies (EUROSIM) is the design of a Modelica language.[1] The goal is to specify a EUROSIM standard concerning a "unified object-oriented language for physical systems modeling". This language was not available to the authors for testing at the time of writing but an example in ASCEND, one of its predecessors, is included in Section 6.4.4. However, the scope of potential application of this language, as defined by its designers, is still limited; one might expect, for example, that this language will not necessarily be well suited for economic applications of the type frequently considered by operations research societies.

One possibility of understanding the term modeling is to define it in input–output terms: given some input values, we can provide a recipe on how to produce output values of the model. The process of producing outputs from given inputs is called the solution process, which uses a solution mechanism. This process might be explicit (e.g., defined by a subsequent evaluation of formulae) or implicit (e.g., defined by using some resolving iterations related to a form of an implicit function theorem). Such an approach to modeling has frequently been used in many software tools and modeling languages; hereafter we refer to this as a simulation-oriented modeling approach.

[1]http://www.Dynasim.se/Modelica.

Another approach to modeling assumes that the definition of inputs and outputs is accompanied by a description of a function defined on outputs (and perhaps also on inputs) that should be optimized; this involves finding the extreme values of the function and the corresponding input (and output) values. This is an optimization-oriented modeling approach; the corresponding tools are called software or languages for mathematical programming (Kuip, 1993).

However, there is a danger in concentrating too much on one of the above approaches. Even if we only want to use model simulation, optimization is needed for estimating model parameters; moreover, as mentioned in previous chapters, optimization might be a powerful tool for various types of model analysis.

Languages of mathematical programming often focus too much on a textbook format of optimization problems, that is: they use the concept of constraints rather than that of model outputs; they provide for a definition of a single optimized function only; and limit the freedom of model definition. As stressed in Wierzbicki (1992b), a more progressive approach to modeling should be based on a combination of simulation and optimization; in Chapter 5 this was referred to as multi-objective optimization and softly constrained simulation modeling (i.e., multi-objective modeling). This approach overcomes the dualism between simulation and optimization models, which is often stressed as essential in textbooks. Optimization algorithms are then treated as instruments for more flexible and diversified simulations, not as goals of modeling. However, a word of warning: the multi-objective modeling approach is more a postulate resulting from the experience of the authors rather than a widely used approach; only a small number of software tools (including some tools presented in the Appendix) exists to support such an approach.

Another source of possible difficulty often not stressed enough is related to the solution mechanism. The spectrum of available implementations of solution mechanisms spans from processors of universal programming languages (e.g., Fortran, C, C++, Pascal), through to processors of general purpose computer algebra systems (e.g., Maple, Mathematica) and general purpose computational environments (e.g., MATLAB), to processors of specialized modeling languages (e.g., GAMS, AMPL, SML, etc.). The pitfalls might result, for example, from differences in assumptions about the properties of a computing process in a relevant processor. The most important aspect of such assumptions is the treatment of values not yet calculated; there are essentially two different ways of such treatment. The values not yet calculated might be treated as unavailable (thus, using them is an error) or as values with some default initial assignment. Authors of various approaches, accustomed to using one of the above assumptions, consider this a self-evident detail and either do not include this assumption in a typical documentation or scatter it in footnotes without stressing its importance. Such neglect can have devastating effects, especially when the distinction (in the definition of a language or in the user's mind) between assignment statements and equality relations is not clear. For example, in balance equations in an economic model, default values of zero stocks are safe. However, in another model describing a phenomenon evolving in time, such default zero values may hide the values, which will remain uncalculated due to errors in the description of the evolution in time. There are also other pitfalls related to possible solution mechanisms, but they are more specific.

Therefore, modeling is an art (Wierzbicki, 1984) that requires intuition and taste, as well as knowledge, in two major areas: in the disciplinary field relevant to the phenomena represented in the model; and in the interdisciplinary field of methods and tools required for specification, verification, and analysis of the model.

In this chapter, we can give the reader only a very short summary of some general aspects of the second field.

6.2 The Taxonomy of Analytical Models

In this book we focus on analytical rather than logical models. Analytical models can be classified in various ways, with most of these classifications overlapping; however, we comment only on selected aspects of these classifications.

6.2.1 Linear versus nonlinear models

Detailed models describing any aspect of our universe are highly nonlinear. However, many phenomena can be successfully approximated with the linearized versions of originally nonlinear models. There is an additional temptation for using linear models because a linear model is nearly always solvable, even for large dimensions of the model; this does not apply to nonlinear models. Thus, in many areas modeling activity started from linear models and was often extended to include nonlinear ones (Kuip, 1993). Nonlinear models have a much richer structure and are harder to classify. Of special interest is the merging of both linear and nonlinear parts in one model. The effectiveness of such an approach depends on the ability of other parts of the modeling environment – especially solution tools called solvers – to use the merits of each part and to arrange a smooth cooperation between them.

6.2.2 Dense, sparse, and other data structures

This classification relates to large-scale models, i.e., models that have a large number of variables and constraints. Most textbook theories are formulated for dense representations of model elements in terms of vectors and matrices. However, the

Jacobian of most large-scale models is fortunately very sparse, i.e., having most constraints dependent only upon a small fraction of the model variables. Owing to this property, models of considerable size may be solved (linear models or linear parts of models used in practice typically have many more variables – higher dimensionality – than nonlinear ones, and even large-scale nonlinear models usually have large linear parts). The time and other resources invested in a correct definition and verification of elements of a large model would be prohibitive for nonsparse models. However, if a model is sparse, a naive application of a textbook approach to its analysis often leads to a very inefficient algorithmic implementation. Almost all elements of a modeling process – from the problem description through solution methods to report generation – must take into account the sparsity of the model.

Another important feature of a model relates to the possibility of representing its structure by a graph. Theoretically, all models can be represented as graphs, but not in a unique way; a good graph representation helps to distinguish the structural features of the model. Although models in the form of a graph can usually be formulated in the analytical form of a linear or nonlinear model, the graph form provides many advantages. These advantages range from the clarity of problem formulation to effective specialized solution algorithms. Other structures help to model specific phenomena (e.g., hierarchy of water flows in a river basin by a tree model; traffic in real transportation or telecommunication networks) and justify the development and application of special solution algorithms. Various applications of specialized structures to modeling are presented by Bisschop and Fourer (1996) and by Bisschop and Kuip (1993).

6.2.3 Continuous versus discrete domains of variables

Variables and parameters in modeling systems are often defined – leaving aside problems resulting from finite accuracy of computer number representation and arithmetics – on continuous, compact subsets of real axes or their Cartesian products. However, there are many practical applications where the corresponding domains of some variables or parameters are additionally constrained to some discrete sets of values (e.g., the available standard sizes of wheels in a mechanical gear or standard diameters of wires in an electric installation). For simple models, such a restriction might simplify model analysis. For complicated models using the optimization-oriented modeling approach, discrete domains of variables make model analysis more difficult. Thus, the ability of handling such situations for large-scale models is limited to a class of linear optimization models – the so-called mixed integer programming case. These will be discussed in more detail in the following chapter.

6.2.4 Static versus dynamic models

Although time might be theoretically treated only as one more model variable or index, the proper setting of a dynamic model involves more fundamental properties, such as causality or a semi-group property (Wierzbicki, 1984). If we define a separate model instance for each moment of time (with the latter treated as an element of a discrete domain), we can naively transform every dynamic model to a static one – at the expense of a dramatic increase of dimensionality of the model. The advantage of a naive representation is that a dynamic model can be treated like this in any modeling system. Most optimization-oriented algebraic modeling languages treat dynamic models this way.

However, there are modeling systems (e.g, MATLAB; see Part-Enander *et al.*, 1996, and ACSL)[2] that are oriented toward the simulation of models of dynamic systems and support specific features of dynamic models (e.g., structural representations by block diagrams; supporting their analysis not only in time-domain but also in frequency-domain). An important issue, when modeling dynamic systems, is the choice between discrete and continuous time.

The simplest type of dynamic model involves an explicit discretization of time resulting in a form equivalent to difference equations. This is also the only viable way for handling such models in a general-purpose algebraic modeling system. A purely continuous treatment of time can be achieved either analytically or by using special methods and tools for symbolic integration of differential equations. An intermediate way consists in applying numeric integration tools. There are many software libraries for solving differential equations, available either as freeware through the Internet (e.g., CVODE)[3] or commercially. Some simulation-oriented modeling systems (e.g., MATLAB) have integration software embedded in them. However, the tendency for making the system trouble-free for an inexperienced user often results in the implementation of a relatively primitive integration method with fixed parameters. Thus, there is a clear-cut dichotomy between easy-to-use systems and more complicated ones. The easy-to-use system is often sophisticated and has a broad spectrum of services but offers little or no control over the choice of an integration method. The complicated system differs in that it enables the user to have full control; however, the user must have sufficient mathematical knowledge, access to the source code of the system, and relevant documentation.

Most software libraries for integrating differential equations are limited to the calculation of trajectories of dynamic models and seldom address other issues of model analysis, although there are specialized tools, say, for discrete event

[2]ACSL Simulation Software, http://www.mga.com.

[3]S.D. Cohen and A.C. Hindmarsh, CVODE User Guide. http://netlb.org/ode/cvode.tar.Z.

simulation. Tools for optimization of dynamic models are very scarce.[4] Although the theoretical knowledge of sensitivity analysis of dynamic models was advanced many years ago, no contemporary software tools are available for this task.

6.2.5 Lumped versus distributed parameters

Until now we have mostly discussed issues related to the lumped parameter dynamic systems that are described by ordinary difference or differential equations. A distributed parameter model is described by partial differential equations, or by difference-differential equations for dynamic models with delays. The exact solutions for partial differential equations can only be obtained by analytical methods or computer algebra methodology (symbolic integration). However, most practical modeling tasks involve models that cannot be solved exactly; thus, approximation methods for solving partial differential equations are usually applied. The most popular approximation methods for partial differential equations are the finite element method (used in DIFFPACK[5] by Bruaset and Langtangen, 1997) and the boundary element method (used in GP-BEST).[6] While these methods are good for solving partial differential equations (i.e., model simulation), other model analysis tasks, such as optimization and sensitivity analyses, are usually not supported sufficiently by the available software. For systems described by partial differential equations, there are various approximative integration methods and corresponding software.

The situation is not as good for dynamic systems with continuous time and with delay, described by difference-differential equations; there are very few standard tools for this case and a good knowledge of advanced mathematical methods is necessary for modeling such systems. However, this difficulty is considerably reduced if the requirement for modeling continuous time is dropped and it is substituted by a (appropriately dense) discretization of time. In a discrete domain of time, systems with delays can be analyzed by standard modeling languages, because a delay means an increase of the order of the difference equation. This leads to longer computations, particularly if optimization or sensitivity analysis issues are investigated. There are special methods that more directly exploit the structure of systems with delay in such cases (Wierzbicki, 1992b); however, the broad spectrum of special approaches to modeling dynamic systems will not be discussed in detail in this chapter.

[4]There are some recent publications on tools for dynamic optimization; however, the authors do not know of any universally applicable tool available commercially or as freeware.

[5]DIFFPACK: generic software for the solution of differential equations, available from http://www.oslo.sintef.no/avd/33/3340/diffpack.

[6]Boundary Element System GP-BEST, Boundary Element Software Technology Corporation, P.O. Box 310, Getzville, New York 14068, USA.

6.2.6 Deterministic models versus models with uncertainty

The majority of modeling tools deals with deterministic models, that is, without representing uncertainty. The uncertainty of the future is sometimes accounted for in the form of scenarios containing various forecasts of otherwise certain parameters. This is, however, only the simplest form of representing uncertainty.

Another form is stochastic modeling, in which a model represents the evolution of probabilistic events over time. As in the deterministic case, there are two distinct approaches to stochastic modeling. The simulation-oriented approach is related to modeling stochastic/differential equations; the modeling system GNANS[7] takes these into account. There are also simulation languages for discrete event systems of stochastic character. The optimization-oriented approach is related to stochastic programming or stochastic optimization, where the goal is to find a solution that is good (in a precisely specified sense of optimizing an expected value) for all scenarios. There is no single modeling system supporting stochastic optimization, but this approach can be used in many programming systems (see the next chapter).

The methods discussed above show only the tip of the iceberg. There are many other theories, methods, and paradigms to model "uncertainty", which, however, are heavily context-dependent (Zimmermann, 1997). The development of universal modeling tools incorporating these notions seems doubtful; even with a limited scope of application they present a challenge for the future.

6.3 Modeling Languages

6.3.1 Languages and cognitive processes

In practical modeling, we often proceed by stages and steps. Two practical steps can be distinguished:

- In the first step, no computer or programming language is used, we just formulate the model using mathematical terminology.
- In the second step, practically no mathematics is used: software tools must be used to represent the model.

These seemingly obvious steps, however, might be overinterpreted with far-reaching consequences if we are not cautious about the meaning of a language and related cognitive processes. Such overinterpretations are related to two historically

[7]B. Martensson, GNANS, A Program for Stochastic and Deterministic Dynamical Systems, http://www.mathematik.uni-bremen.de/~bengt/gnans.html.

sequential approaches to modeling languages, which can also be considered as extremes. In practice, there is a wide spectrum of approaches between these two extremes that are characterized by the following basic assumptions:

- Only a universal programming language has the flexibility to express the diversity of computer models. Although this approach is historically very old, it also occurs recently: an example of this way of thinking is given by Averick *et al*. (1992), who use Fortran to describe a broad spectrum of mainly physical modeling situations, belonging to various classifications discussed in this chapter.
- A modeler should not need to use any programming skills, and the formulation of a model should be "self-evident". If a language is designed for a particular domain of application, any notation sufficiently popular between modelers with the corresponding educational or scientific background will do. However, if a language aims at encompassing multiple areas then a notation closely resembling the mathematical notation commonly found in articles and textbooks is the only feasible base (Kuip, 1993).

The second approach is based on two beliefs, which are, unfortunately, both unconfirmed in practice, and are often not expressed explicitly but are still clearly recognizable:

- No programming skills should be needed for (mathematical) modeling, i.e., any (mathematically) equivalent formulation of a model should lead to a solution (hopefully the same solution and with similar computing costs).
- A modeling language should use a notation that modelers find easy to use for describing problems to each other; thus, the semantics of a modeling language should be natural for modelers yet precise enough to define the corresponding computational problems to a computer system.

When evaluating the importance of such assumptions, however, we must be aware that the choice of modeling language is important because of basic requirements such as the ease of learning and use. A well-recognized phenomenon in linguistics is that the language, in which a thought is expressed, decisively colors and influences the nature of an idea and the understanding of a concept. While this was initially observed in the context of natural languages, it also applies to artificial computer languages, including modeling languages. A modeling language actually creates a way of thinking about the modeling discipline.

However, one should also be aware that no universal, unambiguous mathematical notation exists. Recall that even popular semantic paradigms may create confusion in cases such as the known problem with the priority of the unary minus versus other operators (e.g., What does -2^2 mean?) or the problem with associativity of the power operator (e.g., How to interpret 2^{3^4}?).

The term "programming paradigm" is often used to describe how to perform computations and how they should be organized. The reader should be warned, however, that programming paradigms are often restrictive (similar to puristic ways of proving mathematical theorems, e.g., purely algebraic or purely geometric), and multi-paradigm programming is usually the best practical choice.

Both universal programming languages and specialized modeling languages support either one or more of the following programming paradigms:

- Imperative programming.
- Object-oriented programming.
- Functional programming.

This classification is not exhaustive; other programming paradigms can also be distinguished, such as logic programming, constraints programming, and algebraic programming.

The impact of some of these paradigms on the practice of modeling will be discussed next.

6.3.2 Imperative programming paradigm in modeling

This paradigm has the longest tradition in universal programming languages. It is still the most popular in some disciplines, partly because of the large availability of software resources consistent with this paradigm.

Let us consider a modeling task, related either to optimization or to the simulation of an ordinary differential equation (ODE). A typical program in a high level language then consists of three parts:

- A model subroutine, which evaluates the model outputs (and/or objective functions) in terms of the inputs.
- A collection of analytic subroutines – a library of solvers – for various tasks of model analysis (e.g., optimization, solving ODE).
- A driver program that provides an interface between the parts of these software.

Typically, a modeler has to prepare the first and the third part, while the second part is prepared by specialists. In terms of an interesting classification and analogies given by Kuip (1993), this paradigm corresponds to an algorithmic form of a problem.

This paradigm is often also present (maybe in a residual form) in modeling languages based on other, more modern paradigms, in order to satisfy the need of more traditional users (refer to ASCEND discussed later in this chapter).

6.3.3 Object-oriented programming paradigm in modeling

Although there are several object-oriented universal programming languages, only one counts for complex numerical computations – the C++ language. JAVA is becoming more popular; however, at the time of writing it is unlikely that JAVA will become a real competitor of C++ for complex numerical computations. Therefore, the terminology adopted by C++ will be used here.

Over the last decade C++ has been widely used as a replacement of traditional imperative languages, also in mathematical modeling. A recently adopted standard for C++ (Stroustrup, 1997) includes the Standard Template Library (Muser and Saini, 1995), which allows for using an algebraic-like syntax. Combined with a substantially extended Standard C++ Library (Plauger, 1995), this also greatly simplifies programming work and makes the resulting software much more robust and efficient (in comparison with traditional programming languages). However, the greatest advantage of object-oriented languages over traditional programming languages is their support for creating types of objects that closely correspond to the real-world objects being modeled. Thus methods and data specific to an object are encapsulated and both collections and hierarchies of objects can be easily developed. Types can be parametrized by using templates and therefore software is easier to develop, maintain, and reuse. Moreover, an algebraic-like syntax can be easily implemented in C++ (Nielsen, 1995).

A typical development process of a C++ program can be summarized as follows. A collection of classes corresponding to types (objects) that form a model is designed. Such a set of classes is typically built using both the standard C++ library and some domain-specific libraries. New classes that are required are built using language mechanisms such as inheritance, dynamic typing, and templates. In this way, a specialized programming language can be created for the particular domain. The final program is very simple because the complexity is hidden (distributed) in the definitions of classes. A high numerical efficiency, similar to that achieved in most effective imperative languages, is obtained with careful programming. Moreover, C++ applications can be easily linked with older, well tested numerical libraries developed, e.g., in Fortran. An example of a large, non-linear model implemented in such a way is presented in Chapter 13.

The advantages of the object-oriented methodology were also recognized in the area of modeling languages, e.g., ASCEND, Dymola, gPROMS, NMF, ObjectMath, Omola, SIDOPS+, Smile, U.L.M., VHDL-AMS, and an emerging new language, Modelica. Logic models and programming are not discussed further in this book, and algebraic modeling languages shall be the focus of the next section.

6.4 Algebraic Modeling Languages and Related Conventions

Algebraic modeling languages first appeared in the 1970s in close connection with the optimization-oriented modeling approach. The language GAMS by Brooke *et al*. (1992) has been widely used since then. Such languages have their origins in the operations research community and are tailored to its specific needs; they are now in a period of intensive growth. Here we shall discuss two popular commercial languages: AMPL (Fourer *et al*., 1993) and AIMMS (Bisschop and Entriken, 1993), as well as the object-oriented modeling system ASCEND. As their counterpart, we shall also discuss modeling conventions used in the DIDAS family of multi-objective optimization and model analysis tools (Lewandowski and Wierzbicki, 1989). On application to a testing example, we shall underline the need for caution when using such languages for purposes not necessarily predicted by their designers. Finally, we will present an outline of structured modeling and its implementations.

In practice, we do not usually distinguish between the name of a programming system and the name of its language. The three languages GAMS, AMPL, and AIMMS were designed consecutively in the above-mentioned order; recently the interface of AMPL was redesigned for the windowing environment and it is known as AMPL PLUS (Fourer *et al*., 1996). Each successor benefited from the experience gained using its predecessors and from the progress occurring in computer science and technology. Although successful products usually evolve, older ones remain nevertheless handicapped by typical demands of compatibility with their previous versions.

We should stress here that the traditional name of algebraic modeling languages (AML) is not quite justified semantically. Such languages focus on optimization-based model analysis; however, this is not motivated by semantic requirements, but because of practical reasons. The driving force behind the development of such modeling systems was the existence of a strong market for optimization-based model analysis. This market has well-grounded roots in early applications of operations research to well-structured problems for which single objective optimization was, and still is, the right tool. A general definition of an AML (in the most typical application of an AML) might follow the definition by Kuip (1993): “An algebraic language for mathematical programming is a language to state a mathematical programming problem in a form that closely corresponds to the algebraic notation commonly found in articles and textbooks”. Fourer *et al.* (1996) give a slightly different, possibly more precise definition: “An algebraic modeling language is a popular variety based on the use of traditional mathematical notation to describe objective and constraint functions”.

The authors of both definitions stress the following two features that characterize an algebraic modeling language:

- Its typical application (namely mathematical programming, with its emphasis on one objective and constraint, not on model inputs and outputs).
- The usage of algebraic notation.

The recommended way of modeling in such a system is that the user, using an easy algebraic notation, should define various constraints and one objective. However, this might not be the best way of proceeding for a specialized modeler, who would rather define various model output variables or outcomes depending on model input variables, and then choose which one of them (or more than one) should correspond to objectives and to constraints. Researchers in multi-objective optimization coming from a mathematical programming tradition would then stress that the constraints and objectives should be interchangeable. A sophisticated analyst from a substantive field of applications does not understand the value of this advice, because she/he has always thought in terms of model outcomes that might be either objectives or constraints. Such interchange can also be done with the help of an AML, but the user should be sophisticated enough not to follow the standards of the system too literally. For example, AMPL’s `var` statement allows for the definition of a variable that can serve as an output variable. GAMS and (soon) AMPL also provide for the definition of complementarity conditions as an alternative (or supplement) to objectives. However, such possibilities are rarely used at the moment.

Thus, typical applications of algebraic modeling languages are driven by a classical (and still most popular) way of single-objective optimization-based model analysis and not by a broader variety of approaches using optimization as a tool for diversified methods of model analysis. Because of market demand, the developers of an AML have not paid much attention to other approaches of a model analysis, such as multi-criteria model analysis, soft simulation, or inverse simulation discussed in the previous chapter. As a consequence of the demand-driven development of modeling systems based on an AML, they are difficult to be used for those other approaches. The ease of use of a single-objective function for a model analysis may lead modelers using an AML to focus too much on optimization, while other approaches to model specification and analysis could be more natural or effective for a comprehensive model analysis. Therefore, later in this chapter we give an example of an alternative approach to modeling conventions used in the DIDAS family.

We now examine more closely the group of AMLs. Their implementations allow for the definition of a scalar objective function (to be maximized or minimized) and of a model in the form of constraints. Although AMLs were developed originally for linear models, they are also capable to handle nonlinear and mixed integer models. Some AMLs have syntactic extensions for handling network flow problems. The success of such languages is due to the fact that many real world problems have been successfully analyzed by solving a corresponding mathematical programming problem. Over the years a number of methods and software have been developed for solving various classes of mathematical programming problems. Therefore general-purpose implementations of algorithms are available (see Moré and Wright, 1993, for an example of an optimization software guide). However, while only minimal (or no) direction from the user is needed during the solution process for linear programming problems (including medium-sized mixed integer problems), a definition of nonlinear programming problems (even of a small dimension) may require modeling knowledge and experience. An example of such a problem is presented below.

AMLs are discussed in more detail in several books (Brooke *et al.*, 1992; Bisschop and Entriken, 1993; Fourer *et al.*, 1996), therefore, we adopt a specific approach to their examination. We formulate an illustrative example – apparently (not actually) simple and representative for various modeling tasks (e.g., in the fields of demography, econometrics, or automatic control) – and explain the use of algebraic modeling languages for this example. Then we try to actually solve the problem indicated by this example and see what difficulties it involves.

6.4.1 An illustrative example

Let us consider a simple dynamic system representing a logistic model described by the following difference equations for $t = 0, \dots T-1$ with the initial conditions for x_0 and w_0 given.

$$\begin{aligned}
w_{t+1} &= (1+\alpha) * w_t - \alpha * w_t^2 + 0.5 * x_t, \\
x_{t+1} &= 1.1 * x_t, \\
y_{1,t} &= w_t - 0.5 * x_t, \\
y_{2,t} &= 0.3 * y_{1,t} + 0.15 * w_t \qquad (6.1)
\end{aligned}$$

with $\alpha = 0.5$ and $T = 4$. Two variants or tasks of analysis of the above model will be illustrated, the first one as stated above, that is, a simple process description and its simulation, and the second one will be an optimization variant:

$$\text{minimize } (y_{2,T-1} + 0.3)^2 \ \text{, with respect to } x_0, w_0, \tag{6.2}$$

that is, we force the model to obtain the prescribed final value -0.3 of $y_{2,T-1}$ using the decision variables x_0, w_0. Although simple, this variant has some unpleasant numerical properties when treated by general static optimization methods and tools. In particular, it has nonunique solutions, in fact, an entire nonlinear manifold of initial values x_0, w_0 resulting in $y_{2,T-1} = -0.3$, which is a known trap for some optimization approaches.

6.4.2 AIMMS

The first variant can be represented by the following AIMMS model:

```
ORDERED SET: time:= {t0..t3};
INDEX:       t in time;
PARAMETER:    alpha;
VARIABLES:  w(t) -> (-500, NA, 500), y1(t) -> (-500, NA, 500),
       y2(t) -> (-500, NA, 500), fin,
       x0 ->(-15,4,15),   w0 -> (-15,3,15),
       x(t)->(-15,1.5,15);
PARAMETER: alpha:=0.5;
CONSTRAINTS:
  xt0 .. x('t0') = 0.5,
  xt (t |   t < 't3')  .. x(t+1) = 1.1 * x(t),
  wt0 ..  w('t0') = 3,
  wt (t | t < 't3')
     ..  w(t+1) = (1+alpha)* w[t]- alpha* w[t]**2 + 0.5 * x[t],
  y1t (t)  .. y1(t) = w(t)- 0.5*x(t),
  y2t (t)  .. y2(t) = 0.15*w(t) + 0.3*y1(t);
MODEL: jpw1_1
     subject to: {xt0, xt, wt0, wt, y1t, y2t};
```

The above part of the model describes the actual process modeled and might be called the core model; in order to obtain a full substantive model, we should add the descriptions of various output variables or tasks to this core model. The first simulation task specified in the last subsection can be modeled as:

```
SOLVE  jpw1_1;
DISPLAY ``AFTER solving''    x, w, y1, y2;
```

The second variant – the optimization task – can be represented as follows (only elements added to the above-specified core model are shown):

```
CONSTRAINTS:
  xt0  .. x('t0') = x0,
  fincon .. fin = (y2('t3')+0.3)**2;

MODEL: jpw1_2
     minimize: fin
     subject to: {xt0, xt, wt0, wt, y1t, y2t, fincon};
```

The objective function (in our case, the square of the difference between the desired and actual value of $y2[t3]$) is then minimized by a solver.

An AIMMS program consists of several lower level entities distinguished by corresponding keywords that are case insensitive and allow singular and plural forms. Model entities can appear in any order satisfying the restriction that a symbol must be declared before it is used. Repeated declarations of the same entity are allowed as long as they are not contradictory. Each entity has a default value if uninstantiated.

Sets correspond to indices in the algebraic notation. They can only have a finite number of elements. Sets in AIMMS are generally unordered. Ordered sets are declared by preceding the set keyword with the keyword ordered. The set elements are called labels. The statement time is an example of a set declaration with initialization (in the compilation phase). Index declaration allows for manipulating index elements. Parameters introduce symbolic names for (usually numerical) values in order to make models more general and more readable. Variables correspond to decisions (decision variables). Their values are assigned in the process of solving the model. Their lower bounds, initial values, and upper bounds are defined in the example. Constraints describe relationships (equalities or inequalities) between variables, that is, define a feasible solution. A model declaration either specifies (for an optimization problem) constraints and an objective, or when no objective is given, a system of equations. Solve and Display are examples of executable commands of the AIMMS language.

6.4.3 AMPL

The first task can be represented as the following AMPL model:

```
param  T0 >= 0;
param  TN >  0;
param  alpha > 0;
```

```
set    time:= {T0..TN} ordered;
set    time_1:= {T0..TN-1} ordered;
var  x {t in time};
var  w {t in time} >= -500, <= 500;
var  y1 {t in time} >= -500, <= 500;
var  y2{t in time} >= -500, <= 500;

subject to   xt0: x[T0] = 0.5;
subject to  xt {t in time_1}:  x[t+1] = 1.1* x[t];
subject to  w0:  w[T0] = 3;
subject to  wt {t in time_1}:
 w[t+1] = (1+alpha)* w[t]- alpha* w[t]**2 + 0.5 * x[t];
subject to  y1t {t in time}:   y1[t] = w[t]-0.5*x[t];
subject to  y2t {t in time}:
   y2[t] = 0.15 * w[t] + 0.3 * y1[t];
```

Programs (including keywords) are case sensitive. Keywords appear at the beginning of the declaration and definition of each set, parameter, etc., and not as the marker of the beginning of a new program section only.

There are various other mathematically equivalent formulations of this model. For example, the set declarations may be removed and their definitions substituted instead of their names ($T0..TN$ instead of $time$). More substantial changes will be discussed later.

The model file is accompanied by a data file:

```
param  T0:= 0;
param  TN:= 3;
param  alpha:= 0.5;
```

The second task can be modeled as follows (only the differences in the code are shown below):

```
minimize    fin: (y2[TN] + 0.3)^2;
subject to  xt {t in time_1}:  x[t+1] = 1.1 * x[t];
var  w  {t in time} =  if t = T0 then 3
   else
    (1+alpha)* w[t-1]- alpha* w[t-1]**2 + 0.5 * x[t-1];
subject to  y1t {t in time}:    y1[t] = w[t] - 0.5*x[t];
subject to  y2t {t in time}:    y2[t] = 0.15*w[t] + 0.3*y1[t];
```

A new variant of a declaration, namely, a conditional one for the variable `w`, was additionally presented. The minimize phrase was added and the constraint for the initial value $x[0]$ was removed.

6.4.4 ASCEND

ASCEND (Advanced System for Computation in ENgineering Design) was developed at Carnegie Mellon University, Pittsburgh, PA, USA by Piela *et al.* (1993).[8] In order to illustrate the object-oriented aspects of the language, the illustrative example considered in this section with its two tasks is modeled in the following file as two objects mod_1 and mod_2 derived from a common base class mod_pat.

```
(*  ii2.a4c *)
REQUIRE ‘‘system.a4l";

MODEL mod_pat;
   T0 IS_A integer_constant;
   T1 IS_A integer_constant;
   alpha IS_A generic_real;
   time IS_A set OF integer_constant;
   time_1 IS_A set OF integer_constant;
   T0:== 0;
   T1:== 3;
   time:== [T0..T1];

   x[T0..T1] IS_A generic_real;
   w[T0..T1] IS_A generic_real;
   y1[T0..T1] IS_A generic_real;
   y2[T0..T1] IS_A generic_real;

   FOR t IN [T0..T1-1] CREATE
       r_x[t]:  x[t+1] = 1.1 * x[t];
       r_w[t]:  w[t+1] = (1+alpha)*w[t] -alpha*w[t]^2+0.5*x[t];
       r_y1[t]: y1[t] = w[t]-0.5*x[t];
       r_y2[t]: y2[t] = 0.15*w[t]+0.3*y1[t];
   END FOR;

   r_y1T1: y1[T1] = w[T1]-0.5*x[T1];
   r_y2T1: y2[T1] = 0.15*w[T1]+0.3*y1[T1];

  METHODS
  METHOD clear;
   x[time].fixed:= FALSE;
   w[time].fixed:= FALSE;
```

[8]See also http://www.cs.cmu.edu/afs/cs/project/ascend/home/Home.html.

```
   y1[time].fixed:= FALSE;
   y2[time].fixed:= FALSE;
    END clear;

(*    METHOD calcul;
   FOR t IN [T0..T1-1] DO
        x[t+1]:= 1.1 * x[t];
        w[t+1]:= (1+alpha)*w[t] -alpha*w[t]^2+0.5*x[t];
      y1[t]:= w[t]-0.5*x[t];
      y2[t]:= 0.15*w[t]+0.3*y1[t];
   END FOR;
   y1[T1]:= w[T1]-0.5*x[T1];
   y2[T1]:= 0.15*w[T1]+0.3*y1[T1];
    END calcul; *)
END mod_pat;


MODEL mod_1 REFINES mod_pat;
   r_al:  alpha = 0.5;
   r_xT0: x[T0] = 0.5;
   r_wT0: w[T0] = 3;
END  mod_1;

MODEL mod_2 REFINES mod_pat;
   r_al:      alpha = 0.5;
   r_y2T1fix: y2[T1] = -0.3;
   r_wT0:     w[T0] = 3;
END  mod_2;
```

The compilation of the above file creates the corresponding data structures in the system. There is also a version of the language adapted for the windowing environment. An alternative solution, more convenient in some applications, is to prepare a script file, e.g., as the following:

```
# sii2.a4s

DELETE TYPES;
READ FILE C:/JP_WORK/ASCEND/ii2.a4c;

COMPILE s1 OF mod_1;
RUN s1.clear;
```

```
SOLVE s1;
BROWSE s1;

COMPILE s2 OF mod_2;
RUN s2.clear;
SOLVE s2;
BROWSE s2;
```

and use it to execute the model. The language is equation based, i.e., constraints in the form of equations (and inequalities) are the primary means of expression. However, the imperative paradigm is also present – the method (i.e., function) calcul presents an example for the simulation task in the example (the part related to the function calcul is commented out – i.e., denoted by (* ... *) – in order to stress the essential style of programming in this language). The *. fixed flags indicate that the values of given variables will be calculated in terms of the values given to other variables. The compilation of a model starts with the check of the consistency of the system of equation. After a thorough dependency analysis and eventual reorganization and rewriting of the equations, the problem is exported to the solver. The current ftp distribution of this language uses the QRSLV (a nonlinear algebraic equation solver); according to the manual, several other solvers might also be used (SLV, LSODE, MINOS, LSSlV, OPT(SQP), CONOPT, MAKE MPS).

An interesting feature of the ASCEND language is that a rich family of models with a diversified information flow can be easily obtained.

6.4.5 Modeling in the DIDAS family of DSS tools

This family of decision support system (DSS) tools is closely connected with the reference point optimization approach to multi-criteria optimization and has been referred to in previous chapters. It is also represented in the software attached as an Appendix to this book. We shall consider here only the issues of modeling in two members of this family (using the illustrative example described in Section 6.4.3) aimed at the analysis of nonlinear models, namely an older, closed version DIDAS-N (Kręglewski *et al.*, 1988), and a newer, modular version DIDAS-N++ (Granat *et al.*, 1996).

DIDAS-N is spreadsheet-based, and follows the input–output modeling paradigm, not the objective-constraint optimization paradigm. Any new equation defining a new output variable may depend on input variables, parameters, and previously defined output variables. DIDAS-N consists of two interrelated spreadsheets, one for model editing and simple modeling (including the calculation of all relevant partial or total derivatives of each output) and another for multi-criteria optimization and analysis.

Its language of formulae has no indexed variables, therefore all equations must be manually expanded in the case of dynamic models. They must be properly ordered to preserve the dependence property mentioned above. Models in DIDAS-N are stored in a binary format, therefore the examples will be presented here as read from the spreadsheet. However, the spreadsheet edition relies on self-explaining rules, and the illustrative example can be easily prepared under DIDAS-N. This preparation is equivalent to defining the following equations:

```
x0  = x01     w0  = w01
x1  = 1.1*x0    w1  = (1+a)*w0 - a*(w0)^2 + 0.5*x0
x2  = 1.1*x1    w2  = (1+a)*w1 - a*(w1)^2 + 0.5*x1
x3  = 1.1*x2    w3  = (1+a)*w2 - a*(w2)^2 + 0.5*x2
y10 = w0 - 0.5*x0   y20 = 0.3*y10 + 0.15*w0
y11 = w1 - 0.5*x1   y21 = 0.3*y11 + 0.15*w1
y12 = w2 - 0.5*x2   y22 = 0.3*y12 + 0.15*w2
y13 = w3 - 0.5*x3   y23 = 0.3*y13 + 0.15*w3
fin = (y23 + 0.3)^2
```

where in the first two equations $x01$ and $w01$ are defined as decision variables, while $x0$ and $w0$ are outcome variables determined by them. This convention is used for convenience, since DIDAS-N only displays output variables graphically. During the definitions of model equations, the spreadsheet asks for the declaration of the type of new variables; thus, when defining the equation for $w1$, variable a (used here for simplicity instead of α) has to be declared as a parameter of the model.

During the definition of model equations, one can also define the values of decision variables and parameters; afterwards, model simulation is automatically performed when defining further equations in DIDAS-N. Moreover, it is not only output variables, but also their derivatives with respect to all other variables, that are automatically evaluated and displayed in the spreadsheet (selectively, either partial or total, and numerically while the corresponding symbolic formulae can also be displayed). Thus, model simulation in DIDAS-N includes a sensitivity analysis.

In order to perform the optimization task, we must take into account that DIDAS-N was constructed for multi-objective optimization and analysis and its use for single-objective optimization is somewhat artificial; however, it is possible. Before going to the optimization phase, DIDAS-N requires (similarly to other algebraic modeling tools) the specification of upper and lower bounds for all variables; afterwards, a model can be fixed and the optimization phase started. To obtain the results for the illustrative example, it is sufficient to specify *fin* as a minimized objective and to give the compute command.

In contrast to DIDAS-N, DIDAS-N++ is an open system containing a set of modular tools for various tasks of model definition, analysis, and optimization, both single- and multi-objective. Models in DIDAS-N++ are prepared as text files in a special model definition language (MDL). This language defines only the basic part of the model (also called substantive or core model in this book) and leaves the formulation of a multi-objective optimization problem to other parts of the system. This is different from algebraic modeling languages where both the core model and specific tasks performed with its use appear in one model. Equations appearing in the nonlinear part of DIDAS-N++ must be explicitly resolvable (i.e., contain only defining equations that require no equation solver). However, they can be introduced in any order as they are automatically sorted. The general structure of an MDL program is a generalization of MPS format and was designed to resemble it. The prototype MDL ver. 1.0 uses no concept of sets and no indexed variables; therefore, the corresponding model has the following form:

```
NAME  wi6
parameters alpha = 0.5;
VARIABLES
   x_0  lower = -500; upper = 500; initial  = 0.3;
   w_0  lower = -500; upper = 500;  initial = 3;

EQUATIONS
    x_1 = 1.1 * x_0;      lower = -500; upper = 500;
    x_2 = 1.1 * x_1;      lower = -500; upper = 500;
    x_3 = 1.1 * x_2;      lower = -500; upper = 500;

    w_1 = (1+alpha)* w_0 - alpha* w_0^2 + 0.5 * x_1;
    lower = -500; upper = 500;
    w_2 = (1+alpha)* w_1 - alpha* w_1^2 + 0.5 * x_2;
    lower = -500; upper = 500;
    w_3 = (1+alpha)* w_2 - alpha* w_2^2 + 0.5 * x_3;
    lower = -500; upper = 500;

    y1_0 = w_0 - 0.5*x_0;   lower = -500; upper = 500;
    y1_1 = w_1 - 0.5*x_1;   lower = -500; upper = 500;
    y1_2 = w_2 - 0.5*x_2;   lower = -500; upper = 500;
    y1_3 = w_3 - 0.5*x_3;   lower = -500; upper = 500;

    y2_0 = 0.15* w_0 + 0.3*y1_0;  lower = -500; upper = 500;
    y2_1 = 0.15* w_1 + 0.3*y1_1;  lower = -500; upper = 500;
    y2_2 = 0.15* w_2 + 0.3*y1_2;  lower = -500; upper = 500;
    y2_3 = 0.15* w_3 + 0.3*y1_3;  lower = -500; upper = 500;
ENDATA
```

In this example, equations are already sorted. Such a model is then processed by the model compiler, which produces a file in C++ with functions for the calculation of output variables and their derivatives (the Jacobian matrix), produced by automatic differentiation methods (described in Section 6.5.4). Other files contain the description of model variables, deduced from the model file. They serve as input to other tools, in order to be used when the C++ model file is compiled and linked to other system modules. These also include solvers – in particular, the DIDAS-N solver – thus the optimization results under DIDAS-N++ are similar to those already described for DIDAS-N.

A new version of MDL for DIDAS-N++ has been recently developed to admit indexed variables. In the MDL ver. 2.0, the illustrative example then has the following structure:

```
NAME  wi7
CONSTS N = 3; N_1 = 2;
SETS time = 0..N;     time_1 = 0 .. N_1;
PARAMETER alpha = 0.5;
VARIABLES
  x_0  lower = -500; upper = 500; initial  = 0.3;
  w_0  lower = -500; upper = 500;  initial = 3;
EQUATIONS
 x[0] = x_0;       lower = -500; upper = 500;
 {t in time_1} x[t+1] = 1.1* x[t]; lower = -500; upper = 500;
 w[0] = w_0;       lower = -500; upper = 500;
 {t in time_1}
      w[t+1] = (1+alpha)* w[t] - alpha* w[t]^2 + 0.5 * x[t+1];
      lower = -500; upper = 500;
 {t in time} y1[t] = w[t] - 0.5*x[t];
      lower = -500; upper = 500;
 {t in time} y2[t] = 0.15*w[t] + 0.3*y1[t];
      lower = -500; upper = 500;
END
```

6.4.6 Solving the illustrative example in various systems

AIMMS

The results of the first task – a simple simulation of the illustrative example considered in this section when using AIMMS – are as follows:

```
AFTER solving
Composite table:
    time      x.l      w.l      y1.l     y2.l
!   ----     -----    -----    ------    -----
    t0       0.500    3.000     2.750    1.275
    t1       0.550    0.250    -0.025    0.030
    t2       0.605    0.619     0.316    0.188
    t3       0.666    1.039     0.706    0.368;
```

The first task consists of straightforward computations and can be easily programmed in any universal programming language. Results of such calculations programmed in Pascal and C languages confirm the correctness of the AIMMS results.

The second task – minimizing the square of the difference between the desired and actual value of $y2[t3]$ – needs an optimization solver. The solver used in AIMMS is CONOPT, otherwise a good and reliable tool (refer to Chapter 7). However, in this particular example the results are as follows. After turning on the option Print all messages we obtain in the corresponding Messages window the following statements:

```
 ** Feasible solution.  Convergence too slow. The change in
    objective has been less than 2.9781E-12 for 5 consecutive
    iterations
Warning at line 39.
After 30 iterations CONOPT was interrupted and jpw1_1 is not
    yet optimal.
```

The final value of $y2('t3')$ is equal to -0.645. This example illustrates how cautiously we should use otherwise very good software; in nonlinear programming, for most approaches we can find apparently simple examples such that a given approach fails on them.

AMPL

The results when using AMPL are as follows. Given the model and data, the AMPL generates a specific optimization problem. It then enters a presolve phase that aims to simplify the optimization problem. After setting the option show_stats 1, the following information is obtained:

```
Presolve eliminates 16 constraints and 16 variables.
Adjusted problem:
0 variables;
0 constraints
0 objectives.
Solution determined by presolve.
```

and the solution (with default accuracy omitted here to save space) follows. In this case the usage of a solver is not necessary; the solution is correct.

When solving the second task, AMPL can use several solvers: the authors attempted to use the MINOS solver and the GRG2 solver in the AMPL PLUS environment. The experiments have shown, however, that it is not possible to solve by MINOS (or by GRG2) the optimization problem that results from the above formulation of the model: the system returns false results while announcing that an optimal solution has been found. Robert Fourer suggested the following reformulation of the model,[9] which results in an optimization problem that can be easily solved by both MINOS and GRG2:

```
var  x {t in time};
var w {t in time} = if t=T0 then 3.0 else
 (1+alpha)* w[t-1]- alpha* w[t-1]**2 + 0.5 * x[t-1];
var  y1 {t in time}   = w[t]-0.5*x[t];
var  y2 {t in time}   = 0.15 * w[t] + 0.3 * y1[t];

minimize fin: (y2[TN] + 0.3)^2;
subject to
  xt {t in time_1}:  x[t+1] = 1.1* x[t];
```

The results are:

```
:       x         w           y1           y2          :=
0   -0.276671    3           3.13834      1.3915
1   -0.304338   -0.138336    0.0138336   -0.0166003
2   -0.334772   -0.369241   -0.201855    -0.115943
3   -0.368249   -0.789416   -0.605292    -0.3;
```

Note that this simple example shows that nonlinear models are not invariant as to mathematically equivalent transformations. It should be stressed that this example has no special hidden traps inside it (except for the general difficulty of problems with nonunique solutions) and detailed analysis shows that there are no particularly high sensitivities. Moreover, some other nonlinear solvers (e.g., Numerica)[10] solve it easily. Both AMPL PLUS and MINOS are robust software tools that have been successfully used for solving many – also complex – nonlinear problems. This example is thus not aimed at questioning the quality of this software

[9]The reader is advised to visit the Web page http://www.ampl.com/ampl, which provides useful information on AMPL, and to ftp from the server netlib.bell-labs.com a current update of the AMPL software distributed with the book (Fourer *et al.*, 1993). Note that the example presented in this chapter requires the updated version of the software.

[10]http://www.ilog.com/html/products/numerica/product_numerica_home.html.

tool (for most software tools it is possible to find an example for which it will not work), but rather for illustrating the fact well known to specialists: that various equivalent formulations of a mathematical model may result in dramatic differences for solvers. This in turn supports the view expressed earlier in this chapter that modeling requires not only knowledge but is also an art that calls for intuition and experience.

Finally, we might use this example to demonstrate one good rule of thumb: it is always advisable to use several, or at least two, different solvers for solving an optimization problem and for comparing the results. It is commonly known that out of two good solvers one will perform better for some optimization problems while the second will be better for others. Sometimes it is possible to judge (on the basis of various optimization methods used by solvers) which solver might be better for a particular class of optimization problems; however, often it is not possible (especially for nonlinear problems). Therefore, implementations of AML typically include at least two nonlinear solvers, usually CONOPT and MINOS, and it is easy to compare which solver is better for a particular model (see also Chapter 7).

ASCEND

When using ASCEND, both of the tasks of the illustrative example discussed in this section were solved immediately, without problems, and the first one without using the solver.

DIDAS-N

A simulation task of the illustrative example considered in this section is automatically performed by DIDAS-N, together with model differentiation and thus sensitivity evaluation. Correct results are produced, as follows:

```
x01 =  0.5000  w01 =  3.000   a   =  0.5000
x0  =  0.5000  w0  =  3.000   y10 =  2.750   y20 =  1.275
x1  =  0.5500  w1  =  0.250   y11 = -0.0250  y21 =  0.0300
x2  =  0.6050  w2  =  0.6188  y12 =  0.3162  y22 =  0.1877
x3  =  0.6655  w3  =  1.039   y13 =  0.7064  y23 =  0.3678
fin =  0.3678.
```

When using DIDAS-N the results of the optimization task depend (correctly, because of the nonuniqueness of solutions) on the starting values of $x01$ and $w01$:

```
Starting       x01 =  0.5000  w01 =  3.000
x0  =  0.3112  w0  =  3.307   y10 =  3.152   y20 =  1.442
x1  =  0.3423  w1  = -0.3523  y11 = -0.5234  y21 = -0.2099
```

```
x2  =  0.3675  w2  = -0.4193  y12 = -0.6076  y22 = -0.2452
x3  =  0.4142  w3  = -0.5286  y13 = -0.7357  y23 = -0.3000
fin =  1.968e-12

Starting       x01 =  1.000   w01 = -3.000
x0  =  6.332   w0  =  4.993   y10 =  1.827   y20 =  1.297
x1  =  6.966   w1  = -1.811   y11 = -5.293   y21 = -1.860
x2  =  7.662   w2  = -0.8720  y12 = -4.7036  y22 = -1.542
x3  =  8.428   w3  =  2.143   y13 = -2.071   y23 = -0.3000
fin =  1.489e-13.
```

The various conventions for model definitions discussed in this section, together with their advantages, disadvantages, and various results in applications, justify the conclusion that a standardization of modeling, based both on experience and a good theoretical background, is necessary. The next section on structured modeling provides the background for such a standardization.

6.4.7 Structured modeling

A complete presentation of structured modeling is beyond the scope of this book, therefore we present only an overview of this approach and the corresponding bibliography.

Geoffrion (1989a) envisioned structured modeling as a new generation of modeling environments that would provide a comprehensive, rigorous, principled framework for the entire model life cycle. Model life-cycle phases include: studying the feasibility of a modeling study; designing or adapting a model; making a project plan; developing and verifying the necessary data; preparing a suitable solver; verifying and validating the model and solver; solving the model as many times as necessary; analyzing solution results; developing, reporting, and explaining findings/conclusions/policy implications; documenting the model and solver, data development procedures, and analytical methods; maintaining, updating, and improving these items for the duration of the study; and, finally, terminating the study with proper attention to archives that invite as much reuse as possible. Currently, no modeling environment supports all the model life-cycle phases. However, the structured modeling methodology and prototyping tools will most likely be converted into a next-generation modeling environment.

A modeling environment based on structured modeling will make it much easier to specify, maintain, modify, document, and possibly aggregate models, as well as to apply various techniques to a model analysis.

The executable language SML specified in Geoffrion (1992a, 1992b) is based on the concept introduced by Geoffrion (1987a) and it serves for a model specification according to Geoffrion's conceptual framework (Geoffrion, 1989b). This framework views every analytic model as a collection of objects wherein each object has a definition that is either a primitive or based on the definition of other objects. These definitional dependencies can be represented as a directed graph where the nodes are the objects.

A survey of structured modeling methodology and implementations by Geoffrion (1994)[11] presents the following topics: improved modeling languages; approaches to various kinds of model integration; extensions designed for simulation modeling; and three topics falling under implementation strategies and technologies, namely, host software, language-directed editors, and language-based customization of specialized application packages.

Structured modeling explores the benefits of adapting database principles and technology to modeling environments in order to provide standardized model representations, a well-defined set of model manipulation operations, and support for model administration as presented by Dolk (1988). This is coupled with a rigorous, principled, and general (paradigm-neutral) structured modeling formalism for representing models.

It is now generally agreed that dealing with models of a realistic size requires the use of modern database management system (DBMS) technology, which has advanced immensely in the past two decades. DBMSs can be used far more effectively with a modeling system that is based on principles that are similar to today's dominant DBMS paradigm, namely, the relational paradigm. To understand this possibility, we must recall the evolution of database management theory and technology. The data management revolution occurred in response to severe problems with data reusability associated with file-processing approaches to application development. The need to share data resources resulted in the development of the DBMS, which separates data from the applications that use data. Advances in database technology have been propelled in the past two decades primarily by the development, refinement, and eventual implementation of the relational data model. Two overriding aspects of relational systems were primarily responsible for their ascendancy: first, a simple, table-based perspective that allows users to see data as tables, and second, the existence of an underlying relational theory (Code, 1970) that provides a set-theoretical basis for defining and manipulating tables.

If one considers model management as the modeling counterpart to data management, then model management is at about the same stage of evolution as data management was during its transition from file processing to database processing. The result of this comparison is the insight that what moved database technology forward is now needed to move model management forward: namely, voluntary, de facto standardization around a rigorous, principled representational formalism of great generality. Hence, a next generation modeling system based on structured

[11] See http://www.informs.org./Pubs/ITORMS, Vol. 1 for the updated, on-line version.

modeling can serve as a catalyst for the modeling community and its systems just as relational theory did for the data management community and its systems.

Structured modeling has not yet become as widely used as modeling systems discussed in previous sections of this chapter. However, several implementations of this methodology exist. We provide here references to one recent implementation called OR/SM, which is documented by Wright *et al*. (1997) and by Kang *et al*. (1997). More information about structured modeling can be found at http://www.anderson.ucla.edu/faculty/art.geoffrion.

6.5 Model Analysis and Transformations

This section discusses various requirements of tools for model analysis and transformation, particularly if these tools should become part of a DSS. Experience shows that a DSS must be constructed as a problem-specific software system or package; however, we would like to use generic tools and methods as much as possible or available. This philosophy must be reinterpreted for each case, since we cannot define what is sufficiently problem specific and what is sufficiently generic.

6.5.1 Applicability of various types of modeling tools

Wright *et al*. (1998) recently undertook an empirical study on the needs of business and industry for various features of computer-based modeling environments. These results confirm the common opinion (at least among the operations research specialists) that such needs vary substantially. Therefore, the demand for various types of modeling tools will continue because different types of tools are best for different types of needs.

One can consider a taxonomy of modeling tools based on the range of types of models that can be handled by a tool. The available modeling tools (only some of them were discussed earlier in this chapter) cover the whole range of capabilities, from general purpose modeling systems through modeling languages specialized for various classes of models, to tools specific for a particular application. For simplicity, let us consider just two classes of tools, namely, general purpose modeling systems and problem-specific tools – although tools from the middle of this range, combining generic and problem-specific properties, might sometimes have best properties.

Although there is wide consensus that DSSs should be problem specific (this is illustrated by many examples in this book, including three applications presented in Part III), there are arguments for both a problem-specific and a generic approach. Such arguments are organized in the following groups:

- Functionality of the resulting DSS.
- Efficiency of computations.
- Resources needed for development and maintenance.

The first two are in favor of a problem-specific approach, while the third one is in favor of a generic solution.

Functionality of the Resulting DSS

Functions of a DSS might very much depend on the specific problem. This is due to many reasons. For example, model analysis for decision support requires various types of analytical tasks. For some problems single-objective optimization (which also includes examining various objectives, however only one at a time) combined with a sensitivity analysis provides enough support. However, a more comprehensive analysis that uses various techniques discussed in this book might be needed for other problems, even if an incorporation of all these techniques into the available modeling systems would be very difficult, if at all possible. For example, the model presented in Chapter 11 shows how much can be gained by using multi-objective model analysis techniques – in comparison with detailed and careful single-objective optimization combined with classical simulation and applied to a relatively small and conceptually simple model.

Moreover, a thorough analysis of a complex model requires various levels of modifications of the model by various groups of users. For example, a user of a distributable version of the model needs to access only a very small part of the data and the variables. This can be much easier if it is provided by a problem-specific user interface than by a general purpose modeling system. The model presented in Chapter 12 illustrates a situation when a complex model is composed of several submodels, and when it is practicable to provide separate interfaces to these submodels (because they are maintained and analyzed by various groups of analysts but some users may also want to occasionally analyze a submodel).

Efficiency of Computations

Problem-specific software is usually much more efficient than general-purpose software in processing input data (including more sophisticated checks of data consistency) that is needed for a specification of a complex model and for producing various reports of different levels of details. Thus, problem-specific software is generally superior when it comes to issues of data handling and model interfacing.[12]

[12]This statement refers to the current situation, with no sufficiently standardized or general modeling languages available. When such languages will be developed, this situation might change.

A modeling system does not typically exploit all possibilities of an optimization algorithm used in a solver. This is not an important issue for linear models (because preprocessing is a standard feature of any good LP solver) or for problems of a moderate size. Preprocessing of nonlinear models by a general purpose nonlinear solver is a much more difficult task (Drud, 1997), because a solver does not have complete information about the model (only the structure of the Jacobian and values of its linear elements can be easily recognized by a solver). For large problems it might be necessary to exploit the problem structure in order to reduce optimization time. In such a case, a modeling system can be augmented by a specific software tool for exploiting the structure of optimization problems generated by a modeling language, such as the one by Fragnière *et al*. (1997). However, such tools are not yet very advanced (i.e., they recognize only some typical structures) and are not widely available. Chapter 13 provides a convincing example of a complex nonlinear model for which the model generator performs very efficient preprocessing. Also, this example shows that solution time might be a decisive issue for problems of larger dimension. Because of all these issues, developing a problem-specific model generator might be a better choice for large problems than using a modeling language.

Resources Needed for Development and Maintenance

It is clear that the use of general tools minimizes the resources (i.e., time and money) needed for software development and maintenance. Here we discuss these issues in more detail.

A modeling system greatly simplifies the task of a model specification, especially when compared with the amount of resources needed for developing a problem-specific model generator by using traditional procedural programming languages such as C or Fortran. Although the use of C++ substantially reduces this difference, especially with the recently included Standard Template Library and with a class library for mathematical programming (e.g., by Nielsen, 1995, or by Makowski, 1999, but also enabling the use of a vast range of software developed over the years in C and Fortran), a problem-specific tool typically requires many more resources than an implementation based on a modeling system, if such a system can easily provide the required functionality. However, some models (e.g., those presented in Chapters 12 and 13) are distributed among a large number of users who typically do not want (or cannot afford) to cover costs related to the use of a modeling system.

A modeling system releases the modeler from the complex task of providing codes for computing the values of nonlinear constraints and of nonlinear elements of the Jacobian. The latter task, in particular, used to require a large development time and was prone to human error. However, recent developments in the area of automatic differentiation (see Section 6.5.4) and symbolic computation (see Section 13.4.1 for an example of application) resolved this problem.

Fourer (1983) provides a good overview of the advantages of modeling systems for prototyping, verifying, modifying, and documenting models. However, one should consider how well the actual implementation of a given modeling system fits to a particular modeling problem, and how well the methods of a model analysis offered by this system correspond to actual needs. Moreover, one should be aware of the pitfalls that can trap nonexpert users of modeling systems who may be misled by a belief that when using a modeling system, one does not need modeling expertise (the example provided in Section 6.4.3 illustrates the need for modeling expertise even for the specification of rather simple models). However, modeling systems constantly evolve in order to meet more demanding requirements of users. The developers of these systems typically help the users in dealing with the problems and limitations of a particular system. Therefore, there will be many modeling problems where the application of a modeling system will be a rational choice, and many other problems where the development of a problem-specific software will be a better choice. The latter group of application will increase the demand for modular, robust modeling tools that can be used for the development of such software.

6.5.2 Dependency analysis

In this section, we discuss only general aspects of consistency and dependency analysis for more complex models, not the more classical analysis tasks connected, e.g., with syntax checking. Some analysis of the program defining a model is necessary in each system, but its scope and depth might differ considerably. If a language leans toward the imperative paradigm (i.e., an intermediate variable must be computed to be available), a thorough analysis of dependency and sorting of defining equations is absolutely necessary. Thus, there are two forms of a model in such a language:

- The original one, following the conventions of a descriptive scientific origin.
- The internal one, adapted to the needs of a solver.

It should be stressed that such an analysis is not contained in the compilers of any commercial programming languages known to the authors – the usage of variables with uninstantiated values normally remains undetected.

If a solver admits only explicitly solvable models, such transformation should be accompanied by the detection of loops and their diagnosis, as is done, e.g., in MDL for the DIDAS-N++ system. Even if implicit models are allowed, such loop analyses might be necessary for finding strongly connected components of the dependency graph and for a transformation of equations in order to use effective structural methods for implicit system solving. A module of dependency analysis should be present in each modeling language. However, their sophistication might differ considerably. The algorithms used for this purpose are, as a rule, not documented, hence a user can obtain some insight into their quality only by performing experiments.

6.5.3 Code generation

Many modeling systems are closed, i.e., they can be used only as a whole entity. Although they are often very sophisticated, an assertion that their authors were able to foresee the needs of every modeler is clearly false. One obvious solution to this dilemma is to design an open modeling system, e.g., in the form of modules (ideally in a universal programming language, with source code available), with well defined and documented interfaces between them. Although this solution might be less attractive commercially, we shall comment on its other advantages.

Such a solution has many advantages. An advanced user may configure the system to her/his particular needs. Although a model must be compiled and linked in each new run (if an interface is provided in a text file in a programming language), its execution might be more effective (e.g., if it is executed many times or is computationally intensive). Such an approach was adopted, e.g., in the design of the DIDAS-N++ system (Granat *et al.*, 1996) and a C++ class library for mathematical programming (Nielsen, 1995). As the advantages of this approach are obvious, there is a visible trend to enhance the already existing systems with code generation modules (a target language can often be chosen). Such modules exist in computer algebra systems, such as Maple (Char *et al.*, 1992), Mathematica (Korsan, 1996; Wolfram, 1996), and are announced in the ACSL[13] simulation tools.

6.5.4 Computer differentiation of models

A symbolic differentiation module is a necessary part of any computer algebra system. However, using it in the context of numerical calculations without proper care might produce bad results for models of any practical size. The resulting code might be inefficient due to the repetition of too many common subexpressions or be unreadable due to very long expressions. This problem is addressed by a new branch of numerical analysis, called automatic differentiation (AD; Griewank and Corliss, 1993), and more recently computational differentiation (Bischof *et al.*, 1994; Bertz *et al.*, 1996). AD analysis addresses the issues of computationally efficient calculation of derivatives.

[13]ACSL Simulation Software, http://www.mga.com.

One possible way of viewing an expression, function, or program to be differentiated is in the form of a computational graph. This is a directed (acyclic) graph where leaves denote input and output variables, intermediate nodes denote intermediate variables representing elementary operations, and an arc between two nodes denotes the dependency relation. This graph can be augmented by labeling arcs with values of corresponding partial derivatives. The derivatives of output with respect to input variables can then be calculated by accumulating numerical values of local derivatives. This process can be achieved during the traversal of a computational graph from input towards output nodes, or in the reverse direction – after the preliminary pass during which only values of intermediate variables are calculated. These two variants of algorithms are called forward mode and reverse mode, respectively. A concise overview is presented by Griewank (1994). There are also two main implementation techniques, one based on operator overloading and another where a code in a given language is generated. The first one is relatively easy to implement but numerically inefficient for larger models. In the second technique the most complicated task is the design of an effective simplification algorithm since the naive differentiation gives a redundant code (e.g., with many zeros and ones resulting from the differentiation of constants and independent variables).

The main trend of research in AD concentrated at the beginning mainly on the differentiation of (large) programs in universal programming languages. It was motivated by the huge amount of code that was written without foreseeing the calculation of derivatives. An example of this trend is the ADIFOR for differentiation of programs written in Fortran 77 by Bischof *et al*. (1996).[14]

Another trend in this research was the design of (usually smaller) specialized modeling systems. Examples of implementation of AD algorithms in AMPL appear in Gay (1991, 1996). In this system, differentiation is not visible to the user, it is only used by the solver. A reverse approach was chosen in the DIDAS-N system (Kręglewski *et al*., 1988), where both the values of derivatives and their analytical formulae are visible to the user. The AD algorithm in DIDAS-N utilizes the reverse mode augmented by a thorough analysis of common subexpressions. An example of its results is given below. The reverse mode is also used in DIDAS-N++ in which a text file with a C++ function calculating outputs and their Jacobian is generated.

Since computer algebra packages have powerful simplifying modules, it is relatively easy to extend them by adding an AD module, producing a code in its own language or in a chosen programming language. For example, in Maple ver. 3 there is an implementation in the SHARE library,[15] and a new implementation was also

[14]Current information on ADIFOR is available at http://www.mcs.anl.gov/adifor.

[15]M.B. Monagan and W.M. Neuenschwandler, GRADIENT, Algorithmic Differentiation in Maple, in the distribution package: maplev3/share/autodiff/gradient.tex.

presented by Monagan (1996). In Mathematica (Wolfram, 1996) there is also an AD module.

Although the chain rule of differentiation has been known since the 1600s, a (nearly) optimal AD algorithm still remains a research problem. Examples of issues currently being addressed are the treatment of loops representing iterative calculations, the choice of granularity of elementary operations, exploitation of the structure of the differentiated program, and optimal mixing of forward and reverse modes.

In order to demonstrate the possibilities of automatic differentiation, we return to the illustrative example discussed in previous sections and present its analysis with the help of DIDAS-N (with $\alpha = a = 0.5$, $x0 = x01 = 0.5$, and $w0 = w01 = 3$). As already mentioned, the system automatically computes all derivatives during simulation. The values of the (by default – partial) derivatives are shown in the cells of the spreadsheet of DIDAS-N. If we choose, say, the row corresponding to the output variable $w3$, then nonempty cells correspond only to the variables a, $x2$, $w2$, and display the numeric values of partial derivatives $\frac{\partial w3}{\partial a}$, $\frac{\partial w3}{\partial x2}$, $\frac{\partial w3}{\partial w2}$. However, if we put the cursor on a nonempty cell and push Enter, then a symbolic expression for the derivative is displayed. It is reconstructed from the intermediate code used for numerical calculations. By proceeding this way, the following results are obtained for row $w3$:

Variable	a	x2	w2
Numeric	2.359e-01	5.000e-01	8.812e-0.1
Symbolic	w2-w2^2	0.5	1+a-a*w2^(2-1)*2

The symbolic expression for $\frac{\partial w3}{\partial w2}$ is not sufficiently simplified according to standards of any computer algebra system. However, the main design goal was to make it an effective and easy-to-use multi-objective optimization system and therefore the aesthetic outlook was sacrificed in favor of the speed of computations.

DIDAS-N presents the numeric values of total derivatives, if we use its function Switches. In the example considered above, many more nonempty cells appear in row $w3$ after switching to total derivatives (because an outcome variable might depend on other variables indirectly). We see then that $w3$ depends on a (with a coefficient of –6.208e+00, which summarizes all direct and indirect dependence, different from the direct partial derivative value 2.359e-01 quoted above), on $x01$ and $w01$ (with the same coefficients as on $x0$ and $w0$), on $x1$ and $x2$ (but not on $x3$), on $w1$ and $w2$ (not on itself, $w3$, because we do not use implicit relations), and not on other model variables. This information is very important for more complicated models; in the illustrative example we can check that the output variable *fin* does not depend on $y10$, $y11$, $y12$, $y20$, $y21$, $y22$, though it depends on $y13$, $y23$, and on all other model variables. Because of this and other factors, DIDAS-N is a very good tool for analyzing small- or middle-sized nonlinear models.

An overview of AD tools is given by Bischof and Griewank (1996). The availability of AD changed some trends in optimization: some algorithms previously considered as theoretically attractive but practically nonimplementable are gaining in popularity.

6.6 Specification of a Model for Decision Support

In this section we return to general issues of specification of a substantive (core) model, which is to be used for model-based decision support. Such a model is used for predicting and evaluating the consequences of decisions. The value of a mathematical model as a decision aid comes from its ability to adequately represent reality. Therefore, there is always a trade-off between the requested accuracy (realism) of the model and the costs (and time) of its development and providing it with data. Hence, the requested accuracy should be consistent with accuracy really required for the model. A specification and an implementation of a model require knowledge and experience as well as collaboration between researchers with different backgrounds and with users of a model. Model building is still a mixture of art and science that requires knowledge and experience, including a good understanding of the problem, sound knowledge of model building methodology, and an understanding of solution techniques that will be used for processing the model. Good overviews of problems related to model building, illustrated by many examples, are provided by Huntley and James (1990) and Williams (1990, 1993). The process of specifying the requirements to be met by the modeling process or establishing the specifications that the modeling process must fulfill is sometimes called metamodeling. It is also possible to examine a metamodel through the modeling process, as demonstrated by van Gigch (1991).

6.6.1 The concept of a core model

A core model is typically composed of the following elements (Wierzbicki and Makowski, 1992):

- Decision variables that represent actual decisions (alternatives, choices, options, etc.). For example, in the Regional Water Quality Management Model (see Chapter 11), the decision variables are selections of technologies (which also include the so-called "do nothing option") of waste water treatment plants located at each of the controllable waste emission points. Each technology at each water treatment plant has a corresponding binary variable that indicates whether a given technology is selected.

- Variables defining potential criteria (objectives, goals, performance indices, outcomes), which can be used for evaluating the consequences of implementing the computed or chosen decisions. In the Regional Water Quality Management Model such objectives include various costs (total annual, investment, operational) and ambient water quality indicators (concentration of different waste constituencies, violations of water quality standards) for selected monitoring points and for the whole region.
- Various intermediate and additional variables, such as balance and/or state variables, resources, endogenous (i.e., not controllable) decisions, which are necessary (or make it easier) to formulate the constraining relations and/or ease understanding of the model formulation and interpretation of results. In the Regional Water Quality Management Model such variables include resulting (after selected treatment options) concentrations of waste constituencies in the discharged water and in a river at the monitoring points, and cost components for each treatment plant.
- Constraints (inequalities and equations) that reflect the logical relations between all variables represented in the model. In the Regional Water Quality Management Model constraints include conditions for a sum of binary variables at each plant to be equal to 1 (thus ensuring that exactly one technology is selected for each plant), mass balance equations for waste constituencies at each considered point, nonnegativity constraints for all variables.

A solution of the model is composed of all determined variables of all types (decisions, objectives or criteria, intermediate or additional). A solution that fulfills the constraints is called a feasible solution of the core model. Therefore a set of constraints of a core model indirectly determines a set of feasible decisions and of attainable values of criteria.

The vector of model variables might be composed of all types of variables used for the model specification, namely: decisions, outcomes, objectives or criteria, state variables, and intermediate, parametric, and additional variables. Quite often one variable can be classified as being of more than one type, e.g., a decision variable can also be treated as an outcome variable, an intermediate variable may also be an outcome, and typically all criteria are also outcome variables. The core model defines intermediate and outcome variables by additional equations. A classification of variables is an important issue for a model specification and it is usually convenient to use different symbols for different groups of variables. Readers interested in issues of model specification are advised to consult Wierzbicki (1992b) for more details. However, for the sake of brevity, in this chapter we use a simplified notation, which is aimed at the user of a model, who usually deals with only a small fraction of all variables used in a model, namely, the decision variables and the variables representing objectives (criteria).

The concept of core model, as discussed here, is very similar to the definition of a substantive model introduced in Wierzbicki (1992b) and discussed in earlier chapters; the differences are minor (they were also illustrated by the illustrative example in this chapter) and correspond, e.g., to the issue of whether to include bounds for outcome variables that might become objectives in the core model.

There are two essential differences between a classical specification of an optimization problem and the specification of a core model:

- First, no goal function is specified in the core model, because various goal functions or criteria for multi-criteria model analysis can be specified during the analysis of the model. Traditional single-criterion optimization has this basic drawback, i.e., that the one selected objective function dominates the entire analysis, while most decisions are based upon a consideration of several criteria.
- Second, a properly defined core model should always have a nonempty set of feasible solutions. This is achieved by avoiding the inclusion of any artificial constraints (e.g., those representing a preferential structure of the user) in the core model.

Typically, infeasibility is caused by a representation of a preferential structure of a user by constraints, with constraints that are too tight. As stressed in Chapters 4 and 8, there are a number of approaches to deal with multi-objective problems within the framework of single-objective optimization. One of the most popular approaches is to select one objective as a goal function and to impose constraints on other objectives. This is the essence, for example, of the so-called ϵ-constraint approach, proposed by Haimes and Hall (1974), which replaces (n-1) objectives by constraints with given tolerable levels. Such levels have the interpretation of reservation levels (i.e., values that should be achieved) for these objectives. This is, however, a rather hard requirement; if such constraints are included in the model, they might easily result in an empty set of feasible solutions. There are also other difficulties related to such an approach.

The first difficulty is practical: a sequential conversion of all but one criterion into constraints and consecutive changes of tolerable or desired values of the corresponding constraints is a cumbersome procedure, difficult to be followed even by an experienced model builder. The second difficulty is related to problems with sensitivity analysis and dual solutions (see Chapters 7 and 11). The third drawback is related to cutting off – by constraints that represent criteria – a substantial part of solutions that are actually feasible for a decision problem; thus, solutions in this part will not be analyzed.

All these arguments justify the recommendation not to include any constraints that represent the preferential structure of a user into the core model, but to deal with such constraints in the latter part of analysis, while treating them as soft rather than as hard constraints.

6.6.2 Guidelines for building large models for use in a DSS

Following is a summary of some guidelines based on the experience of the authors with applications in various areas. These guidelines do not pretend to be complete, but experience has shown that disregarding them results in either numerical problems or in unnecessarily complicated model generation, handling, and analysis. We restrict the discussion to linear programming (LP) models (Makowski, 1996), but most of the guidelines also remain valid for mixed integer or nonlinear models.

We illustrate these guidelines first with an additional comment on essential and nonessential matrix coefficients. The problem can be illustrated by the following example. Consider the i-th row of the constraint matrix $\boldsymbol{A}$ from a linear programming model and assume that the matrix $\boldsymbol{A}$ is well scaled (i.e., its coefficients have absolute values close to one)[16] except for the i-th row. Any scaling routine can achieve good scaling of $\boldsymbol{A}$, if all coefficients of the i-th row are of the same (even very small) magnitude. However, if just one a_{ij} is several ranges of magnitude smaller than other coefficients in the i-th row, then there is no way to achieve a good scaling of $\boldsymbol{A}$. This small a_{ij} value has a negligible impact on the value of the i-th row,[17] but even one nonsubstantial coefficient usually causes substantial worsening of scaling of the matrix $\boldsymbol{A}$, which in turn often results in numerical problems for a solver.

There are no simple rules saying which coefficients are essential. This might be decided only by the modeler after careful analysis of each group of constraints. One should be aware that a rule of rejecting coefficients having an absolute value smaller than a given threshold requires specification of such thresholds for each group of constraints.

The following points should thus be considered during the specification and generation of a linear model that will be used for multi-objective analysis aimed at supporting decision making:

[16]A commonly accepted rule of thumb says that a matrix is well scaled if absolute values of coefficients are within the range [0.01, 100].

[17]This is why many modelers tend to neglect problems caused for optimization solvers by the generation of coefficients that are small in the sense of absolute value.

- The data used for the model should be stored, verified, and handled separately from the model specification. The model should be generated either by a problem-specific generator or by a general purpose modeling tool.
- The model should include only substantive constraints. In this context, substantive means constraints that represent all logical and physical relations between variables that should be taken into account while assessing the feasibility of a solution and physical relations between variables.
- Manual scaling of the original data and of the LP matrix coefficients should be avoided. The coefficients should be computed using original data. Any good LP solver scales the problem before attempting to solve it; therefore, one does not need to worry about generating very small or very large coefficients.
- Only such matrix coefficients should be generated that differ essentially from zero. This condition is important although it might be difficult to fulfill, especially if a general purpose modeling tool is used. One should be aware that generation of nonessential, small coefficients may make it impossible to scale the matrix well, which in turn usually results in numerical problems.
- The generated bounds and right-hand side values of constraints should correspond only to logical and physical relations. No additional restrictions or constraints should be introduced in order to reflect acceptability of a solution because this will be accounted for during the model analysis.
- All potential objectives should be defined as outcome variables in the model.
- The model specification should correspond only to the decision problem. For example, one should not generate additional slack variables in order to generate a problem in the standard LP form given by equation (7.2) (see Chapter 7). However, the specification of a model in a form suitable for a specialized solver (e.g., as a dynamic or stochastic problem) usually dramatically decreases the computation time.
- One should avoid specifications of large numbers as infinite values. Such an approach is sometimes used for removing the default finite bounds (specified in addition to the MPS format input files) and it is harmless for the simplex-based solvers. However, it results in problems for the interior point algorithm implementations (see Gondzio and Makowski, 1995b, for more details), therefore it should be avoided.

6.7 Conclusions

Although we have only touched upon various issues of computerized modeling, it should be clear from this overview that this field is complex. As computerized modeling gains in popularity in many scientific disciplines, such as agricultural research (where modeling can essentially decrease the costs of long experiments), or telecommunication networks that can only be developed today with computerized modeling, it is clear that the methodology and standardization of modeling approaches are important subjects that require further research. Hopefully this research will involve a collaboration between researchers from various fields of science, and practitioners who apply and use models for supporting decision making.

Acknowledgments

The authors thank Professor Robert Fourer of Northwestern University, Evanston, USA, for his detailed comments on the draft of this chapter, for checking the examples used throughout, and for suggesting an alternative formulation of one of these examples. The critical and constructive comments have helped the authors to improve the earlier version of this chapter considerably.

The authors also wish to thank Professor Arthur Geoffrion of the University of California, Los Angeles, USA, for sharing his knowledge and experience in modeling, and especially for providing useful information about structured modeling. The authors acknowledge unpublished material of A. Geoffrion, which was adapted for Section 6.4.7.

Chapter 7

Optimization Tools

Janusz Granat, Marek Makowski, and Andrzej P. Wierzbicki

Chapter 6 showed that there is no universal modeling tool; the situation is similar for optimization tools. The research on mathematical programming and computerized optimization tools (which are often called optimization solvers) is now over 50 years old, since it started even before modern computers were invented. This research resulted in a wealth of results covering diverse areas. To present all these results in detail would require more than just a chapter. Thus, we assume that the reader already knows basic concepts and has some theory of mathematical programming (e.g., Luenberger, 1984; Fletcher, 1987; Williams, 1993). We will reintroduce some basic notation and results only in order to comment on specific features of optimization solvers.

Section 7.1 presents some general comments on classes of optimization models and properties of optimization tools. Sections 7.2–7.4 overview linear, mixed integer, and nonlinear programming solvers, respectively. Section 7.5 comments on other classes of optimization models and tools for solving them, and finally, Section 7.6 discusses the application of optimization techniques for decision support.

7.1 Classes and Aspects of Optimization Models and Tools

Attempts are constantly being made to devise a universal optimization tool. Currently, such attempts focus on genetic optimization algorithms, which will be discussed at the end of this chapter. The following complexity rule exists: any universal optimization tool requires computational effort, which increases exponentially with the dimensions of the problem. However, repeatedly it appears that optimization algorithms and the resulting software tools or solvers only work efficiently when they are tailored to a specific class of models.

7.1.1 Classes of optimization models and solvers

A distinction is typically made between the following classes:

(1) Linear programming solvers (for linear programming models, i.e., models with linear objectives and constraints).
(2) Mixed integer programming solvers (for linear programming models with some variables restricted to integer values).
(3) Nonlinear smooth programming solvers (for nonlinear smooth programming models; these are models with nonlinear but differentiable objectives and constraints).
(4) Nondifferentiable optimization solvers (for nonlinear nonsmooth models).
(5) Global optimization solvers (nonlinear smooth as well as nonlinear nonsmooth solvers usually only provide local solutions, whereas special solvers are needed to provide global solutions).
(6) Dynamic optimization solvers (for models with explicit dependence of solutions on time; dynamic programming is only one of the specific algorithms of dynamic optimization).
(7) Stochastic optimization solvers (for models with uncertainty represented by probabilistic distributions).
(8) Fuzzy programming solvers (for models with uncertainty represented by fuzzy sets or possibilistic distributions).

Multi-objective optimization solvers could be added to this list; however, in this book multi-objective model analysis is treated as a methodology that can be used along with any other class of models.

Some other classes mentioned above – i.e., (5), (6), and (7) – are also often combined with some of the earlier classes, mainly (1), (2), and (3). For example, although stochastic optimization has developed methods applicable to more general classes of models (e.g., nonlinear and dynamic) it can be effectively applied to more complicated models only if they are linear. Stochastic linear programming is often combined with dynamic linear programming. Similarly, fuzzy programming can be more effectively applied for piece-wise linear models; more general nonlinear cases require much more computational effort. For these reasons, we only make a short overview of classes (1), (2), and (3) mentioned above, with brief comments on other classes and related problems. There are solvers for solving specific classes of optimization problems, such as network optimization, the traveling salesman problem, scheduling problems, etc., and solvers that use novel approaches like genetic algorithms, simulated annealing, logic constraint programming, etc. In this chapter we concentrate on general purpose analytical solvers.

Information on currently available optimization tools can be found on the internet (e.g., Linear Programming FAQ[1] and Nonlinear Programming FAQ).[2]

7.1.2 General aspects of optimization tools

The problem of describing optimization tools is complicated by several additional factors. One of these is that at least three different approaches to using optimization tools for model analysis can be distinguished.

The first approach is that of a specialized modeler. She/he concentrates on model building and simulation, using tools and model formats specific to her/his specialty. After a model has been built and validated by repeated simulation runs, the modeler might consider specific optimization of some model variables. The modeler then starts to look for an optimization tool and typically adapts the existing tools to her/his needs.

The second approach is that of an optimization specialist. She/he insists that the model should be built according to standard formats and is motivated more by the development of optimization tools of a certain class than by modeling requirements. At the same time, most attention is given to the specifics of a selected technique of optimization and the software package – optimization solver – implementing this technique.

The third approach is a combination of the above two approaches with a basis in multi-objective model analysis; this could be the approach of a "sophisticated" analyst. As a specialized modeler, such an analyst knows the value of tools and model formats specific for a given field of modeling. She/he also knows how to combine such tools with various optimization solvers, and how to treat optimization as a tool of various tasks of model analysis, and not as a goal.

Unfortunately, most of the existing software tools result from one of the first two approaches; these tools can be used within the third approach, but with some difficulties and reservations (see Chapter 6 for a more detailed discussion).

Various other difficulties relate to specific aspects of using optimization tools, examples of which are listed as follows:

[1]Fourer, R., and Gregory, J., 1997, *Linear Programming FAQ*, http://www.mcs.anl.gov/home/otc/faq/linear-programming-faq.html, Usenet sci.answers, anonymous FTP /pub/usenet/sci.answers/linear-programming-faq from rtfm.mit.edu.

[2]Fourer, R. and Gregory, J., 1997, *Nonlinear Programming FAQ*, http://www.mcs.anl.gov/home/otc/faq/nonlinear-programming-faq.html, Usenet sci.answers, anonymous FTP /pub/usenet/sci.answers/nonlinear-programming-faq from rtfm.mit.edu.

- Scaling of variables and constraints, in order to use such scales of variables and functions that the optimization tool works efficiently with them, and presolving (i.e., exploiting a specific model structure to simplify the model for computations with a specific solver). This must be accompanied by postsolving (i.e., reverting to the original form and scaling of the model).
- An initial phase of obtaining a feasible solution that satisfies constraints (in linear and also in some nonlinear solvers) that might be more or less efficient in repetitive optimization runs, together with the problem of a warm restart of a solver (i.e., starting the solver from any point given by the user or by other software modules, not from a point assumed arbitrarily (cold start) in solver implementation).
- The issue of universal heuristics in mixed integer programming (MIP) solvers. These solvers usually include good heuristic algorithms for shortening computations, but such algorithms are good only for specific classes of problems (e.g., for a specific form of objective function) and not for a general selection of any outcome of the model as an objective function.
- The issue of symbolic or automatic differentiation of nonlinear smooth models. Until about 20 years ago, strong emphasis was placed on developing a special class of nonlinear programming algorithms for functions with derivatives that were unknown. However, in mathematical modeling the derivatives of functions that are used are known, only they might be too complicated to obtain any program.[3] Even if separate software did exist for symbolic differentiation, it might not be sufficient to be used with a nonlinear optimization tool. To be really useful, the symbolic differentiation must be integrated with either the modeling system or the optimization tool. However, as already stressed in Chapter 6, not all modeling systems or optimization tools are equipped with automatic differentiation.
- A good optimization solver should be equipped with modules for post-optimal analysis (i.e., sensitivity analysis of the optimal solution and objective function value with respect to various parameters). Post-optimal analysis is related

[3]For example, there is widespread opinion between specialists in nonlinear programming that one should not use Newton–Raphson algorithms, but should use variable metric quasi-Newton algorithms instead, which utilize numerical approximations of the Hessian matrix of second-order derivatives or its inverse. Various reasons exist in the literature to substantiate such an opinion. However, careful testing of these has surprising results. If sufficiently good symbolic differentiation software provides reliable formulae for components of the Hessian matrix, then it might be possible to implement a regularized variant of a Newton–Raphson algorithm in such a way that it is superior to any variable metric algorithm. However, if the formulae for second-order derivatives are to be programmed by hand, than it is almost impossible to escape programming mistakes for problems with more than 10 variables (and thus possibly more than 55 entries in the symmetric Hessian matrix). When taking into account such mistakes, the Newton–Raphson algorithm is much less reliable than a variable metric.

to a dual formulation of the optimization problem and to Lagrange multipliers, which must be reliably determined, if they are used. However, Lagrange multipliers might often be determined nonuniquely, which is a typical trap for such computations. Post-optimal analysis has a special meaning in linear programming, where ranges of parameter changes are investigated that trigger a solution change, and another meaning for other optimization problems (e.g., in nonlinear programming, where local sensitivity coefficients might be determined if symbolic differentiation of the model is additionally used). Very few optimization solvers are equipped with full post-optimal analysis modules.

- A general issue is that of solver robustness: a solver should give reliable results under a wide range of model parameter changes, initial point changes, etc. A careful model analysis might require hundreds of optimization runs with such changes; a solver that works only for some runs, while for others it requires special tuning of its parameters, is clearly not acceptable for the user. Unfortunately, solver robustness cannot be generally expected (except for linear or reasonably small models) and some tuning of solver parameters might often be required.
- Finally, there is the issue of choice between commercial and experimental solvers. Commercially available solvers are typically much more robust than public domain solvers prepared by researchers and presented to the public for experimental use. However, commercial solvers might not be the best attuned for solving a specific type of model or problem that are of interest to the user. Thus, if the user can invest her/his time and find an experimental solver especially developed to take into account the specifics of a given class of applications, such a solver might work many times faster and solve more difficult problems than a commercial one. One should, however, proceed with care, have both commercial and experimental solvers, and compare the results of their applications.

This list does not exhaust all the difficulties that might face a user of an optimization solver, it only lists examples of the most common difficulties. The user must be ready to confront such difficulties. With these precautions in mind, we now discuss specific classes of optimization solvers.

7.2 Linear Programming Solvers

The standard textbook formulation of a linear programming problem is:

$$\underset{\boldsymbol{x} \in X_0}{\text{minimize}} \, (q = \boldsymbol{c}^T \boldsymbol{x}) \tag{7.1}$$

with the set of feasible (admissible) decisions X_0 defined by:

$$X_0 = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{Ax} = \boldsymbol{b} \in \mathbb{R}^m,\ \boldsymbol{x} \geq 0 \in \mathbb{R}^n\}, \tag{7.2}$$

where the minimization can be easily changed to maximization by altering the sign of the objective function. Until now we have considered maximization of an achievement function – or a value function, or utility function – as a standard; however, it is better to consider minimization as a standard case for optimization tools (which simplifies some definitions related to convex optimization problems; see comments in Section 7.4).

All other formats of linear programming problems can be converted to the above form. Inequality constraints are changed into equations by introducing slack variables with an appropriate sign. Absolute values such as $\mid x_j \mid$ (e.g., used to define an objective function in approximation problems) are expressed as $\mid x_j \mid = x_j^+ + x_j^-$, with additional constraints $x_j = x_j^+ - x_j^-,\ x_j^+ \geq 0, x_j^- \geq 0$.

For multi-objective optimization, one has to solve an auxiliary single-criterion problem whose solution provides a Pareto-efficient solution. Such a problem often has a minmax objective function, which cannot be handled by a standard LP solver. However, one can easily reformulate such a problem to the standard LP formulation. Suppose we have k linear objective functions $q_i = \boldsymbol{c}_i^T \boldsymbol{x},\ i = 1, \ldots, k$ and use the minmax aggregation of them (i.e., we want to minimize the maximal of these objective functions).[4] We can then introduce k additional constraints $\boldsymbol{c}_i^T \boldsymbol{x} \leq z,\ i = 1, \ldots, k$ and minimize the additional variable z as a single objective function. If we consider an augmented minmax aggregation by adding the sum of objectives with some coefficient to the maximum of objectives, then it is sufficient to add such a sum to the variable z. Similarly, as shown in Section 5.3, any piecewise linear and convex (concave in the case of maximization) objective function can be converted into linear programming form.

Although it is useful to know how to convert various objective functions into linear programming form, the conversion of all inequalities into equations or, more

[4]In Chapters 4 and 8, and also in other parts of this book, we use a maxmin aggregation when defining an achievement scalarizing function; in the minmax aggregation we simply change the signs of functions. However, we should warn the reader that the maxmin aggregation might result in weakly efficient solutions, unacceptable in practice. To avoid them, it is sufficient, for example, to add the sum of these objectives with a small coefficient to the minimum of these functions (see Chapters 4 and 8). Even good optimization systems, such as Optimization Toolkit in MATLAB (Part-Enander *et al.*, 1996), treat multi-objective optimization without the attention it deserves and instruct the user simply to apply minmax (or maxmin) aggregation without warning that this results in weakly efficient solutions. One can get even worse advice from some Web pages on optimization, where the suggested treatment of a multi-objective optimization problem is to convert it into a single-objective one by summing up the objectives with positive weighting coefficients, which, as shown in other chapters of this book, is possibly the worst method of treating multi-objective problems and leads the user to serious difficulties.

important, the assumption that all variable ranges will be represented by the simple inequality $\boldsymbol{x} \geq 0 \in \mathbb{R}^n$, is another matter. The transformations necessary to satisfy these assumptions result in a large number of slack variables and linear rescaling operations; as a result, the modeler might have considerable difficulties in recognizing and interpreting her/his original model. Thus, good linear programming solvers should accept models in a much more user-friendly format, such as:

$$X_0 = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{b} \leq \boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y} \leq \boldsymbol{b} + \boldsymbol{r} \in \mathbb{R}^m,\ \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}. \quad (7.3)$$

In this formulation, $\boldsymbol{x}$ is the vector of actual decision variables (a different and much less-dimensional vector than $\boldsymbol{x}$ in equation (7.2), as the latter also contains all slack and artificial variables), $\boldsymbol{y}$ is the vector of all intermediary and outcome variables in the linear model, and $\boldsymbol{W}$ might be a lower-triangular matrix (in the former case the model can be directly simulated without using an optimization solver; in the latter, optimization must be used in order to resolve model equations). The vectors $\boldsymbol{l}$ and $\boldsymbol{u}$ represent simple lower and upper bounds on the decision variables. The bounds $\boldsymbol{b}$ and $\boldsymbol{b} + \boldsymbol{r}$ correspond to outcome variables $\boldsymbol{y}$; the parameter $\boldsymbol{r}$ is called the range.

Most modern solvers accept such a format, although some tricks are often necessary to write the model from equation (7.3) in such a way that it is acceptable for a particular solver.[5] Typically, a linear programming solver would not allow for a free selection of objectives between the variables y_j, and thus for an interchange of objectives and constraints; they can be interchanged, but only by modifying the statement that defines the objective function for the solver.

Another highly desirable feature of an LP solver – from a practical point of view – is the handling of piece-wise linear functions. In many applications, a piece-wise linear approximation of a convex problem is solved as an LP problem. Forcing a user to perform a conversion of a problem with piece-wise linear functions to a standard LP problem has several disadvantages. First, it is an error-prone procedure even for experienced specialists in optimization. Second, it substantially increases the amount of data that has been generated and later analyzed by a solver. Third, it does not allow a solver to select the most efficient (for its optimization algorithm) conversion method. Finally, solvers that admit the definitions of piece-wise linear functions can typically find a better starting point for computations.

7.2.1 Two-phase simplex solvers

Most linear programming solvers implement the simplex algorithm that moves from one vertex of an n-dimensional polyhedral set defined by equation (7.2) in $\mathbb{R}^n_+$

[5]For example, in the GAMS modeling system, equation (7.3) can be represented while using special aliased sets in assignments (see Brooke *et al.*, 1992); there are no tricks required to represent equation (7.3) in the AMPL programming language (see Fourer *et al.*, 1993).

to another by changing the so-called basic solution while decreasing the minimized objective function. The basic solution corresponds to a partition of the matrix $\boldsymbol{A}$ into a nonsingular square basis matrix $\boldsymbol{B}$ and the remaining nonbasic part $\boldsymbol{N}$ (with possible reindexing of the columns of the matrix), $\boldsymbol{A} = [\boldsymbol{B},\ \boldsymbol{N}]$. The solution vector x is divided into two vectors: $\boldsymbol{x}_B$ and $\boldsymbol{x}_N$ (and the equation $\boldsymbol{Ax} = \boldsymbol{b}$ is represented by $\boldsymbol{Bx}_B + \boldsymbol{Nx}_N = \boldsymbol{b}$); the $\boldsymbol{x}_B$ and $\boldsymbol{x}_N$ are basic and nonbasic parts of the solution vector $\boldsymbol{x}$, respectively. The $\boldsymbol{x}_N$ vector has components equal to one of the bounds of the decision variable[6] (zero in the standard form equation (7.2), lower or upper bounds in other forms). The partition of the decision vector $\boldsymbol{x}^T = (\boldsymbol{x}_B^T,\ \boldsymbol{x}_N^T)$ defines the corresponding partition of the cost vector $\boldsymbol{c}^T = (\boldsymbol{c}_B^T,\ \boldsymbol{c}_N^T)$ into its basic and nonbasic parts.

The most typical implementation of a simplex solver for large-scale models is based on the revised simplex algorithm in a standard product form. The product form concerns the representation of the inverse of the basic matrix, $\boldsymbol{B}^{-1} = \boldsymbol{E}_l\boldsymbol{E}_{l-1}\dots\boldsymbol{E}_2\boldsymbol{E}_1$, where $\boldsymbol{E}_l$ is an elementary pivot matrix (a unit matrix with one column replaced by a vector η_l which defines the pivot (see equation (7.4)); in actual implementations, only the vectors η_l have to be stored (see Williams, 1993). The revised simplex algorithm computes first the basic solution $\boldsymbol{x}_B = \boldsymbol{B}^{-1}\boldsymbol{b}$, with $\boldsymbol{B}^{-1}$ in the above product form. Then the dual vector $\boldsymbol{\lambda}^T = \boldsymbol{c}_B^T\boldsymbol{B}^{-1}$ (the last also in the product form) is computed in order to define the vector of reduced cost coefficients for nonbasic variables $\boldsymbol{r}_N^T = \boldsymbol{c}_N - \boldsymbol{\lambda}^T\boldsymbol{N}$; the computation of all reduced cost coefficients might be costly, hence only a few of them might be determined in the initial stages of the simplex algorithm, depending on the so-called pricing strategy. If all determined reduced cost coefficients are nonnegative (for x_j on their lower bounds; nonpositive for x_j on upper bounds), further coefficients must be computed; if the entire vector has nonnegative coefficients, the current solution is optimal. Otherwise, one of the negative coefficients with a possibly large absolute value is selected; it determines which nonbasic variable, numbered by s, should enter the basis. Then the corresponding column $\boldsymbol{a}_s$ of the matrix $\boldsymbol{A}$ is transformed by $\boldsymbol{y}_s = \boldsymbol{B}^{-1}\boldsymbol{a}_s$ (again $\boldsymbol{B}^{-1}$ in the product form). If all components of the vector $\boldsymbol{y}_s$ are nonpositive, the optimization problem is unbounded. Otherwise, for all i such that $y_{is} > 0$, the ratios x_{Bi}/y_{is} are computed; the largest ratio determines which variable, numbered by r, should leave the basis. A new pivot element y_{rs} and the corresponding vector:

$$\eta_{l+1} = (-y_{1s}/y_{rs}, \dots, -y_{r-1,s}/y_{rs}, 1/y_{rs}, -y_{r+1,s}/y_{rs}, \dots, -y_{ms}/y_{rs})^T, \tag{7.4}$$

[6]This requirement may be dropped quite easily, as, e.g., in the noncommercial solver SALP of Świętanowski (1995).

as well as the elementary pivot matrix $\boldsymbol{E}_{l+1}$ (a $m \times m$ unit matrix with column number s replaced by the vector η_{l+1}), are thus determined. This completes one iteration of the simplex algorithm.

The above condensed description of one of the most basic optimization algorithms was presented only in order to illustrate the wealth of possible implementation details. Sparsity of matrices and vectors should be taken into account in order to effectively solve problems of very large dimensions; strategies for pricing, but also for testing the impact of round-off errors and restarting the algorithm in case these errors are too big, must be included. As a result, optimization solvers belong to the most complicated software packages. A considerable part of all computing time for bigger computers is spent on running optimization solvers, in particular, the simplex algorithm.

The simplex algorithm has several variants. The most often used variant of the simplex algorithm consists of finding a feasible basic solution in phase 1 (while a sum of model infeasibilities is minimized) and then optimizing the objective function in phase 2.

7.2.2 Big M simplex solvers

Another variant of the simplex algorithm consists in adding a penalty for infeasibilities to the objective function (or subtracting the penalty when maximizing). For historical reasons the penalty coefficient was often denoted by M. Using a capital letter was meant to express the fact that the penalty parameter M had to be a large positive constant (negative when maximizing). Hence the name "big M method".

This constitutes one of the basic forms of an exact penalty function: if the original problem has a bounded optimal solution, then a sufficiently big M exists such that the exact penalty function has its unconstrained minimum at the solution of the original problem. This form of the exact penalty function is a convex (concave when maximizing) piece-wise linear function and can be converted to the standard linear programming format by using a trick similar to that described previously for maxmin objectives.

The difficulty with the exact penalty method lies in choosing a good value of the penalty parameter M. If the penalty is too small the optimal solution of the penalized problem may not be a feasible solution of the original one. Furthermore, even a smaller penalty M can sometimes lead to an unbounded penalty problem. However, an M that is too large is likely to degrade the efficiency of the simplex algorithm. In the worst case it can lead to numerical difficulties and a possible breakdown of the simplex method. It is possible, however, to avoid both these pitfalls by employing a penalty update scheme proposed in Świętanowski (1995) and implemented in a linear optimizer SALP. The initial penalty is chosen to be small and it is increased only when required and only to be as large as necessary.

The advantage is that a warm restart of such a solver is usually much more efficient – thus it is a better option for all applications of linear programming that require many repetitive solutions (not only in multi-objective optimization, but also in hierarchically decomposed linear stochastic optimization problems or in integer programming problems). A noncommercial solver SALP using an adaptive adjustment of M was developed for such applications (see Świętanowski, 1995). For a standard NETLIB library of linear programming problems (see Gay, 1985) with traditional cold starting points, the results obtained with the experimental solver SALP were worse than, but comparable, to those obtained with the commercial solver CPLEX. However, SALP was devised especially for problems requiring many warm restarts, where it shows its advantages (see Ruszczyński and Świętanowski, 1996, for an application requiring numerous restarts). This solver is included in the software described in the Appendix. Recent versions of the commercial solver CPLEX also use a variant of the big M approach.

7.2.3 Dual simplex solvers

The dual simplex method is typically not used alone, but only as an enhancement of other simplex solvers; nevertheless, solvers that are capable of such enhancement can be called dual simplex solvers. Recall that the Lagrangian function for the standard formulation of a linear programming problem (equations 7.1 and 7.2) can be written in two equivalent forms:

$$\begin{aligned} L(\boldsymbol{\lambda}, \boldsymbol{x}) &= \boldsymbol{c}^T \boldsymbol{x} + \boldsymbol{\lambda}^T(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}) \\ &= \boldsymbol{b}^T \boldsymbol{\lambda} + \boldsymbol{x}^T(\boldsymbol{c} - \boldsymbol{A}^T\boldsymbol{\lambda}), \end{aligned} \tag{7.5}$$

where $\boldsymbol{\lambda}$ is a vector of Lagrange multipliers for the constraints $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. For convex and, in particular, for linear programming problems, the Lagrangian function has a saddle point at the optimal solution – a minimum in $\boldsymbol{x}$ and a maximum in $\boldsymbol{\lambda}$ or vice versa. Thus, instead of solving the primary linear programming problem defined by equations (7.1) and (7.2), we can solve the following dual linear programming problem:

$$\underset{\boldsymbol{\lambda} \in \Lambda_0}{\text{maximize}} \; (\; q^* = \boldsymbol{b}^T \boldsymbol{\lambda} \;) \tag{7.6}$$

with the set of dual feasible variables Λ_0 defined by:

$$\Lambda_0 = \{\boldsymbol{\lambda} \in \mathbb{R}^m : \boldsymbol{A}^T \boldsymbol{\lambda} \leq \boldsymbol{c} \in \mathbb{R}^n\}. \tag{7.7}$$

If the primal constraints are all equations, then the dual variables $\boldsymbol{\lambda}$ are unconstrained in their sign (if the primal inequality constraints $\boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}$ are considered instead, the dual variables would be all nonnegative). As mentioned previously, the dual variables are also computed in the primal simplex algorithm, as solutions of the equation $\boldsymbol{B}^T\boldsymbol{\lambda} = \boldsymbol{c}_B$. Similarly, when solving the dual problem (7.6) and (7.7) by a primal simplex algorithm, the primal variable $\boldsymbol{x}$ would be computed anyway. However, instead of solving the dual problem, a slightly different albeit equivalent approach is used as the so-called dual simplex algorithm: a pair of primal and dual solutions is found such that the primal might be infeasible, but it is dually feasible, i.e., conditions (7.7) are satisfied. A simplex iteration similar to that described above is then performed in such a way that the primal solution and the corresponding dual one are modified while the dual feasibility is maintained and the primal infeasibility is decreased. A solution that is both primal and dual feasible is optimal.

The dual simplex algorithm might be more effective than the primal one under certain conditions, and this can be automatically recognized. Some linear programming solvers, such as OSL, can switch automatically or semi-automatically between the primal and the dual simplex method. However, one should also consider the limitations discussed in Section 7.6, related to the nonuniqueness of dual solutions. Therefore, for many practical problems one should select a less effective primal algorithm and apply regularization, rather than use a more effective dual algorithm and have no control over which primal solution will be reported as the optimal one.

7.2.4 Interior point methods for linear programming

Any of the three main variants of the simplex algorithm (two-phase primal, big M primal, and dual) can theoretically go through all the vertices of the simplex before finding an optimal solution – and the number of these vertices increases exponentially with the dimensions $n,\ m$ of the problem (an upper bound on the number of simplex vertices is $\frac{(n+m)!}{n!m!}$ and examples of linear programming problems are known, for which the number of necessary simplex iterations is $2^n - 1$). On average, the number of simplex iterations needed to achieve optimality is not excessive; however, this tends to grow strongly for very large linear programming problems (with $10^4 \ldots 10^6$ variables and constraints). Therefore, starting with the so-called Khachian theorem,[7] nonlinear optimization techniques were adapted for large-scale

[7]In 1980, L. Khachian from the Computing Laboratory in Moscow, Russia, used a nonlinear optimization algorithm of N. Shor from Kiev, Ukraine, to prove that linear programming problems can be solved in a number of iterations that grow polynomially with the dimensions of the problem. This created a renewed interest in large-scale linear programming. In 1984, N. Karmarkar from Bell Laboratories in the USA proposed another, more practical, nonlinear programming approach to linear programming. Further research led to the so-called barrier and interior point methods. It is worthy of note, however, that the idea for using nonlinear optimization algorithms for linear programming problems originated earlier than with Khachian. In 1977, J. Sosnowski (Warsaw, Poland) had already

linear programming problems. Such nonlinear techniques applied to linear problems avoid having to go through many vertices of the simplex and go through the (relative) interior of the set of admissible solutions instead.

A class of linear programming algorithms of this type is known as interior point or barrier methods. In the following, we present a primal–dual interior point algorithm (purely primal or dual versions are not as efficient). The standard form of the linear programming problem is slightly modified by adding an upper bound $\boldsymbol{u}$ on decision variables, $0 \leq \boldsymbol{x} \leq \boldsymbol{u}$, and it is then expressed with the help of slack variables $\boldsymbol{s}$ by writing $\boldsymbol{x} + \boldsymbol{s} = \boldsymbol{u},\ \boldsymbol{x} \geq 0,\ \boldsymbol{s} \geq 0$. Thus, the standard linear programming problem becomes:

$$\underset{(\boldsymbol{x},\,\boldsymbol{s}) \in \mathcal{Z}_0}{\text{minimize}}\ (q = \boldsymbol{c}^T \boldsymbol{x}) \tag{7.8}$$

where

$$\begin{aligned}\mathcal{Z}_0 = \{\boldsymbol{x} \in \mathbb{R}^n,\ \boldsymbol{s} \in \mathbb{R}^n :\ \boldsymbol{A}\boldsymbol{x} &= \boldsymbol{b} \in \mathbb{R}^m;\ \boldsymbol{x} + \boldsymbol{s} = \boldsymbol{u} \in \mathbb{R}^n,\\ \boldsymbol{x} &\geq 0 \in \mathbb{R}^n,\ \boldsymbol{s} \geq 0 \in \mathbb{R}^n\}.\end{aligned} \tag{7.9}$$

The dual to this problem takes the form:

$$\underset{(\boldsymbol{\lambda},\boldsymbol{\psi},\boldsymbol{\omega}) \in \mathcal{Q}_0}{\text{maximize}}\ (q^* = \boldsymbol{b}^T \boldsymbol{\lambda} - \boldsymbol{u}^T \boldsymbol{\psi}) \tag{7.10}$$

where

$$\begin{aligned}\mathcal{Q}_0 = \{\boldsymbol{\lambda} \in \mathbb{R}^m,\ \boldsymbol{\psi} &\in \mathbb{R}^n,\ \boldsymbol{\omega} \in \mathbb{R}^n :\ \boldsymbol{A}^T \boldsymbol{\lambda} + \boldsymbol{\omega} - \boldsymbol{\psi} = \boldsymbol{c} \in \mathbb{R}^n;\\ \boldsymbol{\psi} \geq 0 &\in \mathbb{R}^n,\ \boldsymbol{\omega} \geq 0 \in \mathbb{R}^n\}.\end{aligned} \tag{7.11}$$

In the interior point method, the nonnegativity conditions for $\boldsymbol{x}$ and $\boldsymbol{s}$ in the primal formulation are replaced by logarithmic barrier functions. In the case of minimization, the corresponding penalty function takes the form:

$$F(\boldsymbol{x}, \boldsymbol{s}, \mu) = \boldsymbol{c}^T \boldsymbol{x} - \mu \sum_{j=1}^{n} ln\, x_j - \mu \sum_{j=1}^{n} ln\, s_j, \tag{7.12}$$

where $\mu > 0$ is the barrier parameter (when $\mu \to 0$, the maximal points of (7.12) subject to the constraint $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ converge to the solution of the primal problem).

used a special nonlinear optimization algorithm for linear programming problems; this algorithm is included in the HYBRID solver described briefly in Section 7.2.5 of this chapter. Moreover, this algorithm used earlier ideas of O.L. Mangasarian and R.T. Rockafellar (Seattle and Madison, USA, respectively); in 1976 Rockafellar had already proven finite convergence of an augmented Lagrangian multiplier method when applied to piece-wise linear programming problems (which could also be used to obtain a polynomial estimate of the number of iterations needed).

The optimality conditions for the penalty function, after some transformations, can be written in the form of the following set of equations:

$$\begin{aligned} \boldsymbol{Ax} &= \boldsymbol{b} \\ \boldsymbol{x}+\boldsymbol{s} &= \boldsymbol{u} \\ \boldsymbol{A}^T\boldsymbol{\lambda}+\boldsymbol{\omega}-\boldsymbol{\psi} &= \boldsymbol{c} \\ X\boldsymbol{\Omega e} &= \mu\boldsymbol{e} \\ S\boldsymbol{\Psi e} &= \mu\boldsymbol{e}, \end{aligned} \tag{7.13}$$

where $X,\ S,\ \boldsymbol{\Omega}$, and $\boldsymbol{\Psi}$ are diagonal matrices with the elements $x_j,\ s_j,\ \omega_j$, and ψ_j, respectively, and $\boldsymbol{e} \in \boldsymbol{R}^n$ is a vector composed of all elements equal to one. The last two equations in (7.13) are only a shorthand way for saying that $\omega_j = \mu/x_j$ and $\psi_j = \mu/s_j\ \forall j = 1, \ldots n$; however, these equations are nonlinear (in fact, bilinear) while the three equations preceding them are all linear.

A basic primal–dual interior point algorithm makes a step in Newton's direction $\Delta = (\Delta\boldsymbol{x},\ \Delta\boldsymbol{s},\ \Delta\boldsymbol{\lambda},\ \Delta\boldsymbol{\omega},\ \Delta\boldsymbol{\psi})$, obtained by a linearization of the set of equations (7.13):

$$\begin{bmatrix} A & 0 & 0 & 0 & 0 \\ I & I & 0 & 0 & 0 \\ 0 & 0 & A^T & I & -I \\ \boldsymbol{\Omega} & 0 & 0 & X & 0 \\ 0 & \boldsymbol{\Psi} & 0 & 0 & S \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta s \\ \Delta\boldsymbol{\lambda} \\ \Delta\boldsymbol{\omega} \\ \Delta\boldsymbol{\psi} \end{bmatrix} = \begin{bmatrix} \xi_b \\ \xi_u \\ \xi_c \\ \xi_{\mu 1} \\ \xi_{\mu 2} \end{bmatrix} \tag{7.14}$$

where

$$\begin{aligned} \xi_b &= \boldsymbol{b}-\boldsymbol{Ax} \\ \xi_u &= \boldsymbol{u}-\boldsymbol{x}-\boldsymbol{s} \\ \xi_c &= \boldsymbol{c}-\boldsymbol{A}^T\boldsymbol{\lambda}+\boldsymbol{\psi}-\boldsymbol{\omega} \\ \xi_{\mu 1} &= \mu\boldsymbol{e}-X\boldsymbol{\Omega e} \\ \xi_{\mu 2} &= \mu\boldsymbol{e}-S\boldsymbol{\Psi e}. \end{aligned} \tag{7.15}$$

This set of linearized equations can be reduced to equations only with respect to $\Delta\boldsymbol{x}$ and $\Delta\boldsymbol{\lambda}$ (a so-called augmented system) or to an equation only with respect to $\Delta\boldsymbol{\lambda}$. However, the solution of even the most reduced equation, performed usually by sparse factorization techniques, requires major computational effort in the case of very large m and n. After the determination of the full direction Δ, steps with various step-size coefficients can be made for the primal and dual components of this direction, because the objective of such steps is to diminish equally the primal infeasibility, $g_p = \parallel \boldsymbol{Ax}-\boldsymbol{b} \parallel$, the dual infeasibility, $g_d = \parallel \boldsymbol{c}-\boldsymbol{A}^T\boldsymbol{\lambda}+\boldsymbol{\psi}-\boldsymbol{\omega} \parallel$, and the complementarity gap, $g_c = \boldsymbol{x}^T\boldsymbol{\omega}+\boldsymbol{s}^T\boldsymbol{\psi}$.

Variants of the primal–dual interior point algorithm are possible, depending on possible corrections to the Newton direction. In order to diminish the computational effort necessary for computing this direction, it is possible to decompose this direction into two parts, $\Delta = \Delta_a + \Delta_c$. The term Δ_a (affine scaling) might be obtained by solving (7.14) with the last two components $\xi_{\mu 1},\ \xi_{\mu 2}$ put to zero, and the term Δ_c (centering) – with the first three components $\xi_b,\ \xi_u,\ \xi_c$ put to zero. Such decomposition, although it does not produce a precise Newton's direction, helps to formulate a reasonable value of the barrier parameter μ (see Mehrotra, 1992) and thus offers, on average, a considerable reduction in the number of interior point iterations. When observing that the major difficulty in solving equations (7.13) is related to the last two bilinear sets of equations, other methods of defining the centering term Δ_c can be proposed. These take into account the particular form of the last two equations and combine it with the use of multiple centrality correctors (Gondzio, 1996a).

A good selection of such heuristics for the definition of the centering term results in a reasonably efficient experimental higher order primal–dual method (HOPDM) solver by Gondzio (1995b). This outperforms standard simplex algorithms (e.g., CPLEX),[8] from 7 to 20 times on practical large-scale linear programming problems (taken from IIASA energy studies, see Chapter 14) of about 60,000 variables and constraints. This reduction in computing time – often from more than 10 hours to half an hour – makes it possible to solve large optimization problems repetitively and thus to analyze models much more thoroughly. A version of a HOPDM solver by Gondzio and Makowski (1995a) is included in the software described in the Appendix.

7.2.5 Special LP solvers

Various software products have been developed for multi-objective optimization (e.g., Steuer, 1986); most of them rely on the conversion of the corresponding optimization problem to the standard linear programming form and then on using typical simplex implementations. For example, a relatively simple software package DIDAS-L, used for multi-objective analysis of linear models (good for didactic purposes and included in the software library attached to this book), has a simplex solver specially adapted to the multi-objective case.

However, the conversion of a multi-objective linear model into the standard linear programming format often results in an addition of a large number of slack and artificial variables. In the standard linear programming form (7.2) – or even (7.9), thus this argument applies to interior point methods as well – there are always more variables than constraints, while the number of original variables in the

[8]In its classical simplex version; recently CPLEX also added a barrier method version.

nonstandard, user-friendly model form (7.3) might often be much smaller than the number of constraints. This happens particularly in multi-objective dynamic, or otherwise complicated, linear programming models.

A special nonsimplex solver for such linear programming problems, called HYBRID, has been developed based on a nonlinear programming approach, which is different from those used in the interior point algorithms.[9]

The solver HYBRID accepts linear models in their *native* form (7.3). LP problems solved in HYBRID have the standard (7.1) form of the objective function:

$$\underset{\boldsymbol{x}\in X_0}{\text{minimize}}\ (q = \boldsymbol{c}^T\boldsymbol{x})$$

but the internal representation of the set X_0 has a nonstandard form, e.g.,

$$X_0 = \{\boldsymbol{x} \in I\!\!R^n : \boldsymbol{A}_1\boldsymbol{x} = \boldsymbol{b}_1;\ \boldsymbol{b}_3 \le \boldsymbol{A}_2\boldsymbol{x} \le \boldsymbol{b}_2;\ \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u}\}, \tag{7.16}$$

where some elements of $\boldsymbol{b}_3$ and $\boldsymbol{l}$ are equal to $-\infty$, and some elements of $\boldsymbol{b}_2$ and $\boldsymbol{u}$ are equal to ∞. The data structure and the optimization algorithm do not use slack variables because the optimization algorithm does not require actual conversion of constraints to equality constraints.

To solve this problem, a proximal multiplier method is used. Let $\|\boldsymbol{x}\|$ denote the l_2–norm of $\boldsymbol{x}$ and let $(\boldsymbol{y})_+$ denote the vector composed of components $\max(0, y_i)$. We define the augmented Lagrangian function (actually a shifted penalty function)[10] as:

$$\begin{aligned}\Phi_\rho(\boldsymbol{\lambda},\boldsymbol{x}) &= (1/\rho)\boldsymbol{c}^T\boldsymbol{x} + \frac{1}{2}\|(1/\rho)\boldsymbol{\lambda}_1 + \boldsymbol{A}_1\boldsymbol{x} - \boldsymbol{b}_1\|^2 \\ &+ \frac{1}{2}\|((1/\rho)\boldsymbol{\lambda}_2 + \boldsymbol{A}_2\boldsymbol{x} - \boldsymbol{b}_2)_+\|^2 \\ &+ \frac{1}{2}\|((1/\rho)\boldsymbol{\lambda}_3 + \boldsymbol{b}_3 - \boldsymbol{A}_2\boldsymbol{x})_+\|^2, \end{aligned} \tag{7.17}$$

[9]The basic idea for this algorithm was presented in 1977 (Sosnowski, 1981); hence, it is probably one of the oldest nonlinear programming algorithms applied to linear programming. A theoretical basis of this algorithm using an augmented Lagrangian approach, including the proof of finite convergence of a variant of it applied to piece-wise linear programming problems, is attributed to Rockafellar (1976) and to earlier ideas of Mangasarian (summarized later in Mangasarian, 1981). The basic idea of regularization of optimization problems as used in this algorithm is attributed even earlier to Polyak and Tretiyakov (1972), who in turn exploited the general, earlier concept of Tikchonov regularization. The basic idea of an augmented Lagrangian multiplier iteration also relies on earlier work on an iteratively shifted penalty function (see Powell, 1969, for equality constraints and Wierzbicki, 1971, for the case of inequality constraints).

[10]The parameter $\rho > 0$ is a penalty coefficient. The terms $(1/\rho)\boldsymbol{\lambda}_1,\ (1/\rho)\boldsymbol{\lambda}_2$ in this function can be interpreted as penalty shifts (see Powell, 1969; Wierzbicki, 1971). An augmented Lagrangian function would be obtained as $L_\rho(\boldsymbol{\lambda},\boldsymbol{x}) = \Phi_\rho(\boldsymbol{\lambda},\boldsymbol{x}) - \frac{1}{2}(\|\boldsymbol{\lambda}_1/\rho\|^2 + \|\boldsymbol{\lambda}_2/\rho\|^2 + \|\boldsymbol{\lambda}_3/\rho\|^2)$.

where the last two terms are alternative (only one of them can be different from zero). For the sake of simplifying the presentation of the algorithm, we assume that all components of $\boldsymbol{b}_3$ are equal to $-\infty$. Therefore the $\boldsymbol{\lambda}_3$ multipliers are not included in the remaining part of the presentation of the algorithm (in the actual implementation of this algorithm, $\boldsymbol{\lambda}_2$ and $\boldsymbol{\lambda}_3$ are handled by one variable).

The proximal multiplier method consists of successive iterations of primal and dual solutions $\boldsymbol{x}^{(l)},\ \boldsymbol{\lambda}_1^{(l)},\ \boldsymbol{\lambda}_2^{(l)}$, with any starting $\boldsymbol{x}^{(0)}$ satisfying the simple bounds $\boldsymbol{l} \leq \boldsymbol{x}^{(0)} \leq \boldsymbol{u}$ – hence a warm restart is easy – and $\boldsymbol{\lambda}_1^{(0)} = 0,\ \boldsymbol{\lambda}_2^{(0)} = 0$. The subsequent iterations of this method are defined by minimizing the shifted penalty (augmented Lagrangian) function with a regularizing proximal component added:

$$\boldsymbol{x}^{(l+1)} = \underset{\boldsymbol{l}\leq\boldsymbol{x}\leq\boldsymbol{u}}{\operatorname{argmin}}\ (\Phi(\boldsymbol{\lambda}^{(l)}, \boldsymbol{x}) + \frac{1}{2\rho\gamma^{(l)}}\|\boldsymbol{x} - \boldsymbol{x}^{(l)}\|^2), \tag{7.18}$$

where $\gamma^{(l)} > 0$. The multipliers or dual variables are then modified similarly as for shifted penalty iteration (see Powell, 1969; Wierzbicki, 1971):

$$\begin{aligned} \boldsymbol{\lambda}_1^{(l+1)} &= \boldsymbol{\lambda}_1^{(l)} + \rho(\boldsymbol{A}_1\boldsymbol{x}^{(l+1)} - \boldsymbol{b}_1) \\ \boldsymbol{\lambda}_2^{(l+1)} &= (\boldsymbol{\lambda}_2^{(l)} + \rho(\boldsymbol{A}_2\boldsymbol{x}^{(l+1)} - \boldsymbol{b}_2))_+ \end{aligned} \tag{7.19}$$

until a stopping criterion – involving both primal and dual variables – is satisfied.

The minimized function (7.18) is strongly convex and has a unique minimizer. The regularizing proximal term helps in all cases when the original problem is ill-conditioned (e.g., when some of the linear constraints are almost co-linear). When using this term, we lose the theoretical property of finite convergence (possessed by a multiplier method without this term), but the method behaves better in practice.

The crucial point for the effectiveness of this method is the choice of an algorithm to minimize the piece-wise quadratic function (7.18). An ordinary or preconditioned conjugate gradient algorithm with projections on box-like simple bounds has been used for this purpose in older HYBRID variants (Makowski and Sosnowski, 1989). Theoretically, such an algorithm guarantees that the exact minimizer will be found after a finite number of iterations. However, round-off errors during computations often cause numerical problems for models of large dimensions. Sosnowski (1990) therefore proposed a numerically stable method for minimizing piece-wise quadratic strongly convex functions with box-like constraints. This method combines an active set algorithm with QR factorization. As an option, two implementations of Cholesky factorization are used in current implementations of the HYBRID package; one is a dense factorization with updates, the second is sparse factorization, adapted from the HOPDM solver (Gondzio, 1995; see also Makowski and Sosnowski, 1989).

The HYBRID package is well-tested and robust (the latter is a general property of methods based on shifted penalty iterations). For medium-sized problems, where the number of constraints exceeds the number of variables, it is faster than even the interior point HOPDM solver. The HOPDM solver, however, is better for problems already converted to a standard form with slack variables (such problems might be difficult to be converted back to their original form). The main advantages of HYBRID are the regularization of ill-conditioned problems and the warm start. Moreover, HYBRID was specifically developed to solve problems resulting from the scalarization of multi-objective (and also dynamic) linear programming problems approached by the reference point methodology. Some examples of applications of HYBRID are given in the next chapters. The HYBRID solver is included in the software library attached to this book; it can be used for any (also single-objective) linear programming problem, but is especially effective for problems of the nonstandard, user-friendly format (7.3) or (7.16), or for multi-objective problems.

7.3 Mixed Integer Programming Solvers

7.3.1 Models with integer variables

A mixed integer linear programming (MIP) problem is a linear problem, similar to (7.1):

$$\underset{\boldsymbol{x} \in X_0}{\text{minimize}} \; (q = \boldsymbol{c}^T \boldsymbol{x}),$$

however, it has two kinds of variables: integer variables and continuous variables. Integer variables can only take integer values (in the final solution, not in the solution process), whereas continuous variables can take any real number as a value. Thus, the set X_0 is defined as a specification of the standard formulation:

$$X_0 \;=\; \{\boldsymbol{x} \in I\!\!R^n : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \in I\!\!R^m, \; \boldsymbol{x} \geq 0 \in I\!\!R^n, \; x_j \text{ – integer for } j \in I_0\}, \tag{7.20}$$

where $I_0 \subset \{1, \ldots, n\}$ is the set of integer variables.[11] Alternatively, the set X_0 can be defined in a nonstandard but more user-friendly way:

[11] One can easily convert integer variables into binary variables; however, such conversion might have value only for purely integer or binary problems with $I_0 = \{1, \ldots, n\}$. A purely integer linear programming problem can be considered as a marginal case of a MIP problem and may be solved with the same software; however, specialized algorithms are usually more efficient for these types of problems.

$$X_0 = \{\boldsymbol{x} \in I\!R^n : \quad \boldsymbol{b} \leq \boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y} \leq \boldsymbol{b} + \boldsymbol{r} \in I\!R^m, \ \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u} \in I\!R^n;$$
$$x_j \text{ – integer for } j \in I_0\}. \tag{7.21}$$

In many models, some of the variables represent entities that cannot be partitioned. Many logical relations can be formulated as linear relations with integer or binary variables. Moreover, many nonlinear and nonconvex models can be reformulated as linear programming problems with integer variables (see e.g., Williams, 1991). The option of introducing integer variables into linear programming models therefore makes it possible to analyze many important problems that are not covered by classical linear programming.

Practical analysis of MIP models sometimes results in their multi-objective formulation: any of the outcome variables y_i in equation (7.21) might be chosen as one of several objective functions. The analysis of such problems by a simplistic approach (e.g., by summing up objective functions with positive weighting coefficients) leads immediately into serious difficulties. As a MIP problem is, by definition, not convex, efficient points might exist (and often do in practical problems) that cannot be found by maximizing the weighted sum. However, when using the reference point approach and the augmented maxmin aggregation of objectives, we can transform (as indicated in Section 7.2) the aggregated objective into a function of the form $\boldsymbol{c}^T\boldsymbol{x}$ by adding some artificial variables and changing the set X_0. Thus, the reference point approach – which results in a full characterization of efficient solutions even for nonconvex problems – is ideal for multi-objective MIP problems. It is ideal because it is based on the solution of slightly modified single-objective MIP problems, which allows for finding any efficient solution.

7.3.2 Standard solution procedures for MIP problems

The efficiency of a solution procedure for MIP problems strongly depends on the tightness of linear constraints on integer variables. For example, the set of constraints

$$x_1 + x_2 \leq 1, \quad 0 \leq x_1 \leq 1, \quad 0 \leq x_2 \leq 1, \quad x_1, x_2 \text{ are integers}$$

defines the same integer solutions as the set of constraints

$$0.8x_1 + 0.6x_2 \leq 1.3, \quad 0 \leq x_1 \leq 1, \quad 0 \leq x_2 \leq 1, \quad x_1, x_2 \text{ are integers.}$$

The former provides, however, tighter linear constraints on integer variables than the latter. If the integrality requirements are dropped, the former set of constraints defines the convex hull of integer solutions, whereas the latter defines a larger set. In order to tighten the constraints, additional constraints called cuts are often added to MIP problems. For a more detailed discussion of building tight models with integer variables (see Williams, 1991).

Another aspect that strongly influences the efficiency of a solution procedure is the order in which integer variables are processed during the search for an integer solution. In some solvers, this order depends on the original order of integer variables in the problem. Therefore, it is advisable to introduce integer variables in decreasing order of importance during the formulation of the model or to define appropriate priorities for integer variables.

All commercial and most experimental MIP solvers today are based on a general technique for this class of problems called branch-and-bound (Saltzman, 1994). However, a wide variety of additional techniques has been applied to minimize the total effort involved in the branch-and-bound technique.

The branch-and-bound technique solves the MIP problem by successive optimizations of linear programming problems. It is assumed that the continuous problem (i.e., the MIP problem without integrality requirements) has been solved first. If all integer variables have integer values in the optimal solution to the continuous problem, there is nothing more to do. Suppose that an integer variable, say x_r, has a fractional (noninteger) continuous optimum value $\hat{x}_r$. If $[\hat{x}_r]$ is the integer part of $\hat{x}_r$, then the range:

$$[\hat{x}_r] < x_r < [\hat{x}_r] + 1 \tag{7.22}$$

cannot include any integer solution. Hence, an integer value of x_r must satisfy one of two inequalities:

$$x_r \leq [\hat{x}_r] \quad \text{or} \quad x_r \geq [\hat{x}_r] + 1. \tag{7.23}$$

These two inequalities, when applied to the continuous problem, result in two mutually exclusive linear problems created by imposing the additional constraints $x_r \leq [\hat{x}_r]$ or $x_r \geq [\hat{x}_r] + 1$, respectively, on the original feasible region. This process is called branching and the integer variable x_r is called the branching variable.

As a result of branching, the original problem is partitioned into two subproblems. Each subproblem might be solved as a continuous problem. This can be done in an efficient way when applying the dual simplex algorithm. In particular, if an upper bound on the minimized value of an objective function is already known from solving previous subproblems, the dual simplex algorithm (which uses primal infeasible basic solutions and thus, when solving a minimization problem, increases the objective function value in subsequent iterations) can be stopped and the corresponding subproblem discarded as soon as the objective function value increases above this bound.

If an integer variable fails the integrality requirement in the optimal solution of a subproblem, the branching process may be applied on the subproblem, thus creating a tree of subproblems. Due to this structure, the subproblems might be called nodes (of the subproblems' tree). The original continuous problem is assumed to be node 0 (root of the tree) and the other nodes get subsequent numbers when created.

A node does not need to be further branched if its optimal (continuous) solution satisfies all the integrality requirements. This node, called an integer node, is dropped from the subsequent search while its solution is stored as the best integer solution so far available and its objective value becomes the cutoff value (upper bound when minimizing). A node may also be dropped from further analysis if it is fathomed, i.e., there is evidence (e.g., resulting from the application of the dual simplex algorithm as commented above) that it cannot yield a better integer solution than that available so far. A node is fathomed if it is infeasible and therefore cannot yield any solution. Because a node optimal value is a (lower, when minimizing) bound on the best integer solution value that can be obtained from the node, nodes with noninteger optimal solutions may be fathomed by comparison of their optimal (continuous) value versus the current cutoff value.

It is important to acquire good bounds to fathom nodes at early stages of the search process. Therefore, in advanced implementations of the branch-and-bound techniques, additional penalties are used in fathoming tests. The general idea of the penalties is to estimate the deterioration in the objective value caused by enforcing additional inequalities in branching.

While making the branch-and-bound technique operational, it is necessary to introduce some order in the branching and solving of nodes. For this purpose, the so-called waiting list is usually introduced; it contains all nodes in need of further analysis. The list can be arranged in two ways. If constructed but unsolved nodes are stored on the waiting list, we obtain so-called single branching, where a node selected from the list is solved first and branched next if not fathomed. If solved nodes are stored on the list, we have so-called double branching, where a node selected from the list is branched first and then both the new subproblems are solved and stored on the list if not fathomed. For larger problems, double branching is recommended.

The process of branching continues, where applicable, until each node terminates either by generating an integer solution, or by being fathomed. Thus the branch-and-bound search is completed when the waiting list becomes empty. Three phases can be distinguished during the course of the branch-and-bound search: a search for the first integer solution; a search for the best integer solution; and optimality proof. Computational experiments show that the first two phases are usually completed in a relatively short period of time for typical MIP problems (only a little longer than the time required for a continuous problem solution), whereas the last phase may require much time. The end of the first phase is clearly defined (the first integer solution has been found); however, and unfortunately, the end of the second phase and the beginning of the optimality proof phase is never defined completely until the entire search is completed (because the essence of the optimality proof is testing whether a solution can be found that is better than the last candidate).

Having defined the waiting list, there are still many ways to make the search operational. Most important for the algorithm specification are two operations: the selection of a node for branching and the selection of a branching variable. Both these operations might be arranged in many different ways resulting in various tree sizes and search efficiency. A specification of these two selection operations – known as the branch-and-bound strategy – is crucial for algorithm efficiency on a specific MIP problem. Unfortunately, there is no best strategy (no universal heuristics) for all problems. Therefore, most advanced MIP solvers allow the user to adjust the strategy to the specificity of the MIP problem.

7.3.3 Commercial versus experimental MIP solvers

There are many good commercial MIP solvers; most commercial linear programming solvers have their MIP versions. For example, CPLEX has a special version for MIP problems that can be used either separately or under a GAMS modeling system. The AIMMS modeling system is equipped with a LAMPS solver (see Bisschop and Entriken, 1993, for a description of how the user can influence the branch-and-bound strategy in LAMPS). The AMPL modeling system can optimally be equipped with an OSL solver. Commercial solvers are usually well tested and robust.

However, many good commercial MIP solvers require the use of a mainframe computer. Moreover, because there is no universal, optimal branch-and-bound strategy, experimental solvers might either be better or easier to fine-tune for a specific class of problems. Therefore, it is advisable to use both commercial and experimental solvers and compare the results. In the next section, we present an example of an experimental solver, called MOMIP.

7.3.4 MOMIP solver for multi-objective MIP problems

MOMIP (described by Ogryczak and Zorychta, 1996) is an optimization solver for medium-sized mixed integer linear programming problems, based on the branch-and-bound algorithm. It is designed to take into account single-objective reformulations of multi-objective MIP problems. MOMIP is characterized by the following features:

- Double branching and priorities for branching a variable selection.
- Strengthened penalties on branching variables and special node selection strategies.
- Additional cuts generation.
- Processing and scanning of special ordered sets.

Moreover, MOMIP has various specific features such as a rich list of control parameters that allow the default branch-and-bound strategy provided in MOMIP to be changed for typical applications – for example, to abandon the search if it already seems to be long in a optimality proof phase. However, we briefly comment on the main features of MOMIP.

Branching Strategy

Double branching is implemented in MOMIP because it is more efficient for large problems. A branching variable is selected depending on the predefined order of priorities for variables and on a measure of integer infeasibility of variable values at the optimal solution. A variable value is considered to be integer infeasible (fractional) if it differs from the closest integer by more than an accuracy parameter, say δ. If $f_r = \hat{x}_r - [\hat{x}_r]$ denotes the fractional part of a variable x_r that is supposed to be an integer, then this variable is integer infeasible if:

$$\min(f_r,\ 1 - f_r) > \delta. \tag{7.24}$$

The value $\min(f_r,\ 1 - f_r)$ is called integer infeasibility of variable x_r. A branching variable is selected among integer infeasible variables with the highest priority. In the default strategy, all integer variables have the same priority equal to 0, but the user may specify higher priorities for some variables. In the default strategy, the variable with minimal integer infeasibility (i.e., the variable closest to an integer but not closer than δ) is selected as a branching variable until the first integer solution is found; later, variables with larger integer infeasibility are selected. The user can force MOMIP to apply another rule – say, the variable with maximal integer infeasibility first – by specifying an appropriate control parameter. The minimum integer infeasibility selection rule may lead more quickly to a good first integer solution (because it works like a rounding heuristic) but may be slower when completing the entire branch-and-bound process. The maximum integer infeasibility rule forces larger changes earlier in the tree, which usually tends to produce quicker overall times to find and prove the optimal integer solution.

Node Selection

Nodes are optimized in MOMIP with the dual simplex algorithm. Optimization is abandoned if it becomes clear during the course of the algorithm that the node cannot have a better optimal value than the current cutoff value (thereby it will be fathomed). When a noninteger optimal solution is found, a potential branching variable is selected and the corresponding penalties calculated. As appropriate penalties, so-called strengthened SUB and Gomory penalties based on the Lagrangian relaxation (Zorychta and Ogryczak, 1981) are implemented. If the penalties allow the fathoming of both potential subproblems, the optimized node is also fathomed. If the penalties allow the fathoming of one of the potential subproblems, the constraints of the optimized node are tightened to the second subproblem and the optimization process is continued without explicit branching. Thus a noninteger node is added to the waiting list only if both its potential subproblems cannot be fathomed by the penalties.

MOMIP uses two basic and one mixed node selection rules:

- LIFO (last in first out) is the selection of the latest generated node. The LIFO rule is used as the default strategy in MOMIP.
- BEST is the selection of the best node (node with the best value bound). The user can force MOMIP to use the BEST rule in one or in all search phases, by specifying a control parameter.
- MIXED applies the LIFO rule until the first integer solution is found and then later uses the BEST rule. MIXED can be selected by specifying a control parameter.

The LIFO rule implies that if the branched node has at least one subproblem to be optimized, then one of these subproblems (the one with the better value bound, if there are two) will be selected. If both the subproblems are fathomed or integer, the latest node added to the waiting list is selected. Thus, with the LIFO rule, the waiting list works like a stack. The LIFO rule implies a narrow in-depth tree analysis with a small waiting list. It is a very efficient node selection strategy to use while looking for the first integer solution. In the MOMIP default strategy, it works together with the minimal integer infeasibility branching rule, thus creating a heuristic search for an integer solution close to the continuous one. Both basic node selection rules are implemented in MOMIP as parameterized strategies to prevent an uncontrolled growth of the waiting list.

When the selected node is branched, two of its subproblems have to be optimized. The order of these optimizations can affect the efficiency of the algorithm in two ways. First, if the subproblem optimized as the second one is later selected for branching, then the optimization process can be continued without any restore and refactorization operations. Therefore, we are interested later in optimizing the subproblem that seems more likely to be selected for future branching. Moreover, if an integer solution is found while optimizing the first subproblem, then it speeds up the fathoming of the second one, which might make its optimization short or unnecessary. In MOMIP the subproblem associated with larger integer infeasibility on the branching variable is usually optimized as the first, presuming that the second one will have a better value bound and therefore will be selected for future branching. An exception to this rule is the so-called quasi-integer node, defined as all integer variables that have values relatively close to integers, i.e., all integer infeasibilities are smaller than a specified parameter. In this case, the subproblem associated with a smaller integer infeasibility on the branching variable is optimized first, in order (and with the hope) of setting an integer solution more quickly.

Cuts

The efficiency of a branch-and-bound algorithm greatly depends on the tightness of linear constraints on integer variables. MOMIP allows linear constraints to be tightened by generating additional inequalities (cuts) that are satisfied by all integer solutions but are not satisfied by the optimal solution to the continuous problem. Two types of cuts may be generated in MOMIP as additional constraints for all nodes; the type is selected by specifying a control parameter. In addition, the user may specify the required number of cuts to be generated and added to the problem. MOMIP reoptimizes the continuous problem (with the dual simplex algorithm, which is fast for problems with added constraints) after generating each cut prior to the generation of the next one.

Let $\hat{\boldsymbol{x}}$ be an optimal basic solution to the current relaxation of the MIP problem. An additional constraint is called a cut at $\hat{\boldsymbol{x}}$ if it is satisfied by all feasible integer solutions but is not satisfied by $\hat{\boldsymbol{x}}$. If a cut is introduced during the search for an integer solution, the augmented continuous problem becomes tighter on integer variables and yields a tighter bound on the objective value. In particular, introducing more cuts might result in a shorter optimality proof. MOMIP uses cuts of two types: (i) Gomory u-type mixed integer cuts (Nemhauser and Wolsey, 1988) are used as the default strategy; and (ii) Balas-type cuts are used for mixed binary problems (Balas *et al.*, 1993) if the user specifies such an option.

To define a Gomory cut, the row of the simplex tableau corresponding to a noninteger x_j has to be available. The coefficients of the cut are then computed as simple functions of fractional parts of coefficients in this row. A Balas cut can be applied for mixed binary problems (with 0–1 variables) only; it is efficient in such a case but more complicated and time-consuming. A single Gomory or Balas cut corresponds to a noninteger variable x_j. While selecting various variables, different cuts can be generated. MOMIP selects the noninteger variable with the largest integer infeasibility to generate the cut. The cuts generation process is abandoned if the largest integer infeasibility is less than the quasi-integrality tolerance. Moreover, cuts tighten the linear constraints, but they also increase the density of the coefficients matrix. Therefore, too many cuts might result in denser subproblems and in an efficiency decrease; thus the number of cuts is limited by a control parameter in MOMIP.

Special Ordered Sets

The majority of real-life mixed integer programming models represents multiple choice requirements (Healy, 1964). A multiple choice requirement is usually modeled with a generalized upper bound on a set of zero–one variables (Nemhauser and Wolsey, 1988; Williams, 1991) thus creating a so-called special ordered set (SOS). For instance, the multiple choice requirement:

$$z \in \{a_1,\, a_2,\, \ldots,\, a_r\} \tag{7.25}$$

where a_j represent several options (such as facility capacities), may be modeled as follows:

$$\begin{aligned} a_1x_1 + a_2x_2 + \cdots + a_rx_r &= z \\ x_1 + x_2 + \cdots + x_r &= 1 \\ x_j \geq 0, \quad x_j \text{ integer for } j &= 1, 2, \ldots, r \end{aligned} \tag{7.26}$$

where x_j are zero–one variables corresponding to the choice of the option a_j. Problems with the SOS structure might be solved by using the standard branch-and-bound algorithm for mixed integer programming. However, the standard branching rule:

$$x_k = 0 \quad \text{or} \quad x_k = 1 \tag{7.27}$$

applied on a SOS variable leads to the dichotomy

$$x_1 + x_2 + \cdots + x_{k-1} + x_{k+1} + \cdots + x_r = 1 \quad \text{or} \quad x_k = 1, \tag{7.28}$$

thus creating a very unbalanced branching on the set of the original alternatives (any option different from a_k is selected or option a_k is selected). It causes low effectiveness of the branch-and-bound algorithm. As a result, many special techniques for SOS branching have been developed (see Beale and Tomlin, 1970; Tomlin, 1970;

Beale, 1979; Land and Powell, 1979; Powell, 1985; Tomlin and Welch, 1993) and applied to large mainframe-based MIP codes.

MOMIP, like other portable mixed integer programming codes, is not equipped with special SOS branching rules. However, MOMIP can emulate SOS branching rules due to a special technique of automatic model reformulation (Ogryczak, 1996), thus increasing the efficiency of the branch-and-bound search. The reformulation technique introduces new integer zero–one variables defined as the corresponding partial sums of x_j:

$$y_1 = x_1; \qquad y_j = y_{j-1} + x_j \quad \text{for} \quad j = 2, 3, \ldots, r. \tag{7.29}$$

Note that the standard branching on a y_k variable

$$y_k = 0 \quad \text{or} \quad y_k = 1 \tag{7.30}$$

implies the dichotomy

$$x_{k+1} + x_{k+2} + \cdots + x_r = 1 \quad \text{or} \quad x_1 + x_2 + \cdots + x_k = 1 \tag{7.31}$$

thus emulating the special SOS branching rule and generating a complete analogy with binary branching on the set of original options:

$$z \in \{a_1, a_2, \ldots, a_k\} \quad \text{or} \quad z \in \{a_{k+1}, a_{k+2}, \ldots, a_r\}. \tag{7.32}$$

The variables x_j no longer need to be specified as integer ones (in fact, they should not be specified as integer in order to avoid inefficient branching). Moreover, they can be simply eliminated when replacing the SOS model of multiple choice with the following:

$$\begin{array}{rcl} (a_1 - a_2)y_1 + (a_2 - a_3)y_2 + \cdots + (a_{r-1} - a_r)y_{r-1} + a_r & = & z \\ y_1 \leq y_2 \leq \ldots \leq y_{r-1} & \leq & 1 \\ y_j \geq 0, \quad y_j \ \text{integer for} \ j & = & 1, 2, \ldots, r-1 \end{array} \tag{7.33}$$

where the original values of x_j are defined as the corresponding slacks in the inequalities. The use of variables y_j is called special ordered inequalities (SOI).

The use of SOI instead of SOS does not increase the number of variables (either integer or continuous). SOI modeling increases the number of constraints, but these are very simple, and this does not cause much difficulty. The reformulation of SOS into SOI is controlled in MOMIP by a special parameter.

Special Implementation Features of MOMIP

MOMIP is part of a wider modular library of experimental solvers developed within the Methodology of Decision Analysis (MDA) project at IIASA and attached in the Appendix to this book. The library is a collection of independent modules, implemented as C++ classes, providing all the necessary functions of data input, data transfer, problem solution, and results output. MOMIP has also built in a primal (aside from the necessary dual) simplex algorithm and a standardized data transfer module (LP-DIT, see Makowski, 1994b) for communication with modeling systems.

7.4 Nonlinear Programming Solvers

Nonlinear programming is defined as the optimization of any model that is not linear; by this definition, we cannot even expect that a universal method of nonlinear programming exists. If we were willing to consider mixed integer programming models of arbitrary complexity, we could theoretically approximate any nonlinear programming model with mixed integer programming (and convex nonlinear programming models even with linear programming); thus we would have at least a universal methodology. However, the rationale and advantage of using nonlinear models is precisely that they represent complicated relations in a compact way – and it is often not advisable to lose this compactness for the sake of a uniform methodology.

If we restrict our attention to models described by differentiable functions, i.e., to smooth nonlinear programming, a very broad variety of optimization techniques exists (see Gill *et al.*, 1981; Powell, 1981; Luenberger, 1984; Fletcher, 1987). We provide a very concise overview of some of these techniques.

7.4.1 Basic nonlinear programming techniques

A nonlinear programming problem has the general form:

$$\underset{\boldsymbol{x}\in X_0}{\text{minimize}}\ (q = f(\boldsymbol{x})) \tag{7.34}$$

where

$$X_0 = \{\boldsymbol{x} \in \mathbb{R}^n :\ \boldsymbol{g}(\boldsymbol{x}) \leq 0 \in \mathbb{R}^{m_1},\ \boldsymbol{h}(\boldsymbol{x}) = 0 \in \mathbb{R}^{m_2}\}. \tag{7.35}$$

The function $f :\ \mathbb{R}^n \to \mathbb{R}^1$ is the objective function, the functions $\boldsymbol{g} :\ \mathbb{R}^n \to \mathbb{R}^{m_1}$ and $\boldsymbol{h} :\ \mathbb{R}^n \to \mathbb{R}^{m_2}$ represent the inequality and equality constraints. Minimization is considered as the standard case in nonlinear programming in order to make the description of convex problems consistent: the problem (7.34, 7.35) is called convex if the objective function f and the set of admissible solutions X_0 are convex.[12] The problem (7.34, 7.35) is smooth if all functions $f, \boldsymbol{g}, \boldsymbol{h}$ are differentiable.

Theoretically, we could convert inequality constraints in the problem (7.34, 7.35) into equations by introducing slack variables as for linear programming. However, inequality constraints should be considered a standard part of nonlinear programming, because they might be – if they are strongly nonlinear (i.e., not easily approximable by linear functions) – much more difficult to work with than equality constraints. One of the most important problems in constrained nonlinear programming is the determination of the active set of constraints that specifies the constraints that are active (equal to zero) at the optimal solution. If this active set is empty, the optimal solution is in the interior of the set X_0; we then speak about unconstrained optimization.[13]

Unconstrained Optimization

There are many techniques for unconstrained nonlinear programming. However, if we assume that symbolic differentiation provides the user with reliable formulae for derivatives, we can limit our attention to algorithms utilizing gradients of the objective function. An iteration, numbered here by (l), in all such algorithms consists of two phases:

- *Phase 1.* At the current point $\boldsymbol{x}^{(l)}$, compute the gradient $\nabla f(\boldsymbol{x}^{(l)})$ and use it, together with other data, to determine the current direction of descent $\boldsymbol{d}^{(l)} \in \mathbb{R}^n$ (the so-called steepest descent methods that consistently use only the gradient are among the worst and are not seriously used).
- *Phase 2.* Perform an approximate single-dimensional optimization, known as directional or line search, to determine the next point:

$$\begin{aligned} \tau^{(l)} &\approx \underset{\tau>0}{\operatorname{argmin}}\ f(\boldsymbol{x}^{(l)} + \tau \boldsymbol{d}^{(l)}) \\ \boldsymbol{x}^{(l+1)} &= \boldsymbol{x}^{(l)} + \tau^{(l)} \boldsymbol{d}^{(l)} \end{aligned} \tag{7.36}$$

[12]For this it is sufficient that all component functions of $\boldsymbol{h}$ are linear and all component functions of $\boldsymbol{g}$ are convex. As there are no concave sets in mathematics, we cannot speak of concave problems; if we considered maximization as a standard case, the objective function would have to be concave for the problem to be convex.

[13]In order to simplify this description, we could say that $X_0 = \mathbb{R}^n$, or that there are no constraints for unconstrained optimization. However, in practice fully unconstrained problems do not exist, each decision variable is constrained somehow, hence we should remember that $X_0 \neq \mathbb{R}^n$ but $\hat{\boldsymbol{x}} \in Int\, X_0$ for unconstrained problems.

There are several reliable algorithms to perform the directional search (7.36); see Fletcher (1987). However, the algorithms of unconstrained optimization differ mainly in the determination of the search direction $\boldsymbol{d}^{(l)}$. There are three main classes of such algorithms.

The first class, conjugate direction algorithms, determine the direction in the form:

$$\boldsymbol{d}^{(l)} = -\nabla f(\boldsymbol{x}^{(l)}) + \beta^{(l)} \boldsymbol{d}^{(l-1)}, \tag{7.37}$$

where $\beta^{(l)}$ is a coefficient such that the subsequent directions are at least approximately conjugate, i.e., orthogonal with respect to the Hessian matrix $\boldsymbol{H}^{(l)}$ composed of the second-order derivatives of the objective function f computed at $\boldsymbol{x}^{(l)}$, $\boldsymbol{d}^{T\,(l)} \boldsymbol{H}^{(l)} \boldsymbol{d}^{(l-1)} \approx 0$ (if the minimized function is quadratic and the Hessian matrix is constant, conjugate directions are orthogonal – with respect to the matrix – not only to the last one, but to all previous directions). The advantage of conjugate direction methods is that the Hessian matrix does not need to be computed in order to determine $\beta^{(l)}$; e.g., the formula of Fletcher and Reeves can be used:

$$\beta^{(l)} = \parallel \nabla f(\boldsymbol{x}^{(l)}) \parallel^2 / \parallel \nabla f(\boldsymbol{x}^{(l-1)}) \parallel^2 . \tag{7.38}$$

Conjugate direction methods usually start with $\beta^{(1)} = 0$. The main advantage of conjugate direction methods is that they minimize a quadratic function in a finite number of steps, not larger than the problem dimension. For nonquadratic functions, conjugate directions might accumulate errors after many iterations, and should be restarted.

The second class, variable metric or quasi-Newton algorithms, determines the direction while using a numeric approximation $\boldsymbol{V}^{(l)}$ of the inverse of the Hessian matrix $\boldsymbol{H}^{(l)}$:

$$\boldsymbol{d}^{(l)} = -\boldsymbol{V}^{(l)} \nabla f(\boldsymbol{x}^{(l)}). \tag{7.39}$$

The approximations $\boldsymbol{V}^{(l)}$, called variable metric, can be constructed in various ways, generally using rank-one or rank-two outer product matrices composed of elements of the observed differences $\boldsymbol{s}^{(l)} - \boldsymbol{V}^{(l-1)} \boldsymbol{r}^{(l)}$, where $\boldsymbol{s}^{(l)} = \boldsymbol{x}^{(l)} - \boldsymbol{x}^{(l-1)}$ and $\boldsymbol{r}^{(l)} = \nabla f(\boldsymbol{x}^{(l)}) - \nabla f(\boldsymbol{x}^{(l-1)})$. The simplest rank-one variable metric formula:

$$\boldsymbol{V}^{(l)} = \boldsymbol{V}^{(l-1)} + \frac{(\boldsymbol{s}^{(l)} - \boldsymbol{V}^{(l-1)} \boldsymbol{r}^{(l)})(\boldsymbol{s}^{(l)} - \boldsymbol{V}^{(l-1)} \boldsymbol{r}^{(l)})^T}{(\boldsymbol{s}^{(l)} - \boldsymbol{V}^{(l-1)} \boldsymbol{r}^{(l)})^T \boldsymbol{r}^{(l)}} \tag{7.40}$$

requires safeguards because it becomes ill-defined precisely when the matrix $\boldsymbol{V}^{(l-1)}$ approximates the inverse of $\boldsymbol{H}^{(l)}$ well enough, but can be implemented as an efficient algorithm with such safeguards. The BFGS formula is considered by many specialists to be the most reliable method of unconstrained optimization and is certainly the most widely known for a rank-two variable metric approximation (Fletcher, 1987):

$$\begin{aligned} \boldsymbol{V}^{(l)} &= \boldsymbol{V}^{(l-1)} + \left(1 + \frac{\boldsymbol{r}^{T\,(l)}\,\boldsymbol{V}^{(l-1)}\boldsymbol{r}^{(l)}}{\boldsymbol{s}^{T\,(l)}\boldsymbol{r}^{(l)}}\right)\frac{\boldsymbol{s}^{(l)}\boldsymbol{s}^{T\,(l)}}{\boldsymbol{s}^{T\,(l)}\boldsymbol{r}^{(l)}} + \\ &- \frac{\left(\boldsymbol{s}^{(l)}\boldsymbol{r}^{T\,(l)}\,\boldsymbol{V}^{(l-1)} + \boldsymbol{V}^{(l-1)}\boldsymbol{r}^{(l)}\boldsymbol{s}^{T\,(l)}\right)}{\boldsymbol{s}^{T\,(l)}\boldsymbol{r}^{(l)}} \end{aligned} \tag{7.41}$$

Variable metric formulae usually start with $\boldsymbol{V}^{(0)} = \boldsymbol{I}$ and are restarted after a number of iterations larger than n.

The third class, Newton-Raphson or Newton-type algorithms, require the computation and inversion of the Hessian matrix $\boldsymbol{H}^{(l)}$ in order to determine the Newton-type direction:

$$\boldsymbol{d}^{(l)} = -\boldsymbol{H}^{-1\,(l)}\nabla f(\boldsymbol{x}^{(l)}). \tag{7.42}$$

Such algorithms differ in various details related to the way the Hessian matrix is inverted (and its inverse regularized) and the directional search is safeguarded in order to provide for a broad convergence radius together with a fast convergence close to the optimal solution. We can theoretically obtain a quadratic convergence rate with Newton-type algorithms. This means doubling the negative exponent of the error $\parallel \boldsymbol{x}^{(l)} - \hat{\boldsymbol{x}} \parallel$ in every iteration, hence a few iterations would be sufficient if we are already in the quadratic convergence region. However, Newton-type algorithms might be slow in the beginning stage of convergence and fast only in a neighborhood of the optimal solution.

All such unconstrained optimization algorithms are local, i.e., they converge only to a local minimum in cases when a nonconvex objective function might have many minima. Global optimization algorithms are usually more complicated and require much more computational effort.[14]

Constrained Optimization

Constrained optimization concerns all cases where the set of constraints that are active at the optimal solution is nonempty. Solution techniques for such cases often use the necessary conditions for optimality for the problem (7.34, 7.35). To derive the necessary conditions of optimality in the constrained case, the optimal solution $\hat{\boldsymbol{x}}$ is assumed to be a regular point of the constraining set X_0. There are various

[14]Moreover, there is usually only a given probability of finding the global solution in so-called global algorithms (see later comments). An exception consists of nonlinear problems that can be converted to mixed integer programming linear problems, because the global optimality proof can then be performed with the branch-and-bound procedure.

regularity conditions; a basic one is that the gradients of the constraints active at the optimal solution should be linearly independent.

The necessary conditions of optimality can then be stated in various ways. We present their formulation when using a Lagrangian function:

$$L(\boldsymbol{\lambda}, \boldsymbol{x}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}_1^T \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{\lambda}_2^T \boldsymbol{h}(\boldsymbol{x}), \tag{7.43}$$

where $\boldsymbol{\lambda}^T = (\boldsymbol{\lambda}_1^T, \boldsymbol{\lambda}_2^T)$. The Kuhn-Tucker necessary conditions of optimality can then be written as their primal part:

$$\nabla_x L(\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{x}}) = \nabla f(\hat{\boldsymbol{x}}) + \sum_{i=1}^{m_1} \hat{\boldsymbol{\lambda}}_{1,i} \nabla g_i(\hat{\boldsymbol{x}}) + \sum_{i=1}^{m_2} \hat{\boldsymbol{\lambda}}_{2,i} \nabla h_i(\hat{\boldsymbol{x}}) = 0 \in \mathbb{R}^n \tag{7.44}$$

and their dual part:

$$\begin{aligned} \boldsymbol{g}(\hat{\boldsymbol{x}}) \le 0 \in \mathbb{R}^{m_1}, \;\; \boldsymbol{h}(\hat{\boldsymbol{x}}) &= 0 \in \mathbb{R}^{m_2}; \\ \hat{\boldsymbol{\lambda}}_1^T \boldsymbol{g}(\hat{\boldsymbol{x}}) + \hat{\boldsymbol{\lambda}}_2^T \boldsymbol{h}(\hat{\boldsymbol{x}}) &= 0 \in \mathbb{R}^1 \\ \hat{\boldsymbol{\lambda}}_1 \ge 0 \in \mathbb{R}^{m_1}, \;\; \hat{\boldsymbol{\lambda}}_2 &- \text{ arbitrary.} \end{aligned} \tag{7.45}$$

It should be stressed that if $\hat{\boldsymbol{x}}$ is a regular point of X_0, then Kuhn-Tucker conditions are necessary not only for the optimality of $\hat{\boldsymbol{x}}$, but also for a saddle point (maximum in $\boldsymbol{\lambda}$, minimum in $\boldsymbol{x}$) of the Lagrangian function at $\hat{\boldsymbol{\lambda}}$, $\hat{\boldsymbol{x}}$. However, the Lagrangian function does not usually have a saddle point at $\hat{\boldsymbol{\lambda}}$, $\hat{\boldsymbol{x}}$ (except for convex problems). Because the dual part of necessary conditions can be written as a set of equations in $\mathbb{R}^{m_1+m_2}$ (not as a complicated set of inequalities as it appears from (7.45); see Rockafellar, 1976; Bertsekas, 1982; Wierzbicki, 1982b), a shifted penalty function $\Phi_\rho(\boldsymbol{\lambda}, \boldsymbol{x})$ and an augmented Lagrangian function $L_\rho(\boldsymbol{\lambda}, \boldsymbol{x})$ can be formulated. For the constrained optimization problem (7.34, 7.35), the shifted penalty function and the augmented Lagrangian can be written as:

$$\begin{aligned} \Phi_\rho(\boldsymbol{\lambda}, \boldsymbol{x}) &= (1/\rho) f(\boldsymbol{x}) + \frac{1}{2} \|((1/\rho)\boldsymbol{\lambda}_1 + \boldsymbol{g}(\boldsymbol{x}))_+\|^2 \\ &+ \frac{1}{2} \|((1/\rho)\boldsymbol{\lambda}_2 + \boldsymbol{h}(\boldsymbol{x}))\|^2 \end{aligned} \tag{7.46}$$

$$L_\rho(\boldsymbol{\lambda}, \boldsymbol{x}) = \Phi_\rho(\boldsymbol{\lambda}, \boldsymbol{x}) - \frac{1}{2}(\|\boldsymbol{\lambda}_1/\rho\|^2 + \|\boldsymbol{\lambda}_2/\rho\|^2) \tag{7.47}$$

where $(\boldsymbol{u})_+$ is a vector composed of $\max(0, u_j)$, $\rho > 0$ is a penalty coefficient and $\boldsymbol{\lambda}_1/\rho$, $\boldsymbol{\lambda}_2/\rho$ can be interpreted as penalty shifts (i.e., predicted corrections of starting penalty count for constraint violations). If $\hat{\boldsymbol{x}}$ is an optimal solution satisfying additional regularity conditions,[15] then $\rho > 0$ exists such that the augmented

[15]If second-order sufficient conditions for optimality are satisfied at $\hat{\boldsymbol{x}}$, refer to Rockafellar (1976).

Lagrangian function has a saddle point at the optimal solution. The necessary conditions of the unconstrained saddle point of the augmented Lagrangian function are slightly different from (7.44, 7.45), but equivalent to these equations. For example, the dual conditions (7.45) then take the equivalent form:

$$\begin{aligned} \nabla_{\boldsymbol{\lambda}_1} L_\rho(\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{x}}) &= (\hat{\boldsymbol{\lambda}}_1/\rho + \boldsymbol{g}(\hat{\boldsymbol{x}}))_+ - \hat{\boldsymbol{\lambda}}_1/\rho = 0 \in \mathbb{R}^{m_1} \\ \nabla_{\boldsymbol{\lambda}_2} L_\rho(\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{x}}) &= (\hat{\boldsymbol{\lambda}}_2/\rho + \boldsymbol{h}(\hat{\boldsymbol{x}})) - \hat{\boldsymbol{\lambda}}_2/\rho = 0 \in \mathbb{R}^{m_2} \end{aligned} \tag{7.48}$$

where the second condition reduces simply to $\boldsymbol{h}(\hat{\boldsymbol{x}}) = 0$ and the first implies the nonnegativity of $\hat{\boldsymbol{\lambda}}_1$ (thus the saddle point of the augmented Lagrangian is unconstrained both in $\boldsymbol{x}$ and in $\boldsymbol{\lambda}$, while the saddle point of the Lagrangian (7.43) is constrained in $\boldsymbol{\lambda}_1$ by the inequality $\boldsymbol{\lambda}_1 \geq 0$). If the simplest iteration scheme is applied to the above equations, we obtain the basic penalty shift or multiplier iteration (which can be proven to converge for sufficiently large ρ under appropriate regularity conditions (see Hestenes, 1980, and Powell, 1969, for equality constraints; Wierzbicki, 1971, for inequality constraints):

$$\begin{aligned} \boldsymbol{\lambda}_1^{(l+1)}/\rho &= (\boldsymbol{\lambda}_1^{(l)}/\rho + \boldsymbol{g}(\boldsymbol{x}^{(l)}))_+ \\ \boldsymbol{\lambda}_2^{(l+1)}/\rho &= \boldsymbol{\lambda}_2^{(l)}/\rho + \boldsymbol{h}(\boldsymbol{x}^{(l)}). \end{aligned} \tag{7.49}$$

Most constrained optimization techniques consist of some approximate way of solving the set of Kuhn-Tucker conditions or their augmented equivalent formulation. There are many techniques; we mention only three major classes:

- *Gradient projection and/or reduction techniques*. These generally rely on the fact that the gradient of the Lagrangian function can be interpreted as a projection of the gradient of the objective function on a subspace tangent to the active constraints – or that a reduction of the gradient to a smaller subspace is possible. These techniques – with various specific variants – are used in the best available commercial solvers for nonlinear programming, such as MINOS or CONOPT. Generally, such techniques are advantageous if the nonlinear programming problem is in some sense similar to a linear programming problem – e.g., when the constraints contain a large linear part.
- *Multiplier methods*. These are based on augmented Lagrangian or penalty shift methodology. An example of such a method – HYBRID – has already been described in this book for linear problems of special structure; later, another application of this methodology – DIDAS-N – is presented. Although more robust than gradient reduction for either ill-conditioned problems or for strongly nonlinear constraints, at present such algorithms do not have efficient implementations for very large problems.

- *Trust region methods*. These are based on the proximal point methodology that assumes that any approximation of the original problem with local data might be invalid if we go too far from the current point $\boldsymbol{x}^{(l)}$. Therefore, each problem should have either restricted length of steps $\boldsymbol{x} - \boldsymbol{x}^{(l)}$ or be augmented with a proximal penalty term (such as used in HYBRID) added to the minimized function. For example, in nonlinear constrained optimization, a quadratic approximation of the Lagrangian function can be used and a corresponding quadratic programming problem with restricted steps or a proximal term can be solved. A variant of such methods is called sequential quadratic programming, although the sequential solution of quadratic approximation problems can also be used within other approaches (see Fletcher, 1987, for a detailed description of trust region methods).

7.4.2 Commercial nonlinear programming solvers

There are two good and well-tested commercial nonlinear programming solvers, MINOS and CONOPT. Both are based on similar methodology – the reduced gradient approach – as both were developed to solve linear and nonlinear (but with large linear parts) large-scale problems.

MINOS, however, uses a mixture of reduced gradient, augmented Lagrangian, and sequential quadratic programming in its approach. Let $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ denote parts of decision vectors – including slacks – that correspond to the nonlinear and linear parts of the model and $\boldsymbol{x}^T = (\boldsymbol{x}_1^T, \boldsymbol{x}_2^T)$. The standard form of a MINOS problem can then be written as:

$$\underset{\boldsymbol{x}\in X_0}{\text{minimize}}\,(q = F(\boldsymbol{x}) = f(\boldsymbol{x}_1) + \boldsymbol{c}^T\boldsymbol{x}) \tag{7.50}$$

where

$$\begin{aligned} X_0 = \{\boldsymbol{x} = (\boldsymbol{x}_1^T, \boldsymbol{x}_2^T)^T \in \mathbb{R}^{n_1+n_2} :\ \boldsymbol{g}(\boldsymbol{x}_1) + \boldsymbol{A}_1\boldsymbol{x}_2 &= \boldsymbol{b}_1, \\ \boldsymbol{A}_2\boldsymbol{x}_1 + \boldsymbol{A}_3\boldsymbol{x}_2 &= \boldsymbol{b}_2, \\ \boldsymbol{l} \le \boldsymbol{x} &\le \boldsymbol{u}\}. \end{aligned} \tag{7.51}$$

However, the sign "=" in constraints in (7.51) can be substituted by signs "$\le$", "$\ge$" or both (range inequality), or even by a comment *free* (which means that an outcome variable is used for the reasons of model construction, not optimization). Inside the solver, however, all inequalities are converted to equations using slack variables (which might not be the best strategy for strongly nonlinear models). The objective function f and constraint functions g_i are supposed to be smooth, with gradients given analytically (when used with the modeling system GAMS, MINOS obtains these gradients from a symbolic differentiation in GAMS; however, MINOS also has an option of verifying hand-programmed gradient formulae by a numeric finite difference approximation of derivatives).

If nonlinear terms are absent both in objective and constraint functions, MINOS uses a two-phase primal simplex algorithm. If the nonlinear term is present in the objective function and the constraints are linear, MINOS uses a reduced gradient method. The linear constraints are partitioned by defining the basic $\boldsymbol{x}_B$, the superbasic $\boldsymbol{x}_S$, and the nonbasic $\boldsymbol{x}_N$ parts of decision variables:

$$\boldsymbol{B}\boldsymbol{x}_B + \boldsymbol{S}\boldsymbol{x}_S + \boldsymbol{N}\boldsymbol{x}_N = \boldsymbol{b}. \tag{7.52}$$

The nonbasic variables are equal to their bounds; the superbasic variables might be moving freely like basic variables (because of the nonlinearity of the objective function, there might be more variables than in the linear case). The number of superbasic variables is estimated by the user in MINOS; this number is not greater than the number of variables in $\boldsymbol{x}_1$. The reduced gradient is computed according to the formula:

$$\nabla_{r,x}F(\boldsymbol{x}) = \boldsymbol{Z}^T\nabla_x F(\boldsymbol{x}); \quad \boldsymbol{Z}^T = [-\boldsymbol{S}^T\boldsymbol{B}^{-1T}, \boldsymbol{I}, 0], \tag{7.53}$$

where the matrix $\boldsymbol{Z}$ is represented in a product form resulting from the factorization of $\boldsymbol{B}$. The reduced gradient is then used, together with a variable metric procedure and a line search, to minimize the objective function.

If there are nonlinear parts of constraints, MINOS minimizes an augmented form of the objective function, similar but not equivalent to the shifted penalty function or augmented Lagrangian (because MINOS uses only a quadratic approximation part of it). The nonlinear constraints are linearized:

$$\tilde{\boldsymbol{g}} = \boldsymbol{g}(\boldsymbol{x}_1^{(l)}) + \boldsymbol{J}^{(l)}(\boldsymbol{x}_1 - \boldsymbol{x}_1^{(l)}), \tag{7.54}$$

where $\boldsymbol{J}^{(l)} = \boldsymbol{J}(\boldsymbol{x}_1^{(l)})$ denotes the Jacobian matrix of nonlinear constraints (composed of gradients $\nabla_{x_1}g_i(\boldsymbol{x}_1^{(l)})$ treated as row vectors). For subsequent major iterations denoted here by (l), MINOS minimizes the following quadratic and augmented approximation $F_a^{(l)}(\boldsymbol{x})$ of the original objective function (here we change the sign of Lagrange multipliers when compared to the original MINOS description, in order to maintain consistency with the notation in this chapter):

$$\begin{aligned}\underset{\boldsymbol{x}\in X_0^{(l)}}{\text{minimize}}\ (q_a^{(l)} = F_a^{(l)}(\boldsymbol{x}) \ &= \ f(\boldsymbol{x}_1) + \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{\lambda}_1^T\boldsymbol{J}^{(l)}(\boldsymbol{x}_1 - \boldsymbol{x}_1^{(l)}) \\ &+ \ \frac{1}{2}\rho(\boldsymbol{J}^{(l)}(\boldsymbol{x}_1 - \boldsymbol{x}_1^{(l)}))^T\boldsymbol{J}^{(l)}(\boldsymbol{x}_1 - \boldsymbol{x}_1^{(l)}))\end{aligned} \tag{7.55}$$

where

$$\begin{aligned} X_0^{(l)} = \{\boldsymbol{x} = (\boldsymbol{x}_1^T, \boldsymbol{x}_2^T)^T \in I\!\!R^{n_1+n_2} : \boldsymbol{J}^{(l)}(\boldsymbol{x}_1 - \boldsymbol{x}_1^{(l)}) + \boldsymbol{A}_1\boldsymbol{x}_2 &= \boldsymbol{b}_1, \\ \boldsymbol{A}_2\boldsymbol{x}_1 + \boldsymbol{A}_3\boldsymbol{x}_2 &= \boldsymbol{b}_2, \\ \boldsymbol{l} \le \boldsymbol{x} &\le \boldsymbol{u}\} \end{aligned} \tag{7.56}$$

and the linearized constraints can again be partitioned into the basic, superbasic, and nonbasic parts, the reduced gradient algorithm applied, etc. The Lagrange multipliers are determined in a reduced gradient algorithm similar to the simplex algorithm. Thus, although MINOS uses an augmented quadratic approximation, it does not use the penalty shift (multiplier) iteration (7.49). Therefore, experimental nonlinear solvers that use this iteration might behave differently from MINOS, particularly if constraints are strongly nonlinear. Nevertheless, MINOS is a very efficient code for nonlinear problems with linear or nearly linear constraints.

CONOPT is another commercial solver using reduced gradient methodology in its more pure version. It adds a first phase to find a feasible solution for nonlinear constraints and does not use the augmented form of the objective function (see Drud, 1992, for a more detailed description). Being a newer code, CONOPT solves many nonlinear problems much more efficiently than MINOS (see Chapter 13 for an example) but also solves some less efficiently. The conclusion of Drud (1992) is that both solvers should be used and the results compared (or solvers even used consecutively with a warm restart) for more difficult nonlinear programming problems.

7.4.3 Experimental nonlinear programming solvers and software packages

There are many experimental nonlinear programming codes. Here we briefly mention only three. First is FSQP, which consists of CFSQP (C code) and FFSQP (FORTRAN code) and is available free of charge to nonprofit organizations.[16] CFSQP and FFSQP implement a feasible sequential quadratic programming algorithm and handle constrained nonlinear optimization problems, including problems of the minmax type. They generate iterates that solve quadratic programming approximations of the original problem, but are also forced to satisfy all constraints. CFSQP includes a scheme to efficiently handle problems with a large number of constraints (see Lawrence *et al*., 1996, for a more detailed description of CFSQP).

The second example is the LANCELOT (large and nonlinear constrained extended Lagrangian optimization techniques) optimization package (see Conn *et al*., 1992). This is designed for solving large-scale nonlinear smooth optimization problems with an emphasis on problems that are significantly nonlinear (it

[16]www.isr.umd.edu/Labs/CACSE/FSQP/fsqp.html.

can be inefficient for linear constrained problems, especially in the presence of degeneracy).

The optimization problems in the LANCELOT package have three levels of complication. LANCELOT uses two different optimization algorithms, one for solving a simple-bound constrained minimization problem (SBMIN) and the other for a more general nonlinearly constrained problem (AUGLG).

LANCELOT/SBMIN is an algorithm for solving the minimization problem with simple bounds:

$$\underset{\boldsymbol{x}\in R^n}{\text{minimize}}\, F(\boldsymbol{x}) \tag{7.57}$$

subject to:

$$l_i \leq x_i \leq u_i, \qquad 1 \leq i \leq n. \tag{7.58}$$

It is assumed that F is twice continuously differentiable and any of the bounds may be infinite. A quadratic model of the nonlinear objective function $F(\boldsymbol{x})$ is built in the $(l+1)$ iteration. This model takes the form:

$$m^{(l)}(\boldsymbol{x}) = F(\boldsymbol{x}^l) + g(\boldsymbol{x}^l)(\boldsymbol{x}-\boldsymbol{x}^{(l)}) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}^{(l)})^T B^{(l)}(\boldsymbol{x}-\boldsymbol{x}^{(l)}) \tag{7.59}$$

where $B^{(l)}$ is a symmetric approximation to the Hessian matrix $H(\boldsymbol{x}^{(l)})$ and $g(\boldsymbol{x}^l)$ is the vector of first partial derivatives $(\nabla_{\boldsymbol{x}} F(\boldsymbol{x}^{(l)}))$. A trust region radius $\Delta^{(l+1)}$ is also defined, which determines the trust region:

$$\|\boldsymbol{x}-\boldsymbol{x}^{(l)}\| \leq \Delta^{(l+1)}, \tag{7.60}$$

within which we trust that the values of $m^{(l)}(\boldsymbol{x})$ and $f(\boldsymbol{x})$ will generally agree.

LANCELOT/AUGLG is an augmented Lagrangian type algorithm for solving the more generally constrained minimization problem:

$$\underset{\boldsymbol{x}\in R^n}{\text{minimize}}\, F(\boldsymbol{x}) \tag{7.61}$$

subject to the constraints:

$$c_j(\boldsymbol{x}) = 0,\ 1 \leq j \leq m \tag{7.62}$$

and the simple bounds

$$l_i \leq x_i \leq u_i,\ 1 \leq i \leq n. \tag{7.63}$$

It is assumed that F and the c_j are twice continuously differentiable and any of the simple bounds may be infinite. In the $(l+1)$ iteration the SBMIN algorithm is used to find an approximate minimizer, $x^{(k+1)}$, of the following augmented Lagrangian function (which is a shifted penalty function, see the comments in Section 7.2.5:

$$\Phi(\boldsymbol{x}, \lambda, \boldsymbol{S}, \mu) = F(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i c_i(\boldsymbol{x}) + \frac{1}{2\mu} \sum_{i=1}^{m} s_{ii}(c_i(\boldsymbol{x}))^2, \tag{7.64}$$

where λ_i are Lagrange multiplier estimates, s_{ii} are elements of a matrix $\boldsymbol{S}$ of scaling factors, and μ is a penalty parameter. The estimates of Lagrange multipliers for the next iteration are obtained by the standard penalty shift or multiplier iteration (as in HYBRID or in equation (7.49)).

LANCELOT also supports more complicated forms of objective function, related to specific applications in mechanics and in discretization of distributed parameter problems. This more complicated form is:

$$F(\boldsymbol{x}) = \sum_{i \in I_O} g_i(\sum_{J \in J_i} w_{i,j} f_j(\bar{x}_j) + a_i^T \boldsymbol{x} - b_i) \tag{7.65}$$

subject to the simple bounds:

$$l_i \leq x_i \leq u_i, \quad 1 \leq i \leq n \tag{7.66}$$

and to the additional constraints

$$g_i(\sum_{j \in J_i} w_{i,j} f_j(\bar{x}_j) + a_i^T \boldsymbol{x} - b_i) = 0, (i \in I_E) \tag{7.67}$$

$$0 \left\{ \begin{array}{c} \leq \\ \geq \end{array} \right\} g_i(\sum_{j \in J_i} w_{i,j} f_j(\bar{x}_j) + a_i^T \boldsymbol{x} - b_i) \left\{ \begin{array}{c} \leq \\ \geq \end{array} \right\} r_i, (i \in I_I) \tag{7.68}$$

where I_O, I_E, I_I are index sets; r_i is a given, possibly infinite number; $f_j, j \in J_i$ are nonlinear element functions; $\bar{x}_j, j \in J_i$ are subvectors of $\boldsymbol{x}$, either small or such that f_j has a large invariant subspace; $w_{i,j}$ are weighting coefficients; and $a_i^T \boldsymbol{x} - b_i$ is a linear element for the i-th group.

The optimization problems are specified in the SIF format (standard input format). It is an extension of the MPS, a standard input format for linear programming. The SIF format does not replace a full modeling language such as GAMS or AMPL but is helpful in specifying large nonlinear problems.

The LANCELOT package was written in the standard ANSI FORTRAN 77 and is available on several platforms. The software additionally provides:

- A decoder for translating problems defined in SIF format into a format appropriate for the minimizer.
- Direct and iterative linear solvers for the approximations to Newton's equations.
- Preconditioning and scaling algorithms.
- Quasi-Newton and Newton-type methods for Hessian evaluation.
- A provision for analytical or for finite-difference derivatives.

Another example, more a software package containing a nonlinear optimization code than a separate solver, is the multi-objective optimization system DIDAS-N included in the software library described in the Appendix. This is a closed system containing special user interfaces and model definition parts for multi-objective nonlinear modeling. The user defines the model in a format described in previous chapters, by defining the equations for subsequent model outcome variables:

$$\begin{aligned} y_1 &= f_1(\boldsymbol{x}, \boldsymbol{z}); \\ \ldots &= \ldots \\ y_{j+1} &= f_{j+1}(\boldsymbol{x}, \boldsymbol{z}, y_1, \ldots y_j), \;\; j = 1, \ldots m-1; \\ \ldots &= \ldots \\ y_m &= f_m(\boldsymbol{x}, \boldsymbol{z}, y_1, \ldots y_{m-1}) \end{aligned} \tag{7.69}$$

together with bounds for decision variables and outcomes:

$$x_{ilo} \le x_i \le x_{iup}, \;\; i = 1, \ldots n; \;\; y_{jlo} \le y_j \le y_{jup}, \;\; j = 1, \ldots m. \tag{7.70}$$

Between the outcome variables y_j, the user can select maximized objectives q_i with $i \in I_{max}$, or minimized objectives with $i \in I_{min}$, or stabilized objectives with $i \in I_{stab}$, where $\mid I_{max} \cup I_{min} \cup I_{stab} \mid = k$ is the number of objectives. For each objective, its reference or aspiration level $\bar{q}_i$ (sometimes also a reservation level $\bar{\bar{q}}_i$) is given and might be interactively changed by the user. These reference levels are used to define component achievement functions $\sigma_i(q_i, \bar{q}_i)$, e.g., in the case when only aspiration levels are used (see Chapter 4), but two of them – $\bar{q}_{i,lo}$ and $\bar{q}_{i,up}$ with $\bar{q}_i = (\bar{q}_{i,lo} + \bar{q}_{i,up})/2$ – are defined for stabilized objectives:

$$\begin{aligned} \sigma_i(q_i, \bar{q}_i) &= (q_i - \bar{q}_i)/(q_{i,up} - \bar{q}_i) \quad \text{for } i \in I_{max}, \\ \sigma_i(q_i, \bar{q}_i) &= (\bar{q}_i - q_i)/(\bar{q}_i - q_{i,lo}) \quad \text{for } i \in I_{min}, \\ \sigma_i(q_i, \bar{q}_i) &= \left\{ \begin{array}{c} (\bar{q}_{i,up} - q_i)/(\bar{q}_{i,up} - \bar{q}_i), \text{ if } q_i > \bar{q}_i \\ (q_i - \bar{q}_{i,lo})/(\bar{q}_i - \bar{q}_{i,lo}), \text{ if } q_i \le \bar{q}_i \end{array} \right\} \text{for } i \in I_{stab} \end{aligned} \tag{7.71}$$

A scalarizing smooth achievement function, maximized in DIDAS-N, is then defined as:

$$\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}) = 1 - (\frac{1}{k} \sum_{i=1}^{k} (1 - \sigma_i(q_i, \bar{q}_i))^p)^{1/p} \tag{7.72}$$

where $p > 2$, e.g., $p = 8$. The maximization of this function in DIDAS-N is performed by a special built-in solver. The solver takes advantage of the symbolic differentiation of the entire model (discussed in Chapter 6) to automatically define all needed derivatives. The solver uses a conjugate direction algorithm with projections on the box-like constraints for decision variables for successive optimization runs (major iterations) of a shifted penalty function. This function takes into account the smooth achievement function and shifted penalties, as in (7.46), for the nonlinear constraints on outcome variables. Lagrangian multipliers and penalty shifts are adjusted between major iterations according to the basic penalty shift iteration (7.49).

The solver in DIDAS-N cannot be compared to separate solvers: it is prepared for demonstrative problems of multi-objective optimization and model analysis of small to medium size. However, it works robustly for such problems. Another, modular, and separate version of the nonlinear DIDAS-N++ solver – mentioned in Chapter 6 – has recently been developed.

The best advice for a user of nonlinear solvers – as for other solvers – is to have several solvers (both commercial and experimental) and compare their results.

7.5 Optimization Issues

Issues related to other classes of optimization problems are presented in this section. The dynamic optimization has its own specificity (Wierzbicki, 1984). An essential feature of dynamic models is a special type of model outcome variable, called state variables. When treated with a prior discretization of the time variable, dynamic optimization models convert to bigger static models with a repetitive structure. This structure can be further complicated if time delays (in influence of decision variables and state and outcome variables, or state variables on themselves) are included in the model. A symbolic differentiation and sensitivity analysis of such models requires the solution of the so-called adjoint systems of equations, which should (and in the case of delays, must) be solved in the reverse direction of time. There are many experimental software packages for dynamic optimization. The modeling system GAMS was developed to represent large-scale models that result from dynamic formulations with discretized time; however, it does not include specific methods for solving and analyzing dynamic models.

Another important issue in optimization is nondifferentiability. In some model formulations (e.g., when using hierarchical optimization; see Findeisen *et al.*, 1980) even originally differentiable models might result in nondifferentiable functions. Thus, techniques for solving nondifferentiable or nonsmooth optimization problems have been intensively developed (Kiwiel, 1985). While many experimental codes exist for nondifferentiable optimization, the parameters of these codes often have to be carefully selected and adapted to a given problem in order to obtain a robust optimization tool.

Stochastic optimization has been developed to a stage such that models that are not too large (which, when stochastic phenomena are included in them, might easily become very large) can be practically solved (Ermoliev and Wets, 1988). Experimental codes for stochastic optimization have only been available until now (although some of them are quite effective; see Ruszczyński and Świętanowski, 1996).

Much effort has been put into the development of fuzzy mathematical programming (Zimmermann, 1987). In earlier chapters it was shown how multi-objective optimization closely relates to fuzzy-set formulations. However, objective functions and constraints can be viewed in terms of fuzzy sets. For example, in fuzzy linear programming, constraining equations and inequalities can be treated as fuzzy relations; equivalently, the right-hand side parameter $\boldsymbol{b}$ or bounds on variables and constraints can be treated as fuzzy numbers. The problems of fuzzy linear programming can usually be converted to equivalent problems of traditional, crisp linear programming by using, e.g., the maxmin aggregation. A more difficult instance is obtained when the coefficients in the matrix $\boldsymbol{A}$ are treated as fuzzy numbers, as the equivalent crisp formulation then becomes nonlinear. Nonlinear fuzzy programming has some similarity to multi-objective nonlinear programming.

Increased interest has recently been shown in novel approaches to optimization, such as genetic or evolutionary programming, neural networks, etc. Some impressive examples of these approaches are known, but they led again to hopes for a universal optimization algorithm. We should stress that such hopes were, historically, also attached to the simplex algorithm, then to the so-called dynamic programming method (used in dynamic optimization, but also in various integer optimization problems), and then to the so-called Monte Carlo method which attempted to solve optimization problems by probabilistic sampling of the set of admissible decisions. In each case, the proposed techniques were general, but this generality was usually obtained at the cost of an exponential increase in the computational effort with the problem's dimensions. This complexity rule also applies to novel approaches. For example, genetic programming can be considered a specific implementation of Monte Carlo methods, where the sampling can be interpreted as random mutation and genetic crossing. However, the necessary effort to obtain a global minimum with a high probability must involve a number of samples that will increase exponentially with the problem's dimensions. Thus, hopes for a universal but efficient optimization solver are self-contradictory; solvers will remain efficient only for specific classes of applications.

7.6 Optimization for Decision Support

Finally, we should again warn the reader again that textbook formulations of optimization problems, often used as testing examples for most available solvers, are not necessarily the most appropriate for applications. Here we focus on several problems related to using optimization as a tool for decision support.

Efficient and robust solving of any of the commonly recognized types of optimization problems requires an application of a specialized technique implemented in software, called a solver. A user of a DSS is rarely a specialist in optimization, therefore it is the responsibility of a designer of a DSS to make it possible to use a solver as a robust black-box type tool. However, a user should be aware of a correspondence between a solution provided by a solver and her/his decision problem. This correspondence may or may not be good, depending on the way in which an optimization problem is formulated and its solution is interpreted.

7.6.1 Formulation of an optimization problem

A solver provides an optimal solution of a precisely formulated mathematical programming or optimization problem. The general formulation of an optimization problem is as follows:

$$\underset{\boldsymbol{x} \in X_0}{\text{minimize}}\, f(\boldsymbol{x}) \tag{7.73}$$

where $\boldsymbol{x} \in I\!R^n$ is a vector of variables, $f(\boldsymbol{x})$ is a given objective function,[17] and X_0 is a set of feasible solutions.

For a number of reasons there is no possibility to adequately represent a nontrivial decision problem in the form (7.73). Therefore, for a typical real-world problem, other formulations must be used or at least a sequence or a number of such optimization problems must be solved.

Each instance of an optimization problem may be generated in various ways. For example, the reference point approach to multi-criteria optimization uses a scalarizing achievement function for generating a parametric single criterion optimization problem whose solution is also a Pareto-efficient solution. A direct application of a classical single-criterion optimization to decision support would induce the user to select one criterion as an objective function and to treat other criteria as constraints for which a maximum level of a criterion value is given. Earlier we discussed the advantages of the former approach and the disadvantages of the

[17]The objective function is called sometimes the goal function. However, this term is also associated with goal programming. Therefore we prefer to use the term objective function in this book.

latter; however, there are many other possible approaches and ways of generating an optimization problem.

For almost any way of generating an optimization problem, there are general issues related to the definitions of each of the three main components of the problem (i.e., variables, objective function, the set of feasible solutions). These are discussed in the following subsections.

Definition of Variables

Selecting appropriate variables to represent a real-life problem is one of the crucial elements of a proper model specification (see Chapter 6). For the purpose of the following discussion we assume that one can split the vector $\boldsymbol{x}$ into two parts:

$$\boldsymbol{x} = \{\boldsymbol{x}^u, \boldsymbol{x}^m\} \tag{7.74}$$

where $\boldsymbol{x}^u$ is a subvector of variables that are of interest to the user and $\boldsymbol{x}^m$ is composed of other variables – typically those introduced by a model builder for technical reasons. Clearly, this distinction is not very precise; therefore, the user might interactively change the classification of variables to one of the subvectors of $\boldsymbol{x}$. The primary criterion for this classification is the interest of the user, e.g., her/his need to specify preferred values for variables (used for various purposes, such as a regularization of the problem); such variables should form the subvector $\boldsymbol{x}^u$. Typically, this subvector is mostly composed of decision variables, but might include other variables, such as model outcome variables (as in the DIDAS-N system) or variables of any type that serve as criteria in multi-criteria model analysis.

Objective Function

Most optimization algorithms applied for solving the optimization problem (7.73) were developed under assumptions that were strong enough to guarantee a unique solution. However, most nontrivial optimization problems have practically nonunique solutions, even if an optimization problem can be formulated in a way conforming to requirements of a corresponding mathematical programming problem. The practical nonuniqueness of a solution is caused by a finite numerical precision of computations and can be defined as follows. An optimization problem (7.73) has practically a nonunique solution, if at least one pair of solutions $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ to this problem exist (there might even be a continuum of such solutions)[18] such that:

$$|f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)| \leq \epsilon, \qquad \|\boldsymbol{x}_1 - \boldsymbol{x}_2\| \geq \alpha \tag{7.75}$$

[18]Recall that such a continuum – a nonlinear manifold of nonunique solutions – is characteristic for the illustrative example in Chapter 6.

where ϵ and α are given positive numbers, small and large, respectively. That is, many (very) different solutions exist, for which objective function values differ (very) little. A choice of ϵ and α might depend on the problem, but typically the user does not worry about a loss of an objective function value in the order of a fraction of a percent of its current value. However, she/he understands that a solution is different if it changes by a few percent of its current value. Moreover, sacrificing a fraction of a percent of the objective function value may not be as important for the user as obtaining a solution that is closer to a given reference point. Therefore, for many real-world problems it is reasonable to apply a technique called regularization, i.e., to replace the objective function in (7.73) by:

$$\underset{\boldsymbol{x} \in X_0}{\text{minimize}}(f(\boldsymbol{x}) + \gamma \|\boldsymbol{x}^u - \bar{\boldsymbol{x}}^u\| + \delta \|\boldsymbol{x}^m - \bar{\boldsymbol{x}}^m\|) \tag{7.76}$$

where γ and δ are given positive numbers, and $\|\cdot\|$ is any norm, e.g., Euclidean. The given reference vector $\bar{\boldsymbol{x}}^u$ is composed of desired values of the variables, specified by the user. If such a choice of desired values is not easy, zero is assumed as a desired value, which implies that a minimum-norm solution is the preferred one. The specification of the vector $\bar{\boldsymbol{x}}^m$ is often done by the system analyst who develops the model.

The trade-offs between the values of the objective function and the attainment of the desired solution imply the choice of γ and δ. For example, a good rule of thumb for selecting γ is to define it as:

$$\gamma = \epsilon\ |f(\bar{\boldsymbol{x}})|\ /\ \|\bar{\boldsymbol{x}}^u\|, \tag{7.77}$$

where $f(\bar{\boldsymbol{x}})$ is an estimated value of the optimal solution and $\|\bar{\boldsymbol{x}}^u\|$ is the norm of the reference solution. The above rule should be modified if any of those two values are small (e.g., comparable with tolerances used in optimization). The value of δ is usually set to a small number (for a well-scaled problem it may be of an order of feasibility tolerance).

The replacement of the objective function in (7.73) by an expression such as in (7.76) usually results not only in a unique solution of a given instance of the optimization problem, but also in two additional, desirable effects. First, it helps to avoid the so-called penny switching effect – in which a small change in a model definition results in quite a different solution. Second, it usually improves the numerical properties of the optimization problem, thus at least reducing computation time.

Many practical problems are much more easily solved when using regularization; therefore, regularization is applied as a standard feature in some solvers for numerical reasons (e.g., HYBRID, cf. Section 7.2.5). Regularization applied for purely numerical reasons should not influence the value of an optimal solution, therefore the corresponding regularization coefficients must be very small (a few orders of magnitude smaller than the required optimality tolerance, at least at the end of the optimization process). However, a solver may provide the user with control over regularization coefficients, for the purposes discussed above.

Finally, we should note that the regularization technique applied with larger regularization coefficients is similar to the soft simulation technique discussed in earlier chapters.

Set of Feasible Solutions

The compact notation of the set of feasible solutions X_0 means different types of sets for different types of optimization problems. Here the discussion is restricted to some general problems resulting from the practice of using optimization for decision support.

For any real-world, well-defined decision problem, a set of feasible solutions X_0 should be bounded and nonempty. Yet every optimization specialist knows that, in many cases of actual applications of optimization solvers, the problem being solved has no feasible solution, and in some cases – that the solutions are not bounded. Both cases are lucky situations, because they lead to a verification of model specification. Less often an analysis of optimization processes can lead to an observation that the set of feasible solutions actually does not contain all feasible solutions.

However, it is impossible to verify the correctness of a definition of a set of feasible solutions during optimization. For large and complex decision problems such verification is practically possible only during model specification. This stresses the importance of a thorough model analysis in the early stages of the decision process.

An important issue related to the definition of a set X_0 and its use for decision support relates to a boundary between model specification and optimization problem formulation. The traditional way of model specification, related to single criterion optimization, forces the modeler to include in the model both a definition of a core (substantive) model and this part of the preference structure of the user that is not represented by a single objective function. A core model defines implicitly an original set of feasible solutions, where feasibility is understood in the sense of logical and physical relations that must always hold.

However, in the traditional approach to model specification the set X_0 defined by the core model is additionally restricted by constraints representing the preferential structure of the user; these constraints aim at making a single criterion optimization solution not only feasible in the sense of the core model definition, but also acceptable for the user. Such constraints typically represent bounds on objectives that are not represented in an objective function (e.g., maximum cost, maximum concentration of a pollution). Their representation in a form of hard constraints in a model specification decreases the set of feasible solutions. A part of the set of feasible solutions – which is cut off by these additional constraints – is practically no longer analyzed. This often results in misleading conclusions from model analysis. In extreme cases, the set of feasible solutions might become empty because of additional constraints.

Therefore, we recommend avoiding the practice of including any constraints reflecting the preferential structure of the user into the specification of a substantive or core model. Such preferences can and should be accounted for in a natural way during multi-criteria model analysis. Should a single-criterion optimization be chosen for model analysis, then representing a preferential structure by soft constraints is preferable to the traditional representation by hard constraints. There are at least two desired consequences of using soft constraints. First, the user obtains information and control over the level of fulfillment of constraints for objectives represented as soft constraints. Second, the feasibility set cannot be made empty by the bounds set for such objectives.

7.6.2 Requirements for a solver

For the user of a DSS, a solver should provide a reliable solution of an optimization problem efficiently and quietly. The user often does not know what the specification of the optimization problem is, which is often generated automatically (see Chapter 3). However, solvers are often designed and implemented for users who are specialists in optimization. The issues related to the interface between the user and a solver are further discussed in Chapter 10. Here we briefly discuss selected problems, which might be of interest for both users of DSS's and developers of solvers aimed at being included in a DSS.

- *Presolve analysis.* A solver should perform a presolve analysis (as complete as practicable) of the problem and provide informative diagnostics for any formulation of the optimization problem. In theory, incorrect formulations should be traced during problem generation, but in practice this is not always the case. Therefore, the solver should test whether the problem formulation meets the requirements of the solution method. A more detailed diagnosis can help substantially in improving the model formulation. For example, diagnostic tools for LP problems provide information about: empty (or split) columns and rows; redundant rows and columns; very small or large elements; possible problems with scaling, etc. For nonlinear problems, testing the computation of the Jacobian matrix can be included (if automatic differentiation is not provided) and also the order of magnitude of gradients variations (in order to warn the user about the sensitivity of the problem). There are special modules for presolve analysis of LP problems that often result in a substantial reduction of the problem size (Gondzio, 1997). However, the use of such modules might also result in problems of providing a correct dual solution for the original problem. Therefore, the user should have control over the options of a presolve module.

- *Robustness.* Typically, the user assumes that the solution provided by a solver is the optimal one. However, in practice a solver often provides only a suboptimal solution, or stops computations because of various reasons (typical reasons include: an infeasible or unbounded problem, numerical problems, iteration or time limit, hardware problems, etc). Therefore, a solution file should contain a clear (and not easy to be overlooked) indication of the status of the solution. A solver used as part of a DSS should be well tested in order to deal correctly with numerical problems; therefore, it should never produce messages such as underflow, overflow, core dump, etc. In future, parallel computation techniques might be used in order to improve robustness (Sobczyk and Wierzbicki, 1994).
- *Dual solution.* A general recommendation to the authors of solvers – often not followed in practice – is to avoid generating those components of a dual solution that are not guaranteed to be correct (which is often the case, e.g., if an aggressive presolve analysis results in a substantial reduction of the problem size).
- *Problem definition.* Solvers should not impose requirements on the problem definition that are difficult to be satisfied in practice. An example of such an unrealistic requirement is a feasible initial point for nonlinear problems.
- *Optimization parameters.* Nearly every optimization algorithm can be made much more efficient for a particular class of problems by tuning certain parameters. The default set of parameters is designed for solving many different problems in as short a time as possible. The tuning of parameters requires a sound knowledge of the optimization algorithm applied in the solver, therefore it should be left to optimization specialists. This is also an argument for the necessity of involving such specialists in the implementation of a DSS for any nontrivial problem.
- *Regularization.* Very few solvers provide the user with a possibility of regularization. This is despite the fact that most optimization methods can be easily extended to provide this option, and many algorithms implemented use regularization for numerical reasons. This is probably caused by under-evaluating the practical importance of regularization for model analysis while using optimization techniques, and by the fact that most developers of solvers concentrate on solving standard mathematical programming problems. Our recommendation is to introduce regularization as a standard option to all solvers used for decision support.

- *User interface.* A stand-alone solver, or a solver that is part of a modeling environment, usually allows for much freedom regarding the form of its input/output data, but it is difficult, nevertheless, to integrate stand-alone solvers within larger software systems. A modular solver that is easier to be integrated as part of a DSS built out of modular blocks should also allow for freedom regarding the form of input–output data, thus providing flexible ways for problem formulation, data handling, diagnosis, and access to a solution.
- *Warm start.* A typical way of using optimization in a DSS involves solving long sequences of similar and related problems. If solving one problem takes a substantial amount of computing time, then the use of a previous solution as a starting point often speeds up optimization considerably. Simplex-based LP solvers use the optimal basis for this purpose. Typical interior point solvers cannot use the previous optimal solution as a starting point; however, new techniques for a warm start in interior point algorithms are being developed (Gondzio, 1996b). However, the approaches to the warm start impose severe restrictions on admissible modifications of the problem. In practice such restrictions might be difficult to meet. This subject requires more study and attention to the developers of solvers.

7.6.3 Interpretation of a solution

The interpretation of the results of optimization is typically made either with the help of a user interface to multi-criteria model analysis (cf. Chapter 10) or by problem-specific utilities (cf. Chapters 11 through 14). There is, however, a general aspect of the problem that requires a brief comment; it concerns the interpretation of dual solutions.

Sensitivity analysis procedures that use the dual solutions of optimization problems are recommended by many textbooks on applications of mathematical programming. However, the difficulties and limited reliability of such an approach are not widely known. Comments are restricted to the following two issues:

- The main difficulty is due to the fact that the dual solution has a well-defined interpretation only in the neighborhood of the optimal solution. A specification of this neighborhood is not directly available from the standard output of a solver. Commonly known observations show that users tend to extend the interpretation of a dual solution (e.g., shadow prices for LP problems) far beyond the region in which it is valid – although the bounds of this region can be easily determined by a postoptimal analysis of LP problems (for nonlinear problems this region is typically very small).

- The limited reliability is due to the problem of nonuniqueness of dual solutions and their robustness. It is easy to provide examples of problems for which very different dual solutions can be obtained using different solvers, or even one solver with different options. By very different dual solutions we mean such solutions that have values of some components replaced by zero value, and vice versa. This problem is beyond the scope of this book,[19] but serves to warn against using techniques without fully understanding them.

[19]The authors would like to advise users of dual solutions to read the following: Jansen *et al.* (1993) and Güler *et al.* (1993). The former provide a good summary of the related problems and their experiences with application; the latter present a survey of degeneracy in the interior point methods, which are considered by many users to be free of degeneracy problems.

Chapter 8

Multi-Objective and Reference Point Optimization Tools

Andrzej P. Wierzbicki

This chapter presents in more detail the mathematical background, concepts, and tools for multi-objective optimization and reference point methodology, more generally presented in Chapter 4. Section 8.1 presents a comprehensive discussion of various types of Pareto-optimality and efficiency. Additional material on estimating objective ranges and on objective aggregation is discussed in Section 8.2. A detailed presentation of various types of reference points and achievement functions is contained in Section 8.3. Section 8.4 presents further material on weighted and neutral compromise solutions, together with their relations to reference point approaches. Section 8.5 provides brief comments on tools for multi-objective optimization.

8.1 Pareto-Optimality and Efficiency

While we discussed various types of models in Chapters 5 and 6, we assume here that the general form of a substantive model is:

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{z}); \;\; \boldsymbol{x} \in X_0; \;\; \boldsymbol{z} \in Z_0, \tag{8.1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the vector of decision variables, $\boldsymbol{z}$ is a parameter vector fixed by the modeler, and X_0 is a set of admissible decisions – usually defined by a set of additional inequalities or equations called constraints, or by specifying discrete values of admissible decisions. Recall that $\boldsymbol{y} \in I\!R^m$ is a vector of model outputs or decision outcomes, which also includes various intermediary variables that are useful when formulating the model, even when determining the constraints. Thus, the set X_0 is often defined implicitly. The function $\boldsymbol{f} : I\!R^n \times Z_0 \rightarrow I\!R^m$ that determines model outputs is usually defined implicitly, often by a complicated model structure. In actual applications, substantive models might express dynamic system behavior and uncertainty of results of decisions (while the outcomes $\boldsymbol{y}$ might be understood, e.g., as mathematical expectations of such results). We have discussed more complicated substantive model forms in Chapters 5 and 6; here we assume the more general form in order to shorten theoretical considerations. Moreover, we shall suppress the dependence of the function $\boldsymbol{f}$ on parameters $\boldsymbol{z}$ when not directly needed by defining $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$.

In such a case, $Y_0 = \boldsymbol{f}(X_0)$ is called the set of attainable outcomes. It should be stressed that this set is not given explicitly (even in the simple case when $\boldsymbol{f}$ is given). We can only compute its elements by assuming $\boldsymbol{x} \in X_0$ and then determining the corresponding $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$ by simulating the model.

As explained before, we assume that the modeler or decision maker might choose among model outputs those that are especially interesting; these outputs are called objectives or criteria, and are denoted by $q_i = y_j$, forming an objective vector $\boldsymbol{q} \in I\!R^k$ in the objective space. This vector and space might change during the multi-objective model analysis. The relation between decisions and their objective outcomes (criteria) is denoted $\boldsymbol{q} = \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{z})$, or shortly $\boldsymbol{q} = \boldsymbol{F}(\boldsymbol{x})$. $Q_0 = \boldsymbol{F}(X_0)$ denotes the set of attainable objectives.

In classical multi-objective optimization it is usually assumed that all objectives are either maximized or minimized; as we can change minimization to maximization by changing the sign of an objective, we can equally assume that all objectives are, e.g., maximized. We recall that a Pareto-optimal decision and its objective outcome are such that there are no other admissible decisions and thus attainable objective outcomes that would improve any objective component without deteriorating other components. Alternatively, Pareto-optimal decisions and outcomes are called Pareto-nondominated, since decisions and outcomes that are not Pareto-optimal are called Pareto-dominated (another admissible decision exists with an outcome that is not worse in all components and better at least in one, hence it dominates the former decision outcome).

A closely related, but slightly broader and weaker concept, is that of a weakly Pareto-optimal decision and outcome: in this there are no other admissible decisions that would result in a joint improvement of all outcome components. This concept is too weak for applications. To give an example, suppose we want to buy a book and consider two objectives. One objective (to be maximized) is jointly our interests and the book quality; another objective (to be minimized) is the price. Suppose we select a specific book; this means that we cannot improve on the first objective. If we go to a bookshop and buy the book without checking the prices in other bookshops, we actually follow the concept of weak Pareto-optimality. But the second objective might still be relevant; we might prefer to buy this book (i.e., with no possible improvement of the first outcome component) in a shop that offers lower prices (i.e., improving only the price, not all outcome components). Thus, Pareto-nondominated decisions represent typical preferences better than weakly Pareto-nondominated decisions. However, weak Pareto-optimality is simpler theoretically and in computations; many simpler approaches to multi-objective optimization, such as the maxmin approach, produce only weakly Pareto-optimal results, often without warning the user about the possible difference to Pareto-optimal results.

Sometimes even the concept of Pareto-optimality is too weak for applications, e.g., in cases where we could improve significantly one outcome component at the cost of an infinitesimally small deterioration of another outcome component. The (limits of) ratios of improvements and deteriorations of outcome components, determined at a Pareto-optimal outcome, are called trade-off coefficients; properly Pareto-optimal decisions and outcomes are defined such that the corresponding trade-off coefficients are bounded. Even this concept is sometimes too weak for applications, as the mathematical definition of "bounded" means "anything smaller than infinity". More important for applications are decisions and outcomes that are properly Pareto-optimal with a prior trade-off bound (i.e., such that a finite bound on trade-off coefficients is a priori given and satisfied).

As already mentioned in Chapter 4, in all these specific cases of Pareto-optimality (weak, proper, etc.), the sets of Pareto-optimal decisions and outcomes typically contain many elements. Therefore, Pareto-optimality is a weaker concept than single-criterion optimality and does not tell us which decisions to choose; it tells us only which decisions to avoid. In interactive decision support, however, the nonuniqueness of Pareto-optimal decisions might be considered as an advantage, not a drawback. In order to use this advantage, we need a way of parameterizing the set of Pareto-optimal decisions, or equivalently, of controlling the selection of Pareto-optimal decisions by parameters specified by the user.

When considering such a controlled selection, it is important to realize that Pareto-optimal solutions correspond to part of the boundary of the set of attainable objectives Q_0. This part is called the Pareto boundary or Pareto frontier of Q_0 (see *Figure 8.1*), where the sets X_0, Q_0 and their (weak or ordinary) Pareto frontiers are shown. In this figure, the simplest case of such sets is considered, where the substantive model is defined by linear equations and inequalities – which corresponds to a multi-objective linear programming case – and the sets are convex and polyhedral. The conical sets $\hat{\boldsymbol{q}} + \tilde{D}$ or $\hat{\boldsymbol{q}} + IntD$ in this figure are related to a

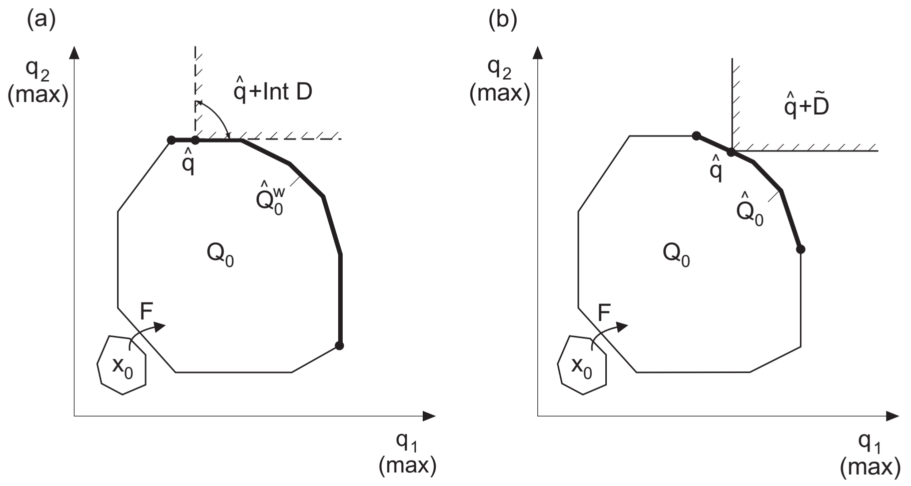


Figure 8.1. Examples of (a) weakly Pareto-optimal and (b) Pareto-optimal objective outcomes for a polyhedral set Q_0.

more formalized and general definition of the concept of Pareto-optimality that is presented below.

The distinction between Pareto-optimal and properly Pareto-optimal decisions and outcomes is not shown in *Figure 8.1*, because all Pareto-optimal outcomes are also properly Pareto-optimal for polyhedral outcome sets (see e.g., Sawaragi *et al.*, 1985). However, this does not suggest that all Pareto-optimal outcomes satisfy a prior bound on trade-off coefficients. Some facets of the polyhedral set Q_0 might be almost parallel to some of the axes. In theory, all points on such a facet might be properly Pareto-optimal; in practice, such a facet might correspond to large trade-offs and be practically equivalent to weakly Pareto-optimal outcomes. This stresses the importance of defining and calculating properly Pareto-optimal outcomes with a prior bound. This is discussed in more detail below.

We stress again that the set Q_0 and its Pareto frontier are not known explicitly to the modeler, even in the example presented in *Figure 8.1*. Thus, it is not easy for the modeler to find decisions $\boldsymbol{x} \in X_0$ that would result in outcomes on the Pareto frontier. It would be even more difficult for the modeler to control the selection of Pareto-optimal decisions and outcomes, i.e., to change decisions in such a way that the outcomes remain Pareto-optimal but move in a desired direction along the Pareto frontier.

Before explaining how such controlled selection can be achieved, we first generalize the concept of Pareto-optimality. This generalization is useful because the modeler might want to specify which objectives q_i will be maximized, which ones will be minimized, and which of them will be stabilized, i.e., kept close to a given reference level $\bar{q}_i$. For the sake of theoretical compactness, we can represent such information as a partial order implied by a positive cone, while the positive cone indicates what we understand by an improvement in the space of objectives. In the case of Pareto-optimality (if all objectives are maximized), the positive cone is the positive "octant" of the objective space:

$$D = \mathbb{R}^k_+ = \{\boldsymbol{q} \in \mathbb{R}^k : \; q_i \geq 0 \; \forall i = 1, \dots k\}. \tag{8.2}$$

A strictly positive cone (assuming an improvement of at least one objective component, as defined for Pareto-optimality) can be expressed as:

$$\tilde{D} = \mathbb{R}^k_+ \setminus \{0\} = \{\boldsymbol{q} \in \mathbb{R}^k : q_i \geq 0 \, \forall i = 1, \dots k; \; \exists i = 1, \dots k : q_i > 0\}. \tag{8.3}$$

A strongly positive cone (assuming an improvement of all objective components, as defined for weak Pareto-optimality) is defined simply as the interior of the positive cone, $IntD = Int\mathbb{R}^k_+$.

In cases where some objectives (from 1 to k_1) are maximized, some (from k_1+1 to k_2) are minimized and some (from $k_2 + 1$ to k) are stabilized, the positive cone can be defined as:

$$\begin{aligned} D \;\; = \;\; & \{\boldsymbol{q} \in \mathbb{R}^k : \; q_i \geq 0, \, i = 1, \dots k_1, \, q_i \leq 0, \, i = k_1 + 1, \dots k_2, \\ & q_i = 0, \, i = k_2 + 1, \dots k\}. \end{aligned} \tag{8.4}$$

Note that the cone describes only changes in objective values, hence $q_i = 0$ means that the objective component is kept equal to its reference level. If we define similarly the strictly positive cone as $\tilde{D} = D \setminus \{0\}$ and the strongly positive cone as $IntD$, we can give a more general definition of Pareto-optimality. Such generalized Pareto-optimality is called efficiency with respect to the cone D; the set of efficient objectives or outcomes is defined as:

$$\hat{Q}_0 = \{\hat{\boldsymbol{q}} \in Q_0 : \; (\hat{\boldsymbol{q}} + \tilde{D}) \cap Q_0 = \emptyset\}, \tag{8.5}$$

and the set of efficient decisions is defined equivalently, while taking into account that $\hat{\boldsymbol{q}} = \boldsymbol{F}(\hat{\boldsymbol{x}})$, as:

$$\hat{X}_0 = \{\hat{\boldsymbol{x}} \in X_0 : \; (\boldsymbol{F}(\hat{\boldsymbol{x}}) + \tilde{D}) \cap \boldsymbol{F}(X_0) = \emptyset\}. \tag{8.6}$$

Note that if $D = \mathbb{R}^k_+$ and $\tilde{D} = \mathbb{R}^k_+ \setminus \{0\}$, the above definition of efficiency coincides with the definition of Pareto-optimality given earlier. This defines that there are no admissible decisions that would improve some component objective or outcome without deteriorating some other component objectives or outcomes. The equivalence of these definitions is also illustrated in *Figure 8.1*.

Similarly, the generalization of weak Pareto-optimality to weak efficiency is obtained by simply replacing, in the above definitions, the strictly positive cone $\tilde{D}$ with the strongly positive cone $IntD$:

$$\hat{Q}_0^w = \{\hat{\boldsymbol{q}} \in Q_0 : (\hat{\boldsymbol{q}} + IntD) \cap Q_0 = \emptyset\}, \tag{8.7}$$

and

$$\hat{X}_0^w = \{\hat{\boldsymbol{x}} \in X_0 : (\boldsymbol{F}(\hat{\boldsymbol{x}}) + IntD) \cap \boldsymbol{F}(X_0) = \emptyset\}. \tag{8.8}$$

Note that if $k > k_2$ (there are stabilized objectives), then the cone (8.4) has an empty interior, hence $\hat{Q}_0^w = Q_0$, and the concept of weak efficiency is redundant in this instance.

In order to define proper efficiency, we must first specify the concept of trade-off coefficients. We assume for simplicity that all objectives are dimension-free and can be directly compared (this assumption is relaxed in the next section). At an efficient point $\hat{\boldsymbol{x}} \in \hat{X}_0$ with $\hat{\boldsymbol{q}} = F(\hat{\boldsymbol{x}}) \in \hat{Q}_0$, if the efficient frontier is smooth at this point, the local trade-off coefficient $t_{ij}(\hat{\boldsymbol{q}})$ between maximized objectives q_i, and q_j is defined as:

$$\begin{aligned} t_{ij}(\hat{\boldsymbol{q}}) &= \lim_{\delta \to 0} \sup_{\boldsymbol{q} \in \hat{Q}_\delta(\hat{\boldsymbol{q}})} \frac{q_i - \hat{q}_i}{\hat{q}_j - q_j}; \text{ with} \\ \hat{Q}_\delta(\hat{\boldsymbol{q}}) &= \{\boldsymbol{q} \in \hat{Q} : \| \boldsymbol{q} - \hat{\boldsymbol{q}} \| < \delta\}. \end{aligned} \tag{8.9}$$

If an objective i or j is minimized, the sign of the appropriate increment in the above equation must be changed. In the case of stabilized objectives, we must consider that they might be either maximized or minimized, with alternative trade-offs.

For nonconvex sets Q_0, it is also useful to define global trade-off coefficients that might be greater (but not in the convex case) than the local ones, i.e.:

$$\begin{aligned} t_{ij}(\boldsymbol{q}) &= \sup_{\boldsymbol{q} \in Q^{(j)}(\hat{\boldsymbol{q}})} \frac{q_i - \hat{q}_i}{\hat{q}_j - q_j}, \text{ with :} \\ Q^{(j)}(\hat{\boldsymbol{q}}) &= \{\boldsymbol{q} \in Q_0 : q_j < \hat{q}_j, \, q_i \geq \hat{q}_i\}, \end{aligned} \tag{8.10}$$

and with the signs of inequalities in the definition of $Q^{(j)}(\hat{\boldsymbol{q}})$ appropriately changed for minimized (or stabilized) objectives.

The computation of trade-off coefficients according to their definitions is a difficult problem (see e.g., Kaliszewski, 1994). We can obtain bounds on trade-off coefficients if we express the concept of proper efficiency in terms of modified positive cones. There are various approaches to such a representation (see e.g., Henig 1982; Sawaragi *et al.,* 1985). It can be shown that properly efficient outcomes and decisions with a prior bound M on trade-off coefficients can be defined as weakly efficient outcomes and decisions with respect to a “slightly broader” positive cone (Wierzbicki, 1986, 1992c; Kaliszewski, 1994). For this purpose, we define first $\varepsilon = 1/(M-1)$ (note that there is no sense in considering $M \leq 1$) and define an ε-neighborhood $IntD_\varepsilon$ of the positive cone D:

$$IntD_\varepsilon = \{\boldsymbol{q} \in I\!R^k : dist(\boldsymbol{q}, D) < \varepsilon \parallel \boldsymbol{q} \parallel\} \tag{8.11}$$

where we could choose any norm in $I\!R^k$ and a (Hausdorff) concept of distance between the point $\boldsymbol{q}$ and the set D in order to obtain an open[1] set $IntD_\varepsilon$. However, in order to obtain the needed bound on trade-off coefficients, it is useful to choose specific norms: l_1 on the right-hand side and mixed l_1 and l_∞ for the distance on the left-hand side of the inequality in the definition of the cone $IntD_\varepsilon$. Let $\boldsymbol{q}^{(-)}$ denote the part of the vector $\boldsymbol{q}$ that is not in the cone D, i.e., a vector with the following coordinates:

$$\begin{aligned} q_i^{(-)} &= \min(0, q_i) \text{ for } i = 1, \ldots k_1 \\ q_i^{(-)} &= \max(0, q_i) \text{ for } i = k_1 + 1, \ldots k_2 \\ q_i^{(-)} &= q_i \text{ for } i = k_2 + 1, \ldots k. \end{aligned} \tag{8.12}$$

The cone $IntD_\varepsilon$ can then be written as:

$$\begin{aligned} IntD_\varepsilon &= \{\boldsymbol{q} \in I\!R^k : \parallel \boldsymbol{q}^{(-)} \parallel_{l_1} + 2\varepsilon \parallel \boldsymbol{q}^{(-)} \parallel_{l_\infty} < \varepsilon \parallel \boldsymbol{q} \parallel_{l_1}\} \\ &= \{\boldsymbol{q} \in I\!R^k : \sum_{i=1}^{k} \mid q_i^{(-)} \mid + 2\varepsilon \max_{1 \leq i \leq k} \mid q_i^{(-)} \mid < \varepsilon \sum_{i=1}^{k} \mid q_i \mid\}. \end{aligned} \tag{8.13}$$

Note that $IntD_\varepsilon \neq \emptyset$ even if $IntD = \emptyset$ (as stressed before, the latter holds for cone D of the form (8.4) including some stabilized objectives). Moreover, if we define weakly efficient solutions with respect to the “broader” cone $IntD_\varepsilon$, we can prove that they are equivalent to properly efficient solutions with (global, not only local) trade-off coefficients bounded a priori by $M = 1 + 1/\varepsilon$; such outcomes and decisions are ε-properly efficient. Thus, the sets of ε-properly efficient outcomes and ε-properly efficient decisions can be defined as:

$$\hat{Q}_0^{p\varepsilon} = \{\hat{\boldsymbol{q}} \in Q_0 : (\hat{\boldsymbol{q}} + IntD_\varepsilon) \cap Q_0 = \emptyset\} \tag{8.14}$$

and

$$\hat{X}_0^{p\varepsilon} = \{\hat{\boldsymbol{x}} \in X_0 : (\boldsymbol{F}(\hat{\boldsymbol{x}}) + IntD_\varepsilon) \cap \boldsymbol{F}(X_0) = \emptyset\}. \tag{8.15}$$

[1]The concept of distance can correspond to different norms in $I\!R^k$ on both sides of the inequality in the definition of the cone $IntD_\varepsilon$, as all norms in $I\!R^k$ are topologically equivalent.

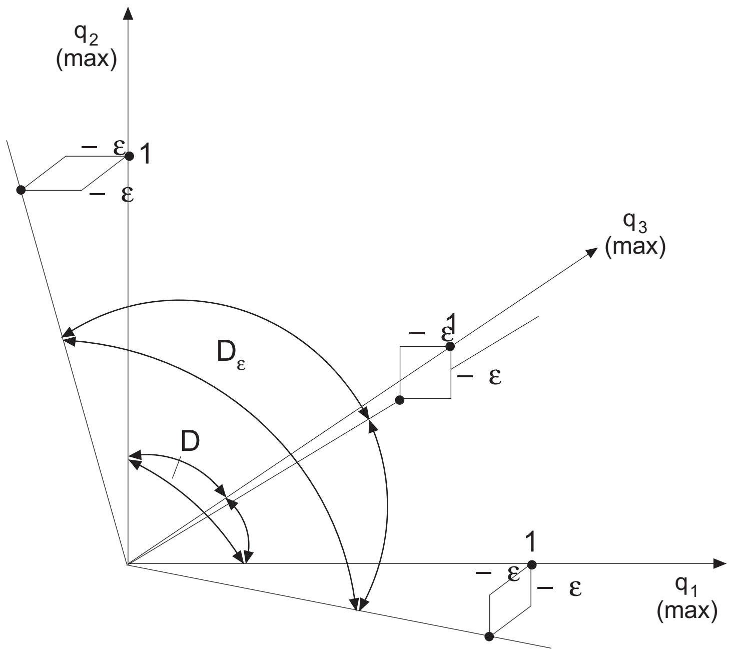


Figure 8.2. An ε-neighborhood of the positive octant in $\mathbb{R}^3$.

The traditional proper efficiency – with only an existential bound on trade-off coefficients – can then be defined as:

$$\hat{Q}_0^p = \bigcup_{\varepsilon>0} \hat{Q}_0^{p\varepsilon}, \; \hat{X}_0^p = \bigcup_{\varepsilon>0} \hat{X}_0^{p\varepsilon}. \tag{8.16}$$

The cone $IntD_\varepsilon$ or its closure D_ε can be better understood by observing that D_ε is simply a "slightly broader" cone than D. For example, if $D = \mathbb{R}_+^k$, in cases of Pareto-optimality, the cone D_ε of the form (8.13) can also be written as:

$$\begin{aligned} D_\varepsilon &= \{\boldsymbol{q} \in \mathbb{R}^k : \boldsymbol{q} = \sum_{j=1}^{k} \lambda_j \boldsymbol{q}_\varepsilon^{(j)}, \, \lambda_j \geq 0\}, \\ \boldsymbol{q}_\varepsilon^{(j)} &= (-\varepsilon, \, -\varepsilon, \ldots 1 + (k-1)\varepsilon_{(j)}, \ldots, -\varepsilon, \, -\varepsilon)^T; \\ D_\varepsilon &= \{\boldsymbol{q} \in \mathbb{R}^k : -q_j \leq \varepsilon \sum_{i=1}^{k} q_i, \, j = 1, \ldots k\} \\ &= \{\boldsymbol{q} \in \mathbb{R}^k : \min_{1 \leq i \leq k} q_i + \varepsilon \sum_{i=1}^{k} q_i \geq 0\}. \end{aligned} \tag{8.17}$$

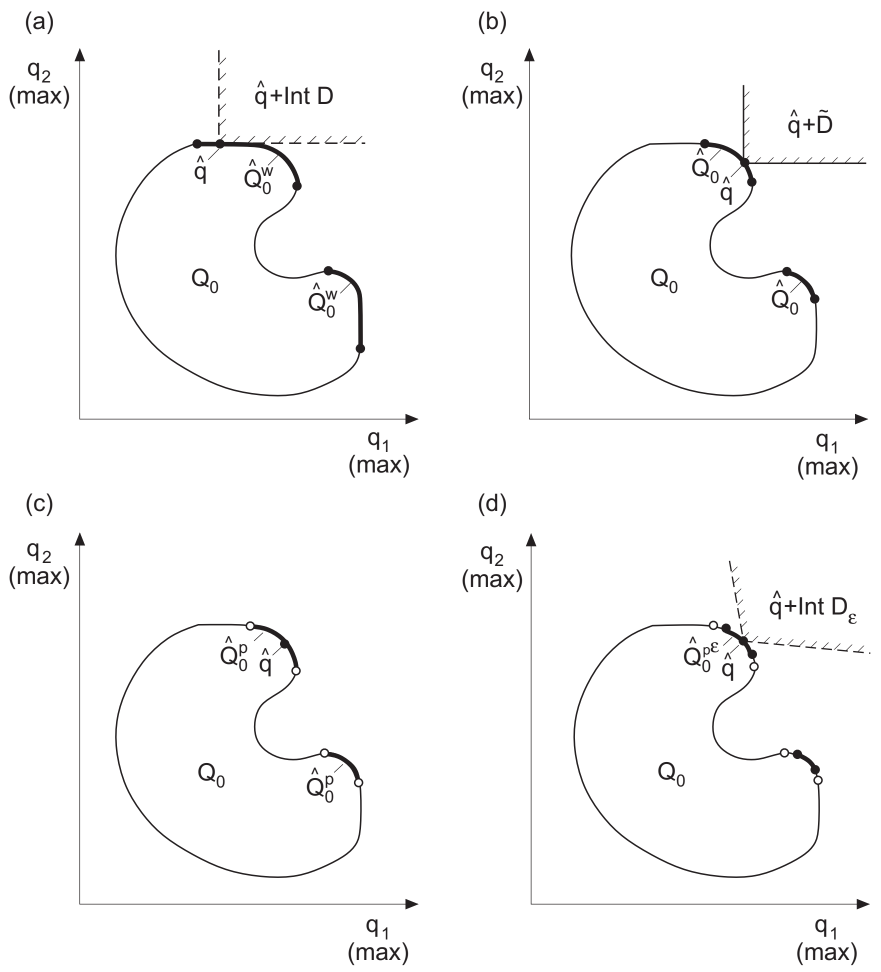


Figure 8.3. Examples of (a) weakly Pareto-optimal, (b) Pareto-optimal, (c) properly Pareto-optimal, and (d) ε-properly Pareto-optimal outcomes for a nonconvex set Q_0.

The last representation is particularly important: D_ε can be represented as a zero-level set of the function

$$\min_{1\le i\le k} q_i + \varepsilon \sum_{i=1}^{k} q_i. \tag{8.18}$$

Figure 8.2 shows an example of the cone D_ε when $k = 3,\ D = \mathbb{R}^3_+$. *Figure 8.3* shows examples of weakly efficient, efficient, properly efficient and ε-properly efficient (i.e., Pareto-optimal) outcomes together with appropriate cones for the case $k = 2,\ D = \mathbb{R}^k_+$, with a nonconvex set Q_0. Note that ε-properly efficient outcomes, although reasonable for applications, might be difficult for computations and controlled selections. In nonconvex cases, the computations might be additionally complicated by the fact that the efficient outcome set (Pareto frontier) might be nonconnected, as illustrated in *Figure 8.3*.

Whether convex or not, the definitions of various types of efficiency imply that:

$$\hat{Q}_0^{p\varepsilon} \subseteq \hat{Q}_0^{p} \subseteq \hat{Q}_0 \subseteq \hat{Q}_0^{w};\ \hat{X}_0^{p\varepsilon} \subseteq \hat{X}_0^{p} \subseteq \hat{X}_0 \subseteq \hat{X}_0^{w}. \tag{8.19}$$

We stress that while simply simulating a model, the modeler selects decisions $\boldsymbol{x} \in X_0$ and then observes their outcomes, not knowing the form of the sets Q_0 or $\hat{Q}_0$ explicitly. Thus, selecting Pareto-optimal decisions and controlling such selections might be difficult for the modeler. Moreover, nonconvexity is typical for more complicated, nonlinear models, or even for linear models but with integer decision variables. Thus, the problems illustrated in *Figure 8.3* are often encountered. The issues of computing efficient decisions and outcomes and controlling their selections will be discussed in the next sections.

8.2 Objective Ranges and Aggregation of Objectives

We already stressed in Chapter 4 that an estimation of objective ranges is crucial for any application of multi-objective model analysis. When some set of variables in a model is specified as objectives q_i, we must know – even if only approximately – the possible ranges of change of these variables. This is also necessary for any aggregation of objectives, i.e., combining them into one function. Since objectives might have various units of measurement, they must be rescaled to dimension-free units before aggregation (e.g., before summation).

We have also stressed that a typical way for performing such an estimation exists. It consists in computing the ideal or utopia point by optimizing each objective separately and then estimating the nadir point (a lower bound on objectives that are maximized, or upper bound on those minimized). While the utopia point components do not usually change, if we alter the number of objectives selected (utopia components might need to be computed for new objectives, but they do not influence old objectives), the nadir point components change.

This is because of the difference in definitions of the utopia and nadir points. The utopia point consists of the best values of objectives in both the Q_0 and $\hat{Q}_0$ sets, but it is simpler to compute the best values in the larger set Q_0. The nadir point consists of the worst values of objectives, but only in the smaller set of efficient outcomes $\hat{Q}_0$; there might be worse values in the set of nonefficient points $Q_0 \setminus \hat{Q}_0$ than at nadir point.

The information contained in the nadir component values is important to the modeler. For example, some maximized objectives might be either unbounded from below or have very low bounds in the model as expressed by the set Q_0. It is interesting how these lower bounds increase if we optimize multi-objectively and consider only the set $\hat{Q}_0$. On the one hand, this set is defined in a much more complicated way than Q_0 and computing the precise lower bounds for $\hat{Q}_0$ is very difficult, with computational complexity exponentially increasing with the number of objectives k. On the other hand, as the modeler usually needs information on the ranges of objectives before analyzing $\hat{Q}_0$ in detail, there is no sense in computing the exact values of such lower bounds. It is sufficient to use approximations of them.

A simple way (although certainly not the best) of approximating nadir components is to take the worst values of objective components that occur during the calculations of the utopia point:

$$\begin{aligned}
q_{i,uto} &= \max_{\boldsymbol{q}\in Q_0} q_i;\;\; \hat{\boldsymbol{q}}^{(i)} = \operatorname*{argmax}_{\boldsymbol{q}\in Q_0} q_i,\;\; i = 1,\ldots k_1 \text{ (max)}\\
q_{i,uto} &= \min_{\boldsymbol{q}\in Q_0} q_i;\;\; \hat{\boldsymbol{q}}^{(i)} = \operatorname*{argmin}_{\boldsymbol{q}\in Q_0} q_i, i = k_1+1,\ldots k_2 \text{ (min)}\\
q_{i,nad}^{(1)} &= \min_{1\le j\le k} \hat{q}_i^{(j)},\;\; i = 1,\ldots k_1 \text{ (max)}\\
q_{i,nad}^{(1)} &= \max_{1\le j\le k} \hat{q}_i^{(j)},\;\; i = k_1+1,\ldots k_2 \text{(min)}.
\end{aligned} \tag{8.20}$$

The worst values $q_{i,nad}^{(1)}$ might be better than the nadir components (they are equal to nadir components only in certain cases, including the case $k = 2$). Thus, in order to estimate the nadir approximately, it is sufficient to increase the range $q_{i,uto} - q_{i,nad}^{(1)}$ somewhat arbitrarily. Counter-examples for any arbitrarily assumed coefficient of increasing this range can be easily found, but taking, for example:

$$q_{i,nad}^{(2)} = 1.25 q_{i,nad}^{(1)} - 0.25 q_{i,uto},\;\; i = 1,\ldots k_2 \text{ (max and min)} \tag{8.21}$$

we obtain a second approximation of the nadir components and an approximation of the range of efficient values of objectives that are sufficient for most applications. There are various ways of further improving estimates of nadir components (see e.g., Lewandowski and Wierzbicki, 1989).

Similarly to trade-off coefficients, it is more difficult to interpret the utopia and nadir point for stabilized objectives. While keeping in mind that such objectives might be either minimized (if above their reference level) or maximized (if below this level), their utopia component can be interpreted as the smallest attainable deviation from the reference level, and their nadir component as the largest attainable deviation. In such a case, however, each change of the reference level would change the utopia and nadir estimates. Therefore, it is sometimes useful to distinguish between objectives that are consistently stabilized and those that are originally maximized or minimized. Adjustments of estimated objective ranges need to be made for the objectives that are stabilized (these might be called goal-type objectives). Objectives that are maximized or minimized may be temporarily stabilized. An initial estimation of a range of the stabilized objective needs to be made, and changed only when this objective is redefined.

As already stressed in Chapter 4, we assume that the modeler has defined – by whatever means – some estimates of ranges of each objective value:

$$q_{i,lo} \le q_i \le q_{i,up} \quad i = 1, \ldots, k \tag{8.22}$$

where $q_{i,up}$ (for maximized objectives, and $q_{i,lo}$ for minimized ones) is at least as high or low as the corresponding utopia point component. The range $q_{i,up} - q_{i,lo}$ should be approximately as large as the range utopia–nadir. Having specified such ranges, we can reduce objectives to dimension-free scales and only then does a discussion of the relative importance of criteria, their weights, and an interpretation of the trade-off coefficients, etc., become meaningful.

Recall that a classical way to compute a Pareto point (for dimension-free objectives or criteria) – assuming that all objectives are maximized – is to maximize a weighted linear scalarizing function:

$$s_1(\boldsymbol{q}, \alpha) = \sum_{i=1}^{k} \alpha_i \frac{q_i - q_{i,lo}}{q_{i,up} - q_{i,lo}}, \quad \alpha_i \ge 0, \quad \sum_{i=1}^{k} \alpha_i = 1 \tag{8.23}$$

where α_i are weighting coefficients. If the objectives are not dimension-free, then we would need additional scaling of weighting coefficients $\beta_i = \alpha_i/(q_{i,up} - q_{i,lo})$. Such weighting coefficients are not dimension-free themselves and thus cannot be interpreted as importance weights.[2]

Each maximal point of such functions (i.e., $s_1(\boldsymbol{F}(\boldsymbol{x}), \alpha)$ maximized with respect to $\boldsymbol{x} \in X_0$) is weakly Pareto-optimal (properly Pareto-optimal if $\alpha_i > 0$, $\forall i = 1, \ldots k$). This results from a general sufficient condition for efficiency or Pareto-optimality: for a maximal point of a function to be efficient (in the sense of a strictly or strongly positive cone $\tilde{D}$ or $IntD$) it is sufficient that the function is strictly monotone with respect to this cone (i.e., $\boldsymbol{q}' - \boldsymbol{q}'' \in \tilde{D}$ or $\boldsymbol{q}' - \boldsymbol{q}'' \in IntD$ implies $s_1(\boldsymbol{q}', \alpha) - s_1(\boldsymbol{q}'', \alpha) > 0$).

If the set Q_0 is convex, then a converse statement is also true: for each weakly Pareto-optimal (and thus also for each Pareto-optimal or properly Pareto-optimal) decision $\hat{\boldsymbol{x}}$ there exist weighting coefficients $\hat{\alpha}_i \ge 0$, $\forall i = 1, \ldots k$ such that the maximum of the function $s_1(\boldsymbol{F}(\boldsymbol{x}), \hat{\alpha})$ with respect to $\boldsymbol{x} \in X_0$ is attained at $\hat{\boldsymbol{x}}$. This is a result of another general property – the Hahn-Banach theorem on the existence of linear functions separating disjoint convex sets – which is generally

[2]This point is not sufficiently addressed in many books on multi-objective decision support, although they do often concentrate on the use of importance weights (e.g., Janssen, 1992).

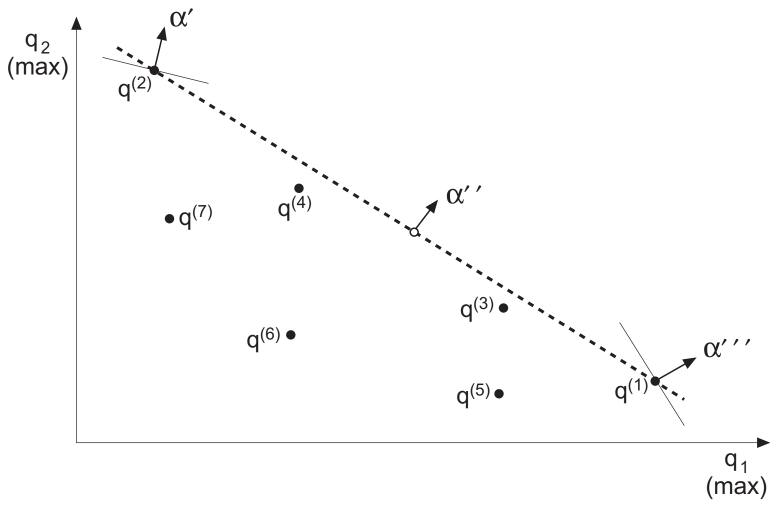


Figure 8.4. An illustration of the inadequacy of a linear aggregation of objectives for discrete sets of decisions and outcomes.

used for deriving the necessary conditions of optimality (not only in the multi-objective case).

Thus, it might seem that the weighted linear scalarizing function covers all that is needed for multi-objective optimization. However, there are at least two important reasons to consider such functions as highly inadequate. First is that the necessary condition of efficiency stated with the help of the linear scalarizing function does not account for nonconvex (in particular discrete) sets Q_0. As an example, consider a set Q_0 with seven discrete outcomes of seven possible decisions, each evaluated by two criteria q_1 and q_2 (both assumed to be maximized), presented in *Figure 8.4*. Out of these seven outcomes, $\boldsymbol{q}^{(5)}$, $\boldsymbol{q}^{(6)}$, and $\boldsymbol{q}^{(7)}$ are Pareto-dominated, the remaining are all Pareto-optimal. However, if we use the maximization of a linear weighted combination of q_1 and q_2 to select the Pareto-optimal decisions, we obtain only the extreme cases $\boldsymbol{q}^{(1)}$ and $\boldsymbol{q}^{(2)}$. The more balanced cases $\boldsymbol{q}^{(3)}$ and $\boldsymbol{q}^{(4)}$, although Pareto-optimal, will never be selected, regardless of how the weighting coefficients are changed.[3] We say that such decision outcomes are not supported by a linear function.

The second reason for the inadequacy of a weighted sum for multi-objective analysis is the fact that maximal points of linear scalarizing functions depend discontinuously on the weighting coefficients, even in the most classical, convex case

[3]This decision could involve the selection of a partner for marriage, e.g., where q_1 denotes sex-appeal and q_2 intelligence (or the quality of cooking in a more sexist interpretation). This is known as the Korhonen paradox: if we used only linear approximations of our preferences, we would be attracted only by extremes and most of us would remain unmarried.

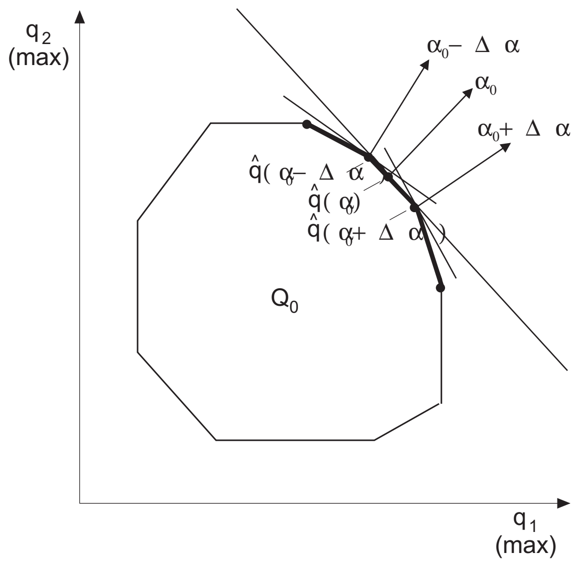


Figure 8.5. An illustration of the discontinuity of maximal points of a linear aggregation of objectives in the case of convex polyhedral sets of decisions and outcomes.

of multi-objective linear programming with polyhedral sets X_0 and Q_0. This is illustrated in *Figure 8.5*: changing the weighting coefficients means changing the slope of the line $s_1(\boldsymbol{q}, \alpha) =$ const. At the value α_0 of weighting coefficients, this line includes a facet of the set Q_0; slight changes of α close to α_0 result in jumps toward the corners or end-points of this facet.

Both these reasons justify a conclusion: the weighted sum should be used as a way of aggregating multiple objectives only if we can be sure that this would not result in paradoxical conclusions. Since we can be sure only in special cases, it is usually better to consider other functions as means of aggregation. In multi-objective optimization theory, and particularly in goal programming approaches (see e.g., Charnes and Cooper, 1977), much attention has been given to the use of a norm that characterizes a distance of an attainable outcome $\boldsymbol{q} \in Q_0$ from a reference point $\bar{\boldsymbol{q}} \in \mathbb{R}^k$:

$$s_2(\boldsymbol{q}, \bar{\boldsymbol{q}}) = \| \ \boldsymbol{q} - \bar{\boldsymbol{q}} \ \| \ . \tag{8.24}$$

Even if we assume that all objectives are maximized, the distance function should be minimized. Various norms (Euclidean, l_1, l_p, l_∞ called Chebyshev, and mixed l_1 and l_∞ called augmented Chebyshev) can be used.

Unfortunately, for reasons already discussed in Chapter 4, the use of such a distance function is not consistent with the concept of Pareto-optimality or efficiency.

If we would like to use the reference point as the main parameter to control the selection of Pareto-optimal solutions, we might like to admit the use of any reasonable reference point, at least in the range $\boldsymbol{q}_{lo}$, $\boldsymbol{q}_{up}$, including such reference points that are attainable or not, i.e., $\bar{\boldsymbol{q}} \notin Q_0$ and $\bar{\boldsymbol{q}} \in Q_0$. However, the minimal points of such a distance function for $\bar{\boldsymbol{q}} \in Q_0 \setminus \hat{Q}_0$ are not Pareto-optimal, for the simple reason that a norm changes its monotonicity properties at zero, i.e., at $\boldsymbol{q} = \bar{\boldsymbol{q}}$. Therefore, goal programming techniques based on the minimization of such distance functions do not result, in general, in Pareto-optimal solutions. Pareto-optimality can be secured, even if Q_0 is convex, only if $\bar{\boldsymbol{q}} \notin Q_0 \setminus \hat{Q}_0$ and $\bar{\boldsymbol{q}}$ are "above to the north-east" of Q_0 (see e.g., Wierzbicki, 1986).

In a more general, nonconvex case, Pareto-optimality of the minimal points of a distance function results from the assumption that $\bar{\boldsymbol{q}} \in \boldsymbol{q}_{uto} + D$, which is the case of the displaced ideal of Zeleny (1974). However, such reference points are very distant from the attainable outcome set and thus are difficult to be used as the main parameters controlling the selection of Pareto-optimal outcomes. For this reason, weighting coefficients were additionally used in the displaced ideal and other similar techniques in order to modify the distance function and to control the selection of its minima. For example, if an l_p norm is used with weighting coefficients, the corresponding distance function has the form:

$$s_3(\boldsymbol{q}, \bar{\boldsymbol{q}}, \alpha) = (\sum_{i=1}^{k} \alpha_i \mid q_i - \bar{q}_i \mid^p)^{1/p}. \tag{8.25}$$

The use of weighting coefficients in a distance function creates, however, problems similar to their use in a linear function.[4]

Since reference points usually provide much better interpretations for the modeler than weighting coefficients, it would be better to use the reference points as the main parameters controlling the selection of Pareto-optimal points, provided we could secure Pareto-optimality of the resulting decisions. Therefore, as already stressed in Chapter 4, instead of a distance function we must use a different function. This is called an achievement scalarizing function or an order-consistent achievement scalarizing function which, as opposed to a distance function, preserves the monotonicity even if the point $\boldsymbol{q} = \bar{\boldsymbol{q}}$ is crossed and has certain additional properties. In the next section, we explain how such a function can be constructed for general cases of maximized, minimized, and stabilized objectives.

[4]If $p > 1$, then the use of weighting coefficients to control the selection of efficient points might not so easily result in discontinuity, as in the linear case; however, the dependence of efficient points on weighting coefficients does not remain transparent enough. There are cases where changing weighting coefficients produces the reverse effects from those expected (see e.g., Nakayama, 1994).

8.3 Reference Points and Achievement Functions

We assume that for each objective that can be maximized, minimized, or stabilized, reference levels in the form of either aspiration levels $\bar{q}_i$ (which would be good to achieve) or reservation levels $\bar{\bar{q}}_i$ (which should be avoided if at all possible) are specified by the modeler. These reference levels will be used as the main interaction parameters by which the modeler controls the selection of decisions and their outcomes. The values of these reference levels are subject to reasonability constraints only, given lower and upper bounds $q_{i,lo},\ q_{i,up}$ for each objective:

$$\begin{array}{lllllllll} q_{i,lo} & < & \bar{\bar{q}}_i & < & \bar{q}_i & < & q_{i,up}, & i = 1,\ldots,k_1 & \text{(max)} \\ q_{i,lo} & < & \bar{q}_i & < & \bar{\bar{q}}_i & < & q_{i,up}, & i = k_1+1,\ldots,k_2 & \text{(min)}\,. \end{array} \tag{8.26}$$

For stabilized outcomes we can use two pairs of reservation and aspiration levels: one “lower” pair $\bar{\bar{q}}_{i,lo} < \bar{q}_{i,lo}$ as if for maximized outcomes and one “upper” pair $\bar{q}_{i,up} < \bar{\bar{q}}_{i,up}$ as if for minimized ones.

A way of aggregating the objectives into an order-consistent achievement function[5] consists in specifying partial achievement functions $\sigma_i(q_i, \bar{q}_i)$ or $\sigma_i(q_i, \bar{q}_i, \bar{\bar{q}}_i)$. These functions should:

- Be strictly monotone with respect to the specified partial order – increasing for maximized objectives, decreasing for minimized ones, increasing below the (lower) aspiration level, and decreasing above the (upper) aspiration level for stabilized ones.
- Assume value 0 if $q_i = \bar{q}_i \ \ \forall i = 1,\ldots,k$ and only aspiration levels are used, or assume value 0 if $q_i = \bar{\bar{q}}_i \ \ \forall i = 1,\ldots,k$ and assume value 1 if $q_i = \bar{q}_i \ \ \forall i = 1,\ldots,k$, if both aspiration and reservation levels are used.

This seeming inconsistency results from the fact that the number 0 is more important than the number 1: if only aspiration levels are used, we just check, with the help of the sign of an achievement function, whether they could be reached. In such a case, it is useful to define partial achievement functions with a slope that is larger if the aspiration levels are closer to their extreme levels:

$$\begin{array}{ccl} \begin{array}{c}\sigma_i(q_i,\bar{q}_i)\\ \text{(max)}\end{array} & = & (q_i-\bar{q}_i)/(q_{i,up}-\bar{q}_i) \\[2ex] \begin{array}{c}\sigma_i(q_i,\bar{q}_i)\\ \text{(min)}\end{array} & = & (\bar{q}_i-q_i)/(\bar{q}_i-q_{i,lo}) \\[2ex] \begin{array}{c}\sigma_i(q_i,\bar{q}_i)\\ \text{(stab)}\end{array} & = & \left\{\begin{array}{l} (\bar{q}_i-q_i)/(q_{i,up}-\bar{q}_i),\ \text{if } q_i > \bar{q}_i \\ (q_i-\bar{q}_i)/(\bar{q}_i-q_{i,lo}),\ \text{if } q_i \le \bar{q}_i \end{array}\right\} \end{array} \tag{8.27}$$

[5]For a more detailed theory of such functions, see e.g., Wierzbicki (1986).

where $\bar{q}_{i,lo} = \bar{q}_{i,up} = \bar{q}_i$ was assumed for stabilized objectives. An alternative way is to use piece-wise linear functions, e.g., to change the slope of the partial achievement function depending on whether the current point is above or below the aspiration point:

$$
\begin{array}{ccl}
\underset{\text{(max)}}{\sigma_i(q_i,\bar{q}_i)} & = & \left\{\begin{array}{lccccc} (q_i-\bar{q}_i)/(q_{i,up}-\bar{q}_i), & \bar{q}_i & \le & q_i & \le & q_{i,up} \\ \beta(q_i-\bar{q}_i)/(\bar{q}_i-q_{i,lo}), & q_{i,lo} & < & q_i & \le & \bar{q}_i \end{array}\right\} \\
\underset{\text{(min)}}{\sigma_i(q_i,\bar{q}_i)} & = & \left\{\begin{array}{lccccc} (\bar{q}_i-q_i)/(\bar{q}_i-q_{i,lo}), & q_{i,lo} & \le & q_i & \le & \bar{q}_i \\ \beta(\bar{q}_i-q_i)/(q_{i,lo}-\bar{q}_i), & \bar{q}_i & < & q_i & \le & q_{i,up} \end{array}\right\} \\
\underset{\text{(stab)}}{\sigma_i(q_i,\bar{q}_i)} & = & \left\{\begin{array}{lccccc} (q_{i,up}-q_i)/(q_{i,up}-\bar{q}_i), & \bar{q}_i & \le & q_i & \le & q_{i,up} \\ (q_i-q_{i,lo})/(\bar{q}_i-q_{i,lo}), & q_{i,lo} & \le & q_i & < & \bar{q}_i \end{array}\right\}
\end{array}
\tag{8.28}
$$

where the coefficient $\beta > 0$ is selected in such a way that the functions are not only monotone, but also concave (thus can be expressed as minima of their component linear functions, which is helpful for their use together with linear models). Again, $\bar{q}_{i,lo} = \bar{q}_{i,up} = \bar{q}_i$ was assumed for stabilized objectives, though we could make a different assumption, define $\bar{q}_i = 0.5(\bar{q}_{i,lo} + \bar{q}_{i,up})$ and define the piece-wise partial achievement function for stabilized objectives similarly as for maximized or minimized ones.

If both aspiration and reservation levels are used, it is more useful to define the partial achievement functions as piece-wise linear functions, e.g., of the form:

$$
\begin{array}{ccl}
\underset{\text{(max)}}{\sigma_i(q_i,\bar{q}_i,\bar{\bar{q}}_i)} & = & \left\{\begin{array}{cccccc} 1+\alpha(q_i-\bar{q}_i)/(q_{i,up}-\bar{q}_i), & \bar{q}_i & \le & q_i & \le & q_{i,up} \\ (q_i-\bar{\bar{q}}_i)/(\bar{q}_i-\bar{\bar{q}}_i), & \bar{\bar{q}}_i & < & q_i & < & \bar{q}_i \\ \beta(q_i-\bar{\bar{q}}_i)/(\bar{\bar{q}}_i-q_{i,lo}), & q_{i,lo} & \le & q_i & \le & \bar{\bar{q}}_i \end{array}\right\} \\
\underset{\text{(min)}}{\sigma_i(q_i,\bar{q}_i,\bar{\bar{q}}_i)} & = & \left\{\begin{array}{cccccc} 1+\alpha(\bar{q}_i-q_i)/(\bar{q}_i-q_{i,lo}), & q_{i,lo} & \le & q_i & \le & \bar{q}_i \\ (\bar{\bar{q}}_i-q_i)/(\bar{\bar{q}}_i-\bar{q}_i), & \bar{q}_i & < & q_i & < & \bar{\bar{q}}_i \\ \beta(\bar{\bar{q}}_i-q_i)/(q_{i,lo}-\bar{\bar{q}}_i), & \bar{\bar{q}}_i & \le & q_i & \le & q_{i,up} \end{array}\right\} \\
\underset{\text{(stab)}}{\sigma_i(q_i,\bar{q}_i,\bar{\bar{q}}_i)} & = & \left\{\begin{array}{cccccc} \beta(\bar{\bar{q}}_{i,up}-q_i)/(q_{i,up}-\bar{\bar{q}}_{i,up}), & \bar{\bar{q}}_{i,up} & \le & q_i & \le & q_{i,up} \\ (\bar{\bar{q}}_{i,up}-q_i)/(\bar{\bar{q}}_{i,up}-\bar{q}_i), & \bar{q}_i & < & q_i & < & \bar{\bar{q}}_{i,up} \\ (q_i-\bar{\bar{q}}_{i,lo})/(\bar{q}_i-\bar{\bar{q}}_{i,lo}), & \bar{\bar{q}}_{i,lo} & < & q_i & < & \bar{q}_i \\ \beta(q_i-\bar{\bar{q}}_{i,lo})/(\bar{\bar{q}}_{i,lo}-q_{i,lo}), & q_{i,lo} & \le & q_i & \le & \bar{\bar{q}}_{i,lo} \end{array}\right\}.
\end{array}
\tag{8.29}
$$

The coefficients α, β should be positive and chosen in such a way that partial achievement functions are not only monotone, but also concave. Other forms of

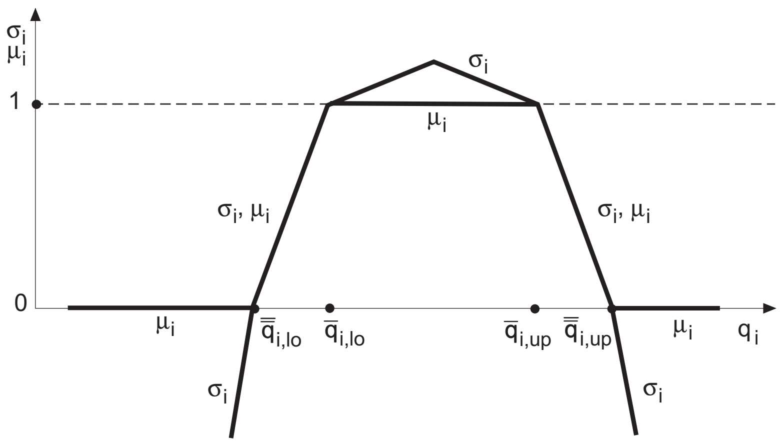


Figure 8.6. The difference between a partial achievement function $\sigma_i(q_i, \bar{q}_i, \overline{\overline{q}}_i)$ and a corresponding fuzzy membership function $\mu_i(q_i, \bar{q}_i, \overline{\overline{q}}_i)$.

piece-wise linear partial achievement functions satisfying these conditions are also possible, e.g., an achievement function for stabilized objectives might be defined as greater than 1 inside the interval $[\bar{q}_{i,lo}; \bar{q}_{i,up}]$ if $\bar{q}_{i,lo} < \bar{q}_{i,up}$ (see *Figure 8.6*).

If the values of $\sigma_i(q_i, \bar{q}_i, \overline{\overline{q}}_i)$ are restricted to the interval [0;1], then they can be interpreted as fuzzy membership functions $\mu_i(q_i, \bar{q}_i, \overline{\overline{q}}_i)$ (see e.g., Zadeh, 1978; Seo and Sakawa, 1988; Zimmermann and Sebastian, 1994), which express the degree of satisfaction of the modeler with the value of the objective q_i. More complicated forms of such fuzzy membership functions can also be used (see e.g., Vincke, 1992; Fodor and Roubens, 1994; Granat *et al.*, 1994); for simplicity, we do not consider these more complicated forms.

A partial achievement function can be viewed as a nonlinear transformation of the objective range satisfying some monotonicity requirements. The essential issue is how to aggregate these functions in order to obtain a scalarizing achievement function with good properties for multi-objective analysis. There are several ways of executing such an aggregation. One way is to use fuzzy logic and select an appropriate representation of the fuzzy “and” operator.[6] The simplest operator of this type is the minimum operator:

$$\mu(\boldsymbol{q}, \bar{\boldsymbol{q}}, \overline{\overline{\boldsymbol{q}}}) = \bigwedge_{1 \le i \le k} \mu_i(q_i, \bar{q}_i, \overline{\overline{q}}_i) = \min_{1 \le i \le k} \mu_i(q_i, \bar{q}_i, \overline{\overline{q}}_i) \tag{8.30}$$

[6]In selecting the fuzzy “and” operator for aggregation, we assume that all objectives are similarly important and noncompensative. This assumption is fully justified in multi-objective model analysis (we do not ask the modeler for reasons why she/he has selected a given set of objectives), but it might not be satisfied in other cases of aggregation of attributes.

which, however, would result only in weakly Pareto-optimal or weakly efficient outcomes when used for multi-objective analysis. To secure ε-properly efficient outcomes, we augment this operator by some linear part (compare the last expression for the cone D_ε in (8.17)). The corresponding overall membership function would then have the form:

$$\mu(\boldsymbol{q}, \bar{\boldsymbol{q}}, \overline{\overline{\boldsymbol{q}}}) = (\min_{1 \le i \le k} \mu_i(q_i, \bar{q}_i, \overline{\overline{q}}_i) + \varepsilon \sum_{i=1}^{k} \mu_i(q_i, \bar{q}_i, \overline{\overline{q}}_i))/(1 + k\varepsilon). \tag{8.31}$$

An interpretation in terms of membership functions can be used in a graphic interaction with the modeler; however, membership functions $\mu_i(q_i, \bar{q}_i, \overline{\overline{q}}_i)$ and $\mu(\boldsymbol{q}, \bar{\boldsymbol{q}}, \overline{\overline{\boldsymbol{q}}})$ are not strictly monotone if they are equal to 0 or 1. Therefore, inside a multi-objective optimization system, a slightly different overall achievement function must be used, with values not restricted to the interval [0;1]:

$$\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}, \overline{\overline{\boldsymbol{q}}}) = (\min_{1 \le i \le k} \sigma_i(q_i, \bar{q}_i, \overline{\overline{q}}_i) + \varepsilon \sum_{i=1}^{k} \sigma_i(q_i, \bar{q}_i, \overline{\overline{q}}_i))/(1 + k\varepsilon). \tag{8.32}$$

In both of the above equations, $\varepsilon > 0$ is the same coefficient as the one used when defining the proper ε-efficiency, with a prior bound $M = 1 + 1/\varepsilon$ on corresponding trade-off coefficients. This bound limits trade-off coefficients not between various objectives q_i and q_j, but between their transformed values $\sigma_i(q_i, \bar{q}_i, \overline{\overline{q}}_i)$ and $\sigma_j(q_j, \bar{q}_j, \overline{\overline{q}}_j)$. In order to obtain bounds on original trade-off coefficients between q_i and q_j, it is necessary to consider the corresponding slopes of partial achievement functions. However, if these slopes have prior bounds, the original trade-off coefficients will also have prior bounds.

The above derivation of an order-consistent achievement function from a fuzzy "and" operator is not the only one possibility. Simpler versions of order-consistent achievement functions were used originally. Some of these versions can be considered as a simplification of function (8.32). For example, suppose only aspiration levels $\bar{q}_i$ are used, all objectives are maximized and dimension-free, and the partial achievement functions have a simple form $\sigma_i(q_i, \bar{q}_i) = q_i - \bar{q}_i$. The order-consistent achievement function then takes the form already discussed in Chapter 4:

$$\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}) = \min_{1 \le i \le k} (q_i - \bar{q}_i) + \varepsilon \sum_{i=1}^{k} (q_i - \bar{q}_i). \tag{8.33}$$

We do not have to subdivide this function by $1 + k\varepsilon$ because only the value 0 of this function, and not 1, is significant. This function can be seen as a prototype order-consistent achievement scalarizing function. It is monotone with respect to the cone $IntD_\varepsilon$ and its zero-level set represents this cone (see (8.17)):

$$\bar{\boldsymbol{q}} + IntD_\varepsilon = \{\boldsymbol{q} \in \mathbb{R}^k : \ \sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}) > 0\}. \tag{8.34}$$

Other order-consistent achievement functions similar to (8.32) were also used in reference point methodology or other similar approaches to multi-objective optimization (see e.g., Sawaragi *et al.*, 1985; Steuer, 1986; Wierzbicki, 1986, 1992b).

As function (8.32) is also strictly monotone with respect to the cone $IntD_\varepsilon$, we have the following:

• *Sufficient condition for ε-proper efficiency.* For any $\bar{\boldsymbol{q}}, \bar{\bar{\boldsymbol{q}}}$ (with components strictly contained in the ranges $[q_{i,lo}; q_{i,up}]$) a maximal point of $\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}, \bar{\bar{\boldsymbol{q}}})$ with respect to $\boldsymbol{q} \in Q_0 = \boldsymbol{F}(X_0)$ is a properly efficient objective vector with a prior bound on trade-off coefficients and, equivalently, a maximal point of $\sigma(\boldsymbol{F}(\boldsymbol{x}), \bar{\boldsymbol{q}}, \bar{\bar{\boldsymbol{q}}})$ with respect to $\boldsymbol{x} \in X_0$ is a properly efficient decision with a prior bound.

In order to derive a corresponding necessary condition, consider $\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}, \bar{\bar{\boldsymbol{q}}})$ as a function not of $\boldsymbol{q} = (q_1, \ldots q_i, \ldots q_k)^T$ but of their transformed values $\boldsymbol{w} = (w_1, \ldots w_i, \ldots w_k)^T, w_i = \sigma_i(q_i, \bar{q}_i, \bar{\bar{q}}_i)$. In the transformed space, the reservation point $\bar{\bar{\boldsymbol{w}}} = 0$, because $\bar{\bar{w}}_i = \sigma_i(\bar{\bar{q}}_i, \bar{q}_i, \bar{\bar{q}}_i) = 0$ (if only aspiration levels are used, a similar argument applies to them). Denote by $\rho(\boldsymbol{w}) = \sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}, \bar{\bar{\boldsymbol{q}}})$ the achievement scalarizing function in the transformed space. Then, according to (8.17), we can write the cone D_ε (with its vertex shifted to $\bar{\bar{\boldsymbol{w}}}$, which is conveniently equal to 0 in this case) in the following form:

$$\bar{\bar{\boldsymbol{w}}} + D_\varepsilon = \{\boldsymbol{w} \in \mathbb{R}^k : \ \rho(\boldsymbol{w}) \geq 0\}. \tag{8.35}$$

When taking into account the monotonicity of $\rho(\boldsymbol{w})$, we obtain similarly to (8.34) the following:

$$\bar{\bar{\boldsymbol{w}}} + IntD_\varepsilon = \{\boldsymbol{w} \in \mathbb{R}^k : \ \rho(\boldsymbol{w}) > 0\}. \tag{8.36}$$

Suppose $\hat{\boldsymbol{q}} = \boldsymbol{F}(\hat{\boldsymbol{x}})$ is a properly efficient outcome of an admissible decision $\hat{\boldsymbol{x}} \in Q_0$ with such bounds on trade-off coefficients that they are less than $M = 1 + 1/\varepsilon$ in the transformed space of $w_i = \sigma_i(q_i, \bar{q}_i, \bar{\bar{q}}_i)$. Let us shift the reservation point to this properly efficient point, $\bar{\bar{\boldsymbol{q}}} = \hat{\boldsymbol{q}}$. According to the definition of the ε-proper efficiency, the cone $\bar{\bar{\boldsymbol{w}}} + IntD_\varepsilon$ cannot intersect the (transformed by $w_i = \sigma_i(q_i, \bar{q}_i, \bar{\bar{q}}_i)$) set Q_0. However, relation (8.36) indicates that, in such a case, $\hat{\boldsymbol{w}} = \bar{\bar{\boldsymbol{w}}} = 0$ corresponding to $\hat{\boldsymbol{q}} = \bar{\bar{\boldsymbol{q}}}$ will be a maximal point of $\rho(\boldsymbol{w})$ in the transformed set Q_0, or, equivalently, $\hat{\boldsymbol{q}}$ will be a maximal point of $\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}, \bar{\bar{\boldsymbol{q}}})$ with respect to $\boldsymbol{q} \in Q_0$.

This method of deriving the necessary conditions of efficiency is actually an adaptation of the concept of separation of sets to the case of nonlinear separating functions that represent conical sets. The function $\rho(\boldsymbol{w})$ separates (by a cone) the sets $\bar{\bar{\boldsymbol{w}}} + D_\varepsilon$ and the transformed Q_0, if $\bar{\bar{\boldsymbol{q}}} = \hat{\boldsymbol{q}}$ (see Wierzbicki, 1992b). Therefore, we have the necessary condition for ε-proper efficiency:

• *Necessary condition for ε-proper efficiency.* For any properly efficient $\hat{\boldsymbol{q}} = \boldsymbol{F}(\boldsymbol{x})$ with appropriate prior bounds on trade-off coefficients, there exist $\overline{\overline{\boldsymbol{q}}}$ and/or $\bar{\boldsymbol{q}}$ such that $\hat{\boldsymbol{q}}$ maximizes $\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}, \overline{\overline{\boldsymbol{q}}})$ with respect to $\boldsymbol{q} \in Q_0 = \boldsymbol{F}(X_0)$.

It is possible to prove even more (see Wierzbicki, 1986): the modeler can influence the selection of $\hat{\boldsymbol{q}} = \boldsymbol{F}(\hat{\boldsymbol{x}})$ Lipschitz-continuously by changing $\overline{\overline{\boldsymbol{q}}}$ and/or $\bar{\boldsymbol{q}}$ (except in cases when the set of properly efficient objectives is disconnected). This selection is continuously controllable.

Moreover, the scaling of the partial achievement functions and the scalarizing achievement function is such that the modeler can easily draw the following conclusions:

• *The attainability of reservation and/or aspiration points.* If the maximal value of $\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}, \overline{\overline{\boldsymbol{q}}})$ with respect to $\boldsymbol{q} \in Q_0 = \boldsymbol{F}(X_0)$ is below 0, it indicates that the reservation point is not attainable, $\overline{\overline{\boldsymbol{q}}} \notin Q_0 = \boldsymbol{F}(X_0)$, and also that there are no points $\boldsymbol{q} \in Q_0$ dominating $\overline{\overline{\boldsymbol{q}}}$, i.e., $\{\boldsymbol{q} \in I\!R^k : \boldsymbol{q} \geq \overline{\overline{\boldsymbol{q}}}\} \cap Q_0 = \emptyset$. If this maximal value is 0, it indicates that the reservation point is attainable and properly efficient. If this maximal value is 1, the aspiration point $\bar{\boldsymbol{q}}$ is attainable and properly efficient. For values between 0 and 1, the values of all criteria are better than the corresponding components of the reservation point. For values above 1, they are better than the aspiration point. If we use aspiration levels alone, there is only one critical value 0 of the achievement function corresponding to the aspiration point $\bar{\boldsymbol{q}}$.

This property justifies the name "achievement function" because its values measure the achievement as compared to aspiration and reservation points. The name "order-consistent" achievement scalarizing function is used to indicate that the function is strictly monotone with respect to the cone $IntD_\varepsilon$, hence it preserves the (partial) order implied by the cone, and its zero-level-set corresponds to the set $\overline{\overline{\boldsymbol{q}}} + D_\varepsilon$, therefore it represents the order implied by the cone.

The achievement function $\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}, \overline{\overline{\boldsymbol{q}}})$ – and other similar functions – is nondifferentiable. Moreover, the maximum of this achievement function is in most cases attained at its corner, i.e., at the point of nondifferentiability. This is indicated in *Figure 8.7(a,b,c)*, where the loci of the corner points of such achievement functions are shown (indicated by dashed lines) in three cases. In *Figure 8.7(a)*, the partial achievement function is based only on aspiration levels and has the simple form (8.27). In *Figure 8.7(b)*, the partial achievements function is also based only on aspiration levels, but has the slightly more complicated piece-wise linear form (8.28). Finally, in *Figure 8.7(c)*, the partial achievement functions are based on both aspiration and reservation levels and have the more complicated form (8.29). In all cases, it is assumed that the loci of the corner points of achievement functions intersect the properly efficient frontier $\hat{Q}_0^{p\varepsilon}$; in such a case, the maxima of the achievement functions coincide with these intersection points (the consequences of the converse assumption are shown in *Figure 8.9*). Note that the changes of $\bar{\boldsymbol{q}}$

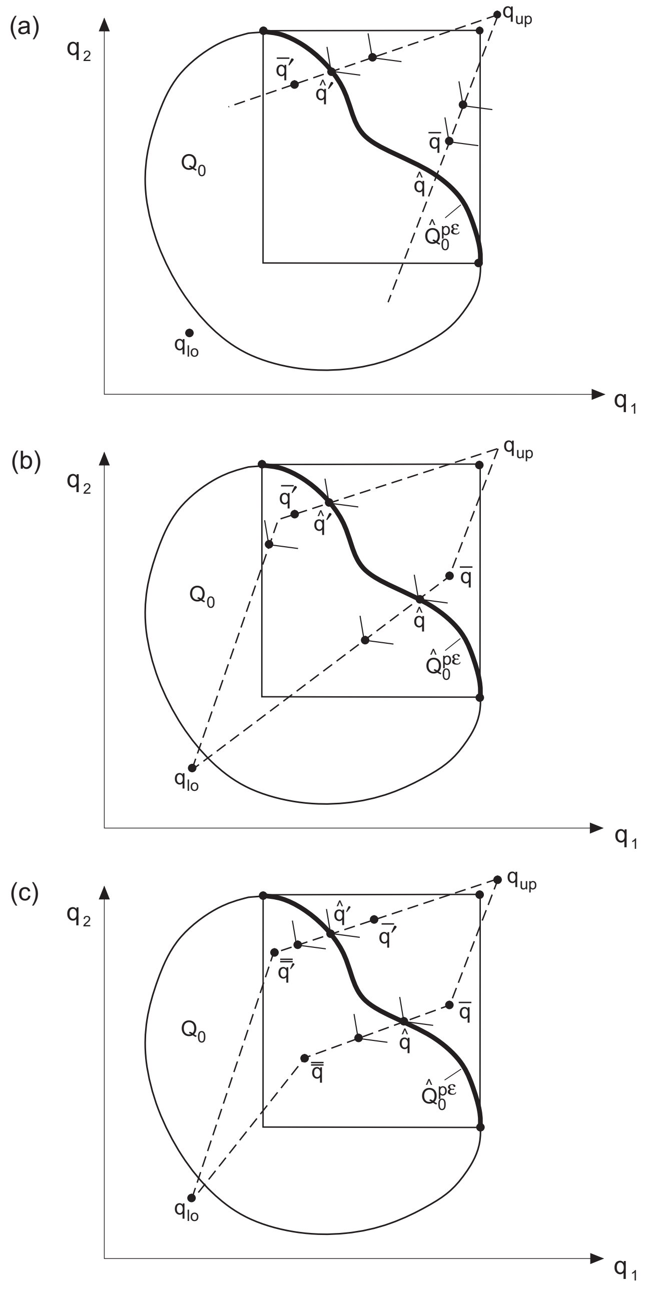


Figure 8.7. The impact of various types of achievement functions (with partial achievement functions composed of linear, two pieces of linear, and three pieces of linear functions, cases *a, b, c*, respectively) on controlling the selection of properly efficient solutions by aspiration and/or reservation levels.

in *Figure 8.7(a,b)*, or the changes of $\bar{\boldsymbol{q}}$ and $\bar{\bar{\boldsymbol{q}}}$ in *Figure 8.7(c)* make it possible to smoothly and completely explore the set $\hat{Q}_0^{p\varepsilon}$. However, in order to do so, we must be able to maximize a nondifferentiable function.

In the case of linear models, the nondifferentiability of the achievement function $\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}, \bar{\bar{\boldsymbol{q}}})$ is unimportant, because the function is concave and its maximization can be equivalently expressed as a linear programming problem by introducing dummy variables (see e.g., Steuer, 1986; Lewandowski and Wierzbicki, 1989, or Sections 7.2 and 9.3.4). In the case of nonlinear models, however, optimization algorithms for smooth functions are more robust (i.e., work more reliably without the necessity of adjusting their specific parameters to obtain results) than algorithms for nonsmooth functions. Therefore, there are two approaches to the maximization of such achievement functions. The first is to introduce additional constraints and dummy variables as for linear models. The second is a useful modification of the achievement function by its smooth approximation, which can be defined, e.g., when using an l_p norm (with $p > 2$, because a circle or a ball do not approximate the piece-wise linear achievement function very effectively; it would be best approximated by very large p, but usually $p = 4 \ldots 8$ suffices, as larger p result in badly conditioned optimization problems). We provide an example of such an approximation only for the case of using the aspiration point $\bar{\boldsymbol{q}}$ alone, assuming that partial achievement functions $\sigma_i(q_i, \bar{q}_i) \leq 1$ (e.g., $\sigma_i(q_i, \bar{q}_i) = 1$ if $q_i = q_{i,up}$ for maximized objectives):

$$\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}) = 1 - (\frac{1}{k}\sum_{i=1}^{k}(1 - \sigma_i(q_i, \bar{q}_i))^p)^{1/p} \tag{8.37}$$

although a similar formula can also be given when using both $\bar{\boldsymbol{q}}$ and $\bar{\bar{\boldsymbol{q}}}$ (see Granat *et al.*, 1994).

Note that $\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}})$ is not, in general, a norm or a distance function between $\boldsymbol{q}$ and $\bar{\boldsymbol{q}}$; it might be equivalent to such a distance function only if all objectives are stabilized. As discussed above, a norm would not preserve needed properties of monotonicity for maximized or minimized objectives. Therefore, the use of order-consistent achievement functions can be interpreted as generalized goal programming, improved by providing efficient solutions for arbitrary reservation and/or aspiration points and equivalent to goal programming in the specific case of stabilizing all objectives. The possibility of treating the reference point approach and order-consistent achievement functions as a generalization of goal programming was noted by Korhonen (Korhonen and Laakso, 1986).

However, the reference point approach has relations to many other techniques of multi-objective optimization. A popular technique (see e.g., Polak, 1976) is to maximize the minimal of maximized objectives or of their increases above postulated reservation levels.[7] This corresponds to maximizing the achievement function (8.33) with $\varepsilon = 0$ which, however, guarantees only weak Pareto-optimality.

Another frequently used technique, underlying the surrogate worth trade-off method by Haimes and Hall (1974), is to optimize one objective while imposing constraints on other objectives with some assumed constraining levels (which can be interpreted as reservation levels). Such a technique – without additional tests – might also result in weak Pareto-optimality; moreover, if the reservation levels are not attainable, this results in a badly posed optimization problem, with an empty set of admissible solutions. The achievement function (8.33) can be interpreted[8] also as a penalty function for such constrained formulation, thus regularizing badly posed cases.

Throughout our discussion we have viewed reference points (reservation and/or aspiration) as if they were simple collections of their components. However, for more complicated models – e.g., with a dynamic structure – it is often advantageous to use reference profiles or reference trajectories of the same outcome variable changing, e.g., over time. Suppose that a model describes the ecological quality of forests in a region or country (Kallio *et al.*, 1980), the expected demand for wood, forestry strategies, and the projected prices for some longer time – say, the next fifty years because of the slow growth dynamics of forests. The modeler would then interpret all model variables and outcomes as their profiles over time or trajectories, rather than as separate numbers in given years. Mathematically, we can represent such a profile as a vector, e.g., in fifty-dimensional space, hence the methodology presented above is fully applicable. From the modeler's point of view, however, it is much easier to interpret model outcomes and their reference points as entire profiles (see *Figure 8.8*).

Psychological studies show that it is too difficult to evaluate jointly more than seven to nine outcomes. However, this applies to separate variables (not to their corresponding profiles or trajectories). Because each profile or trajectory is represented by one variable, only changing parametrically or over time, its mental evaluation is not as difficult as that of a large number of separate variables. Such profiles become aggregated mentally (particularly if graphically presented) into more complex entities, and we can again evaluate and compare perhaps not seven to nine, but perhaps five to seven of such profiles – each containing a much larger amount of data. In this sense, comparing a reference profile with the actual outcome profile is

[7]The technique of maximizing the minimal of maximized objectives was rediscovered many times. It is now popularly used, e.g., in the Optimization Toolbox of MATLAB (however, with no warning to the user that it might result in weak Pareto-optimality).

[8]The reference point approach was originally developed from investigations of penalty techniques applied to multi-objective optimization (see Wierzbicki, 1980a).

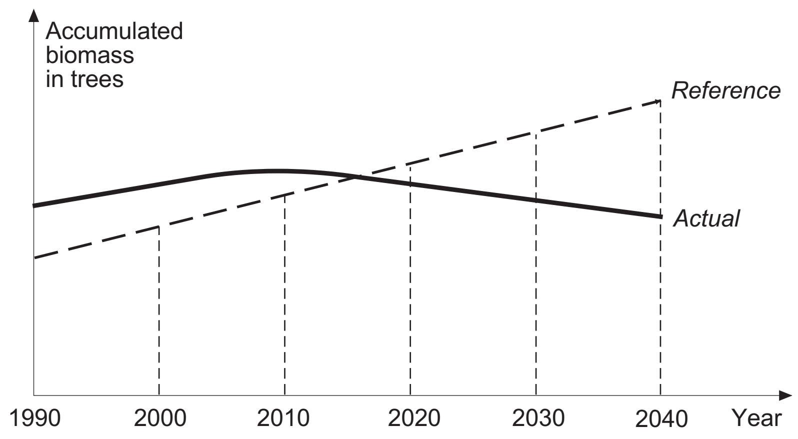


Figure 8.8. An example of objective trajectory profiles, reference and actual, respectively.

easy for the modeler. The possibility of aggregating and comparing large amounts of data into such profiles is another advantage of reference point methodology.

There are more issues related to applying multi-objective optimization to dynamic models. For example, the modeler might like to specify the growth ratios or the increments of selected model outcomes as an additional objective trajectory; the dynamic properties of the model might be exploited when preparing or executing optimization of model outcomes. For a more detailed discussion of multi-objective optimization of dynamic models see Makowski and Sosnowski (1993a) and Wierzbicki (1988b).

8.4 Neutral and Weighted Compromise Solutions

A neutral compromise solution is typically understood in multi-criteria analysis to be a decision with outcomes located somewhere in the middle of the efficient set; however, a more precise meaning of “somewhere in the middle” specifies the type of compromise solution. This notion was investigated in detail first by Zeleny (1974), who has shown that a compromise solution might be defined by:

- Selecting a concept of a norm in the objective space, together with weighting coefficients that might be used to modify this concept; for a neutral compromise solution, equal weighting coefficients should be used.
- Selecting such a reference point that the minima of the distance from the reference point (defined by the selected norm) are efficient.

Assume, for simplicity, that we maximize all objectives, with efficiency equivalent to Pareto-optimality and $D = I\!R^k_+$. Zeleny has shown that, in order to be sure of the efficiency of solutions that minimize the distance (even if the set Q_0 is not convex), the reference point $\bar{\boldsymbol{q}}$ for a scalarizing function $s(\boldsymbol{q}, \bar{\boldsymbol{q}}) = \| \boldsymbol{q} - \bar{\boldsymbol{q}} \|$ should be taken at the utopia point. This point is called the ideal point, and is denoted by $\boldsymbol{q}_{uto}$. The point $\bar{\boldsymbol{q}}$ can be set to a point "above to the north-east" (in the case of maximization of all objectives) of $\boldsymbol{q}_{uto}$, at a "displaced ideal", or simply be an upper bound $\bar{\boldsymbol{q}} = \boldsymbol{q}_{up} \in \boldsymbol{q}_{uto} + D$. Thus, when minimizing a metric or distance related to an l_p norm with $1 \le p < \infty$, properly efficient (Pareto-optimal) compromise solutions are obtained. The Chebyshev (l_∞) norm results in only weakly efficient solutions, unless its minimization is supplemented by a lexicographic test. Dinkelbach and Isermann (1973) have shown that an augmented Chebyshev norm – with a linear part added such as in achievement functions discussed earlier in this chapter – results in properly efficient solutions.

As we stressed earlier, in order to use a norm, we must be sure that all objective components are of the same dimension or dimension-free. However, the increments of objectives still need to be rescaled – from $\mid q_i - q_{i,up} \mid$ to $\mid q_i - q_{i,up} \mid / \mid q_{i,up} - q_{i,lo} \mid$ (where $q_{i,lo}$ is a lower bound, e.g., an approximation of the nadir point component). After such rescaling, when fixing the reference point at the upper bound point, we can define neutral compromise solutions (actually, neutral compromise solution outcomes, while neutral compromise solution decisions are just the decisions needed to reach these outcomes) with equal weighting coefficients as:

$$\begin{aligned}
\hat{\boldsymbol{q}}^{(p)}_{neu} &= \operatorname*{argmin}_{\boldsymbol{q} \in Q_0} \left(\sum_{i=1}^{k} \frac{\mid q_{i,up} - q_i \mid^p}{\mid q_{i,up} - q_{i,lo} \mid^p} \right)^{1/p}, \quad 1 \le p < \infty \\
\hat{\boldsymbol{q}}^{(\infty)}_{neu} &= \operatorname*{argmin}_{\boldsymbol{q} \in Q_0} \left(\max_{1 \le i \le k} \frac{\mid q_{i,up} - q_i \mid}{\mid q_{i,up} - q_{i,lo} \mid} \right), \\
\hat{\boldsymbol{q}}^{(1,\infty)}_{neu} &= \operatorname*{argmin}_{\boldsymbol{q} \in Q_0} \left(\max_{1 \le i \le k} \frac{\mid q_{i,up} - q_i \mid}{\mid q_{i,up} - q_{i,lo} \mid} + \varepsilon \sum_{i=1}^{k} \frac{\mid q_{i,up} - q_i \mid}{\mid q_{i,up} - q_{i,lo} \mid} \right), \qquad (8.38)
\end{aligned}$$

the last one with some small $\varepsilon > 0$. As noted above, $\hat{\boldsymbol{q}}^{(\infty)}_{neu}$ might be only weakly efficient, and not uniquely defined in such a case. However, we can then select an efficient solution by additional testing, e.g., by an additional lexicographic optimization (see Ogryczak *et al.*, 1989).

The neutral solution $\hat{\boldsymbol{q}}^{(1,\infty)}_{neu}$ can also be obtained in a different way, since $q_{i,up} \ge q_i \ge q_{i,lo}$ and

$$\frac{q_{i,up} - q_i}{q_{i,up} - q_{i,lo}} = \frac{1}{2} - \frac{q_i - \bar{q}_{i,mid}}{q_{i,up} - q_{i,lo}}, \quad \bar{q}_{i,mid} = \frac{q_{i,up} + q_{i,lo}}{2}, \qquad (8.39)$$

therefore:

$$\begin{aligned} \max_{1\le i\le k} \frac{\mid q_{i,up} - q_i \mid}{\mid q_{i,up} - q_{i,lo} \mid} + \varepsilon \sum_{i=1}^{k} \frac{\mid q_{i,up} - q_i \mid}{\mid q_{i,up} - q_{i,lo} \mid} &= \\ - \min_{1\le i\le k} \frac{q_i - \bar{q}_{i,mid}}{q_{i,up} - q_{i,lo}} - \varepsilon \sum_{i=1}^{k} \frac{q_i - \bar{q}_{i,mid}}{q_{i,up} - q_{i,lo}} &+ \frac{1}{2}(1 + \varepsilon k). \end{aligned} \tag{8.40}$$

Hence, minimizing the distance (induced by the augmented Chebyshev norm) from the upper bound point is equivalent to maximizing the following order-consistent achievement function (which is just a rescaled version of (8.33)):

$$\begin{aligned} \sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}_{mid}) &= \min_{1\le i\le k} \frac{q_i - \bar{q}_{i,mid}}{q_{i,up} - q_{i,lo}} + \varepsilon \sum_{i=1}^{k} \frac{q_i - \bar{q}_{i,mid}}{q_{i,up} - q_{i,lo}}, \\ \bar{q}_{i,mid} &= \frac{q_{i,up} + q_{i,lo}}{2} \end{aligned} \tag{8.41}$$

with the reference point $\bar{\boldsymbol{q}}_{mid}$ located precisely in the middle of the ranges between upper bound and lower bound point.

However, such neutral solutions as defined above might serve only as a starting point for interaction with the modeler. A more general concept is that of weighted compromise solutions. First the decision maker must state what the relative importance of criteria is, then this relative importance is expressed in terms of weighting coefficients $\alpha_i > 0,\ \forall i = 1, \ldots k,\ \sum_{i=1}^{k} \alpha_i = 1$. Usually, the responses of the decision maker can be interpreted in terms of the ratios $\omega_{ij} = \alpha_i/\alpha_j$, and we must apply a method of converting these responses into the vector α. A broadly applied method, the AHP (analytical hierarchy process) approach, involves exploiting the properties of eigenvalues of rank-one matrices in order to shorten computations needed for such a conversion (see Saaty, 1980). A best approximation of the vector α – assuming some random errors in the responses $\omega_{i,j}$ of the decision maker – would require more complicated computations.

Once the weighting coefficients α have been determined, the weighted compromise solutions $\hat{\boldsymbol{q}}_{\alpha}^{(p)}$ are defined by:

$$\begin{aligned} \hat{\boldsymbol{q}}_{\alpha}^{(p)} &= \operatorname*{argmin}_{\boldsymbol{q}\in Q_0} \left(\sum_{i=1}^{k} \alpha_i \frac{\mid q_{i,up} - q_i \mid^p}{\mid q_{i,up} - q_{i,lo} \mid^p} \right)^{1/p}, \quad 1 \le p < \infty \\ \hat{\boldsymbol{q}}_{\alpha}^{(\infty)} &= \operatorname*{argmin}_{\boldsymbol{q}\in Q_0} \left(\max_{1\le i\le k} \alpha_i \frac{\mid q_{i,up} - q_i \mid}{\mid q_{i,up} - q_{i,lo} \mid} \right), \\ \hat{\boldsymbol{q}}_{\alpha}^{(1,\infty)} &= \operatorname*{argmin}_{\boldsymbol{q}\in Q_0} \left(\max_{1\le i\le k} \alpha_i \frac{\mid q_{i,up} - q_i \mid}{\mid q_{i,up} - q_{i,lo} \mid} + \varepsilon \sum_{i=1}^{k} \alpha_i \frac{\mid q_{i,up} - q_i \mid}{\mid q_{i,up} - q_{i,lo} \mid} \right). \end{aligned} \tag{8.42}$$

Whereas the concept of weighted compromise results in sufficient conditions of Pareto-optimality – all weighted compromise solutions $\hat{\boldsymbol{q}}_{\alpha}^{(p)}$ for $1 \le p < \infty$ are properly efficient (Pareto-optimal) – necessary conditions are more complicated. For a given p and a given properly efficient outcome $\hat{\boldsymbol{q}} \in \hat{Q}_0^p$, we cannot be sure that we will find such a vector of weighting coefficients $\hat{\alpha} \in \text{Int}\, \mathbb{R}_+^k$ that $\hat{\boldsymbol{q}}_{\hat{\alpha}}^{(p)} = \hat{\boldsymbol{q}}$. It can only be shown that we can find $\hat{\boldsymbol{q}}_{\alpha}^{(p)}$ in some neighborhood of the given $\hat{\boldsymbol{q}}$ by jointly changing α and p (see e.g., Sawaragi *et al.,* 1985). Thus, for a given $p,\ 1 \le p < \infty$, we cannot say that we will obtain any properly efficient outcome desired by the decision maker by changing weighting coefficients alone.

Moreover, the character of the dependence of $\hat{\boldsymbol{q}}_{\alpha}^{(p)}$ on α is not easy to interpret. In some applications, this might be a procedural advantage (see comments in Chapter 9). However, in the case of a decision maker who is a modeler, the lack of a clear interpretation of this dependence is a disadvantage.

Fortunately there is one case in which the dependence of a weighted compromise solution on weighting coefficients is easy to interpret, namely the case of Chebyshev norms. Here we discuss the augmented Chebyshev norm and $\hat{\boldsymbol{q}}_{\alpha}^{(1,\infty)}$. Suppose we choose a weighting coefficient vector α with $\alpha_i > 0,\ \sum_{i=1}^{k} \alpha_i = 1$, and a scalar coefficient $\eta > 1/\alpha_i\ \forall i = 1, \ldots k$ in order to assign to each α_i an aspiration level:

$$\bar{q}_i = q_{i,up} - \frac{q_{i,up} - q_{i,lo}}{\eta \alpha_i}. \tag{8.43}$$

Then we obtain $q_{i,lo} \le \bar{q}_i < q_{i,up}\ \forall i = 1, \ldots k$ owing to the inequality satisfied by η; however, the aspiration levels $\bar{q}_i$ might change with η, in which case equation (8.43) describes a line segment in $\mathbb{R}^k$ ending at $\boldsymbol{q}_{up}$ as $\eta \to \infty$. Conversely, for any aspiration point $\bar{\boldsymbol{q}}$ such that $q_{i,lo} \le \bar{q}_i < q_{i,up}\ \forall i = 1, \ldots k$ we can set:

$$\begin{aligned} \alpha_i &= \frac{q_{i,up} - q_{i,lo}}{q_{i,up} - \bar{q}_i} \Bigg/ \sum_{j=1}^{k} \frac{q_{j,up} - q_{j,lo}}{q_{j,up} - \bar{q}_j} \\ \eta &= \sum_{j=1}^{k} \frac{q_{j,up} - q_{j,lo}}{q_{j,up} - \bar{q}_j}, \end{aligned} \tag{8.44}$$

which defines the inverse to the transformation (8.43). We can interpret this inverse transformation in the following way: the ratios $\omega_{ij} = \alpha_i/\alpha_j$ of importance of criteria are defined by selected aspiration levels as an inverse ratio of their relative distances from upper bound levels:

$$\omega_{ij} = \frac{\alpha_i}{\alpha_j} = \frac{q_{j,up} - \bar{q}_j}{q_{i,up} - \bar{q}_i}. \tag{8.45}$$

The transformation (8.44) has been used by Steuer and Choo (1983) in a procedure using the augmented Chebyshev norm and $\hat{\boldsymbol{q}}_{\alpha}^{(1,\infty)}$. However, this transformation can be controlled interactively by the decision maker who specifies aspiration points (called definition points by Steuer and Choo) that are used to define the weighting coefficients. The outcomes of such a procedure are properly efficient; Steuer and Choo show that any properly efficient outcome can be obtained by this procedure for convex sets Q_0.

However, we can show more: under transformations (8.43) and (8.44), the weighted compromise solution $\hat{\boldsymbol{q}}_{\alpha}^{(1,\infty)}$ can be equivalently obtained by the maximization of an order-consistent achievement scalarizing function with such aspiration levels used as a reference point. This is because we have:

$$\alpha_i \frac{q_{i,up} - q_i}{q_{i,up} - q_{i,lo}} = \left(\frac{q_{i,up} - q_i}{q_{i,up} - \bar{q}_i} \right) \Big/ \eta = \left(1 - \frac{q_i - \bar{q}_i}{q_{i,up} - \bar{q}_i} \right) \Big/ \eta. \tag{8.46}$$

Therefore, since $q_{i,up} \geq q_i \geq q_{i,lo}$:

$$\max_{1 \leq i \leq k} \alpha_i \frac{\mid q_{i,up} - q_i \mid}{\mid q_{i,up} - q_{i,lo} \mid} + \varepsilon \sum_{i=1}^{k} \alpha_i \frac{\mid q_{i,up} - q_i \mid}{\mid q_{i,up} - q_{i,lo} \mid} =$$

$$(- \min_{1 \leq i \leq k} \frac{q_i - \bar{q}_i}{q_{i,up} - \bar{q}_i} - \varepsilon \sum_{i=1}^{k} \frac{q_i - \bar{q}_i}{q_{i,up} - \bar{q}_i} + 1 + \varepsilon k)/\eta. \tag{8.47}$$

Hence, minimizing the weighted distance (induced by the augmented Chebyshev norm) from the upper bound point is equivalent to maximizing the following order-consistent achievement function (again defined as in (8.33) with rescaling):

$$\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}) = \min_{1 \leq i \leq k} \frac{q_i - \bar{q}_i}{q_{i,up} - \bar{q}_i} + \varepsilon \sum_{i=1}^{k} \frac{q_i - \bar{q}_i}{q_{i,up} - \bar{q}_i} \tag{8.48}$$

with the aspiration point $\bar{\boldsymbol{q}}$ defined as the transformation (8.43) of the vector of weighting coefficients α with any (sufficiently large) value for parameter η.

Moreover, even if the set Q_0 is not convex (or even if it is a discrete set), we can take any properly efficient (Pareto-optimal) outcome $\hat{\boldsymbol{q}}$ with trade-off coefficients scaled down by the deviations from the upper levels and bounded by:

$$t_{ij}(\hat{\boldsymbol{q}}) \frac{q_{j,up} - \hat{q}_j}{q_{i,up} - \hat{q}_i} \leq (1 + 1/\varepsilon). \tag{8.49}$$

At any such point, we can define a reference point and weighting coefficients – by taking $\bar{\boldsymbol{q}} = \hat{\boldsymbol{q}}$ and applying the transformation (8.44) – in such a way that the

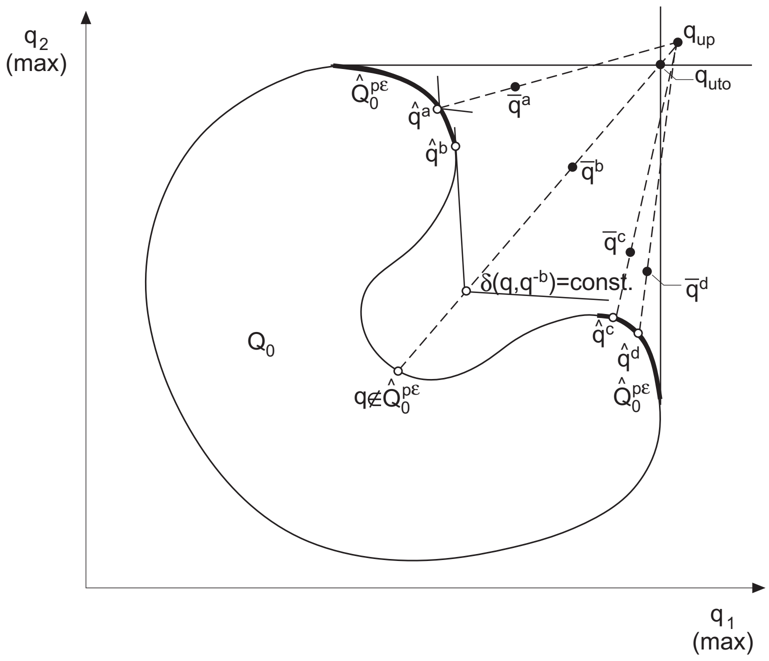

q2 (max)
q1 (max)
qup
quto
$\hat{Q}_0^{p\varepsilon}$
$\hat{q}^a$
$\bar{q}^a$
$\hat{q}^b$
$\bar{q}^b$
$\bar{q}^c$
$\bar{q}^d$
$\hat{\delta}(q,\bar{q}^b)$=const.
$\hat{q}^c$
$\hat{q}^d$
$\hat{Q}_0^{p\varepsilon}$
Q0
$q \notin \hat{Q}_0^{p\varepsilon}$

Figure 8.9. Controlling the selection of properly efficient solutions by aspiration levels.

maximal point of $\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}})$, equal to the weighted compromise solution $\hat{\boldsymbol{q}}_\alpha^{(1,\infty)}$, coincides with $\hat{\boldsymbol{q}}$. This can be shown by an appropriate modification of the argument on separating the set Q_0 and the conical set $\hat{\boldsymbol{q}} + \text{Int } D_\varepsilon$ by a level set of the function $\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}})$, as discussed in Section 8.3 (see also Wierzbicki, 1992b). Equivalently, for any $\hat{\boldsymbol{q}} \in \hat{Q}_0^p$, we can choose a sufficiently small ε such that (8.49) is satisfied. We can then choose an aspiration point (e.g., equal to $\hat{\boldsymbol{q}}$, or at any point on the line segment joining $\boldsymbol{q}_{up}$ and $\hat{\boldsymbol{q}}$) together with the corresponding weighting coefficients, such that the point $\hat{\boldsymbol{q}}$ is obtained either by maximizing the achievement function or by minimizing the weighted and augmented Chebyshev distance from the upper bound point.

We must stress, however, that this equivalence does not mean that the order-consistent achievement function is a distance function. The order-consistent achievement function compares points $\boldsymbol{q}$ and $\bar{\boldsymbol{q}}$, not $\boldsymbol{q}$ and $\boldsymbol{q}_{up}$, and cannot be a distance between $\boldsymbol{q}$ and $\bar{\boldsymbol{q}}$, as it must preserve monotonicity at $\boldsymbol{q} = \bar{\boldsymbol{q}}$. In fact, it is a more complicated function of these two variables than a distance function. The equivalence only means that this more complicated function can be constructed while using the distance between other points, $\boldsymbol{q}$ and $\boldsymbol{q}_{up}$.

These arguments are illustrated in *Figure 8.9.* If the line segment defined by equation (8.43) with changing parameter η (represented by the dashed lines in

Figure 8.9 and corresponding to the loci of corner points of achievement functions, as illustrated in *Figure 8.8*) intersects the set $\hat{Q}_0^p$ at a point $\hat{\boldsymbol{q}}$, then we can take a sufficiently small ε such that the minimum of the weighted and augmented Chebyshev distance and the maximum of the achievement function $\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}})$ coincide with the intersection point. This occurs at every properly efficient $\hat{\boldsymbol{q}}$ – as in the case of reference points $\bar{\boldsymbol{q}}^a$, $\bar{\boldsymbol{q}}^c$ and $\bar{\boldsymbol{q}}^d$ and the corresponding $\hat{\boldsymbol{q}}^a$, $\hat{\boldsymbol{q}}^c$ and $\hat{\boldsymbol{q}}^d$. However, this does not occur for every reference point and weighting coefficient vector, because the corresponding line segment might not intersect the set $\hat{Q}_0^p$, as in the case of $\bar{\boldsymbol{q}}^b$. The maximal point $\hat{\boldsymbol{q}}^b$ of the achievement function $\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}^b)$ is properly efficient, but does not belong to the corresponding line segment. As long as the corresponding line segment intersects the set of ε-properly efficient outcomes, small changes of reference points (e.g., between $\bar{\boldsymbol{q}}^c$ and $\bar{\boldsymbol{q}}^d$) result in small changes of efficient points (e.g., between $\hat{\boldsymbol{q}}^c$ and $\hat{\boldsymbol{q}}^d$). Thus, we can control the selection of efficient outcomes continuously. This does not hold, however, when the set of efficient outcomes is disconnected and the corresponding line segment does not intersect this set, as in the case of the reference point $\bar{\boldsymbol{q}}^b$. The set Q_0 in *Figure 8.9* is not convex; because the above discussion does not assume the convexity of the set of attainable outcomes, the conclusions are also valid for nonconvex and for discrete sets.

While using the equivalence discussed above, Kok and Lootsma (1985) proposed applying a technique similar to the AHP for determining the weighting coefficients and then combining them with the Chebyshev or augmented Chebyshev distance minimization in order to obtain a weighted compromise solution. This clearly has an advantage when compared to the use of other norms, as we can interpret the ratios of weighting coefficients as in (8.45). We can thus inform the decision maker that if she/he specifies an objective j to be, for example, five times more important than objective i, then she/he is likely to obtain the relative deviation of the objective j from its best possible level in the desired proportion. This is measured in the percentage of the entire range of changes of this objective, which is five times smaller than the relative deviation of the objective i. In this statement, "likely to obtain" is used because of the possibility of such special cases as shown in *Figure 8.9*; in more regular cases, if a point with such deviation ratios is contained in the set of properly efficient outcomes (with an appropriate bound on trade-off coefficients), we obtain precisely the desired proportion.

However, if the decision maker is a modeler interested in multi-objective model analysis, she/he does not need to specify these importance ratios. The modeler could just as well directly specify aspiration levels and use the optimization of an achievement scalarizing function. It might be helpful to know that if the aspiration level for the i-th objective deviates from the upper level five times more (in the sense relative to objective ranges) than the one for the j-th objective, then this proportion is likely (except in cases of disconnected sets of efficient outcomes) to be preserved in the outcomes obtained by maximizing the achievement function. However, this information is not crucial, as long as the achieved efficient outcomes can be continuously influenced by changing aspiration or reference levels. In many experiments using various multi-objective techniques for model analysis, modelers consistently prefer to use reference levels, not weighting coefficients. Thus, a typical exploration of the Pareto boundary by a modeler consists simply in varying reference points, maximizing the corresponding achievement function, and learning what efficient decisions and objective outcomes can be obtained in this way.

8.5 Tools for Multi-Objective Optimization

The possibility of using an aggregation of objectives and of achievement scalarizing functions might lead to the conclusion that no special tools are needed for multi-objective optimization – after all, we finally use a single-objective optimization. However, as already indicated in previous chapters, multi-objective optimization requires special tools, for many reasons. One of these reasons is the parametric character of multi-objective optimization: single-objective optimization runs are made repetitively, with changing parameters, and all issues of using old solutions for a hot start of a new optimization run must be considered here. Another reason might be the special, usually nondifferentiable, character of an achievement scalarizing function, with the resulting issues concerning the best way of reformulating and solving optimization problems with such functions. The third reason might be the relation of multi-objective optimization to a regularization of optimization problems and to softly constrained and inverse simulation of models.

For all of these reasons, optimization solvers might be either well adapted to be applied in multi-objective optimization, or not very well adapted. We have already described several optimization tools that were developed with the intention of being used in repetitive optimization runs during multi-objective analysis of a substantive model of a given class, in relation to applications presented in this book. For linear substantive models, these are, for example: the experimental solver SALP by Świętanowski (1995) – the simplex big M method with a special provision for hot starts; the interior point solver HOPDM, particularly in its version by Gondzio and Makowski (1995a); and the proximal point solver HYBRID, in several variants (Makowski and Sosnowski, 1989, 1993b). For mixed integer-linear models, the solver MOMIP by Ogryczak and Zorychta (1993) was developed with the purpose of performing well also for problems resulting from reformulations of achievement scalarizing functions. Nonlinear solvers developed especially for multi-objective optimization are less numerous, since the solver in the DIDAS-N system, based on shifted penalty algorithm and tested as quite robust for nonlinear multi-objective optimization on problems of smaller and medium size, is contained in a closed system. This same solver is contained as a module in the DIDAS-N++ system, but is less extensively tested for larger nonlinear models. All of this software was developed for the type of applications presented in this book and is available on the Web as described in the Appendix.

This software is only a part of experimental or commercial software that was either developed for or can be applied to multi-objective optimization. In Part III of this book we show that, in actual applications, it is always worthwhile to use alternative tools and compare results obtained with their help. For large nonlinear models several commercial nonlinear solvers might be used, e.g., when analyzing the extensions of RAINS models, including topospheric ozone emissions (described in Chapter 13), both CONOPT and MINOS solvers were applied.

We hope that the applications presented in this book will motivate software developers to adapt various available solvers more to the needs of multi-objective optimization and model analysis.

Chapter 9

Tools Supporting Choice

Andrzej P. Wierzbicki and Hirotaka Nakayama

There are two basic approaches to multi-objective decision support. One of these assumes that the main objective of a decision support system (DSS) is to support learning by the decision maker (DSS user, modeler); this was the motivation for presenting most of the techniques and tools in this book. A different and older approach assumes that the main objective of a DSS is to support choice.

If the essence of an interactive procedure is related to learning, then the preferences and value functions of the decision maker change during this procedure. Therefore, an interactive procedure should concentrate more on exploring the set of attainable decision outcomes than on the convergence to a final decision. If we assume, however, that the decision maker has already learned enough about the decision situation and its models, two cases may arise. In the first case the decision maker already knows which decision to choose and does not need further support. In the second case the decision maker knows only approximately which decisions to choose and some further support is required. In practice, most cases fall between these two limiting cases: further support might be useful and welcome if it does not require too much additional time. As most of the book concentrates on the first case, we devote this chapter to the second one.

Various approaches to supporting choice can be divided into two basic classes. The first might be called automated choice; in this approach a preferential model is constructed detailed enough to infer a final decision.[1] If we are supporting repetitive, tactical, or operational decisions in which the details of the substantive model might change but the preferences of the decision maker do not, then it is advantageous to automate choice. In the next section we outline some of the classical methods of doing this.

However, most of the applications and experiences reported in this book concern a second class, usually related to strategic or innovative decisions. In this class interactive choice is supported, that is, we do not construct a very detailed preferential model and the participation of the decision maker in selecting the final decision is essential. These approaches can also be subdivided into various subclasses, and are discussed in further sections of this chapter. In Section 9.2 we describe interactive procedures with proven convergence. Section 9.3 describes various methods of generating options for interactive choice, usually without proven convergence but often more effective in practice. Finally, Section 9.4 presents a new method of adding a convergent final phase to practical and effective methods of interactive option generation.

9.1 Automating Choice

The various cases of multi-objective decision support described above can be represented by a decision tree (see *Figure 9.1(a)*). This decision tree represents only a classification of various cases – "states of the world". Decision theory often assumes a probabilistic interpretation of these states of the world and assigns probabilities to them. Thus, the decision tree in *Figure 9.1(a)* might mean the following, for example: after half of the time that the decision maker has for making a decision, she/he might want to continue learning with probability p_1; has finished learning and does not need any other support with probability p_2; has finished learning and wants interactive decision support with probability p_3; or has finished learning but wants the DSS to make the decision for her/him with probability p_4. To be useful in supporting decisions in a specific situation, this tree should be augmented by further branches representing possible decisions, e.g., denoted by x_i (where i denotes consecutive discrete decision options, not components of a decision vector). For example, suppose the decision is related to how many pages should be written on "Automating Choice" (e.g., 2, 5, 10, 20, or 50 pages?).[2] This might be represented by the simpler tree in *Figure 9.1(b)*, where each branch corresponds to

[1]Because many interpretations of classical approaches to decision analysis exist, we are aware that there might be some objections to classifying them as methods of "automating choice". However, if we build a full model of a utility or value function of a human decision maker, we might just as well replace her/him by an automaton. In some cases, this is advantageous: automata do not tire and are always consistent.

[2]This is a classical topic in decision analysis and much has already been written about it.

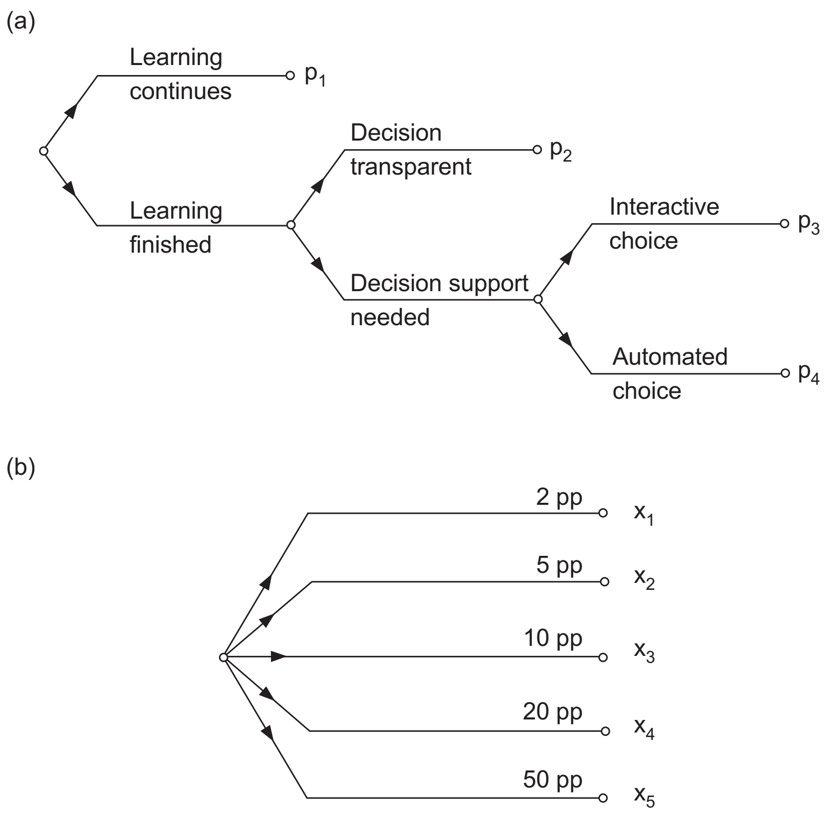


Figure 9.1. Decision trees of (a) states of the world; and (b) possible decisions (e.g., how many pages (denoted by pp) are needed to write about automated choice).

a decision; this simpler tree should then be attached to each branch of the tree from *Figure 9.1(a)*.

A classical method of decision analysis is to assess probabilities p_j of events leading to the branches of the tree that represent various states of the world, and estimate values or utilities $U_j(x_i)$ related to the outcomes of different decisions x_i given the state j of the world. The expected utility is then defined as:[3]

$$EU(x_i) = \sum_{j=1}^{J} p_j U_j(x_i) \tag{9.1}$$

and a final decision that corresponds to the maximal expected utility is selected.

[3]We should note that the concept of utility is reserved in decision theory for expected utility based on probabilistic representation of uncertainty. In cases without such representation of uncertainty it is preferable to speak of a value function.

An essential result of expected utility theory (see e.g., Rios, 1994) is the specification of reasonable[4] axioms. Under these axioms the relation (9.1) can be simplified by assuming $U_j(x_i) = U(x_i)$, generating an expected utility form that is linear in probabilities:

$$EU(x_i) = \sum_{j=1}^{J} p_j U(x_i). \tag{9.2}$$

Even in this simpler form, however, the determination of the function $U(x_i)$ is a complicated task. In the simplest, single-objective case when there is no complicated substantive model and the decisions x_i can be characterized by their single outcome (e.g., money won at a lottery) there are known techniques of identifying the function $U(x_i)$ by repeated questions about preferences between various hypothetical lotteries. Because the form of the function $U(x_i)$ can be complicated – expressing risk aversion, risk proneness, or a combination of such attitudes (e.g., risk proneness at small stakes, risk aversion at high stakes) – the number of necessary questions can be high even in this simplest case. For more complicated cases, where the outcomes of decisions x_i can be characterized by several attributes, sophisticated techniques of identifying the utility function have been developed (see Keeney and Raiffa, 1976; Keeney, 1992). Although the number of questions and answers needed for such an identification is usually considerable, such techniques might be useful (e.g., for environmental assessments under uncertainty).

There are at least two types of decision situations where it is necessary to invest the effort needed for identifying a utility function. One of them involves clarifying preferences; Keeney (1992) calls this "value focused thinking". The other situation involves automating or semi-automating repetitive decisions, where a DSS could suggest – in changing conditions – decisions that maximize a previously identified utility function.

Because of the complexity in identifying a utility function for a probabilistic representation of uncertainty, alternative approaches using other uncertainty representations were developed. One such approach is AHP, the analytic hierarchy process (see Saaty, 1980), where instead of a utility function a value function of the decision maker is approximately estimated. Similar to expected utility approaches, the proposed process concentrates first on structuring the decision problem by using models in the form of decision trees to describe relations between various aspects of the decision situation – forces, actors, objectives, scenarios, etc. An essential

[4]Theoretically at least. Research in psychology and various paradoxes – such as the Allais paradox – show that one of the basic axioms of expected utility theory, the axiom of independence, does not necessarily describe the observed behavior of real decision makers. Thus, various alternative models of nonlinear dependence of expected utility on probabilities have been proposed (see e.g., French, 1986).

part of the process is to use models and interact with decision makers to determine hierarchical priorities of objectives.

A simplification and comparison of this process to expected utility approaches is as follows:

- AHP works with averaged outcomes of decisions, i.e., objectives q_i (where i denotes the component of the objective vector $\boldsymbol{q}$), and not with the probabilistic distributions of outcomes.
- The essence of AHP is a way of identifying weighting coefficients in a linear approximation of the value function:

$$v(\boldsymbol{q}) = \sum_{i=1}^{k} \alpha_i q_i. \tag{9.3}$$

AHP uses evaluations of the importance of objectives expressed as ratios; the decision maker must ask how much more important objective q_i is than another objective q_j. The answers to such questions, w_{ij}, are treated as evaluations of the ratios of weighting coefficients, $\omega_{ij} = \alpha_i/\alpha_j$. Given a vector $\alpha \in \mathbb{R}^k$ with $\alpha_i > 0 \ \forall i = 1, \ldots, k$, the ratios of vector components $\omega_{ij} = \alpha_i/\alpha_j$ form a matrix Ω with the following properties:

$$\Omega = [\omega_{ij}];\ \omega_{ij} > 0,\ \omega_{ji} = 1/\omega_{ij},\ \omega ii = 1 \ \ \forall i, j = 1, \ldots, k. \tag{9.4}$$

The matrix Ω of such properties is consistent with a vector $\alpha \in \mathbb{R}^k$, if its elements $\omega_{ij} = \alpha_i/\alpha_j$. The answers w_{ij} of the decision maker form another matrix $W = [w_{ij}]$ which, even if it satisfies all properties of the matrix Ω as specified by (9.4), need not be consistent with any vector in this sense (for $k > 2$ it is easy to construct examples of such inconsistency). Thus, the question is: given matrix W, how can a consistent matrix Ω (and its underlying vector of weighting coefficients) be constructed, which is close to the given W?

AHP solves this problem by exploiting analytical properties of matrices consistent with a vector. Matrix Ω has rank 1, $k - 1$ eigenvalues equal to 0 and one positive eigenvalue $\lambda_{max}(\Omega) = k$; the eigenvector corresponding to this eigenvalue is $c\alpha$, where c is an arbitrary constant. Matrix W, even if inconsistent with any vector, has similar properties to Ω; thus, it can be shown that $\lambda_{max}(W) \geq k$. The approach applied in AHP consists in computing the largest eigenvalue of W and the eigenvector $\beta \in \mathbb{R}^k$ corresponding to it, and normalizing the vector β to obtain the vector of weighting coefficients α:

$$\beta \in \mathbb{R}^k : \ W\beta \ = \ \lambda_{max}(W)\beta$$

$$\alpha_i \;=\; \beta_i / \sum_{j=1}^{k} \beta_j, \;\forall i = 1, \ldots, k. \tag{9.5}$$

After estimating the vector α, the linear function (9.3) can be maximized over the set of admissible decisions, which results in properly efficient decisions and outcomes.[5] The resulting decisions are presented to the decision maker.

The advantage of AHP is that relatively simple and psychologically well-substantiated questions about importance ratios are considered. Moreover, a reasonable number of questions is needed to establish weighting coefficients, and the computation of eigenvalues and eigenvectors is standard and fast. Thus, the technique can be used interactively: if the decision maker does not like the proposed decision, she/he can modify the answers concerning the importance of criteria.

However, the character of the dependence of decisions obtained on the stated importance ratios is not easy to interpret. As commented in Chapter 5, this might be a procedural advantage in some situations. If a group of high-level decision makers cannot agree on selecting a decision, it might be advantageous to propose to them a rational procedure of automating decisions without clearly predictable outcomes. They should come to an agreement by voting on the comparative importance of criteria. The weighting coefficients would then be determined as an outcome of the vote and a weighted compromise solution computed and presented to the group. However, in many other cases – particularly if the decision maker is a modeler – the lack of a clear interpretation of this dependence is disadvantageous.

Moreover, the use of a linear approximation of a value function has various disadvantages (these are described in earlier chapters). Lootsma *et al.* (1993) proposed another approximation of a value function: multiplicatively aggregated improvements of objectives as compared with their lower bounds (e.g., the nadir point component). A suggested way of aggregation is to use a weighted geometric mean. This can take the form:

$$P(\boldsymbol{q}) = \prod_{i=1}^{k} \left(\frac{q_i - q_{i,lo}}{q_{i,up} - q_{i,lo}} \right)^{\gamma_i}, \;\; \gamma_i > 0 \;\forall i = 1, \ldots, k, \; \sum_{i=k}^{k} \gamma_i = 1 \tag{9.6}$$

where γ_i play the role of weighting coefficients; and $q_{i,up}$, $q_{i,lo}$ are upper and lower estimated bounds of objective ranges. An interesting property of the weighted geometric mean $P(\boldsymbol{q})$ is well known (see e.g., Sawaragi *et al.*, 1985): at a given point $\boldsymbol{q}$, the proportion of partial derivatives of the geometric mean, scaled by the inverse

[5]AHP is usually applied to problems with a discrete, finite number of decision options with directly specified outcomes or attributes. In principle, however, it could be applied to more complicated models, where the objectives y_i are defined as (possibly nonlinear) functions of decisions x.

ratio of the deviations from the lower bounds, is equal to the ratio of weighting coefficients:

$$\left(\frac{\partial P}{\partial q_i}\Big/\frac{\partial P}{\partial q_j}\right)\frac{q_i - q_{i,lo}}{q_j - q_{j,lo}} = \frac{\gamma_i}{\gamma_j}. \tag{9.7}$$

The ratio γ_i/γ_j can be again interpreted as a measure of relative importance of objective i over objective j and identified with the help of multiplicative AHP proposed by Lootsma (see e.g., 1993). The weighted geometric mean of improvements $P(\boldsymbol{q})$ is a strictly monotone function of $\boldsymbol{q}$, hence its maxima over $\boldsymbol{q} \in Q_0$ are efficient (Pareto-optimal), even properly efficient because of the boundedness of (9.7).

Another disadvantage of AHP is that the approximation of importance ratios through eigenvalues and eigenvectors does not have clear statistical significance. An alternative procedure with statistical significance might be as follows. Suppose there are $N \geq 1$ expert decision makers who evaluate importance ratios $w_{ij} = \alpha_i/\alpha_j$ (or $w_{ij} = \gamma_i/\gamma_j$). They then make random errors in these evaluations. Concerning a statement about ratios, it is reasonable to assume that these errors have a lognormal distribution, i.e., that the logarithms $\log w_{ij} - \log \alpha_i + \log \alpha_j$ have a normal distribution with a zero mean. Under such an assumption, the most probable weighting coefficients can be determined as solutions of the following nonlinear optimization problem:

$$\min_{\alpha \in A_1} \sum_{n=1}^{N} \sum_{i=1}^{k} \sum_{j<i} (\log w_{ij}^n - \log \alpha_i + \log \alpha_j)^2$$

$$A_1 = \{\alpha \in \mathbb{R}^k : \alpha_i \geq 0,\ \sum_{i=1}^{k} \alpha_i = 1\}, \tag{9.8}$$

where w_{ij}^n is the evaluation of the importance ratio α_i/α_j given by the n-th expert. The advantage of this formulation over classical AHP is that all known techniques of statistical validation can be used. Solving problem (9.8) requires slightly more computing power than determining eigenvalues and eigenvectors, but this does not limit applications if using modern computers.

9.2 Interactive Procedures with Proven Convergence

We turn now to techniques that do not require a full identification of a utility or value function, but rely on the continuing participation of the decision maker instead. These are interactive procedures of supporting choice. From these procedures, we address those that have proven convergence.

9.2.1 Geoffrion-Dyer-Feinberg procedure

The first of the interactive procedures with guaranteed convergence was described by Geoffrion *et al.* (1972). Only brief comments are made on this procedure. It is applicable for general, nonlinear, but differentiable substantive models; it assumes that the decision maker also has a nonlinear but differentiable value function. The procedure is an extension of the Frank-Wolfe nonlinear optimization algorithm, where interaction with the human decision maker replaces the modules of function and gradient determination. The determination of the gradient of a value function is a complicated task, requiring many pairwise comparison questions to be answered by the decision maker. Thus the procedure, though theoretically elegant, has not found many applications in multi-objective decision support. It can be stated that in order to be effective, an interactive procedure must concentrate more on the details of interaction with the decision maker, and less on imitating mathematical programming algorithms.

However, this procedure is a prototype of other convergent procedures that exploit the concepts of convergence of mathematical programming algorithms. An unconstrained nonlinear programming algorithm based on directional searches converges under two general assumptions (see e.g., Luenberger, 1973):

- The directions of search should give uniform improvement, i.e., the angle between these directions and the gradient should be uniformly acute (in the case of maximization).
- The step-size coefficients determining the length of steps along the directions should not converge to zero too fast. These can be specified by various tests (of Goldstein or Wolfe) or expressed by the assumption that, if they converge to zero, then the sequence of points generated by the algorithm converges to the maximum or at least to a stationary point of the function.

The Frank-Wolfe algorithm produces directions of uniform improvement, and the second assumption sufficient for convergence is fulfilled in the Geoffrion-Dyer-Feinberg procedure because of the details of a specific line search used in the procedure.

9.2.2 Zionts-Wallenius procedure

The procedure of Zionts and Wallenius (1976) was designed for a much more limited class of models, but greater attention was paid to the details of interaction with the decision maker. Substantive models were restricted to multi-objective linear programming models, where the set Q_0 is polyhedral. As a preferential model, the linear value function

$$s(\boldsymbol{q}, \alpha) = \alpha^T \boldsymbol{q} = \sum_{i=1}^{k} \alpha_i q_i \tag{9.9}$$

was chosen, but only as an approximation of any concave value function.

The procedure starts at any efficient extreme point of Q_0, for example, from a neutral point corresponding to $\alpha_i = \frac{1}{k}$, $\forall i = 1, \ldots, k$. The decision maker is asked what is preferable:

- Other extreme points of Q_0 that are adjacent to the current point but distinctly different from it, with a given difference threshold.
- Trade-off vectors generated in a multi-objective linear programming algorithm and corresponding to the edges leading to adjacent extreme points that are too close, below a given difference threshold.

The adjacent extreme points are compared to the current one, but the trade-off vectors are evaluated in more general terms of preference. Subsequent experience has shown that decision makers might feel uncomfortable when answering such general questions. Because the decision maker is asked whether she/he likes improvements in a given direction (e.g., if the adjacent extreme points are too close), another way of phrasing the question could be to present distinctly different points on the same straight line, not necessarily belonging to Q_0.

After such questioning, either a preference for an adjacent point or for a direction is established, otherwise all edges leading to adjacent extreme points of the current point are evaluated as nonprofitable. Conceptually, this would suffice to construct a convergent interactive procedure. Indeed, moving to the preferred extreme point results in an increase of the value function. As the procedure moves among a finite number of extreme points only, the convergence (assuming compact Q_0) is finite. This is precise for linear value functions and only approximate for concave value functions that might have maxima on the facets, not at some extreme points of Q_0.

However, in the Zionts-Wallenius procedure, the data obtained from the answers of the decision maker are used additionally for updating the set of weighting coefficients $A^{(l)}$, where the upper index (l) denotes the iteration number. At the starting point, $A^{(0)} = \{\alpha \in \mathbb{R}^k : \alpha_i \geq 0, \sum_{i=1}^{k} \alpha_i = 1\}$ is assumed. Each increase of preference along a given direction, $\boldsymbol{q} \succ \boldsymbol{q}^{(l)}$, indicates that the set $A^{(l+1)}$ might be constructed from $A^{(l)}$ by adding the constraint $\alpha^T(\boldsymbol{q} - \boldsymbol{q}^{(l)}) \geq \delta$, where

$\delta > 0$ is an accuracy threshold. Trade-off vectors can be treated similarly as directions $\boldsymbol{q} - \boldsymbol{q}^{(l)}$. Errors in interaction might be corrected by deleting the oldest remembered constraints on α.

Given an updated set $A^{(l+1)}$, the next extreme efficient point $\boldsymbol{q}^{(l+1)}$ is generated not as the preferred adjacent extreme point, but as a solution to the problem:

$$\max_{\boldsymbol{q} \in Q_0} \sum_{i=1}^{k} \alpha_i^{(l+1)} q_i \tag{9.10}$$

where $\alpha^{(l+1)}$ is an (arbitrary) element of $A^{(l+1)}$.

A reasonable modification of the Zionts-Wallenius procedure would be to let the decision maker choose between $\boldsymbol{q}^{(l+1)}$ and the preferred adjacent extreme point. The original Zionts-Wallenius procedure, though it pays much attention to the interaction with the decision maker, is complicated in details; many further modifications, often simplifying this procedure, have been proposed (see e.g., Zionts and Wallenius, 1983). The main advantage of this procedure, viewed from a contemporary perspective, is that the properties of the set Q_0 and the thresholds of accuracy are used interactively with the decision maker to produce a procedure with finite convergence.

9.2.3 Korhonen-Laakso procedure and Pareto Race

Korhonen and Laakso (1986) proposed a visual interactive method for controlling the selection of efficient outcomes and decisions, incorporated in a software tool called Pareto Race.

The essence of this method is a directional search for improvements of the satisfaction of the decision maker, using the outcomes of efficient decisions (or value function). Starting from an efficient objective outcome point $\hat{\boldsymbol{q}}^{(l)},\ k = 1$, the search is a repetition of the following three phases:

- The determination of a direction $\boldsymbol{d}^{(l)} \in \mathbb{R}^k$ that would point toward some improvements of the decision maker's satisfaction.
- The computation of several efficient objective outcomes and decisions corresponding to reference points distributed along the direction.
- The choice of the step-length in the direction, i.e., a decision and objective outcome determined in a previous phase that provides the best improvement of the satisfaction of the decision maker; the resulting point becomes the next $\hat{\boldsymbol{q}}^{(l+1)}$.

Thus, the general concept of the Korhonen-Laakso procedure is similar to that of the Geoffrion-Dyer-Feinberg procedure, with some essential differences. The direction $\boldsymbol{d}^{(l)}$ is chosen in Pareto Race by the decision maker, but she/he might do it either arbitrarily (and take full responsibility for the convergence of the procedure) or be supported by an appropriate procedure. Such a procedure might consist in asking the decision maker, for each objective component q_i, how much this component should be – as compared to $\hat{q}_i$ – increased or decreased in relative terms (in % of the known range $[q_{i,lo}, q_{i,up}]$). One way of doing this is with a prepared questionnaire form, asking the decision maker whether the objective q_i should be:

- Strongly increased ($d_i^{(l)} = 0.1(q_{i,up} - q_{i,lo})$);
- Moderately increased ($d_i^{(l)} = 0.03(q_{i,up} - q_{i,lo})$);
- Slightly increased ($d_i^{(l)} = 0.01(q_{i,up} - q_{i,lo})$);
- Kept constant ($d_i^{(l)} = 0$);
- Slightly decreased ($d_i^{(l)} = -0.01(q_{i,up} - q_{i,lo})$);
- Moderately decreased ($d_i^{(l)} = -0.03(q_{i,up} - q_{i,lo})$);
- Strongly decreased ($d_i^{(l)} = -0.1(q_{i,up} - q_{i,lo})$),

for all $i = 1, \ldots, k$ consecutively.

Given a direction $\boldsymbol{d}^{(l)}$, several reference points can be chosen as follows:

$$\begin{aligned} \bar{\boldsymbol{q}}^{(l,j)} &= \hat{\boldsymbol{q}}^{(l)} + \tau_j \boldsymbol{d}^{(l)}, \\ \tau_j &= 2^j,\ j = -3, -2, -1, 0, 1, 2, 3, \end{aligned} \tag{9.11}$$

which can be additionally projected on the ranges $[q_{i,lo}, q_{i,up}]$, if necessary. The efficient objective outcomes that are presented to the decision maker for evaluation are obtained in the Pareto Race method by a maximization of an order-consistent achievement function (of a form similar to those discussed in Chapter 4):

$$\hat{\boldsymbol{q}}^{(l,j)} = \underset{\boldsymbol{q} \in Q_0}{\operatorname{argmax}}\, \sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}^{(l,j)}). \tag{9.12}$$

These points can be considered as projections[6] of the reference points $\hat{\boldsymbol{q}}^{(l,j)}$ (changing along a direction) on the properly efficient frontier $\hat{Q}_0^{p\varepsilon}$. Thus, another essential difference from the Geoffrion-Dyer-Feinberg procedure – and similar to

[6]It should be stressed, however, that these projections can be interpreted as results of a distance minimization only if $\bar{\boldsymbol{q}}^{(l,j)}$ are "above to the north-east" of $\hat{Q}_0^{p\varepsilon}$. If $\bar{\boldsymbol{q}}^{(l,j)} \in Q_0$ (which might occur in nonconvex cases), then these projections represent a uniform improvement of all objective components rather than distance minimization, because an order-consistent achievement function is generally not a norm.

the Zionts-Wallenius procedure – is that Pareto Race moves among efficient points.

The choice of a step-length in the direction $\boldsymbol{d}^{(l)}$ can be prepared by graphically displaying these projections as profiles of changes of component objectives $\hat{q}_i^{(l,j)}$ along changing j. An important feature of the Pareto Race tool is its graphically supported interaction with the decision maker, who uses such profiles in order to select some j for which the improvement of all objective components is best balanced. This choice constitutes the third stage of the method and determines the next starting point for a new iteration:

$$\hat{\boldsymbol{q}}^{(l+1)} = \hat{\boldsymbol{q}}^{(l,j)} \quad \text{for selected } j. \tag{9.13}$$

If the decision maker is not satisfied with any j – because $\hat{\boldsymbol{q}}^{(l,j)}$ is no better than $\hat{\boldsymbol{q}}^{(l)}$ – the DSS should ask her/him to specify another direction $\boldsymbol{d}^{(l)}$. This will hopefully lead to an improvement, in which case the decision maker takes the responsibility for the convergence. The search stops if the decision maker cannot suggest a direction of improvement. Korhonen and Laakso (1986) presented a convergence proof of this procedure under appropriate assumptions.

9.2.4 Stochastic quasi-gradient procedures

The interactive procedures described in the previous sections have some corrective properties in case of errors made by the decision makers; however, they do not take into account such errors systematically. A way exists, however, of obtaining slow but sure convergence for directional search methods even if the evaluations of the decision maker are subject to random errors. This way is known from the old technique of stochastic approximation, or from the more general[7] technique of stochastic quasi-gradient of Ermoliev (see e.g., Ermoliev and Wets, 1988). It consists in applying a harmonic sequence of fixed step-length coefficients for each iteration (l):

$$\begin{aligned} \tau_{(l)} &= \tau_{(1)}/l; \quad l = 1, 2 \ldots; \\ \bar{\boldsymbol{q}}^{(l+1)} &= \hat{\boldsymbol{q}}^{(l)} + \tau_{(l)} \boldsymbol{d}^{(l)}; \\ \hat{\boldsymbol{q}}^{(l+1)} &= \underset{\boldsymbol{q} \in Q_0}{\operatorname{argmax}} \, \sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}^{(l+1)}). \end{aligned} \tag{9.14}$$

The step to $\hat{\boldsymbol{q}}^{(l+1)}$ can either always be made (which results in a not monotone but still convergent variant of the procedure) or can only be made if the decision maker accepts $\hat{\boldsymbol{q}}^{(l+1)}$ to be better than $\hat{\boldsymbol{q}}^{(l)}$. If the step is not made, the decision maker has to redefine the direction $\boldsymbol{d}^{(l)}$. Owing to the fixed but harmonically

[7] Also taking into account nondifferentiable and constrained cases.

decreasing coefficients, the procedure converges under the assumption that the directions specified by the decision maker approximate (with random errors but without bias) the stochastic quasi-gradient of her/his value function (see Ermoliev and Wets, 1988). Michalevich (1986) has used such a technique to propose an interactive procedure of decision support with proven convergence.

The convergence of such techniques is ensured by sufficiently slow convergence such that the stochastic errors of evaluations can be averaged. Even if this idea is attractive and theoretically powerful, practical experiments show that decision makers find such convergence too slow.

9.3 Generating Options for Interactive Choice

Several approaches exist that aim to accelerate convergence of interactive decision support procedures. These approaches are based on presenting a larger sample of decision options and their outcomes to the decision maker. It is assumed that the decision maker is able to choose the most preferred outcome of such a sample. Theoretically, this is a difficult cognitive task; therefore, it has often been argued that a technique of discrete multi-attribute choice should be used additionally, not to produce the final answer, but as an iteration of an interactive process. Thus, we might repetitively apply previously described techniques of AHP or multiplicative AHP, or other techniques. These techniques, such as ELECTRE or PROMETHEE (see Roy and Vincke, 1981; Brans and Vincke, 1985), are based on outranking relations rather than on value functions.

Practically, however, if the decision maker can perform the more difficult cognitive task, her/his time can be saved. The time spent by a decision maker is often the most important attribute of a decision process. Moreover, we have already stressed that learning by the decision maker means changes of her/his value function during the decision process. It is important, therefore, to develop methods that do not require consistency of value judgments. Such procedures might be preferred because of their practical efficiency, even if they typically have no proven convergence. In practice, results might be obtained faster with such procedures than with procedures that have guaranteed convergence. Therefore, in the following sections we briefly outline some of these procedures.

9.3.1 Reference point trials

The basic principle behind reference point methods (see e.g., Wierzbicki, 1980a) is to support the decision maker – or user of a DSS – by generating properly efficient decisions and their outcomes, corresponding to a given reference point $\bar{q}$ in objective space. As explained in Chapter 4 and used in many other related methods (for example, in Pareto Race), efficient points $\hat{\boldsymbol{q}}$ are obtained by maximizing an order-consistent achievement function:

$$\hat{\boldsymbol{q}} = \operatorname*{argmax}_{\boldsymbol{q} \in Q_0} \sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}). \tag{9.15}$$

The dependence of $\hat{\boldsymbol{q}}$ on $\bar{\boldsymbol{q}}$ can be interpreted as either a projection of $\bar{\boldsymbol{q}}$ on Q_0 if $\bar{\boldsymbol{q}}$ is located to the north-east (in the case of maximization of all objectives) of the properly efficient set $Q_0^{p\varepsilon}$, or as a uniform improvement of $\bar{\boldsymbol{q}}$ if it is located to the south-west of $Q_0^{p\varepsilon}$ (e.g., if $\bar{\boldsymbol{q}} \in IntQ_0$). It can also be interpreted differently in other situations. The interpretation of such a dependence is slightly more complicated if we use both aspiration points $\bar{\boldsymbol{q}}$ and reservation points $\bar{\bar{\boldsymbol{q}}}$ and an appropriate order-consistent achievement function depending on both of them (see Chapters 4 and 5 for definitions of these achievement functions). In the following definition:

$$\hat{\boldsymbol{q}} = \operatorname*{argmax}_{\boldsymbol{q} \in Q_0} \sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}, \bar{\bar{\boldsymbol{q}}}), \tag{9.16}$$

where the maximization over attainable objectives $\boldsymbol{q} \in Q_0$ means actually the maximization over admissible decisions $\boldsymbol{x} \in X_0$ with the substitution $\boldsymbol{q} = \boldsymbol{F}(\boldsymbol{x})$.

Usually, we try to locate aspiration points $\bar{\boldsymbol{q}}$ to the north-east (in the case of maximization of all objectives) of the properly efficient set $Q_0^{p\varepsilon}$ and the reservation points $\bar{\bar{\boldsymbol{q}}}$ to the south-west of this set. $\hat{\boldsymbol{q}}$ then becomes just a properly efficient point on (or close to) the intersection of the line segment joining $\bar{\bar{\boldsymbol{q}}}$ and $\bar{\boldsymbol{q}}$ and the properly efficient frontier (see *Figure 9.2(a)*). Even if we do not succeed in an appropriate location of the aspiration and the reservation point, we still obtain efficient points $\hat{\boldsymbol{q}}$. However, these are located only on (or close to) the line segments joining either the aspiration point and the utopia point (see *Figure 9.2(b)*), or the reservation point and the approximate nadir point (see *Figure 9.2(c)*). Reference point trials consist thus in changing the reference point (or aspiration and reservation points) and using these changes to learn about the shape of the efficient frontier.

Such pure trials of reference points and corresponding efficient points do not contribute much to supporting choice, if the decision maker is not sure about her/his preferences. However, such trials contribute greatly to learning about the decision situation. In order to illustrate this point, we consider the example used already in Chapter 6 together with an economic interpretation. Suppose an investor must provide a sum of money to cover two investments, $x01$ and $w01$; the investor is interested in various outcomes of these investments after three years. Suppose the joint negative effect (risk, additional cost, etc.) of such investments can be modeled as $sum = y13 + y23$, an outcome to be minimized. Assume also that it is desirable to obtain a specific value 0.3 of the outcome $y23$.[8] Suppose, additionally, that

[8]The sign of this specific value as compared to the example considered in Chapter 6 is changed to make it easier to interpret positive quantities.

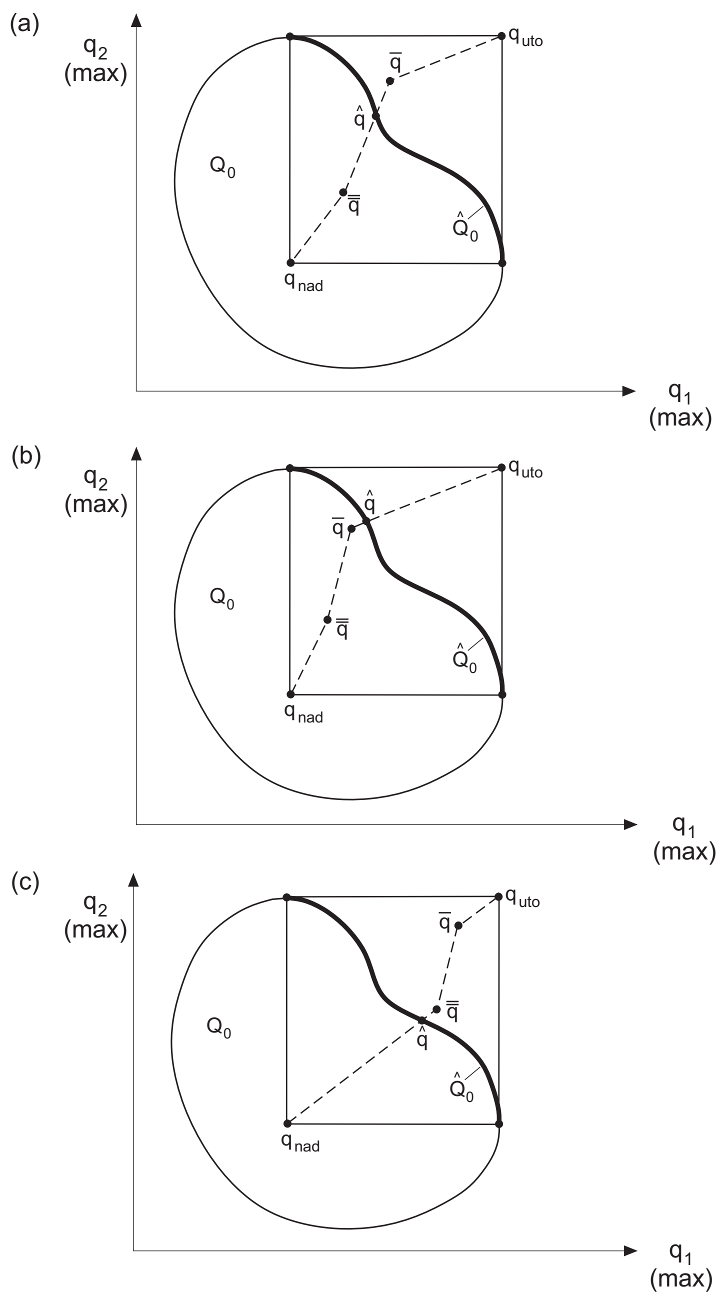


Figure 9.2. Various cases of locating aspiration and reservation points versus the efficient frontier. In case (a) aspiration ($\overline{q}$) is not attainable and reservation ($\overline{\overline{q}}$) is attainable; in (b) both $\overline{q}$ and $\overline{\overline{q}}$ are attainable; and in (c) both $\overline{q}$ and $\overline{\overline{q}}$ are not attainable.

a specialist in finance prepared a complicated model predicting various outcomes and results of the initial investments, and that the model can be summarized by the equations already used in Chapter 6, augmented by an equation for the sum of negative effects:

```
x0 = x01;                 w0 = w01
x1 = 1.1*x0;              w1 = (1+a)*w0 - a*(w0)^2 + 0.5*x0
x2 = 1.1*x1;              w2 = (1+a)*w1 - a*(w1)^2 + 0.5*x1
x3 = 1.1*x2;              w3 = (1+a)*w2 - a*(w2)^2 + 0.5*x2
y10 = w0 - 0.5*x0;       y20 = 0.3*y10 + 0.15*w0
y11 = w1 - 0.5*x1;       y21 = 0.3*y11 + 0.15*w1
y12 = w2 - 0.5*x2;       y22 = 0.3*y12 + 0.15*w2
y13 = w3 - 0.5*x3;       y23 = 0.3*y13 + 0.15*w3
fin = 100*(y23 - 0.3)^2;  sum = 100*(y13 + y23).
```

In this equation the coefficients 100 are introduced as a simple rescaling, the parameter $a = 0.5$ and we use the initial values $x01 = 0.5$ and $w01 = 3.0$ (e.g., as a starting point for optimization).

For this illustrative example let us assume that the investor believes in the expertise of the specialist who constructed the model,[9] rather than question the validity of the model and possible means of its validation. The investor simply wishes to use the model to decide what the initial investment and its actual division between $x01$ and $w01$ should be, and what final values of fin and sum are acceptable for her/him.

In an approach that concentrates on automating decisions, the investor would initially be asked about her/his preferences and trade-offs (e.g., how much they are willing to increase the negative end effect sum in order to diminish fin by a unit. At this point, however, the investor would be justified in refusing to answer such questions; first the ranges of reasonable decisions and decision outcomes must be learned. For this purpose, reference point trials are the best tool.

The model presented above can be easily programmed, e.g., when using DIDAS-N (see Chapter 6). However, before proceeding to multi-objective model analysis, DIDAS-N requests that each variable has defined bounds.[10] Suppose we choose $[-500, 500]$ as the range of all variables except $x01$, $w01$, and sum, which are assumed to be bounded to the range $[0, 500]$ (the variable fin is also naturally bounded below by 0, but this is not introduced as an additional bound). After defining the bounds and fixing and saving the model, DIDAS-N changes the format of its spreadsheet and we can then proceed to multi-objective model analysis. We define

[9]This is a very strong assumption; however, we use an arbitrary model to illustrate the diverse problems of investigating various model properties, and do not consider the issue of model validity.

[10]A disadvantage of DIDAS-N is that it requires model formulation and modification in a traditional (not "user-friendly") way.

that the variables fin and sum should be minimized. The system automatically computes the utopia point and estimates the nadir point (by minimizing separately these variables) as well as computes a neutral compromise point (by assuming an aspiration and a reservation point on the line segment utopia-nadir and maximizing an achievement function). With the standard setting of DIDAS-N options,[11] we obtain then the following results:

```
variable        utopia       neutral       nadir
fin           1.170e-15    6.765e-01    2.587e+00
sum           1.863e-14    3.381e+00    1.282e+01.
```

It is also possible to obtain either fin or sum (but not both) that are sufficiently small. sum does not have a unique minimum but an entire nonlinear manifold of minima in this model; the issue is whether there is a point in this manifold where fin is also zero. The above results suggest a negative answer. Therefore, we would like to investigate the efficient frontier in more detail. After fixing and saving the above results as the basis for a multi-objective problem of model analysis, DIDAS-N again changes the format of its spreadsheet and we can specify interactively aspiration and reservation levels. The results of such a session are as follows:

```
Res\_No  fin_asp     fin     fin_res    sum_asp     sum     sum_res
0      1.00e-01  1.48e-01  1.00e+00   4.00e+01  4.11e+01  6.00e+01
I      2.00e-01  4.11e-01  2.00e+00   2.00e+01  2.23e+01  4.00e+01
II     4.00e-01  6.33e-01  4.00e+00   1.00e+01  1.04e+01  2.00e+01
III    6.00e-01  9.30e-01  6.00e+00   5.00e+00  4.97e+00  1.00e+01
IV     1.20e+00  9.71e-01  6.00e+00   2.00e+00  1.17e+00  4.00e+00
V      2.40e+00  9.61e-01  6.00e+00   1.00e+00  0.0       2.00e+00
VI     1.00e-01  7.95e-06  2.00e-01   1.00e+00  0.0       2.00e+00.
```

A significant cusp can be observed at the results No. V, VI (better visible in *Figure 9.3*, where these results are presented graphically). The model is not convex and all results 0 ... IV correspond to local minima of the achievement function. The region of these local minima can be abandoned upon reaching results V and VI. Thus, there are two leafs of the efficient frontier, as presented in *Figure 9.3*: the first leaf is local and the second leaf (practically coinciding with the axes $sum = 0$, $fin > 0$ and $fin = 0,\ sum > 0$) represent global minima; we know that these minima are global because $sum \geq 0$ and $fin \geq 0$.[12]

[11] The accuracy of optimization (the admissible value of the norm of the gradient of the minimized function) is 0.0001 and the limiting number of iterations 1,000.

[12] There are no methods of proving with absolute certainty that a minimum is global in the case of nonlinear, nonconvex models; we can only apply Monte Carlo or genetic algorithms to prove that a minimum is global with a given probability. However, the computational effort needed for such proof increases sharply with the problem's dimensions and with the required probabilistic accuracy.

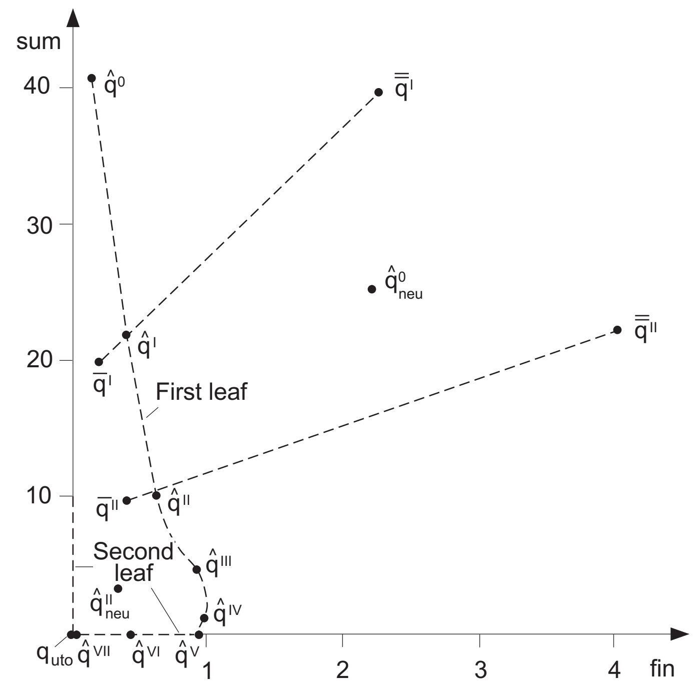


Figure 9.3. Approximation of an efficient frontier: a badly conditioned nonlinear case.

The initial results can easily lead to erroneous conclusions but a more detailed model analysis corrects these results. It is possible to achieve sum and fin both sufficiently small, however, the difficulty is to find values of $x01$ and $w01$ that produce this global minimum (circa $x01 = 4.35$ and $w01 = 4.49$ are required for this purpose). This is a common case with nonconvex models. Moreover, the investigated model – or rather the multi-objective analysis problem – is badly conditioned. This can be seen from *Figure 9.3*: the originally computed neutral compromise point, denoted by $\hat{\boldsymbol{q}}^o_{neu}$, is not locally efficient. This is because it is not a truly optimal solution, but only a solution obtained with a given accuracy 10^{-4} of optimization. This solution does not correspond to a minimum but to a point on a function plateau. We can test this by changing slightly the aspiration or reservation levels displayed by DIDAS-N at the neutral solution and by letting the system optimize anew. At a true minimum, after a few iterations the optimization would stop at a similar solution; in the case considered here, the optimization starts and goes through many iterations to reach a quite different solution. After two such tests, the point denoted by $\hat{\boldsymbol{q}}^{II}_{neu}$ in *Figure 9.3* is reached and we would probably go even further toward zero when the accuracy of optimization is increased.

This does not mean, however, that the entire model considered here is always badly conditioned. Suppose, for example, that the investor changed her/his mind and asked a different question: whether it would be possible to obtain $y13 = 0.3$ and $y23 = 0.3$ at the same time and, if not, what the possible compromises would be. We know that there are many pairs $x01,\ w01$ for which $y23 = 0.3$ is obtained; however, between them there might also be a pair that results in $y13 = 0.3$. To analyze this problem, we augment the model by an additional outcome variable aux:

```
aux = 100*(y13 - 0.3)^2
```

and start a multi-objective analysis problem by specifying that both fin and aux should be minimized. The results of such an analysis are presented graphically in *Figure 9.4*. We have a similar problem with computing the starting point $\hat{\boldsymbol{q}}^0_{neu}$ but, after two tests of slightly changing aspiration levels, the neutral solution stabilizes at a point indicated by $\hat{\boldsymbol{q}}^{II}_{neu}$. (In this instance the initial point for $x01,\ w01$ can also be changed: DIDAS-N has a warm start and the initial point can be easily changed by switching between model edition and multi-objective analysis.) There is a stable efficient frontier indicated in *Figure 9.4*, together with some examples of aspiration and reservation points and resulting efficient points. Naturally, we cannot prove that this is the global efficient frontier; however, after a detailed analysis of the model (in particular, after several changes of initial points for $x01, w01$), we can be reasonably sure that there are no other minima. Other advantages of the reference point approach can be observed in this case: note that the results denoted by $\hat{\boldsymbol{q}}^V$, $\hat{\boldsymbol{q}}^{VI}$, $\hat{\boldsymbol{q}}^{VII}$ in this figure correspond to very high trade-off coefficients between fin and aux, hence they could not be as easily obtained when using, for example, a weighted sum approach.

The basic reference point approach is not aimed at directly supporting choice, but more at interactive option generation and learning through a multi-objective analysis of substantive models of decision situation. However, for cases of nonlinear or otherwise complicated models, an informed choice is impossible without a thorough analysis of the model of decision situation. An ad hoc decision support, concentrating on an early definition of trade-offs or weighting coefficients without understanding the more basic features of the decision situation, could result in erroneous decisions. Alternatively, a pure reference point approach must be augmented by other methods in order to fully support choice. We describe some of these methods in the next sections.

9.3.2 Reference ball

A basic concept in reference point methodology is to approximate the preferred decision outcome by a collection of points. Such a collection might consist of a central point $\hat{\boldsymbol{q}}^{(l)}$ (where the upper index (l) denotes the iteration number). At

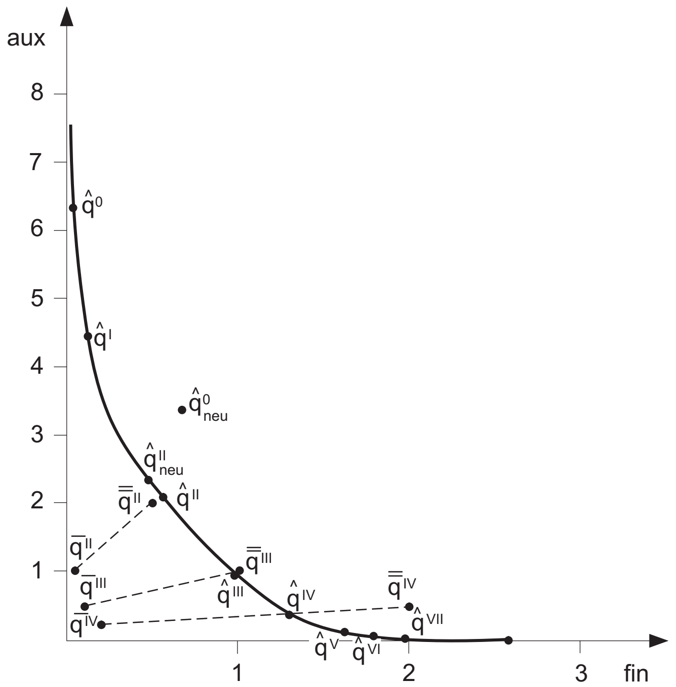


Figure 9.4. Approximation of an efficient frontier: a reasonably conditioned nonlinear case.

$l = 1$ this point could be selected, e.g., as a neutral point in the efficient frontier, and also as a number of points scattered around the central point in a natural way:

$$\begin{aligned}
\delta^{(l)} &= 0.1/l, \quad l = 1, 2, \ldots \\
\bar{\boldsymbol{q}}^{(l,j)} &= \hat{\boldsymbol{q}}^{(l)} + \delta^{(l)}\boldsymbol{e}^{(j)}, \ \ \boldsymbol{e}^{(j)} = (0, \ldots, 0, 1_{(j)}, 0, \ldots, 0)^T \\
\hat{\boldsymbol{q}}^{(l,j)} &= \operatorname*{argmax}_{\boldsymbol{q} \in Q_0} \sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}^{(l,j)})
\end{aligned} \tag{9.17}$$

where $\bar{\boldsymbol{q}}^{(l,j)}$ is one of the scattered reference points and $\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}^{(l,j)})$ is an order-consistent achievement function. The resulting $m + 1$ points $\hat{\boldsymbol{q}}^{(l)}$, $\hat{\boldsymbol{q}}^{(l,j)}$, $j = 1, \ldots, m$ are efficient, because of the properties of the achievement function. The decision maker then selects the most preferred one as the next central point $\hat{\boldsymbol{q}}^{(l+1)}$. The harmonic decrease of the radius of scattering $\delta_{(l)}$ was selected in order that the procedure might be convergent, even in the presence of random evaluation errors. However, no formal proof of convergence was given; moreover, practical trials have shown that this procedure is too slow for a typical decision maker.

This procedure illustrates the idea of a reference ball: i.e., given a current point $\hat{\boldsymbol{q}}^{(l)}$, scatter a given number of additional points in a ball of (relative) radius $\rho^{(l)}$

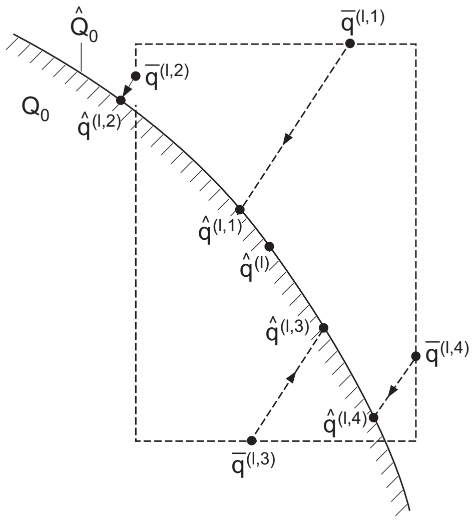


Figure 9.5. Reference box and resulting efficient outcomes.

centered at the current point and use the additional points as reference points. If we take the Chebyshev norm to define the ball, it becomes a box centered at $\hat{\boldsymbol{q}}^{(l)}$:

$$\begin{aligned} B(\hat{\boldsymbol{q}}^{(l)}, \rho^{(l)}) &= \{\bar{\boldsymbol{q}} \in \mathbb{R}^m : |\bar{q}_i - \hat{q}_i^{(l)}| \le \rho^{(l)}(q_{i,up} - q_{i,lo}) \ \forall i = 1, \dots, k\} \\ &\quad \rho^{(l)} \in (0,\, 1). \end{aligned} \tag{9.18}$$

Suppose we choose some number P of randomly generated points $\bar{\boldsymbol{q}}^{(l,j)}$, $j = 1, \dots, P$ on the boundary of this box. To generate points on the box boundary, it is sufficient to select randomly the index of the component i and then to generate randomly (e.g., with a uniform distribution) an aspiration component:

$$\bar{q}_i \in [\hat{q}_i^{(l)} - \rho^{(l)}(q_{i,up} - q_{i,lo}),\ \hat{q}_i^{(l)} + \rho^{(l)}(q_{i,up} - q_{i,lo})] \tag{9.19}$$

and then project this component on the box boundary:

$$\begin{aligned} \bar{q}_i^{(l,j)} &= \hat{q}_i^{(l)} + \rho^{(l)}(q_{i,up} - q_{i,lo}) \ \text{ if } \ \bar{q}_i \ge \hat{q}_i^{(l)} \\ \bar{q}_i^{(l,j)} &= \hat{q}_i^{(l)} - \rho^{(l)}(q_{i,up} - q_{i,lo}) \ \text{ if } \ \bar{q}_i < \hat{q}_i^{(l)}. \end{aligned} \tag{9.20}$$

For other components $i+1, \dots, m,\ 1, \dots, i-1$ of the vector $\bar{\boldsymbol{q}}^{(l,j)}$ we repeat the generation as in (9.19), but not necessarily the same projection as in (9.20). If we repeat the projection, we obtain a random sequence of corner points of the box. Having generated the points $\bar{\boldsymbol{q}}^{(l,j)},\ j = 1, \dots, P$, we might filter them additionally by selecting points most distant from the box center.

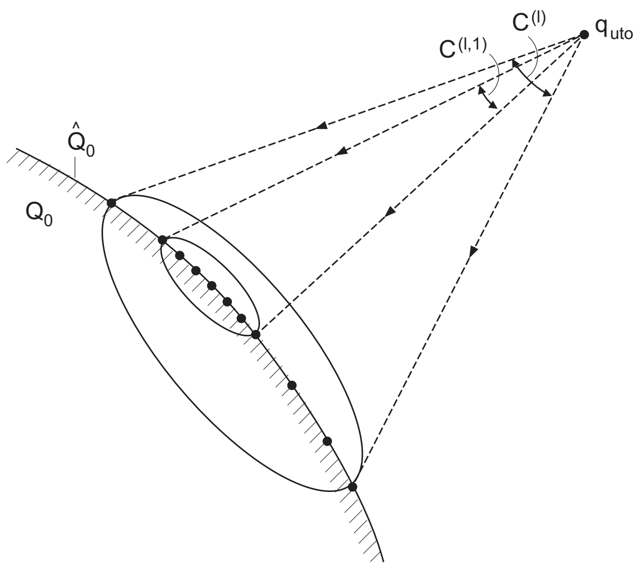


Figure 9.6. Contracted cone illuminating efficient surface.

We proceed then to compute the corresponding properly efficient outcome points by repeated maximization of an order-consistent achievement function:

$$\hat{\boldsymbol{q}}^{(l,j)} = \operatorname*{argmax}_{\boldsymbol{q}\in Q_0} \sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}^{(l,j)}),\ j = 1, \ldots, P. \tag{9.21}$$

The resulting properly efficient outcomes $\hat{\boldsymbol{q}}^{(l,j)}$ will not necessarily be the most distant from $\hat{\boldsymbol{q}}^{(l)}$ (see *Figure 9.5*), even if we would randomly choose aspiration points only in the box corners. Therefore, an additional filtering of these outcomes might be performed. A selected number of spread points $\hat{\boldsymbol{q}}^{(l,j)},\ j = 1, \ldots, P'$, and the center $\hat{\boldsymbol{q}}^{(l)}$ are presented to the decision maker who is supposed to select the most preferred one as the next $\hat{\boldsymbol{q}}^{(l+1)}$.

9.3.3 Contracted cone

A similar idea is employed in the contracted cone procedure of Steuer and Choo (1983). The original procedure was to define a contracting set of weighting coefficients that could be interpreted as a cone of directions in objective space, starting at the upper bound or utopia point and "illuminating" a part of the efficient surface. The decision maker would choose a point in the "illuminated" part, around which a new but contracted "narrower" cone would be centered (see *Figure 9.6*).

The contracted cone procedure was defined originally in terms of weighting coefficients and an augmented Chebyshev norm specifying the distance from the utopia point; later, it was interpreted in terms of contracting directional cones. As discussed in Chapter 4, we can equivalently define this procedure in terms of a ball of reference points with a contracting radius and use the maximization of an order-consistent achievement function.

The convergence of such a procedure depends critically on the choice of the sequence $\rho^{(l)}$. We can choose either an exponential or a harmonic sequence:

$$\rho^{(l)} = 1/2^l \text{ or } \rho^{(l)} = 1/2l, \ l = 1, \ldots \infty. \tag{9.22}$$

Steuer and Choo (1993) proposed the exponential sequence. Although their procedure has found many successful applications, exponential sequences might result in a lack of convergence in special cases.[13] If we choose the harmonic sequence instead (see the comments on stochastic quasi-gradient) the procedure converges. However, the convergence with the harmonic sequence is rather slow.

9.3.4 Satisficing trade-off

The satisficing trade-off method (STOM) was developed by Nakayama (1984) as one of the aspiration level approaches. In STOM, the aspiration point at the (l)-th iteration $\bar{\boldsymbol{q}}^{(l)}$ is modified as follows:

$$\bar{\boldsymbol{q}}^{(l+1)} = \boldsymbol{T} \circ \boldsymbol{P}(\bar{\boldsymbol{q}}^{(l)}), \tag{9.23}$$

where $\boldsymbol{P}$:

$$\begin{aligned} \boldsymbol{P}(\bar{\boldsymbol{q}}^{(l)}) &= \hat{\boldsymbol{q}}^{(l)} = \boldsymbol{F}(\hat{\boldsymbol{x}}^{(l)}) \\ \hat{\boldsymbol{x}}^{(l)} &= \underset{\boldsymbol{x} \in Q_0}{\operatorname{argmax}} \ \sigma(\boldsymbol{F}(x), \bar{\boldsymbol{q}}^{(l)}). \end{aligned} \tag{9.24}$$

In this approach, $\boldsymbol{P}$ is the operator of projecting the aspiration point $\bar{\boldsymbol{q}}^{(l)}$ on the Pareto surface by maximizing an achievement function $\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}^{(l)})$, and $\boldsymbol{T}$ is an operation of aspiration modification, defined by the decision maker but supported by an analytical computation of trade-off coefficients that might be a by-product of the optimization (9.24).

Thus, the procedure is composed of two stages. First, given the aspiration point $\bar{\boldsymbol{q}}^{(l)}$, we determine a Pareto-optimal solution corresponding to it (the operator $\boldsymbol{P}$). Second, if the decision maker is not satisfied with the shown objectives $\hat{\boldsymbol{q}}^{(l)} = \boldsymbol{P}(\bar{\boldsymbol{q}}^{(l)})$ (which implies the decision $\hat{\boldsymbol{x}}^{(l)}$), she/he is required to make a trade-off among criteria $q_i, i = 1, \ldots, k$. Suppose that the decision maker wants to

[13]Consider the following single-dimensional example: maximize $q = -(x-2)^2$ starting with $x^{(1)} = 0$ and with the sequence of limited steps $x^{(l+1)} - x^{(l)} = 1/2^l, \ l = 1, \ldots \infty$. Obviously, $\lim_{l \to \infty} x^{(l)} = 1$, which falls short of $\hat{x} = 2$.

improve some of the criteria. Because $\hat{\boldsymbol{q}}^{(l)} = \boldsymbol{P}(\bar{\boldsymbol{q}}^{(l)})$ is Pareto-optimal, no feasible solution exists that makes all criteria better than $\hat{\boldsymbol{q}}^{(l)}$. Thus, the decision maker has to agree to relax other criteria for the compensation of the improvement; this defines the operator $\boldsymbol{T}$. Based on this trade-off, a new aspiration level is decided as $\boldsymbol{T} \circ \boldsymbol{P}(\bar{\boldsymbol{q}}^{(l)})$. Such a process is continued until the decision maker is satisfied with the obtained criteria and solution.

The satisficing trade-off method provides certain devices that make the trade-off analysis easier by using the sensitivity analysis and parametric optimization techniques in traditional mathematical programming. We turn now to a more detailed presentation of STOM.

The operation $\boldsymbol{P}$ is performed by an auxiliary scalar optimization solver. This optimization is equivalent to maximizing an achievement function in the augmented maxmin form:

$$\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}^{(l)}) = \min_{1 \le i \le k} \sigma_i(q_i, \bar{q}_i^{(l)}) + \varepsilon \sum_{i=1}^{k} \sigma_i(q_i, \bar{q}_i^{(l)}). \tag{9.25}$$

This is applied in STOM, as well as in other reference or aspiration point methods, because it yields a properly Pareto-optimal solution (not merely a weakly Pareto-optimal solution), as shown in Chapter 5. Here ε is a sufficiently small positive number, say 10^{-6}. When partial achievement functions $\sigma_i(q_i, \bar{q}_i^{(l)})$ are used in STOM, they are usually defined as:

$$\begin{aligned} \sigma_i(q_i, \bar{q}_i^{(l)}) &= (q_i - \bar{q}_i^{(l)})/(q_{i,up} - \bar{q}_i^{(l)}) \quad \text{for } i \in I_{max}, \\ \sigma_i(q_i, \bar{q}_i^{(l)}) &= (\bar{q}_i^{(l)} - q_i)/(\bar{q}_i^{(l)} - q_{i,lo}) \quad \text{for } i \in I_{min}, \end{aligned} \tag{9.26}$$

where I_{max}, I_{min} are the index sets of maximized and minimized objectives, and $q_{i,up}$, $q_{i,lo}$ are estimated during utopia and nadir computations. In the case described above, the weighting (actually scaling) coefficients $\alpha_i = 1/(q_{i,up} - \bar{q}_i^{(l)})$ or $\alpha_i = 1/(\bar{q}_i^{(l)} - q_{i,lo})$ change with $\bar{\boldsymbol{q}}^{(l)}$ and thus from iteration to iteration. However, we can also use constant scaling factors $\alpha_i = 1/(q_{i,up} - q_{i,lo})$ while defining the partial achievement functions as follows:

$$\begin{aligned} \sigma_i(q_i, \bar{q}_i^{(l)}) &= (q_i - \bar{q}_i^{(l)})/(q_{i,up} - q_{i,lo}) \quad \text{for } i \in I_{max}, \\ \sigma_i(q_i, \bar{q}_i^{(l)}) &= (\bar{q}_i^{(l)} - q_i)/(q_{i,up} - q_{i,lo}) \quad \text{for } i \in I_{min}. \end{aligned} \tag{9.27}$$

The use of variable scaling factors as in (9.26) gives the user slightly stronger control over the outcomes of optimization (9.24) than the use of constant scaling factors as in (9.27). Optimization (9.24) concerns a nonsmooth scalarized objective function; thus, it is often performed by solving the following equivalent optimization problem:

$$\underset{(z,\boldsymbol{x})\in\mathcal{Z}_0}{\text{maximize}}\ (q = z + \varepsilon\sum_{i=1}^{k}\sigma_i(F_i(\boldsymbol{x}),\bar{q}_i^{(l)})), \tag{9.28}$$

where:

$$\mathcal{Z}_0 = \{(z,\boldsymbol{x}) \in I\!R^{n+1} :\ \boldsymbol{x} \in X_0,\ \sigma_i(F_i(\boldsymbol{x}),\bar{q}_i^{(l)}) \geq z,\ \forall i = 1,\ldots,k\}. \tag{9.29}$$

During this equivalent optimization, in most solvers it is easy to obtain also Lagrange multipliers $\lambda_i,\ i = 1,\ldots,k$ for the additional constraints defining the set $\mathcal{Z}_0$; this sensitivity information is useful for the next stage of the procedure.

The next stage – the operation $\boldsymbol{T}$ – is performed in cases when the decision maker is not satisfied with the solution for $\boldsymbol{P}(\bar{\boldsymbol{q}}^{(l)})$. The decision maker is requested to specify her/his new aspiration levels, defining the point $\bar{\boldsymbol{q}}^{(l+1)}$, or at least to specify which objectives should be improved, which can be relaxed and which should be maintained at their current levels (are acceptable). Let $I_I^{(l)}$, $I_R^{(l)}$, and $I_A^{(l)}$ denote the index sets of the objectives that are to be improved, can be relaxed, and are acceptable as they are, respectively.

For simplicity of description, consider all objectives as maximized. Clearly, $\bar{q}_i^{(l+1)} > \bar{q}_i^{(l)}$ for all $i \in I_I^{(l)}$. For $i \in I_A^{(l)}$, we can usually set $\bar{q}_i^{(l+1)} = \bar{q}_i^{(l)}$. For $i \in I_R^{(l)}$, the decision maker has to agree to decrease the value of $\bar{q}_i^{(l+1)}$. It should be noted that an appropriate sacrifice of q_j for $j \in I_R^{(l)}$ is needed in order to attain the improvement of q_i for $i \in I_I^{(l)}$.

Theoretically, it is possible for the decision maker to specify new aspiration levels of all objective functions. A natural tendency of human beings, however, is to usually specify the amount of relaxation not sufficient to compensate for the requested improvement. In practical problems, moreover, we often encounter cases with a large number of objective functions, for which the decision maker might get tired when specifying new aspiration levels for all objective functions.

Therefore, it is more practical that the decision maker specifies only the requested improvement rather than both improvement and relaxation. At this point, the sensitivity information obtained when solving the equivalent mathematical programming problems (9.28) or (9.29) becomes useful. When analyzing a linear approximation of the Pareto frontier determined with the help of Lagrange multipliers λ_i, we can determine the necessary assignment of sacrifice for q_j, $j \in I_R^{(l)}$:

$$\Delta q_j = \frac{1}{N(\lambda_j+\varepsilon)\alpha_j}\sum_{i\in I_I^{(l)}}(\lambda_i+\varepsilon)\alpha_i\Delta q_i, \quad j \in I_R^{(l)}, \tag{9.30}$$

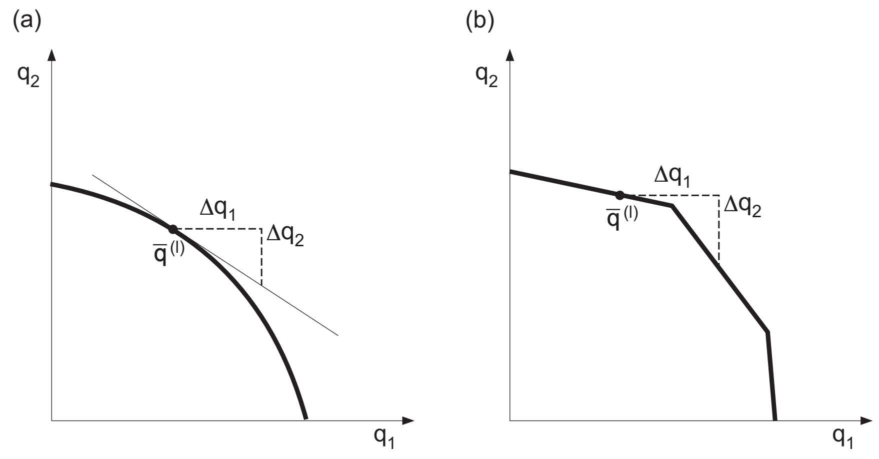


Figure 9.7. Graphical illustration of (a) automatic; and (b) exact trade-off.

where N is the number of elements of the set $I_R^{(l)}$, and α_i are the weighting or scaling coefficients (e.g., $\alpha_i = 1/(q_{i,up} - \bar{q}_i^{(l)})$) used in the partial achievement functions (9.26) or (9.26). The quantities Δq_i are the required improvements of the aspirations $\bar{q}_i^{(l+1)} - \bar{q}_i^{(l)}$ for $i \in I_I^{(l)}$, and the quantities Δq_j are the necessary sacrifices in aspirations $\bar{q}_j^{(l)} - \bar{q}_j^{(l+1)}$ for $j \in I_R^{(l)}$. These sacrifices are assigned in proportion to $(\lambda_j + \varepsilon)\alpha_j$ for all $j \in I_R^{(l)}$ (see e.g., Nakayama, 1984).

When using the above automatic trade-off method, the burden of the decision maker can be much reduced, particularly for problems with a large number of objectives. Naturally, if the decision maker does not agree with the automatically assigned Δq_j, she/he can modify it in a manual way.

For linear or quadratic models, we can evaluate the exact trade-off in a form similar to (9.30). This implies that we can calculate exactly the amount of relaxation needed for the new aspiration point to be on Pareto frontier (Nakayama, 1994). For a graphical illustration of the difference between an automatic and an exact trade-off see *Figure 9.7*. The main idea in computing exact trade-offs is that a parametric optimization technique is used instead of the simple linear approximation. When using this approach, a new Pareto-optimal solution can be obtained faster because the scalarized optimization problem does not need to be solved again. Therefore, a graphic presentation of computer outputs can be quickly presented to the decision maker, who can see the trade-off among criteria in a dynamic way, e.g., as an animation. This is helpful for the decision maker to form her/his judgment easier. A similar idea is implemented in the Korhonen-Laakso procedure (1986) presented in a previous section.

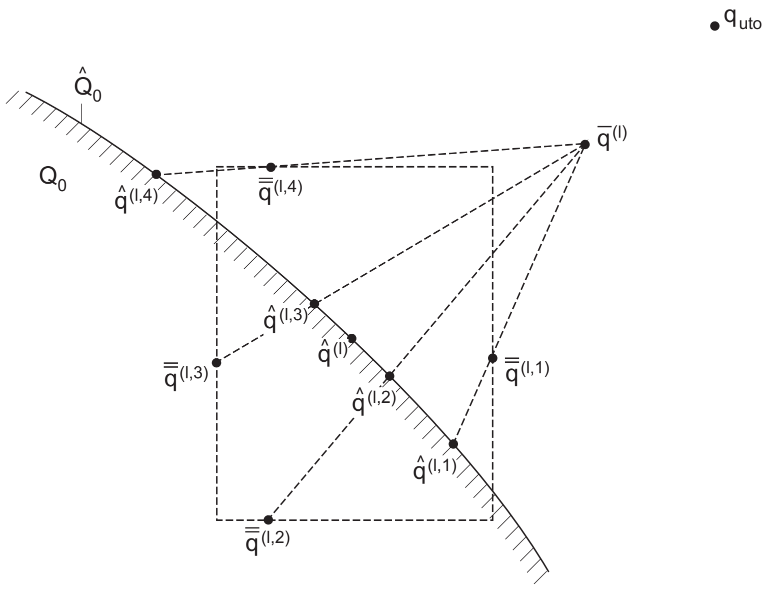


Figure 9.8. Modified light beam search method using aspiration and reservation points.

9.3.5 Light beam search method

An idea similar to the contracted cone method was applied by Jaszkiewicz and Słowiński (see e.g., 1994) when illuminating part of an efficient surface in their light beam search method. They noted, however, that the use of an order-consistent achievement scalarizing function instead of the Chebyshev metric method makes it possible to further modify the ideas of Steuer and Choo (1983). They located “the light source” at any nonattainable reference point (to the north-east of the efficient frontier, not necessarily at the utopia point) and generated the cone of light with the help of additional weighting coefficients.

We should comment that, when using both aspiration and reservation levels instead of one reference point in an order-consistent achievement function (see e.g., Granat *et al.*, 1994), we can use an aspiration point to locate the source of light and appropriate changes in reservation levels to generate the cone of light. For example, if an aspiration point $\bar{\boldsymbol{q}}^{(l)}$ and a reservation point $\bar{\bar{\boldsymbol{q}}}^{(l)}$ are used to obtain an efficient point $\hat{\boldsymbol{q}}^{(l)}$ by maximizing an order-consistent achievement function $\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}^{(l)}, \bar{\bar{\boldsymbol{q}}}^{(l)})$, then a spread of reservation points around $\hat{\boldsymbol{q}}^{(l)}$ such as described in (9.19) and (9.20) will generate an appropriate cone of light. This is illustrated in *Figure 9.8*.

Another important aspect of the light beam search method of Jaszkiewicz and Słowiński was the construction of an outranking relation in the sense of Roy (see

Vincke, 1992), in the illuminated part of the efficient frontier. This feature of the light beam search method is not presented in detail; instead, we propose (in Section 9.4) another method that combines the construction of an outranking relation directly with the specification of reference levels. The proof of convergence of this method indicates how to examine convergence of methods such as the light beam search (or other interactive procedures with accelerated convergence) that, until now, have lacked convergence proofs.

9.4 Outranking Trials

Another approach exists that illustrates the convergence of interactive procedures of multi-objective decision support. This is a new convergent procedure (see Wierzbicki, 1997a) that uses a combination of an outranking relation of the type introduced by Roy (see e.g., Roy and Vincke, 1981) and the reference point methodology.

The proposed method is as follows. Suppose the decision maker (the user of a decision support system) has performed as many experiments of multi-objective optimization with the substantive model as she/he wished, but finally selected a number of objectives to be optimized, defined what to do with them (maximize, minimize, stabilize) and specified a properly efficient outcome $\hat{\boldsymbol{q}}$ (e.g., summarizing the experience gained in the initial experiments) as a starting point for the procedure.

Starting from this outcome, the decision maker asks for help in checking whether a more satisfactory decision (eventually the most satisfactory) outcome can be found. We assume that by asking for help in convergence, the decision maker has learned enough about the substantive model and has formed a preferential model. Although this preferential model might be expressed by a value function, we assume that the decision maker does not care about small differences of value and does not want to compare decisions that result in too widely changing objectives. Thus, we shall use another preferential model in the form of an outranking relation. Both the value function and the outranking relation are assumed to be consistent with each other (which will be discussed in detail later) and with the partial order of objectives specified by the decision maker. For simplicity of description, we assume that the partial order corresponds to the maximization of all objectives, although the method can be easily generalized for minimized objectives and, with some additional details, also for stabilized objectives.

Therefore, at this point of procedure, the decision maker is asked to specify some parameters of her/his outranking relation. As an example we could use the outranking relation of the type used in ELECTRE 3 or 4 (see Vincke, 1989). However, we shall modify these outranking relations to exploit the full possibilities of reference point methods. The outranking relation proposed here is constructed as follows. For each component objective q_i, while knowing the range of changes of this objective and the point $\hat{\boldsymbol{q}}$, the decision maker is asked to specify four incremental threshold values – two obligatory, and two optional. If a range of reference levels (an aspiration level $q_{i,asp}$ and a reservation point $q_{i,res}$ such that $q_{i,res} < \hat{q}_i < q_{i,asp}$) is given, then these thresholds can also be defined as corresponding parts of this range:

- The veto threshold $\Delta q_{av,i} > 0$, e.g., $\Delta q_{av,i} = \beta(\hat{q}_i - \bar{q}_{i,res}),\ \beta \in (0,1]$.
- The negative indifference threshold $\Delta q_{ni,i} > 0$, $\Delta q_{ni,i} < \Delta q_{av,i}$ or $\Delta q_{ni,i} = \xi \Delta q_{av,i},\ \xi \in (0,1)$.
- The component outranking threshold $\Delta q_{co,i} > 0$, e.g., $\Delta q_{co,i} = \beta(\bar{q}_{i,asp} - \hat{q}_i),\ \beta \in (0,1]$.
- The positive indifference threshold $\Delta q_{pi,i} > 0$, $\Delta q_{pi,i} < \Delta q_{co,i}$ or $\Delta q_{pi,i} = \xi \Delta q_{co,i},\ \xi \in (0,1)$.

The number of objective components increased beyond the positive indifference threshold is then compared with the number of components decreased beyond the negative indifference threshold, in order to determine specific outranking relations. All of the above threshold values might depend on the current point; however, this dependence shall not be indicated explicitly, it is up to the decision maker to correct them if needed.

We assume that all component objectives are measured on ratio scales. Because the decision maker already learned about their ranges of change $q_{i,up} - q_{i,lo}$, all of the above thresholds might be also expressed in percentage terms of these ranges. We do not assume that any one objective is more important than another, although moderate ratios of importance of objectives might be expressed implicitly by the selection of thresholds and their ratios. We assume that the decision maker might specify separate negative and positive indifference thresholds (although in ELECTRE the thresholds $\Delta q_{ni,i}$ and $\Delta q_{pi,i}$ are usually equal). We base this assumption on psychological studies (see e.g., Kahneman and Tversky, 1982) showing that loss and gain are usually valued differently.

The specification of four outranking thresholds for each objective component might require too much effort from the decision maker. Therefore, we suggest that she/he specifies only two of them: the veto threshold $\Delta q_{av,i}$ and the component outranking threshold $\Delta q_{co,i}$ for each i. Their suggested values might be just a given percentage (e.g., 5%) of the objective ranges, or a given part β of the distances to the reservation and aspiration levels. If the decision maker is reluctant to define specific values for the other (indifference) thresholds $\Delta q_{ni,i}$ and $\Delta q_{pi,i}$, she/he might define a number $\xi \in (0, 1)$ that defines them generally, with the suggested value $\xi = 1/k$, where k is the number of objectives.

For any given pair of $\hat{\boldsymbol{q}}$ (which is interpreted as the current efficient point in objective space) and $\boldsymbol{q}$ (a compared point in objective space), the specified outranking thresholds might be used to define the following five index sets:

$$\begin{aligned}
I_+ &= \{i \in \{1,\ldots,k\} : q_i > \hat{q}_i + \Delta q_{pi,i}\} \\
I_- &= \{i \in \{1,\ldots,k\} : q_i < \hat{q}_i - \Delta q_{ni,i}\} \\
I_0 &= \{1,\ldots,k\} \setminus (I_+ \cup I_-) \\
I_c &= \{i \in \{1,\ldots,k\} : q_i > \hat{q}_i + \Delta q_{co,i}\} \\
I_v &= \{i \in \{1,\ldots,k\} : q_i < \hat{q}_i - \Delta q_{av,i}\}.
\end{aligned} \tag{9.31}$$

The decision maker is also asked to define an integer number k_0 in comparing the numbers of objectives increased and decreased beyond indifference thresholds, with a suggested value being, for example, the smallest integer greater than $k/6$. The component $\mathcal{C}$, normal $\mathcal{P}$ and weak $\mathcal{Q}$ outranking relations between $\boldsymbol{q}$ and $\hat{\boldsymbol{q}}$ can then be defined as follows:

$$\begin{aligned}
\boldsymbol{q}\,\mathcal{C}\,\hat{\boldsymbol{q}} &\Leftrightarrow (I_c \neq \emptyset) \wedge (I_- = \emptyset) \\
\boldsymbol{q}\,\mathcal{P}\,\hat{\boldsymbol{q}} &\Leftrightarrow (|\,I_+\,| - |\,I_-\,| \geq k_0) \wedge (I_v = \emptyset) \\
\boldsymbol{q}\,\mathcal{Q}\,\hat{\boldsymbol{q}} &\Leftrightarrow (0 \leq |\,I_+\,| - |\,I_-\,| < k_0) \wedge (I_v = \emptyset),
\end{aligned} \tag{9.32}$$

where $|\,I_+\,|$, $|\,I_-\,|$ denote the numbers of elements of sets I_+, I_-. Thus, the normal outranking of $\hat{\boldsymbol{q}}$ by $\boldsymbol{q}$ means that there is no veto. This also means that the number of objectives "in favor of $\boldsymbol{q}$" exceeds the number of objectives "in favor of $\hat{\boldsymbol{q}}$" at least by k_0, which is a number threshold defined by the decision maker. For small k, a natural choice is $k_0 = 1$. The weak outranking relation means that there is no veto and that the number of objectives in favor of $\boldsymbol{q}$ is at least equal but does not exceed by k_0 the number of objectives in favor of $\hat{\boldsymbol{q}}$. This weak outranking is interpreted, similarly to ELECTRE, as a case where the decision maker might hesitate.

For example, consider the case $k = 2,\ \ k_0 = 1$; hesitation may ensue if one component improves beyond its positive indifference threshold and the other one deteriorates beyond its negative indifference threshold but not beyond the veto threshold. For $k = 2$, normal outranking means that one component improves beyond its positive indifference threshold but the other one does not deteriorate beyond its negative indifference threshold. The component outranking is stronger than the normal one: one component improves beyond its component outranking threshold, greater than the positive indifference threshold, and the other one does not deteriorate beyond its negative indifference threshold. It is easy to consider similar cases, say, for $k = 3,\ \ k_0 = 1$.

It is convenient to consider both the outranking relations $\mathcal{P}$ and $\mathcal{Q}$ as represented by one parametric outranking relation $\mathcal{PQ}_{k_1}$ with a given parameter $k_1 \in \{0, \ldots, k-1\}$:

$$\boldsymbol{q}\, \mathcal{PQ}_{k_1}\, \hat{\boldsymbol{q}} \;\Leftrightarrow\; (|\,I_+| - |\,I_-| = k_1) \wedge (I_v = \emptyset). \tag{9.33}$$

The outranking relations considered above are slightly more sensitive than those used in ELECTRE. They will be used to produce decision options and their outcomes that are suspected of outranking the current decision and its outcome. The final choice, whether the decision option really outranks the current one, is reserved for the decision maker who plays a sovereign role in this procedure.

The essence of this procedure is the use of reference point methodology to produce points suspected of outranking the current one. This is based on a basic property of an order-consistent achievement scalarizing function of the type (8.48):

$$\bar{\boldsymbol{q}} \in Q_0 \Rightarrow \left\{ \begin{array}{rcl} \max_{\boldsymbol{q}\in Q_0}\; \sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}) & \geq & 0; \\ \hat{\boldsymbol{q}} = \operatorname{argmax}_{\boldsymbol{q}\in Q_0} \sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}) & \geq & \bar{\boldsymbol{q}} \end{array} \right\} \tag{9.34}$$

Thus, if $\max_{\boldsymbol{q}\in Q_0} \sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}) < 0$, then $\bar{\boldsymbol{q}} \notin Q_0$. Therefore, in order to test for $\boldsymbol{q}\, \mathcal{C}\, \hat{\boldsymbol{q}}$, it is sufficient to test the attainability of the reference points $\bar{\boldsymbol{q}}^{(j)}$ with component reference levels:

$$\bar{q}_i^{(j)} = \left\{ \begin{array}{ll} \hat{q}_i + \Delta q_{co,i}, & i = j \\ \hat{q}_i - \Delta q_{ni,i} & i \neq j \end{array} \right\},\; j \in \{1, \ldots, k\} \tag{9.35}$$

If the maximum of $\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}^{(j)})$ is nonnegative, we take the point maximizing this function as suspected for component outranking and present it to the decision maker for acceptance. If the maximum is negative, no component outranking point exists for the objective j. In this case, and also in the case when the decision maker does not accept a suggested point, we check for the next $j \in \{1, \ldots, k\}$; k maximizations must therefore be performed of the order-consistent achievement function (8.48).

The testing for normal and weak outranking is more complicated. Parametric outranking $\mathcal{PQ}_{k_1}$ with k_1 decreasing from $k-1$ to 0 is tested for. In order to do this, sets $I_+^{(j)}$, $I_-^{(j)}$, $I_0^{(j)}$ are defined such that:

$$|\,I_+^{(j)}| - |\,I_-^{(j)}| = k_1, \tag{9.36}$$

where (j) denotes one of all possible subdivisions of the set $\{1, \ldots, k\}$ into such sets. The number of the possible subdivisions, denoted here by $p(k, k_1)$, can be generated in a combinatorial way. This number denotes all possible partitions of the set $I = \{1, \dots, k\}$ into three sets $I_+^{(j)}$, $I_-^{(j)}$, $I_0^{(j)}$, including the cases in which the sets $I_-^{(j)}$, $I_0^{(j)}$ are empty.

For example, if $k = 3,\ k_1 = 1$, then $p(k, k_1) = 6$, since there are the following partitions:

$$\begin{array}{lcllllll} j & = & 1 & 2 & 3 & 4 & 5 & 6 \\ I_+^{(j)} & = & \{1,2\} & \{1,3\} & \{2,3\} & \{1\} & \{2\} & \{3\} \\ I_-^{(j)} & = & \{3\} & \{2\} & \{1\} & \emptyset & \emptyset & \emptyset \\ I_0^{(j)} & = & \emptyset & \emptyset & \emptyset & \{2,3\} & \{1,3\} & \{1,2\}. \end{array} \tag{9.37}$$

If $k = 4,\ k_1 = 1$, then $p(k, k_1) = 12$, etc.; it is not difficult to write a computer program generating all partitions $I_+^{(j)}$, $I_-^{(j)}$, $I_0^{(j)}$, $j = 1, \dots, p(k, k_1)$ for any k and k_1. For each such partition a specific reference point can be defined (for each k_1, the count is started anew with $j = 1, \dots, p(k, k_1)$:

$$\bar{q}_i^{(j)} = \left\{ \begin{array}{ll} \hat{q}_i + \Delta q_{pi,i}, & i \in I_+^{(j)} \\ \hat{q}_i - \Delta q_{av,i} & i \in I_-^{(j)} \\ \hat{q}_i - \Delta q_{ni,i} & i \in I_0^{(j)} \end{array} \right\}, \quad \begin{array}{l} i = 1, \dots, k, \\ j = 1, \dots, p(k, k_1). \end{array} \tag{9.38}$$

For each $j = 1, \dots, p(k, k_1)$, the order-consistent achievement function $\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}^{(j)})$ needs to be maximized again. Using the basic property of such functions we note that if any of these maxima is greater or equal to zero, then an attainable outcome outranking the point $\hat{\boldsymbol{q}}^{(l)}$ might exist. Such an outcome can be chosen as the maximal point of this function. Conversely, if all these maxima are smaller than zero, then the reference points $\bar{\boldsymbol{q}}^{(j)})$ are not attainable for all $j = 1, \dots, p(k, k_1)$ and there are no attainable points outranking the point $\hat{\boldsymbol{q}}$.

If points exist that are suspected of outranking, all of them (for a given k_1) are presented to the decision maker. The sum of numbers $p(k, k_1)$ has an upper bound 2^k and thus the total number of points to be tested in outranking trials might be as many as 2^k, if the decision maker wants to check all possibilities for $k_1 = k - 1,\ k - 2, \dots, 1, 0$. However, the decision maker has the right and motivation to stop earlier, if the stronger outranking tests give negative results. Conversely, solving such a large number (e.g., $2^k = 128$ if $k = 7$) of maximization problems is not a strong restriction today. If necessary, scalably parallel computers can be applied, special parallel computation algorithms exploited, or computing power distributed in computer networks applied.

The procedure of outranking trials can be organized as follows:

1. *Redefine the outranking thresholds.* At a given current outcome point $\hat{\boldsymbol{q}}^{(l)}$, where (l) denotes the iteration number, check if the decision maker would like to redefine the outranking thresholds. If yes, correct these thresholds.

2. *Check for component outranking.* Using reference point methodology, check whether an admissible decision exists with outcomes $\boldsymbol{q}$ that component outrank the current point. If yes, present it for acceptance to the decision maker. If $\boldsymbol{q}$ is accepted, shift current outcome point $\hat{\boldsymbol{q}}^{(k+1)} = \boldsymbol{q}$ and start a new iteration. If $\boldsymbol{q}$ is not accepted (for a possible case of different objective components), again check whether another admissible decision exists with outcomes that component outrank the current point and, if yes, present it to the decision maker. Repeat until either a new point is accepted and a new iteration started, or all possible cases of component outranking are tested negatively.
3. *Check for normal outranking.* Proceed as above, but test for the existence of possible cases of normal outranking. To do this, test for parametric outranking with k_1 decreasing from $k-1$ to k_0, until a new iteration is started or all possible normal outranking cases are tested negatively.
4. Proceed as above for k_1 decreasing from $k_0 - 1$ to 0, testing for the existence of possible cases of weak outranking. Test until a new iteration is started or all possible weak outranking cases are tested negatively.
5. *Stop criterion.* If all cases were tested negatively (either outranking decisions do not exist or the decision maker does not accept them as outranking), stop; the current point is the most satisfactory according to the accepted outranking relation. Otherwise return to point 1.

The use of three different outranking relations simplifies computations in the initial phases of the procedure. One decision option that is accepted as outranking by the decision maker is sufficient to start a new iteration. Therefore, we start by testing for the strongest relation and proceed to a weaker one when the results of the stronger one are negative. When we come to the weak outranking, we test only for decisions that might be suspected of outranking. If there are too many suggestions, the decision maker has the right to stop the procedure at any point (on the understanding that further trials will produce only results that are less strongly outranking than former trials).

In order to accelerate convergence, the above procedure might be modified as follows: when checking for outranking of a given type, points suspected of outranking are presented to the decision maker jointly, not separately. For example, in the case of component outranking, for all $j \in \{1, \ldots, k\}$ such that the maxima of $\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}^{(j)})$ are nonnegative. The decision maker can either select any of these points or reject them all.

This modification is especially important when testing for normal and weak outranking, because there might be many such tests. It can be seen that one iteration of outranking trial might be long, particularly if the decision maker wants to check all possible cases of outranking. However, the advantage of outranking trials is their assured and finite convergence.

In order to investigate convergence, the decision maker might decline to accept a possible outranking option because she/he believes that this option does not increase the value function sufficiently. Thus, we can use the concept of a value function in order to analyze the convergence of outranking trials; we must only assume that there is a finite indifference threshold for a meaningful increase of the value function. Because the decision maker decides exclusively on all parameters and details of the procedure, she/he can act consistently with the value function and with the indifference threshold. The decision maker can do this by defining sufficiently small outranking thresholds to produce all options suspected of outranking and accepting only those from such options that result in a meaningful increase of the value function, above the indifference threshold. Thus, the condition of consistence of an outranking with a value function is defined as follows: outranking relations defined by (9.32) are consistent with a continuous, monotone value function $v(\boldsymbol{q})$ if there is a finite, positive indifference threshold Δv such that, if $v(\boldsymbol{q}) - v(\hat{\boldsymbol{q}}^{(l)}) \geq \Delta v$, then $\boldsymbol{q}$ outranks $\hat{\boldsymbol{q}}^{(l)}$ at least in the weak sense.

Assuming outranking-value consistence, the proof of the convergence of outranking trials is immediate. If the set Q_0 is compact, the continuous value function can only finitely increase in this set. Because each iteration in which we find an outranking point increases this function at least by Δv, the procedure must stop after a finite number of iterations at a final point where no outranking point can be found. Earlier it was shown that if no such point can be found, then no attainable outranking point exists, due to the basic property of the order-consistent achievement function used for seeking such points. Because of outranking-value consistence, no point exists that would increase the value function more than Δv when compared to the final point of the procedure.

The use of a value function and an outranking-value consistence property is necessary to investigate the convergence in classical terms; without such assumptions, the convergence can be deduced in the more general terms of outranking relations. The concept of convergence of outranking trials might be only local, because the use of veto thresholds implies a refusal to compare decisions with widely different objective components. However, if the assumed value function is concave and the set Q_0 is convex, local convergence is equivalent to the global one.

While it remains to be tested practically whether the outranking trials procedure is acceptable for a typical decision maker, the procedure provides an alternative theoretical approach to the issue of convergence of some interactive procedures of multi-objective optimization and decision support. For example, the convergence of the contracted cone or light beam search procedures could possibly be investigated using similar concepts applied to outranking trials.

Chapter 10

Interfaces

Marek Makowski and Janusz Granat

This chapter discusses the role and development of user interfaces in decision support. The role is multiple and involves the following:

- Organizing the decision support system (DSS) as well as helping the user to organize the decision process.
- Conveying the commands of the user to the DSS and providing for other forms of interaction.
- Displaying to the user the results of analysis prepared by the DSS.

The development of user interfaces has resulted in the use of interactive graphics and, in a sense, multi-media tools for constructing modern, user-friendly DSS interfaces. The role of user interfaces in organizing the DSS and in helping in the organization of the decision process is discussed in Section 10.1. An overview of approaches to graphical user interfaces in DSSs is presented in Section 10.2. Section 10.3 concentrates on the current trend of using interactive graphics for DSSs and for multi-criteria model analysis. Section 10.4 comments on the use of interactive graphics for a reference point approach based on aspirations and reservations, together with a brief presentation of a modular DSS interface implementing this approach. Section 10.5 presents another implementation of a DSS interface applied to the definition, edition, and multi-criteria analysis of nonlinear models. Finally, Section 10.6 concludes this chapter.

10.1 User Interfaces in Decision Support

If a DSS should support learning and enhance the intuition of its user, and if intuition formation is deeply related to a gestalt-type processing of graphical images and other impressions (see Chapter 2), then the role of a modern, multi-media interface between the DSS and its user is essential. Such an interface has multiple functions and goals, the most important being:

- To organize parts of the decision process and to help the user in the organization of other parts.
- To convey either the commands or the information from the user to the DSS and to provide various other forms of interaction.
- To display to the user results of the DSS analysis in a way that helps the user best.

All this should be done in a clear and user-friendly way. These requirements have been discussed frequently in the literature (e.g., Jones, 1996); here we comment on the three functions outlined above.

It has long been recognized that the decision support limited to analyzing a simple alternative and preparing a choice between two options is not sufficient in computerized decision support (Hall and Dracup, 1970), and that decision support tools are needed for assisting in the development of alternatives. Thus, various prototypes of decision processes have been developed and used in DSSs. However, the development of more advanced and complicated decision processes results in the temptation to impose this more complex process on the user, and to make the user follow all stages of the process meticulously. Some decision processes – particularly the ones that focus more on automating decisions than on learning by the decision maker – lose consistency if all their stages are not carefully followed. Each decision process, even the most user friendly, has its own internal logic, implying which steps are really necessary. Thus the user interface has to organize the DSS (i.e., indicate to the user which steps of the decision process are necessary and in what sequence they should be executed), or to automatically execute those steps of the decision process that do not require the user's attention.

However, a good DSS cannot violate the sovereignty of its user too much (if it does, it will simply not be used). Common advice is that a DSS (as represented by its user interface) should be user friendly and, in particular, should: (i) help but not restrict the user; (ii) guide but not bound the user; (iii) contain no hidden assumptions about the user; and (iv) respond to various needs and expectations of the user.

Note that the last item implies that a good DSS should be problem specific and be developed in close consultation with its future user, since the needs and expectations of the user are different for different applications. Each user has her/his own habitual domain (Yu, 1985), therefore the DSS should help in broadening the domain, but should not be inconsistent with it, otherwise its advice will be useless. It can be observed that various professions and their specializations have their own language and culture, consisting of special "jargon" of verbal expressions, their own customs of graphical representations, even their own preferences in using certain types of models and analytical approaches. Ideally, a user interface in a DSS should be adapted to the language and culture of its user;[1] however, this is costly and a compromise between the value of cultural diversity and the efficiency of standardized solutions must be made.

The second function of a user interface – to convey either the commands of, or the information from, the user to the DSS – is closely related to the first, organizing function, and to the third, displaying function. Commands of the user – their character and form – depend strongly on the decision process adopted in the DSS and its organization. For example, throughout most of this book we assume that such commands will have the form of specifying reference (aspiration and reservation) levels for criteria; interfaces based on this assumption are described in more detail in this chapter. However, there might also be other forms of communication between the user and the DSS: selecting alternative modes of analysis, and performing or skipping certain phases of the decision process. In DSSs concentrated on automating decisions, the commands of the user are limited; instead, the DSS requires information from the user. Again, the form of providing such information is also important.

The third, displaying function of an interface might be combined – especially in modern interfaces using interactive graphics – with the command and information function; later we show how this can be done. However, the displaying function is also important on its own. The issue of how to combine text and tables of numeric results, hypertext, graphics and animation, multi-media, and virtual reality tricks in order to best support the decision maker is widely discussed in the literature (Cleveland and McGill, 1984; Jones, 1996). These methods of display should result in a cognitive fit (Vessey and Galletta, 1991).

Decision support for a nontrivial problem typically requires the analysis of more than one model (e.g., see Grigg, 1996, for a comprehensive discussion of water management problems). A user interface should therefore allow for an integrated analysis of several models. Moreover, each model typically has at least three distinct representations: (i) a "natural" representation suitable for communication with people (often managers) without specific training in modeling; (ii)

[1]Preserving cultural diversity – not only related to national and regional cultures, but also to those resulting from various professions – is an important goal of human civilization, especially when globalization and computerization trends promote cultural uniformization.

a mathematical representation suitable for analytical use; and (iii) a computer-executable representation (see Fourer, 1983). In practice, the number of model representations is higher because each of the three above-mentioned representations typically has a number of forms (e.g., there are several forms of the so-called standard formulation of a linear programming problem; a variety of "natural representations" of various types of models is much larger). Moreover, various methods of model analysis often require a specific model representation.

There are also certain relations between computerized modeling and visualization: we might use visual approaches for model formulation, especially in the early phases of conceptual model building, or various visual representations of modeling results. However, we limit the discussion of visualization to more specific issues of decision support.

Finally, the recent fast development of Web-based tools provides new dimensions for interfaces for decision support. For example, the recently developed DecisionNet by Bhargava *et al*. (1997) illustrates the capabilities of configurable Web-based systems that can be customized to meet specific modeling needs. DecisionNet adopts an electronic market-based approach to model-based decision support wherein various elements (such as data, models, solvers) are offered by various providers on demand.

10.2 Graphical User Interfaces

A user interface is one of the most important components of a DSS, decisive in determining the effectiveness of the interaction with the user. Advances in computer technology make it possible to use graphics for interaction and data visualization, which significantly improves this interaction. When using a graphical user interface, various functions can be performed:

- The selection of actions and options by using typical objects for interaction (menu, buttons, checkbox, etc.).
- The presentation of data in numerical form and also in various static graphical forms.
- More advanced functions when applying dynamic graphical interaction, combining the above and other functions.

The term graphical user interface (GUI) is most often used in the context of the first function. The second function relies on data visualization; and the third combines visualization with the additional features of direct manipulation of graphical objects and the selection of actions during such manipulation.

There is an essential difference between classical methods of computer graphics and graphics in DSSs. Classical methods of computer graphics are used mainly for data presentation and usually assume that extensive information exists but in a relatively simple form. Thus, the problem is how to present this information to the user and how to quicken the display of this information. For example, most of the current graphical data analysis belongs to the class of exploratory data analysis (EDA) methods (Tukey, 1977).

In DSSs based on analytical modeling, the data for the user are usually obtained as the result of complicated simulation or optimization runs. In such cases, the question is how to present data in order to provide the user with the acquired knowledge about the model or problem being solved as quickly and effectively as possible. The proper selection of the form of data presentation can significantly reduce not only the difficulty of the analysis, but also the calculation time.

The current state of user interfaces in DSSs is affected by research in graphical perception, data visualization, and modern graphical direct-manipulation interfaces. Lucas (1981) investigated the impact of graphical data presentation on the efficiency of decision making. His research does not conclusively state whether tabular or graphical data presentation is better. However, he suggests considering decision style when planning the form of data presentation. The tabular form might be sufficient for an analytical approach, but for a heuristic approach the graphical presentation of data is more useful. Additionally, Lucas stresses the importance of user involvement in designing the graphical interface.

DeSanctis (1984) carried out extensive research comparing various forms of data presentation, and suggested that the issue of efficacy of data presentation is problem specific. Jarvenpaa and Dickson (1988) provide an initial set of guidelines for an effective use of graphics technology when supporting business decision making. They suggest using graphs for a quick summary of data, detecting trends over time, comparing points and patterns of different variables, and information forecasting – particularly if a vast amount of information is presented and relatively simple impressions are to be drawn. Tables should be used for reading individual data values. Further, these authors suggest what forms of graphical presentation are best adapted to a given task. Their suggestions are as follows:

- To use line or bar charts for summarizing data; line graphs are preferred over bars if the speed of comprehension is important.
- To group line or bar charts for showing trends over time.
- To group bar charts for presenting parts of the whole; they suggest not to use pie charts or segmented bar charts for this purpose.
- To group line or bar charts for comparing patterns of variables; they suggest not to use segmented line or bar charts in this case.

- To use horizontal rather then vertical bars for making comparisons between variables.
- To place data values on the top of bars for point reading.

Tan and Benbasat (1993) carried out three laboratory experiments to assess the relative strengths and weakness of bar, symbol, and line graphs for performing a variety of elementary information extraction tasks using two dependent variables: time and accuracy.

The discussion so far has related to a static form of graphical presentation. The use of interactive graphics is more efficient for data analysis. Eick and Wills (1995) describe the core features of interactive graphics, investigate how familiar plots can be made interactive, and show examples of interactive graphics for general and specific data analysis. They define interactive graphics as the class of techniques for exploring data that allow the user to manipulate graphical representation of data. They stress the following advantages of interactivity:

- Clarity (e.g., the user can control what is seen by manipulating graphical objects; information is presented only on demand).
- Robustness (e.g., the user can examine diverse ways of viewing the data).
- Power (e.g., combining the views and impact of the value of one view on another).

10.3 Interactive Graphics in Multi-Criteria Model Analysis

Multi-criteria model analysis (MCMA) for decision support requires an enhancement of graphical methods that are typically used for user interface and data analysis. An interaction screen designed for a special purpose helps in understanding the decision problem being analyzed. In this section we outline several graphical methods used in multi-criteria model analysis, which are summarized in *Table 10.1*.

In the method proposed by Korhonen and Laakso (1986), a linear graph is used not only for data presentation but also to support the selection of the best solution from solutions obtained by projecting the reference direction on the Pareto surface. It is important to stress that graphical interaction is the essential part of the proposed algorithm, not just an alternative form of data presentation. This method was further developed in the Visual Interactive Goal programming (VIG) system (Korhonen and Laakso, 1984). In this system, the user chooses the reference direction and can dynamically observe the values of criteria on a bar graph. This method is called Pareto Race.

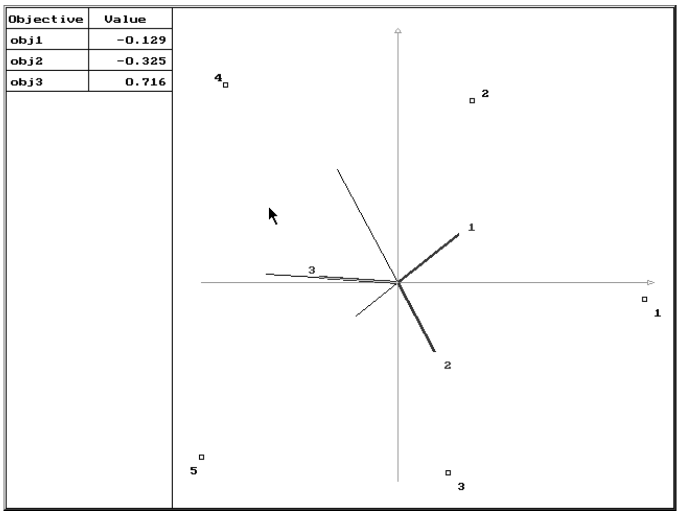
Table 10.1. Graphical methods and authors used in MCMA.

Methods	Authors
VIG	Korhonen and Laakso (1984)
VIMDA	Korhonen (1986)
TRIMAP	Clímaco (1987)
VISA	Belton and Vickers (1988)
VICO	Korhonen (1991)
PCA	Mareschal and Brans (1988)
Dynamic BIPLOT	Lewandowski and Granat (1991)
A descriptive approach	Ng (1991)
MCView	Vetschera (1989)
P-P-F	Granat (1993)
Cone Contraction	Jaszkiewicz and Słowiński (1992a)
Light Beam Search	Jaszkiewicz and Słowiński (1992b)
GO-2	Hemming and Troutt (1992)
TOMMIX	Clímaco and Antunes (1992)
MC-VISA	Choi and Kim (1992)
VIDEA	Belton and Vickers (1992)
Expert Choice	Exp (1995)
ISAAP	Granat and Makowski (1998)

The TRIMAP system analyzes three-criteria linear problems and is oriented toward progressive and selective learning about the Pareto-optimal set. Two graphs are presented to the user. The first one displays the space of weighting coefficients and shows the indifference regions (for these coefficients) corresponding to each Pareto-optimal solution already computed. The second one displays orthogonal projections of Pareto-optimal solutions on a plane. This approach was also applied in the TOMMIX system (Clímaco and Antunes, 1992). However, the latter system also includes alternative methods for solving multi-criteria optimization problems: STEM, Zionts-Wallenius, interval criterion weights, Pareto Race.

The concept of the visual interactive modeling was applied in the visual interactive modeling (VIM/VISA) system (see Angehrn and Lüthi, 1990; Bodily, 1991; Liu and Hou, 1992). The VIM system allows a visual interactive model to be built and used, and consists of three essential components: a mathematical model, graphical presentation of the status of the model, and methods for changing the status of the model. The VISA system uses a multi-attribute utility function approach for analyzing the decision problem.

The vectors, which are presented to the user in multi-criteria optimization, are multi-dimensional. There are various approaches to the visualization of multi-dimensional data; none of them is ideal. The selection of the form of presentation is problem specific. One possible approach is to parameterize a displayed



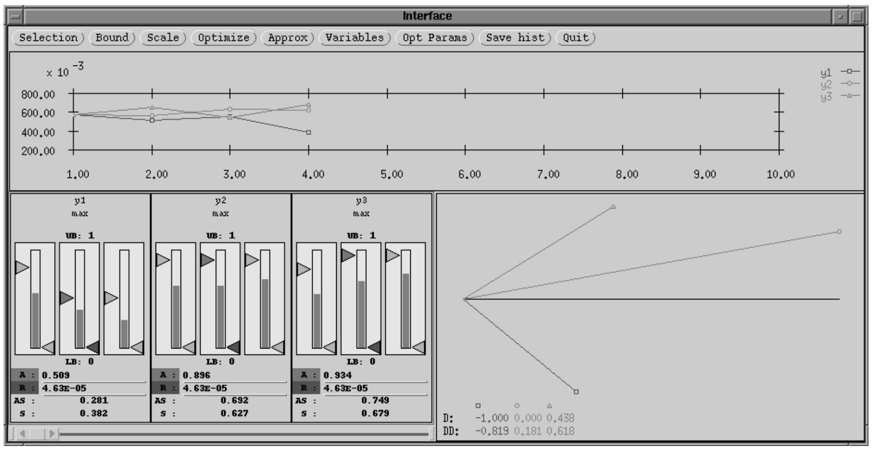
Objective	Value
obj1	-0.129
obj2	-0.325
obj3	0.716

Figure 10.1. An example of a dynamic BIPLOT screen.

figure by coordinates of the visualized vector. This approach was applied in the VICO system (Korhonen, 1991). The method was called "Harmonious Houses", since houses – either harmonious or distorted – were chosen to represent multi-dimensional vectors. There are also figures other than houses (e.g., faces, stars) that can be parameterized for this purpose.

Statistical data analysis can also be used to help in multi-dimensional vector representation. The crucial idea in this class of methods is to project multi-dimensional points on a two-dimensional plane. Mareschal and Brans (1988) applied Principal Component Analysis (PCA) for simultaneous presentation of the criteria and alternatives on the dimensional plane. Criteria are represented by arrows projected on the principal component (PC) plane beginning in the origin of the coordinate system and alternatives are displayed as points on the same plane. Lewandowski and Granat (1991) presented a similar technique for building a graphical interactive user interface. This technique uses the BIPLOT method for static graphical data analysis, which was proposed by Gabriel (1971) and extended to dynamic graphical interaction. An $n \times m$ matrix Y of data is considered, representing an assembly of multi-dimensional vectors. In multi-criteria analyses, this matrix might represent a selected set of Pareto-optimal solutions with columns corresponding to objectives and rows corresponding to solution options. Especially important for using BIPLOT as an interactive interface are the following properties:

- Elements of the data matrix can be approximated as orthogonal projections of vectors on principal component directions.



y1 max UB: 1 LB: 0 A : 0.509 R : 4.63E-05 AS : 0.281 S : 0.382

y2 max UB: 1 LB: 0 A : 0.896 R : 4.63E-05 AS : 0.692 S : 0.627

y3 max UB: 1 LB: 0 A : 0.934 R : 4.63E-05 AS : 0.749 S : 0.679

```
     □      ○     △
D:  -1.000 0.000 0.438
DD: -0.819 0.181 0.618
```

Figure 10.2. An example of a PPF interface.

- Correlation coefficients between variables are approximated by the angle between vectors.
- The variance of a variable is approximated by the length of a corresponding vector h.

An example of the BIPLOT screen is shown in *Figure 10.1.* Criteria are represented by thick lines projected on a two-dimensional plane[2] and beginning at the origin of the coordinate system; alternatives are displayed as points on the same plane. The numerical values of objectives are displayed on the left side of the screen. The user can set a new reference point by moving the cursor and clicking the mouse button. The thin lines show the approximated values of objectives.

Vetschera (1989) proposed further modifications of the selection of a projecting plane, since important preference information could be lost in the previous approaches. Vetschera assumed that preference information is represented by a linear utility function and the projection matrix is built as the combination of a standard PCA projection matrix and a preference preserving projection matrix. Ng (1991) adapted statistical techniques for graphical analysis of trade-offs between objectives.

Most interfaces concentrate on the current state of the interaction process. Granat (1993) proposed extending the information presented to the user by a graph representing the history of the process of interaction as well as a graph predicting the changes of the solution after changing reference points. This interface is called Past-Present-Future (PPF; see *Figure 10.2*). The graphic user interface consists of three windows.

[2]The method of choosing the plane is described in Lewandowski and Granat (1991).

The first window, called Past, shows the history of the process of interaction. The values of objectives are presented for each iteration. The second window, called Present, shows three bars for each objective. The first bar presents the last solution for the last aspiration level, and the second bar presents a new aspiration level, which can be changed dynamically on the screen during the process of interaction, also using an approximation of the solution based on directional derivatives. When the aspiration levels marked on the first and second bars are compared, the user obtains information about the direction of changes of the aspiration level. The third bar shows the new efficient solution, which is obtained as a result of a new optimization run. Additionally, the numerical values of aspiration levels (A), approximation of the solution (AS), and the solution (S) are presented to the user. The third window, called Future, presents the prediction of changes of objectives when the aspiration point is changed in the specified direction based on values of directional derivatives. Additionally, the numerical components of the direction vector (D) and the directional derivatives (DD) are presented to the user. The prediction of solution might be useful when the decision problem being analyzed is represented by large models. In such cases and without additional help, the optimization time and consequently the system response time might be unacceptable for effective interaction with the user.

Another approach to interaction graphics is applied in the Cone Contraction method (Jaszkiewicz and Słowiński, 1992a) and the Light Beam Search method (Jaszkiewicz and Słowiński, 1992b). These methods use the outranking relation for modeling user preferences. The interaction procedure in the first method is composed of two stages. The first stage consists of modeling preferences by a fuzzy outranking relation. In the second stage, the user scans Pareto-optimal solutions by using interactive graphics. The samples of efficient points (called profiles) are presented to the user. The profiles are presented in a three-dimensional bar form or in the trajectory form. The second method uses the reference point method for finding the so-called middle point and then its neighborhood is presented to the user. The outranking relation is used as a local preference model in the neighborhood of the middle point.

Two other systems, GO-2 (Hemming and Troutt, 1992) and VIDEA (Belton and Vickers, 1992), apply bar charts for interactive data modification. MC-VISA displays the values of the objectives obtained during directional search. Various well-designed graphical presentations are used in the commercial package Expert Choice (Exp, 1995), which uses an analytic hierarchy process for analyzing multicriteria problems. An animation of graphical presentations is used in this package as a tool for training the user of the system.

Further integration of interactive graphics for specification of user preferences is presented in more detail in the following section.

10.4 The Use of Aspirations and Reservations in Model Analysis

Earlier chapters discussed various approaches to multi-criteria model analysis for decision support, particularly the reference point approach. Here we recall some details of the aspiration/reservation-based reference point approach that are important for the interactive graphics user interface.

A critical step in multi-objective analysis is the generation of part of the Pareto-optimal solution set that is interesting for the user. Efficient or Pareto-optimal solutions are those where an improvement in the value of one criterion cannot be attained without worsening the value of at least one other criterion. Generating the entire Pareto set is practically impossible. Therefore, most multi-objective methods facilitate the generation of some selected set of Pareto solutions – in our case, implicitly defined by aspirations and reservations specified by the user. Additional tools for analyzing these solutions and for generating other subsets of Pareto-optimal solutions based on these results are also needed. Since aspirations are usually not attainable, the decision maker must learn, while using the mechanisms of the method, how to adjust them in order to find a feasible solution that best meets her/his expectations.

The reference point method is based on the concept of satisficing behavior (also called bounded rationality), in which the decision maker attempts to attain aspiration levels, usually by first trying to improve the criterion that shows the worst performance (March and Simon, 1958; Wierzbicki, 1982a). This method has a number of advantages over other multi-objective optimization methods, as discussed in detail in Chapters 4 and 5.

In order to explain the construction of the Interactive Specification and Analysis of Aspiration-based user Preferences (ISAAP) tool, the reference point methods could be based on the following steps:

1. The user or decision maker (DM) specifies a number of criteria (objectives). In typical applications there are 2–7 criteria.
2. The DM specifies an aspiration point $\bar{\boldsymbol{q}} = \{\bar{q}_1, \ldots, \bar{q}_k\}$, where $\bar{q}_i$ are aspiration levels (the desired values for each criterion) and k is the number of criteria. Additionally, the DM specifies a reservation point $\bar{\bar{\boldsymbol{q}}}$, which is composed of the worst values of criteria that a DM would like to consider.
3. The underlying formulation of the problem is the maximization of a (piece-wise linear) achievement function. This can be interpreted either as a value function of the DSS specified in response to the specific aspiration and reservation levels, or as an ad-hoc, nonstationary approximation of the value function of the decision maker, dependent on these levels. The problem is then transformed by

the DSS into an auxiliary parametric single-objective problem, the solution of which gives a Pareto-optimal point.[3]

4. The DM explores various Pareto-optimal points by changing the aspiration point $\bar{\boldsymbol{q}}$ and reservation point $\bar{\bar{\boldsymbol{q}}}$ for each criterion. Additionally, a DM may stabilize a criterion (i.e., specify a desired value instead of minimizing or maximizing the value of this criterion) or temporarily remove a criterion from the analysis. This results in the computation of a Pareto optimal point with respect to the remaining "active" criteria, but values of inactive criteria are still available for review.
5. The procedure described in points 2, 3, and 4 is repeated until a set of satisfactory solutions is found.

As shown by Korhonen and Laakso (1986), and also by Ogryczak and Lahoda (1992), the reference point approach may be considered as an extension of goal programming (Charnes and Cooper, 1967), which was the precursor of most multi-criteria optimization and model analysis methods. The reference point approach to multi-objective optimization can also be used for inverse simulation: instead of repeatedly adjusting the decision variables in order to determine acceptable states (expressed as constraints in the classical approach to optimization), the user chooses desired states (in terms of ranges of values of objectives) and the DSS determines for her/him the resulting values of the decision variables. The reference point approach also takes into account soft constraints often needed in the single-criterion optimization. Namely, one can replace a soft constraint (or group of constraints) by an objective, and then set the aspiration level equal to the desired value of the constraint and the reservation level to the worst acceptable value. Thus, violations of soft constraints can be treated as criteria (to be minimized) in the multi-objective approach.

Geometrical aspects of the reference point approach are shown in *Figure 10.3*, which illustrates a Pareto-optimal frontier (between points D and E) for a two-criteria minimization problem. Utopia and nadir points (denoted by U and N, respectively) are composed of the best and worst values of criteria in the Pareto frontier. For a given aspiration point A, various Pareto-optimal points can be obtained by the minimization of various scalarizing achievement functions.

[3]We refer to properly Pareto-optimal solutions with a prior bound on trade-off coefficients as Pareto solutions (unless otherwise mentioned). A Pareto-optimal point in objective space is composed of values of all criteria for a corresponding Pareto-optimal solution. If a specified aspiration level $\bar{q}$ is not attainable, then the Pareto-optimal point is the nearest (in the sense of a Chebyshev weighted norm) to the aspiration level. If the aspiration level is attainable, then the Pareto-optimal point is uniformly better than $\bar{q}$. Properties of the Pareto-optimal point depend on the localization of the reference point (aspiration and reservation levels) associated with the criteria.

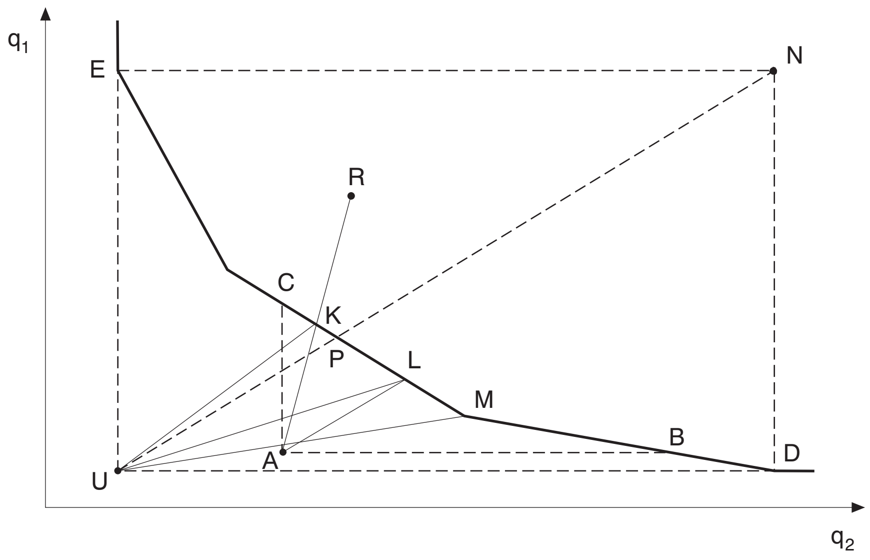


Figure 10.3. Illustration of a Pareto-optimal surface for a two criteria case.

An achievement function of the form discussed in Chapters 4 and 8 is used here:

$$\sigma(\boldsymbol{q}, \bar{\boldsymbol{q}}, \bar{\bar{\boldsymbol{q}}}) = \min_{1\leq i\leq k} \sigma_i(q_i, \bar{q}_i, \bar{\bar{q}}_i) + \epsilon \sum_{i=1}^{k} \sigma_i(q_i, \bar{q}_i, \bar{\bar{q}}_i), \tag{10.1}$$

where $q, \bar{q}, \bar{\bar{q}}$ are vectors of values of criteria, aspiration levels, and reservation levels, respectively, and ϵ is a given small positive number. Maximization of the function (10.1) provides a properly Pareto-optimal solution with the trade-off coefficient smaller than $(1 + 1/\epsilon)$. Partial or component achievement functions $\sigma_i(\cdot)$ are strictly monotone (decreasing for minimized and increasing for maximized criteria, respectively) functions of the objective vector component q_i and have the same analytical form for the three types of criteria, i.e., minimized, maximized, and goal. The functions $\sigma_i(\cdot)$ have the following values for utopia, aspiration, reservation, and nadir points, respectively:

$$\sigma_i(q_i^U, \cdot) = 1 + \bar{\beta}, \quad \sigma_i(\bar{q}_i, \cdot) = 1, \quad \sigma_i(\bar{\bar{q}}_i, \cdot) = 0, \quad \sigma_i(q_i^N, \cdot) = -\bar{\eta} \tag{10.2}$$

where $\bar{\beta}$ and $\bar{\eta}$ are given positive parameters, typically equal to 0.1 and 10, respectively. However, in order to correctly handle aspiration and reservation values close to utopia and nadir values, respectively, these parameters should be dynamically adjusted (see Granat and Makowski, 1998, for details),

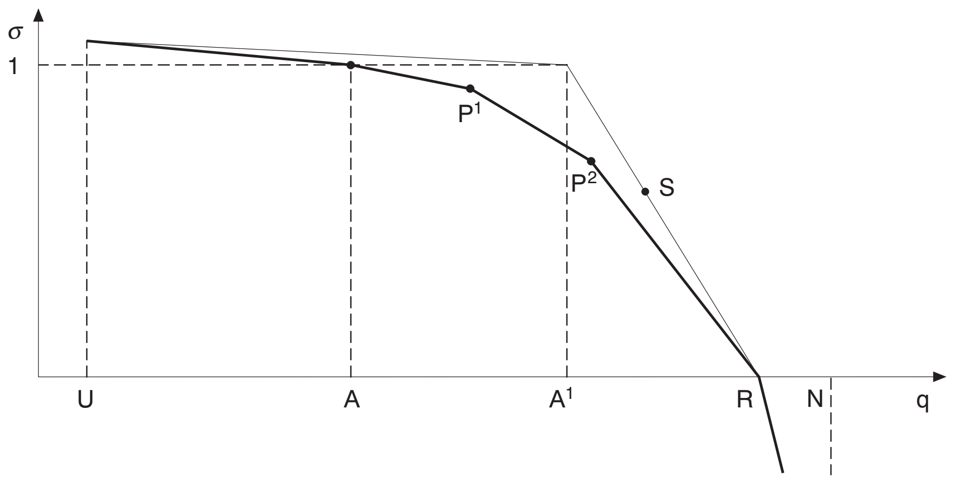


Figure 10.4. Component achievement scalarizing function for a minimized criterion.

As shown in *Figure 10.3*, using the achievement scalarizing function (10.1) and selecting the points A and R for the aspiration and reservation levels, respectively, would result in a Pareto-optimal solution denoted by K. With the aspiration level A, the application of different scalarizing functions (e.g., specified by using different reservation points) can result in any Pareto solution between the points B and C. For example, a scalarizing function that uses weighting coefficients corresponding to the difference between the utopia and nadir points would result in the point L, while another scalarizing function using only the utopia and the aspiration point for the definition of weighting coefficients would result in the point M. However, using the utopia and the nadir point for the automatic calculation of weighting coefficients has a number of drawbacks (Makowski, 1994c), hence we have implemented an approach based on aspiration and reservation levels. Finally, the neutral compromise solution, found when using aspirations and reservations on the utopia–nadir line, is denoted by the point P.

The component achievement functions (CAF) σ_i used in the scalarizing achievement function (10.2) have a more general form than that presented in Chapter 5, namely, they are piece-wise linear functions with possibly many segments (see *Figure 10.4*). Such functions allow either for specification of aspiration and reservations levels only or for additional specification of preferences (for the criteria values between aspiration and reservation levels) in terms of fuzzy sets, as described by Granat and Makowski (1998). Therefore, the piece-wise linear functions σ_i are defined by segments σ_{ji}:

$$\sigma_{ji} = \alpha_{ji} q_i + \beta_{ji}, \qquad q_{ji} \leq q_i \leq q_{j+1,i} \qquad j = 1, \ldots, p_i, \tag{10.3}$$

where p_i is the number of segments for the i-th criterion. Practical applications show that sometimes it is useful to set $\bar{q}_i = q_i^U$ and/or $\bar{\bar{q}}_i = q_i^N$. Therefore, in order to also handle CAFs composed of only one segment (in cases when an aspiration level is set to the utopia value and a reservation level is equal to an approximation of the nadir), the number of segments p_i has to be greater or equal to one.

The coefficients defining the segments are given by:

$$\alpha_{ji} = \frac{\sigma_{j+1,i} - \sigma_{ji}}{q_{j+1,i} - q_{ji}} \tag{10.4}$$

$$\beta_{ji} = \sigma_{ji} - \alpha_{ji} q_{ji} \tag{10.5}$$

where points (σ_{ji}, q_{ij}) are interactively defined with the help of the software described in Section 10.4. Concavity of the piece-wise linear functions $\sigma_i(q_i)$ defined by segments (10.3) can be assured by the condition:

$$\alpha_{1i} > \alpha_{2i} > \ldots > \alpha_{p_i i}. \tag{10.6}$$

Note that the CAF σ_i defined by (10.3) take the same form for minimized and maximized criteria. However, one should add (in addition to the condition (10.6) that assures concavity) a condition:

$$\alpha_{1i} < 0 \qquad i \in I^{\min} \tag{10.7}$$

$$\alpha_{p_i i} > 0 \qquad i \in I^{\max} \tag{10.8}$$

where $I^{\min}$ and $I^{\max}$ are sets of indices of criteria minimized and maximized, respectively. The conditions (10.7) and (10.8) are fulfilled automatically for the CAF σ_i specified with the help of ISAAP.

A goal-type criterion can be used when a distance from a given target value (which can be changed during the interaction) is to be minimized. For this type of criterion a CAF is composed of two parts: the first part is defined for the criterion values smaller than the target value, and the second part for the criterion values larger than the given target. This function is illustrated in *Figure 10.5*. The conditions specified for maximized and minimized criteria hold for the first and second function, respectively. There is only one point i, for which $\alpha_{i-1,i} > 0$ and $\alpha_{i,i+1} < 0$. The criterion value for this point corresponds to a target value (denoted by T) for the goal-type criterion. The function shown in *Figure 10.5* is symmetric, but for many applications an asymmetric function is appropriate and therefore both types of functions for the goal type of criteria are supported by ISAAP. *Figure 10.10* (on page 302) illustrates an asymmetric CAF.

The methodological details described previously have been implemented in the software package MCMA, with a particular module called ISAAP that specifies

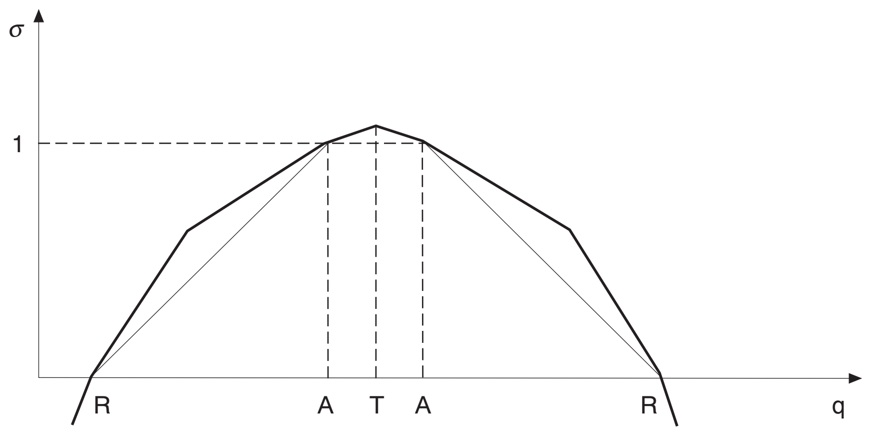


Figure 10.5. Component achievement function for a goal type (or stabilized) criterion.

the achievement functions through an interactive graphical interaction based on aspiration and reservation levels. The software package MCMA, together with ISAAP, is included in the software available from IIASA (refer to the Appendix for details). Examples of its application are discussed in more detail in Chapters 11 and 12.

Multi-criteria model analysis is composed of three stages. First, a core model is specified and generated. The core model contains only constraints that correspond to logical and physical relations between the variables used in the model. Those variables should also include variables that represent potential criteria (goals, objectives, performance indices). Second, in the preparatory stage of the MCMA, a user selects (from the core model variables) a set of criteria that will be used for the analysis of the model, and specifies a type for each criterion. The selected type declares that a criterion is either minimized or maximized or targeted at a given value.[4] After the selection of a set of criteria, MCMA automatically performs a series of optimizations in order to compute the utopia point and an approximation of the nadir point.[5] The preparatory stage is finished with computation of a neutral

[4]Note that a variable can also represent more complicated forms of criteria (e.g., following a trajectory, minimization of a distance). Examples of various types of criteria (which are formally represented by a variable, whose value is either minimized or maximized) and the way to handle so-called soft constraints can be found in Makowski (1994c).

[5]The utopia and nadir points (in the space of criteria) are vectors composed of the best and worst values of the criteria in the efficient set. As indicated in earlier chapters, computation of a nadir point for problems with more than two criteria may be very difficult. The nadir point plays a minor albeit informative role in the approach presented here and there is no justification for spending resources in order to get a better approximation. Hence, as an approximation of nadir, we assume the worst value of a corresponding criterion obtained during the analysis.

compromise solution for which the aspiration and reservation levels are (automatically) set to the utopia and to an approximation of the nadir points, respectively. The third stage is an interactive procedure used for helping the user in selecting an efficient solution that best corresponds to her/his preferences. During such a procedure, a user specifies goals and preferences, including values of criteria to be achieved and to be avoided. The vectors composed of those values are called aspiration and reservation levels, respectively. Such a specification defines CAFs that are used for selection of a Pareto-optimal solution. The solution is achieved by generation of additional constraints and variables, which are added by MCMA to the core model thus forming a parametric optimization problem. The solution of this problem is a Pareto-optimal solution that is nearest (in the sense of a measure defined by the aspiration and reservation levels) to the specified aspiration levels (or uniformly better than these levels, if they are attainable).

The first stage is problem specific and a core model should be generated either by using a problem-specific generator (refer to Chapters 11 and 12) or a general purpose modeling tool, such as AMPL (Fourer *et al*, 1996) or GAMS (Brooke *et al*., 1992). The MCMA software can be used for the second and third stage of the analysis of a core model that conforms to specific rules (see Chapter 6).

In order to demonstrate capabilities of MCMA with a real-world example, the basic functions of MCMA are presented in Chapter 12 using the land use planning problem. In particular, Section 12.5.2 provides a detailed illustration of criteria definition and a simple way of specifying preferences by selecting only aspiration and/or reservation levels based on an analysis of previously obtained Pareto solutions (a reader who is not familiar with aspiration/reservation-led model analysis may want to read the tutorial example provided in Section 12.5.2 before continuing with this section).

The third stage of the analysis is supported within MCMA by the modular tool ISAAP. In the following, we provide an overview of some more advanced functions of ISAAP focusing on the specification of piece-wise linear CAFs (10.3) and on other features of ISAAP that are essential for the analysis of non-toy problems.

There are three more advanced possibilities of specifying preferences by using CAFs (10.3) (as opposed to just moving aspiration and/or reservation points by clicking a mouse, illustrated in Section 12.5.2); these possibilities are:

- To change the shape of the CAF.
- To provide exact value specifications of aspiration and reservation levels and targets (the latter for stabilized criteria).
- To add/remove extra points that define additional segments of the CAF.

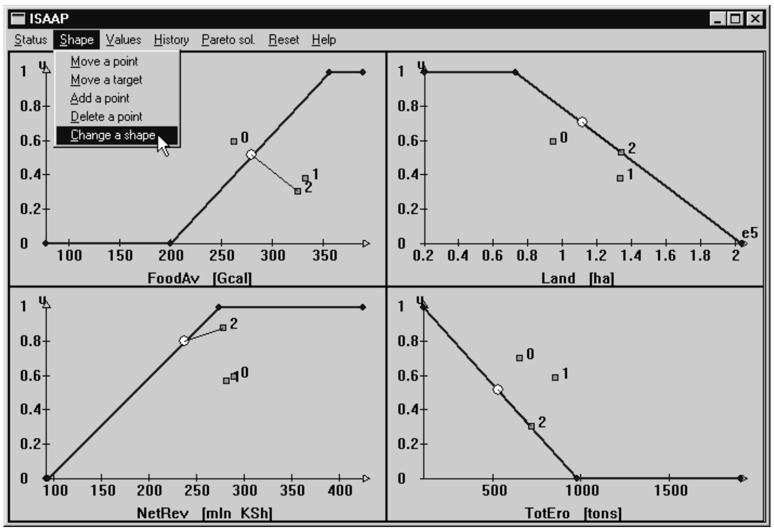
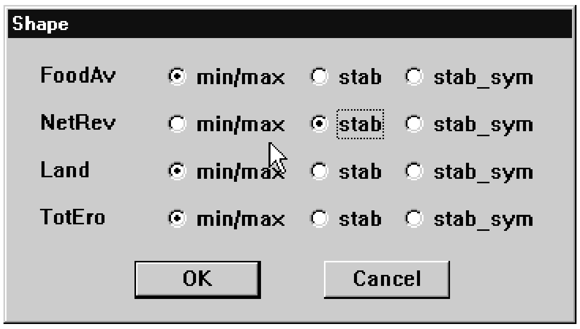


Figure 10.6. The starting stage of analysis for more advanced functions of ISAAP (e.g., with the selection of a menu changing the shape of a function).



Shape

FoodAv	[✓ min/max]	stab	stab_sym
NetRev	min/max	[✓ stab]	stab_sym
Land	[✓ min/max]	stab	stab_sym
TotEro	[✓ min/max]	stab	stab_sym

OK Cancel

Figure 10.7. Dialog for changing the shape of a function.

The shape of a CAF can be altered by selecting the `Change a shape` item from the `Shape` menu (as shown in *Figure 10.6*, which presents the starting stage of analysis for more advanced functions of ISAAP). This then activates the dialog illustrated in *Figure 10.7*. In this dialog, the `NetRev` criterion has been stabilized. The stabilized criterion has a shape illustrated in *Figure 10.8* (this can also be changed by clicking the mouse). The dialog presented in *Figure 10.7* provides a selection of two types of a stabilized criterion: `stab` and `stab_sym`, which differ by the way in which the shape of the corresponding function can be modified. The `stab_sym` is used in cases where a symmetric CAF function is desired. In this instance, which is a typical one, modifications of the function are done in such a way

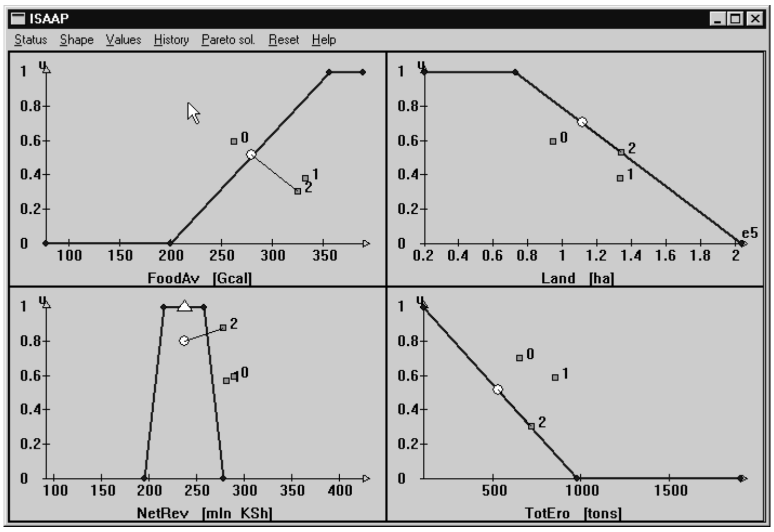
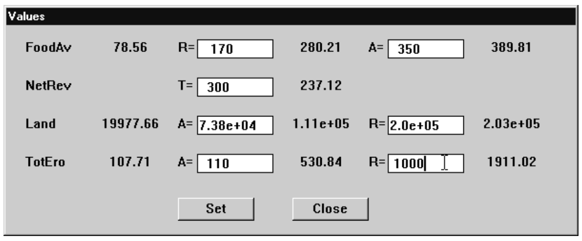


Figure 10.8. ISAAP window after changing the shape of a function.



FoodAv	78.56	R= 170	280.21	A= 350	389.81
NetRev		T= 300	237.12		
Land	19977.66	A= 7.38e+04	1.11e+05	R= 2.0e+05	2.03e+05
TotEro	107.71	A= 110	530.84	R= 1000	1911.02

Figure 10.9. Dialog for setting values from the keyboard after modification of the target, aspiration, and reservation values.

that symmetry is assured. However, for some criteria a nonsymmetric shape may be appropriate. This can be achieved by using the `stab` type.

The specification of the aspiration and reservation levels with the aid of the mouse is useful, but it does not set precise values. Usually this is acceptable; however, the user can set precisely the values of aspiration and reservation levels (and target values for stabilized criteria) by using the `Values` option. This dialog is illustrated in *Figure 10.9*, which contains values specified from the keyboard. The dialog allows for setting (for each criterion) either a pair of reservation and aspiration values, or a target value for a stabilized criterion. This dialog contains static text for a minimized or maximized criterion with values of utopia and approximation of nadir point components (denoted by U and N, respectively). For any type of criterion the current value (denoted by V) is displayed as static text. The values are set (and the plots are updated accordingly) after clicking on the `Set` button.

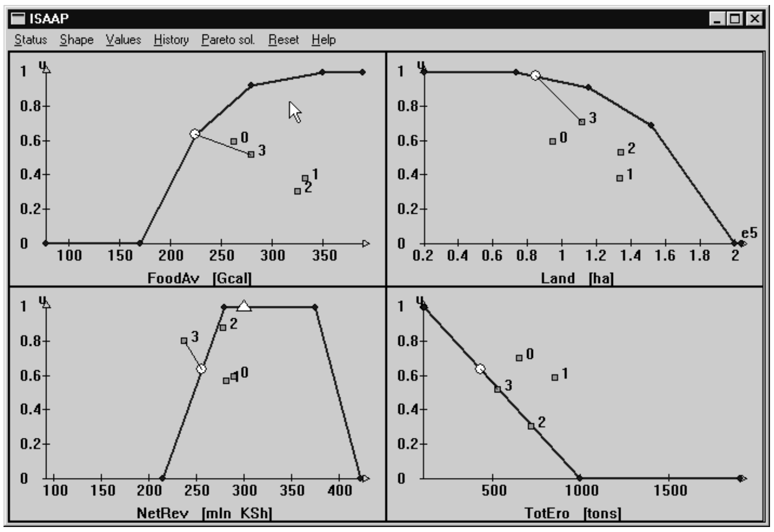
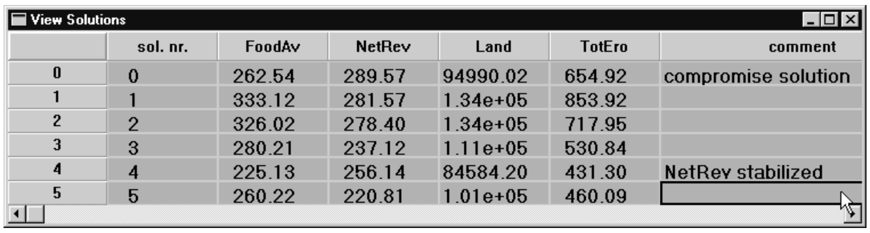


Figure 10.10. Illustration of CAF defined more precisely with additional segments for the criteria `FoodAv` and `Land`, and for the stabilized (nonsymmetric) criterion `NetRev`.

New values of aspiration and reservation levels for any criteria can be set by clicking the mouse near a point that one wants to move. However, more advanced users may want to specify (in addition to the pairs of aspiration and reservation levels) the piece-wise linear CAF. An example of such a function is shown in *Figure 10.10*. One can add more segments to any CAF by selecting the `Add a point` item from the `Shape` menu. After selecting this option, the background of the ISAAP window changes from the default gray color to green and it will remain green until one of the following options from this menu is selected. (Other options also change the background color to alert a user that the default action of the mouse should be restored by selecting the `Move a point` item.) In this mode (indicated by the green background) each click of the mouse results in adding one point to the CAF (at the current location of the mouse pointer). Once enough points have been added, `Move a point` should be selected from the `Shape` menu. This results in switching back to the default mode of MCMA (where a click of the mouse moves the nearest point to the place currently pointed at by the mouse). Additional point(s) no longer needed for the definition of the piece-wise linear functions can be removed by (temporarily) switching the mode by selecting `Delete a point` from the `Shape` menu. In this mode (indicated by the red background) each click of the mouse removes one point (closest to the current location of the mouse pointer) of the CAF.

After several iterations, the plots of the achievement functions and the points that mark previous solutions may resemble those in *Figure 10.10*. The circles mark



	sol. nr.	FoodAv	NetRev	Land	TotEro	comment
0	0	262.54	289.57	94990.02	654.92	compromise solution
1	1	333.12	281.57	1.34e+05	853.92	
2	2	326.02	278.40	1.34e+05	717.95	
3	3	280.21	237.12	1.11e+05	530.84	
4	4	225.13	256.14	84584.20	431.30	NetRev stabilized
5	5	260.22	220.81	1.01e+05	460.09	

Figure 10.11. History of solutions in the form of a spreadsheet.

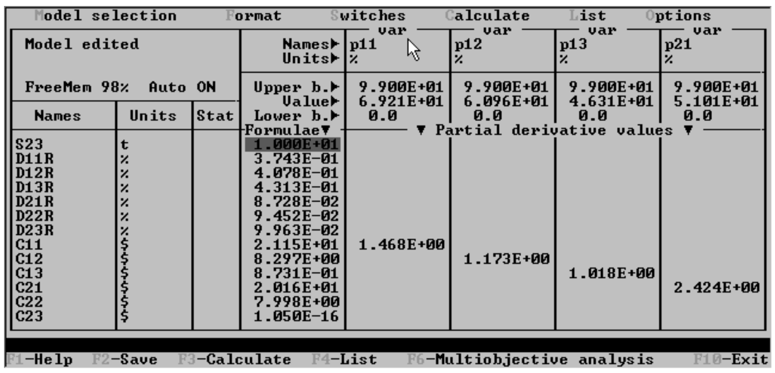
the values of criteria corresponding to the new solution and the thin lines connect the criteria value obtained in the previous and last solutions. The small squares represent the previously calculated solutions. The history of the interactions is provided both graphically and in the form of a spreadsheet. Complete information about the previous optimization run is available in two forms of spreadsheets that can be displayed by choosing the `History` item from the main menu. The first spreadsheet contains triples for each criterion composed of the criterion value and the values of aspiration and reservation levels. This information (available by the `View ASR` from the `History` menu) usually does not fit completely on a screen and therefore the spreadsheet has to be horizontally scrolled. Therefore, another form (composed only of criteria values, see *Figure 10.11*) is available via the `View solutions` item from the `History` menu of MCMA.

The last field of the history allows short comments to be attached to selected solutions.

In concluding this overview of ISAAP and MCMA, we mention several other implemented functions that are essential for the analysis of real problems:

- The continuation of analysis that was broken in order to provide enough time for a more detailed analysis of the already obtained solutions, discussions, reflections, etc.
- Storing the full solution for more detailed analysis, which is provided by a software that produces a problem-specific report (see Chapter 12).
- Rearranging and changing labels of displayed solutions (this is necessary, when the number of solutions obtained is too large for them all to be displayed).
- Changing the status of selected criteria (usually done at some stage in the analysis of problems with a larger number of criteria).

All functions of MCMA are documented and illustrated by a detailed tutorial example in Granat and Makowski (1998). This example is also available online with the MCMA software (see the Appendix for information about software availability).



```
 Model selection     Format     Switches     Calculate     List    Options
                                      var          var          var          var
 Model edited              Names▸ p11          p12          p13          p21
                           Units▸ %            %            %            %

 FreeMem 98%  Auto ON    Upper b.▸ 9.900E+01   9.900E+01   9.900E+01   9.900E+01
                            Value▸ 6.921E+01   6.096E+01   4.631E+01   5.101E+01
 Names      Units  Stat Lower b.▸  0.0         0.0         0.0         0.0
                        Formulae▼        ▼ Partial derivative values ▼
 S23        t            1.000E+01
 D11R       %            3.743E-01
 D12R       %            4.078E-01
 D13R       %            4.313E-01
 D21R       %            8.728E-02
 D22R       %            9.452E-02
 D23R       %            9.963E-02
 C11        $            2.115E+01  1.468E+00
 C12        $            8.297E+00              1.173E+00
 C13        $            8.731E-01                          1.018E+00
 C21        $            2.016E+01                                      2.424E+00
 C22        $            7.998E+00
 C23        $            1.050E-16

F1-Help  F2-Save  F3-Calculate  F4-List  F6-Multiobjective analysis   F10-Exit
```

Figure 10.12. A model editing spreadsheet: definition and editing of nonlinear models.

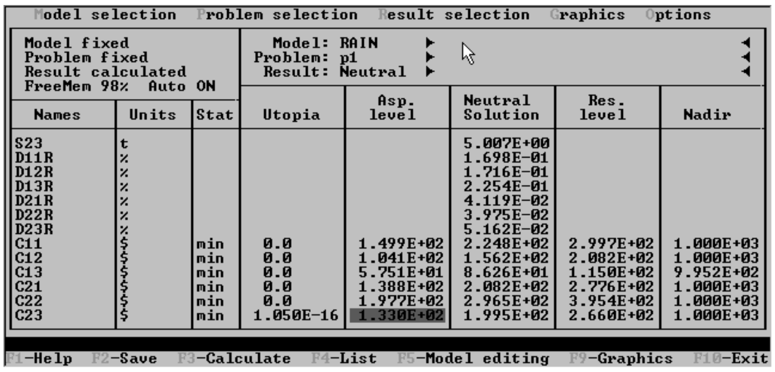
10.5 An Interface Supporting Multi-Criteria Nonlinear Model Analysis

This section presents another user interface, with only limited graphics, applied in the IAC-DIDAS-N system (Dynamic Interactive Decision Analysis and Support System for Multi-Criteria Analysis of Nonlinear Models); see Kręglewski *et al.* (1991). The DIDAS-N system supports the definition and edition of substantive models as well as multi-objective problem definition and analysis in an easy but flexible standard format of a spreadsheet; in addition, the system also contains a nonlinear solver.

Two specialized spreadsheets are used in DIDAS-N. The first one, a model editing spreadsheet (see *Figure 10.12*), is used for model definition and the second one, an interactive analysis spreadsheet (see *Figure 10.13*), is used for the problem definition and interactive scanning of the efficient frontier.

The spreadsheets used in the IAC-DIDAS-N system are specialized. They differ from the standard ones (such as Lotus or Excel) in two respects. First, there are predefined types of contents of all cells in the DIDAS-N spreadsheets: there are cells dedicated for storing text, other cells for storing numbers, and others for storing formulae. Second, IAC-DIDAS-N has an integrated compiler with a symbolic differentiation facility, which compiles the formulae and produces binary codes for calculations of formula values and all derivatives. Therefore, two kinds of procedures are defined for spreadsheet cells with formulae: compilation and calculation. IAC-DIDAS-N uses a compiler that directly generates a binary code for a numeric coprocessor; calculations of formulae and derivative values are more than ten times faster than in a previous version of the system that used an interpreted internal code.

The top screen line in both spreadsheets contains pull-down menu entries and the bottom line contains the meanings of the function keys. Both spreadsheets are



```
Model selection  Problem selection  Result selection  Graphics  Options

Model fixed               Model: RAIN
Problem fixed           Problem: p1
Result calculated        Result: Neutral
FreeMem 98%  Auto ON
```

Names	Units	Stat	Utopia	Asp. level	Neutral Solution	Res. level	Nadir
S23	t				5.007E+00		
D11R	%				1.698E-01		
D12R	%				1.716E-01		
D13R	%				2.254E-01		
D21R	%				4.119E-02		
D22R	%				3.975E-02		
D23R	%				5.162E-02		
C11	$	min	0.0	1.499E+02	2.248E+02	2.997E+02	1.000E+03
C12	$	min	0.0	1.041E+02	1.562E+02	2.082E+02	1.000E+03
C13	$	min	0.0	5.751E+01	8.626E+01	1.150E+02	9.952E+02
C21	$	min	0.0	1.388E+02	2.082E+02	2.776E+02	1.000E+03
C22	$	min	0.0	1.977E+02	2.965E+02	3.954E+02	1.000E+03
C23	$	min	1.050E-16	1.330E+02	1.995E+02	2.660E+02	1.000E+03

```
F1-Help  F2-Save  F3-Calculate  F4-List  F5-Model editing  F9-Graphics  F10-Exit
```

Figure 10.13. An interactive analysis spreadsheet: interactive problem definition and scanning of efficient frontier.

built from two partially independent parts. The first three columns from the left are common for both spreadsheets and have as many rows as there are outcome variables in the model. These three columns are used to enter and display the definition of outcomes: name, unit, and status, from left to right, respectively. However, the status is the element of a problem definition. Therefore, it can only be accessed from the interactive analysis spreadsheet. The fourth column of the model editing spreadsheet contains outcome formulae.

The status of the model, problem, and result are displayed in the upper left corner of both spreadsheets, together with the amount of free memory in relation to the amount of free memory available after the system initialization. Another status value displayed in the upper left corner of the screen is the flag Auto `ON/OFF`, which controls the automatic spreadsheet recalculation after each change of any cell.

The second part of the model editing spreadsheet has as many columns as variables, parameters, and outcomes; a type designator (var, par, or out) is displayed at the top of each column, respectively. Each column contains the name, unit, upper bound, value, lower bound, and values of derivatives, from top to bottom, respectively. The first five items are elements of the model definition, except for values of variables that are accessible and can be changed in any phase of the work. Cells with values of outcomes are only for display purposes and their contents cannot be edited. The texts "Partial derivative values" or "Total derivative values" are alternatively displayed just above the area with derivative values. Values of either partial or total derivatives are then displayed in the spreadsheet.

The model debugger can be used if a formula for either an outcome variable or a derivative is displayed in the editor window.

The second part of the interactive analysis spreadsheet has seven columns. The contents of these seven columns depend on the type of outcome and can be different for different rows of the spreadsheet. Descriptions of the columns change according to the type of outcome that the spreadsheet marker is currently pointing to.

This interactive spreadsheet also allows for computation of efficient decisions and outcomes – interactively following various instructions of the user – and their presentation for analysis. For this purpose, IAC-DIDAS-N contains a specialized nonlinear solver. Following the experience with previous versions of nonlinear DIDAS-N systems, a robust nonlinear programming algorithm, based on shifted penalty functions and projected conjugate directions techniques, was further developed for IAC-DIDAS-N.

A multi-objective problem definition usually admits many efficient decisions and outcomes; the user should first learn about the ranges of changes of outcomes and bounds on efficient outcomes. The main function of IAC-DIDAS-N in the initial analysis phase is the calculation of these bounds. The user can request the system to optimize any objective separately; however, there is also a special command that automatically performs all necessary calculations.

The command "utopia" results in subsequent computations of the best possible outcomes for all objectives treated separately (such outcomes are rarely jointly attainable, hence the name utopia point for the point in outcome space composed of such outcomes). During utopia calculations approximations of some worst possible efficient values are also obtained. The point in outcome space composed of the worst efficient values is called the nadir point. The exact calculation of this point is a very difficult computational task – for nonlinear models there is no constructive method for such a calculation. The approximation of nadir point components obtained during utopia point calculations is too optimistic. The user can change, according to her/his knowledge, the obtained approximations of nadir values.

After the initial phase, the user has a reasonable knowledge of the range of possible outcomes and thus can specify her/his reference levels: aspiration levels to be attained and reservation levels to be avoided in any case. The utopia and the nadir points could be used as initial values for the aspiration point and the reservation point, respectively. However, because the neutral solution has also been calculated, the system suggests another, more adequate initial aspiration and reservation points. Namely, an unattainable outcome point closer to the efficient solutions than the utopia point, and an attainable outcome closer to the efficient solutions than the nadir point, respectively.

One iteration of IAC-DIDAS-N analysis is composed of three elements. First, it uses the aspiration and reservation levels as parameters in a special achievement function coded in the system. Second, it uses its solver to compute the solution of a nonlinear programming problem equivalent to maximizing this achievement function. Finally, it responds to the user with an attainable, efficient solution and outcomes that strictly correspond to the user-specified references.

If the aspirations are unattainable while the reservations are attainable (i.e., a typical and recommended case), then the response of the system is a solution with attainable, efficient outcomes that are either between the aspiration and reservation points or uniformly as close as possible to the aspiration point. If the aspirations are too low (i.e., if they correspond to attainable but inefficient outcomes that can be improved), then the response of the system is a solution with outcomes that are uniformly better than the aspirations. If the reservations are too high (i.e., if they correspond to outcomes that are not attainable), then the response of the system is an efficient solution with outcomes that are uniformly worse than the reservations. The precise meaning of the uniform approximation or improvement depends on scaling units for each objective. These units are defined automatically in the system in relation to the differences between the utopia point, the current aspiration point, and the current reservation point. Therefore, the scaling units are implicitly defined by the user. This automatic definition of scaling units has many advantages for the user who is not only freed from specifying them but also has better control over the selection of efficient outcomes by changing the reference levels.

After scanning several representative efficient solutions and outcomes controlled by changing references, the user typically learns two things. She/he either learns enough to subjectively select an actual decision (which need not correspond to the decisions proposed in the system, since even the best substantive model can differ from a real decision situation) or to select an efficient decision proposed by the system as a basis for actual decisions.

10.6 Conclusions

This chapter has been somewhat akin to a journey: we started with general remarks on functions and the philosophy of user interfaces, to general features of graphical user interfaces, and ended with a detailed analysis of specific interfaces. It should be stressed again that the selection of the most fitting interface is problem and user specific. However, since the software for customizing user interfaces is not yet developed, we must rely on standardization. Such a standardized interface, using interactive graphics and relying on an aspiration/reservation-based reference point approach, is contained in the software package MCMA, together with the interactive module ISAAP. However, we do not conclusively say that this is the best possible interface for all DSSs, or even for all reference point approaches in DSSs. An older spreadsheet-type interface of DIDAS-N also has its advantages, at least for other stages of the process of model analysis. Thus, while aiming to achieve a balance between standardization and diversity of problems and users, developing good user interfaces will – at least, for some foreseeable time – remain an art.

Part III

Environmental Applications

Chapter 11

River Basin Water Quality Management

Marek Makowski and László Somlyódy

This chapter presents some results of the development of decision support systems (DSSs) for regional water quality management, prepared by the Water Resources Project (WAT) in cooperation with the Methodology of Decision Analysis Project (MDA) at the International Institute for Applied Systems Analysis (IIASA). After an overview of several DSSs developed by the WAT Project, we concentrate on the application of multi-criteria model analysis methodology to the Nitra River basin in Slovakia. In this application, a model-based, aspiration-led methodology for multi-criteria decision support has been applied and compared with single-criterion model analysis techniques.

This chapter provides a formulation of the mathematical model (including the embedded water quality model) used for decision support and an overview of results that illustrate the applied approach and provide some insights into this case study. Several reusable, modular software tools have been used for the development of a DSS, whose structure is outlined in Section 3.6. Only two software modules have been developed specifically for this application, namely a problem-specific generator to produce the core part of the mathematical programming model and a reporting module.

After a broad introductory section, the remaining part of this chapter is organized as follows: Section 11.2 contains a brief overview of the river basin water quality management problem, Section 11.3 documents the formulation of the mathematical model and Section 11.4 provides an overview of the DSS implementation, involving the use of single- and multi-criteria optimization techniques for a model analysis. A discussion of results illustrating the multi-criteria model analysis methodology is given in Section 11.5; finally, the last section summarizes the advantages and limitations of a multi-criteria approach as observed during this application.

11.1 Introduction

Decision making in water quality management typically involves the consideration of a number of goals; these goals cannot be aggregated into a single criterion to be used as a performance measure for ranking alternatives. Multi-criteria optimization has long been applied to water resources problems. Early work was categorized by Cohon and Marks (1975; for follow-up discussion see Cohon and Marks, 1977; and Krzysztoforowicz *et al*., 1977) and later reviewed by Haimes and Hall (1974), Cohon (1978), and Changkong and Haimes (1983). More recently, multi-criteria decision analysis techniques have been used to derive reservoir operating rules (e.g., Lewandowski *et al*., 1985; Ikebuchi and Kojiri, 1992; Laabs and Schultz, 1992), to design groundwater remediation plans (Shafike *et al*., 1992), and to assist in water resources conflict negotiations (Thiessen and Loucks, 1992). A comprehensive overview of recent methodologies is given by Hipel (1992).

The WAT Project at IIASA focused, however, less on the methodology of multi-criteria analysis but more on analyzing various strategies for river basins in the Central and Eastern European (CEE) region. Due to the ongoing socioeconomic transition and lack of financial resources, this region urgently needs efficient policies and tools to develop these resources. The WAT Project developed two river basin DSSs and applied them to four river basins to evaluate various management strategies (Somlyódy, 1996).

The driving principle behind the development was to use the best state-of-the-art knowledge in the fields of water quality control (hydraulics, water quality, uncertainty and parameter estimation, alternative evaluation of treatment technologies, economic instruments, optimization methods, etc.) and to integrate them under the available capabilities of the computers to achieve an interdisciplinary tool of policy development. Policies included uniform emission reduction, least-cost strategies (e.g., to satisfy local dissolved oxygen criteria), strategies to meet requirements of regional nutrient emission load reductions (keeping in mind the eutrophication problems of, e.g., the Baltic or the Black Sea, which call for a basin-wide coordinated action plan, in addition to local measures), economic instruments, and a number of mixed strategies.

An attempt was made to use some new but generic model components such as the one-dimensional diffusive wave approach to compute flows in the river system, dynamic programming, and the so-called Hornberger and Spear uncertainty analysis method. The perceived advantage of the latter two is that they are not model structure-specific. Decision variables are typically discrete – 0 or 1 – and are associated with each possible municipal wastewater treatment plant (MWWTP) alternative – upgrading, new design, alternative technologies – for all the emissions. Alternatives enter the procedure via control matrices belonging to a particular emission. Each line of a matrix includes effluent quality of the pollutants considered and the associated costs (investment, operation, maintenance and repairs, and the total annual cost). Industrial emissions or nonpoint sources can formally be represented by similar tables (although their derivation is not a straightforward task).

After careful preparation, two public domain DSSs were developed assuming their implementation in a PC environment. Detailed descriptions of the tools can be found in their respective manuals (DeMarchi *et al*., 1996; Ivanov *et al*., 1996). The spreadsheet tool for river environment assessment management and planning (STREAMPLAN) is a DSS that allows the evaluation of river basin water quality policies on the basis of local and regional water quality goals, effluent standards, costs, financing, economic instruments, municipal water management issues, and the generation of treatment plant alternatives.[1] The decision support system for evaluating river basin strategies (DESERT) is a flexible Microsoft Windows-based tool (written in C++) for decision support of water quality management at the river basin scale. This software provides a powerful instrument for developing least-cost and mixed river basin policies and their analysis under conditions deviating from the design scenario. While STREAMPLAN is stronger in terms of offering policy options and economic analysis, DESERT offers more process descriptions and modeling methodologies.[2]

A brief comparison of the two DSSs is given in *Table 11.1*. STREAMPLAN and DESERT complement one another in various ways. For instance, calibration for hydraulics and water quality simulation can be taken from DESERT, then a broader range of policies and municipal wastewater treatment alternatives can be assessed by STREAMPLAN; detailed water quality simulation of STREAMPLAN strategies can be checked through simulation in DESERT.

Both systems use single-criterion optimization and the multi-criteria aspects are considered by systematically changing values of the constraints representing water quality goals. However, in the course of further development, multi-criteria assessment possibilities were also evaluated; this possibility is discussed in more detail later in this chapter.

[1]See http://www.iiasa.ac.at/Research/WAT/docs/stream.html.
[2]See http://www.iiasa.ac.at/Research/WAT/docs/desert.html.

Table 11.1. Comparison of the major features of STREAMPLAN and DESERT.

	STREAMPLAN	DESERT
Focus of the assessment	River basin, municipal, and treatment plant level	River basin level
Hydraulics	Uniform steady flow	Steady and unsteady flow
Water quality	Six water quality constituents (C, N, P); linear reactions and steady state (transfer coefficient approach)	Large number of constituents (C, N and P), nonlinear, steady, and unsteady state conditions, flexibility to select the model structure (C++ language)
Calibration	None	Generalized parameter estimation and uncertainty analysis
Optimization	Mixed integer linear programming	Dynamic programming (nonlinear model and binary variables)
Generation of MWWTP alternatives	Included	None
Water quality strategies	Regional nutrient emission reduction, economic instruments (effluent charges, water price, fee, etc.), and a large number of combined policies	
	Both have uniform emission reduction, effluent standards (including minimum treatment level), ambient criteria based on least-cost policies, mixed strategies	

In selecting a set of water pollution control measures such as technologies to be applied at MWWTPs, a group of economic criteria must be considered (e.g., various types of costs: investment, operation and maintenance) and water quality criteria (dissolved oxygen levels and concentrations of different pollutants). If we select one of the criteria to be optimized and represent the other by hard constraints, the system will overlook possibilities for slightly violating a constraint on a criterion in order to substantially improve other criteria. Thus, a fully multi-criteria approach must also be tested.

Another difficulty is related to the explosion of dimensionality typical for such an approach to discrete and dynamic optimization problems.[3] Because the number

[3]The reader must be warned that this is one of the typical traps in practical optimization. At a preliminary stage of research, we naturally consider only conceptual, simple examples. We are then apt

of feasible alternatives can be astronomical even for a small region (e.g., eleven MWWTPs each with seven technology options lead to about two billion alternatives), a rational way is needed to select a manageable number of alternatives to be considered in more detail.

Thus, another variant of a DSS was developed in cooperation with WAT and MDA Projects at IIASA for regional water quality management (and subsequently referred to as RWQM). It is based on the method of multi-criteria decision analysis described in detail in Chapter 4 (i.e., aspiration/reservation led multi-criteria model analysis). The purpose of RWQM is to facilitate the exploration of the set of efficient solutions and, ultimately, aid the decision maker in evaluating trade-offs and selecting a preferred set of wastewater treatment technologies.

11.2 The Problem

The scope of the problem considered here is a river basin, or a region composed of several basins, in which untreated or inadequately treated municipal and industrial wastewater emissions should be reduced in order to improve ambient water quality. At each discharge point, one technology out of a set of possible technologies can be implemented in order to meet the desired water quality goals in the region. In this selection of technologies, or strategy development, decision makers must evaluate the trade-offs among a large number of alternatives based on, among other things, effluent and/or ambient water quality standards and goals; capital investment and annual operating costs; and the principles of equity, uniformity, and efficiency. There is also a need to set strategies that are realistic in the short term yet consistent with long-term planning goals. This strategy development can be performed in many different ways, depending on the underlying principles and methodologies employed.

The traditional approach, as used in developed countries, is based on the selection of generally uniform effluent standards which, in turn, are often based on given technologies. This is the well-known policy of “best available technology”. Under such an approach, both ambient water quality standards and budget requirements are considered only indirectly. The following two conditions must be met:

- If effluent standards are defined stringently enough, then ambient water quality will be “good enough”.

to believe that the discrete character of variables will simplify optimization and that some conceptually elegant approaches, such as dynamic programming, will be appropriate. It then turns out that we should use slightly more complicated models, and the dimensionality explodes: the discrete character of variables complicates optimization and dynamic programming requires very long computational time.

- Enough money (or willingness to pay) is available to achieve "safe" environmental conditions (without raising the issues of how safe they are and how much should be paid for them).

Unfortunately, such a robust and uniform policy may not be an affordable option for countries in Central and Eastern Europe for the coming decade or longer. Rough estimates suggest that the per capita cost required to reach surface water quality comparable to western standards is approximately equal to the per capita GDP of these countries – a few thousand US dollars. Thus, a more promising approach is to specify ambient water quality goals and to look for a regional least-cost policy. Such a nonuniform strategy can lead to significant cost savings (as shown in Somlyódy, 1994; Somlyódy *et al*., 1994b) although implementation is less straightforward than for a policy based on effluent standards. The present trend in the developed countries is to focus more strongly on regional ambient water quality impacts and thus to combine effluent and receiving water quality criteria. The reason for this change is the realization of past overexpenditures and some failures of effluent-based control programs (e.g., due to underestimating the effect of nonpoint source pollution).

11.3 Model Formulation

The river water quality model presented here is quite simple. It is based on the concept of linear transfer coefficients (Loucks *et al*., 1981), which are derived from first-order rate equations and the (linear) extended Streeter-Phelps model incorporating dissolved oxygen (DO), carbonaceous oxygen demand, and nitrogenous oxygen demand (e.g., Thomann and Mueller, 1987). Steady-state hydraulics are considered, based on a "critical design flow" (Somlyódy *et al*., 1994b). Complete mixing downstream of each emission and tributary confluence and uniform flow along the river between these points are assumed.

For the Nitra River system (shown in *Figure 11.1*) a set of locations or points is defined, each of which is characterized by at least one of the following:

- An emission point, at which wastewater is discharged. The amount of discharged pollutants depends on the treatment technology chosen in the decision process.
- An abstraction point, at which water is withdrawn from the river. At these points one can consider a "negative" emission, whereby the constituent loads are reduced proportionally to the reduction in river flow.
- A monitoring point, at which concentrations of water quality constituents are compared to given standards.

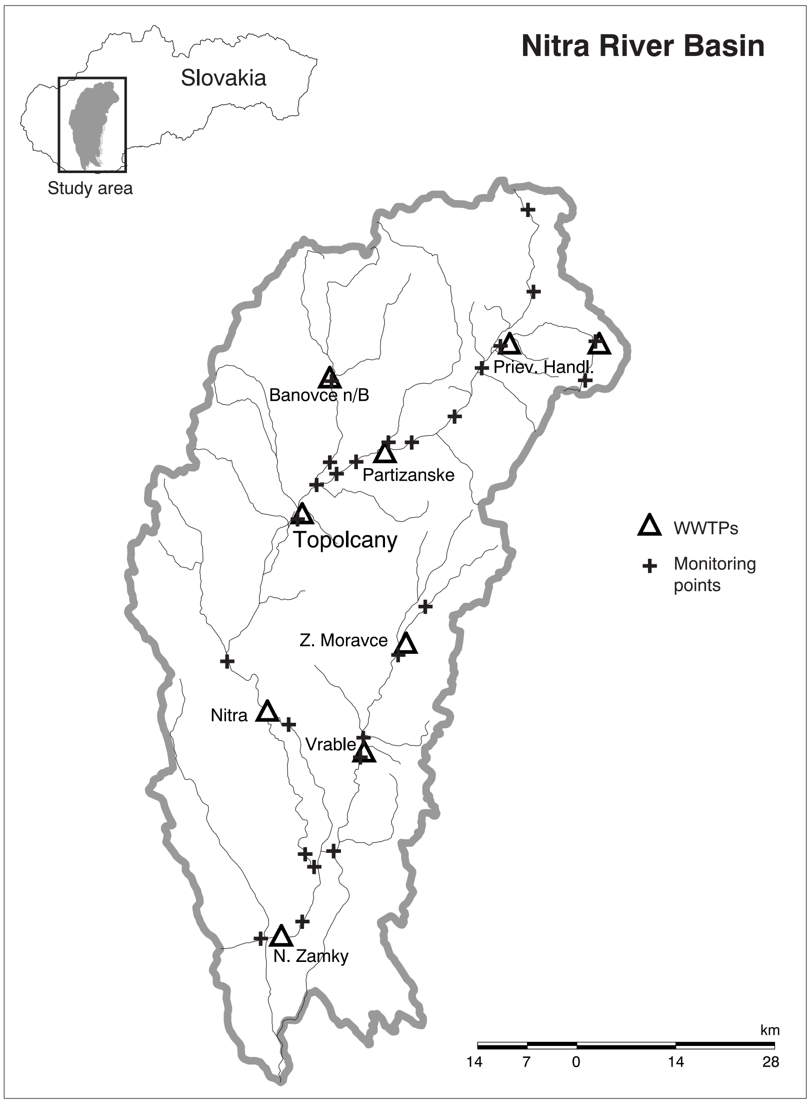


Figure 11.1. Nitra River basin, Slovakia.

- A confluence point, which represents the junction of two rivers. Constituent loads are the sum of loads from both rivers.
- A weir point, where DO is added to the river due to the increase in turbulence downstream of a weir or small dam.
- Other points where hydraulic and hydrologic data exist and therefore new travel times and transfer coefficients can be calculated. The loads of constituents do not change at these points.

Each of these points is called a node, denoted by the subscript j. At every node the equations that define water quality (i.e., mass balances of constituents) are given. Overall, four water quality constituents (of which three are real state variables) are considered. In the equations the subscript l is used to denote the respective constituents:

0. DO (dissolved oxygen).
1. CBOD (carbonaceous biochemical oxygen demand).
2. NBOD (nitrogenous biochemical oxygen demand.
3. NH_4 (ammonia).

NBOD is calculated directly from NH_4, assuming that all of the nitrogen consumes oxygen. Sediment oxygen demand (SOD), denoted by $l = 4$, is considered a parameter to be calibrated.

The decision variables are the treatment technologies to be implemented at the nodes where wastewater emissions occur. These are denoted by x_{jn}, where n is the technology choice at emission node j. These technology options include the option of no treatment (with raw waste concentrations and no cost, which is actually only a theoretical alternative since minimum treatment levels would most likely be required), as well as the option of maintaining the existing technology (with the operating cost but no investment cost). Since only one technology can be implemented at each point, the following constraint is imposed:

$$\sum_{n \in N(j)} x_{jn} = 1 \qquad x_{jn} \in \{0,1\}, \qquad j \in E, \tag{11.1}$$

where $N(j)$ is the set of technologies considered for emission node j, and E is the set of nodes where emissions occur.

Auxiliary variables (the values of which depend on decision variables) in the model include variables related to water quality and variables related to cost. Focusing on the first set, we consider the water quality constituent concentrations resulting from the implementation of the n-th technology at the j-th emission node, em_{jnl} (mg/l). The emission load of the l-th constituent at the j-th node is denoted by e_{jl} (g/s) and is defined by:

$$e_{jl} = q_j \sum_{n \in N(j)} x_{jn} em_{jnl} \qquad l \in \{1,3\}, \qquad j \in E, \tag{11.2}$$

where q_j (m^3/s) is the waste flow rate. Note that due to equation (11.1), for each j exactly one out of $N(j)$ binary variables, x_{jn}, will be equal to one while the others will be equal to zero.

Next, the ambient constituent concentrations must be defined. The ambient concentration of DO (mg/l), typically the most important water quality indicator, is affected by several constituents, as well as by its saturation level (see Thomann and Mueller, 1987, for details). The DO concentration (denoted for the j-th node by aq_{j0}) is given by the extended Streeter-Phelps model, analytically integrated stretch by stretch as follows:

$$\begin{aligned} aq_{j0} \;=\; & [1/(Q_j + W_j)] * \Big(\sum_{i \in I(j)} \Big(b_{i0} + Q_i * (DOsat_j \\ & -TC_{i0}(DOsat_i - aq_{i0}) - \sum_{l \in \{1,2,4\}} TCp_{il} aq_{il}) \Big) + ioxy_j \Big). \end{aligned} \qquad (11.3)$$

In this, the set $I(j)$ is composed of indices of nodes located immediately upstream of the j-th node (this set contains two elements for confluence nodes and one element otherwise), aq_{il} (mg/l) are the upstream concentrations of oxygen-demanding constituents (CBOD, NBOD, SOD), and the remaining right-hand side quantities are given (or computed from given data): $DOsat_j$ (mg/l) is DO saturation level at the j-th node, TC_{i0} is a dimensionless transfer coefficient for the DO deficit (defined as $DOsat_i - aq_{i0}$), TCp_{il} are dimensionless transfer coefficients for CBOD, NBOD, and SOD, respectively; Q_j (m^3/s) is the river flow just below node j, W_j (m^3/s) is the withdrawal occurring at the j-th node, b_{i0} (g/s) is the background level of DO entering the river upstream of node j, and $ioxy_j$ (g/s) is the DO emission at node j. Thus, the summation term represents the DO coming from upstream, which consists of oxygen transfer from the upstream node(s) as well as "background" oxygen from groundwater infiltration flow (for simplicity, we assume that background loads of other constituents do not affect DO until the next reach downstream). This upstream mass is then mixed with the DO load from the wastewater emission, $ioxy_j$, hence the division by the total flow $Q_j + W_j$. Calculation of the transfer coefficients on the basis of the Streeter-Phelps equations (exponential terms expressing transformations due to decay and reaeration over the travel time) and other parameters of the model are documented in Makowski *et al.* (1995).

Ambient concentrations of the other constituents such as CBOD and NBOD (denoted by aq_{jl}) are defined by:

$$aq_{jl} = \Big(\sum_{i \in I(j)} (b_{il} + TC_{il} aq_{il} Q_i) + e_{jl} \Big) / (Q_j + W_j) \qquad l \in \{1, 3\}, \qquad (11.4)$$

where, as in equation (11.3), the first term in this equation represents the background load of constituent l which accounts for nonpoint or noncontrollable source pollution, the second term represents the load of the constituent l arriving from the upstream reach(es), and the third term represents the emission load of constituent l at node j. Cross-impact transfer coefficients, TCp_{il}, are not included here since these constituents are not affected by the DO level unless anoxic conditions exist.

Based on these ambient constituent concentrations, the following three indices of water quality are defined:

$$DO = \min_{j \in M}(aq_{j0}), \tag{11.5}$$

$$BOD = \max_{j \in M}(aq_{j1}), \tag{11.6}$$

$$NH_4 = \max_{j \in M}(aq_{j3}), \tag{11.7}$$

where aq_{jl} (defined by (11.3) or (11.4)) is the ambient concentration of the l-th constituent at node j, and set M contains indices of monitoring nodes.

Finally, several cost variables are defined in the model. Corresponding to the n-th treatment technology implemented at the j-th node are an investment cost IC_{jn} and an operation and maintenance cost OMC_{jn}. The investment costs Inv_j for the j-th emission point are defined by:

$$Inv_j = \sum_{n \in N(j)} x_{jn} IC_{jn} \qquad j \in E. \tag{11.8}$$

The operation and maintenance costs OM_j are given by:

$$OM_j = \sum_{n \in N(j)} x_{jn} OMC_{jn} \qquad j \in E. \tag{11.9}$$

The total annual cost (TAC) of each technology is determined from the two previous cost components as:

$$TAC_j = [r(r+1)^m/((r+1)^m - 1)]Inv_j + OM_j \qquad j \in E, \tag{11.10}$$

where r is a given discount rate, m is a given capital recovery period, and the multiplier of the first term is called the uniform series capital recovery factor. One may also want to consider the sums of respective costs for the whole region:

$$Tot\_Inv = \sum_{j \in E} Inv_j, \tag{11.11}$$

$$Tot\_OM = \sum_{j \in E} OM_j \tag{11.12}$$

$$Tot\_TAC = \sum_{j \in E} TAC_j. \tag{11.13}$$

The treatment alternatives at various MWWTPs, along with the corresponding effluent concentrations and costs, are discussed in detail in Somlyódy *et al*. (1994b). Each alternative was developed on the basis of technological calculations using physical, biological, and chemical processes, as well as their combination. These lead to well-known methods such as mechanical–biological treatment with or without denitrification, mechanical–biological treatment with chemical addition to remove phosphorus and/or to increase the capacity of the plant, biological–chemical treatment with denitrification, and so forth. The alternatives identified also depend on whether upgrading an existing facility or constructing a new plant is considered a viable option. For each MWWTP, four to nine alternatives were developed and denoted numerically, indicating improving effluent quality and increasing costs (i.e., 0 = no treatment, 1 = operation of the existing plant, 2 = chemical upgrading of an overloaded mechanical–biological plant, 3 = biological upgrading of plant 2, 4 = plant 3 with P removal, 5 = a combination of the upgraded plant 4 and a new plant, and 6 = a new advanced treatment plant with P and N removed; see Somlyódy *et al*., 1994b).

11.4 Model Analysis

The studies of the WAT Project resulted in considerable experience in modeling regional water management problems and, in particular, the problem stated above. However, as already stressed in Chapters 6 and 7, there are two basic types of approaches for the analysis of mathematical models: simulation and optimization. In simulation, decision variables are inputs and goals are outcomes. Therefore, this technique is good for exploring, for enriching the intuition of a decision maker, and for the verification of a model. Simulation is also good for providing a decision maker with detailed information about the consequences of applying certain decisions. One can thus consider simulation as an alternative-focused method of analysis, in which the user examines the effects of implementing prespecified decision alternatives. In contrast, optimization can be considered as a goal-oriented (value-focused) approach that is directed toward creating alternatives. Optimization is driven by a desire to reach a set of goals expressed in terms of values of the objective(s). Therefore, goals are a driving force, and the values of decision variables are outcomes. An interchangeable use of both simulation and optimization can be achieved by a multi-criteria approach, as stressed, for example, in Chapter 5. Such an approach has obvious advantages, especially in the learning phase of using a DSS. Before discussing this approach, we first briefly summarize the use of simulation and single-objective optimization for the Nitra River basin case study.

11.4.1 Simulation and single-objective optimization

The regional least-cost approach generally leads to the use of simulation and single-objective optimization for decision support (Somlyódy *et al*., 1994a). This requires the development of a core model that relates wastewater emissions, treatment decisions, and the resulting ambient water quality. It is important that the core model includes only physical and logical relations, and not the preferential structure of the decision maker (hence the core model is a type of a substantive model as defined in Part I of this book). Simulation with the core model, or scenario analysis, generally allows the application of a more complex, physically based water quality model than the optimization approach. However, for decision problems with a large number of alternatives, as noted before, the simulation model may have to be run a great number of times in order to identify a "best" (or even acceptable) solution. Optimization can provide a tool for screening the large number of alternatives and choosing a "best" solution with respect to a particular objective function and set of constraints.

The single-objective optimization-based DSS DESERT described previously was also applied to the Nitra River basin (DeMarchi *et al*., forthcoming; Jolma *et al*., 1997). The core model included a hydraulic model (diffusive wave approach) and a sequence of linear and nonlinear water quality models considering dissolved oxygen, nitrogen, and phosphorus balances. For optimization, dynamic programming was employed with a least-cost formulation. Multi-criteria aspects of the problem were handled by changing constraints and their right-hand side values (see Somlyódy *et al*., 1994a, 1994b, for details).

There are several techniques to deal with de facto multi-criteria problems within the framework of single-criterion optimization, some of them discussed in earlier chapters. For example, the ϵ-constraint approach, in which (n-1) objectives are placed into constraints with given tolerable levels (which can be interpreted as reservations for the criteria that have to be achieved), was proposed in Haimes and Hall (1974). This hard requirement can be relaxed by representing requirements for the values of criteria as "soft" constraints, which can be treated as a special case of multi-criteria optimization (see Makowski, 1994c, for details). Here we do not discuss approaches based on the idea of converting a multi-criteria problem into a single-criterion one by summing up weighted criteria. This approach has a number of drawbacks (as discussed in detail in Makowski, 1994c; Nakayama, 1994; and also in earlier chapters). Here we only consider two main disadvantages. First, such an approach does not allow us to find all Pareto-optimal (efficient) solutions. Second, using weights can be counterintuitive, as one can find examples in which, for certain regions of the efficient frontier, there is no positive correlation between increasing the weight for a criterion and the corresponding improvement of the criterion value.

For the case in which certain ambient water quality standards are implemented (such as maintaining DO concentrations above 5 mg/l in order to achieve a Class II water quality status), the single-objective optimization based approach would probably provide the most straightforward decision support (Somlyódy *et al.*, 1994a, 1994b). The objective would be to minimize the cost of meeting each of the "hard" water quality constraints. Similarly, if the budget for policy implementation was fixed, the objective would be to maximize some measure of water quality given the prescribed budget (though different water quality constituents could represent different objectives). However, if neither the ambient water quality standards nor the budget is set, as in most of the countries of Central and Eastern Europe, the problem becomes a truly multi-objective one. In this case, with single-objective optimization, the decision maker would typically select one criterion to be optimized and then systematically vary the constraint levels (which correspond to the other criteria). This was done for the Nitra River basin when applying DESERT in order to learn the multi-objective aspects of the problem. However, such an approach might become tedious and, similarly as in scenario analysis (although to a lesser extent), several acceptable solutions could be overlooked.

11.4.2 Multi-criteria model analysis

The multi-criteria approach used for the Nitra River basin application attempts to alleviate some of these difficulties. Hard constraints for environmental criteria and the sequential analysis of a set of single-objective solutions (necessary in a single-objective approach) are replaced by an interactive specification of the preferences of the decision maker or DSS user expressed in terms of aspiration and reservation levels. Moreover, the simplicity and flexibility of such an approach allow the user to learn more effectively about the decision situation during the process of analysis. One possible disadvantage of the implemented approach is that simple, linear models were used rather than more physically based (i.e., nonlinear) ones. However, an original nonlinear model can also be approximated by a piece-wise linear model (which was not done in the application presented here, but can be done if the obtained modeling accuracy is not sufficient). Multi-criteria analysis of a nonlinear model would also be possible. However, the development of a nonlinear model would require more resources and its application would require more powerful hardware (or would not be interactive). Moreover, the usefulness of complex models for management is often limited by uncertainty in system identification and by the lack of high-quality field data. This is indeed true for the case study described here, as well as for many other river basins in Central and Eastern Europe. It should also be noted that decisions are often made on the basis of a few aggregated indicators (as opposed to many variables used for a related scientific analysis) for which case the assumption of linearity is generally acceptable.

To facilitate the multi-criteria analysis of the core model, the RWQM system was constructed from several modular tools and implemented with the principle of reusability in mind. The functional structure of RWQM is illustrated in *Figure 3.2* and its components are described in Section 3.6. A particular implementation of the methodology discussed in Chapter 4 applied to the multi-criteria model analysis (later referred to as the MCMA method) in the Nitra case study is presented in Chapter 10. The MCMA method is applied to a core model, which has to fulfill the requirements specified in Section 6.6.

The core model for the Nitra River basin described in Section 11.3 conforms to those requirements. The core model is composed of equations (11.1) through (11.13), which involve only the logical and physical constraints and the definitions of variables. Therefore, none of the decision variables is constrained by a quantity which is actually an exogenous decision variable (such as a maximum available budget or an acceptable constituent concentration). Due to the nature of the MCMA method, one can examine various Pareto-optimal solutions that represent compromises between costs and water quality. The basic advantage of this approach is that it provides a natural way to examine a number of Pareto-efficient solutions without facing the risk of infeasibility.

Before the interactive part of multi-criteria analysis starts, a user selects the criteria to be considered. The discussion presented here is based on results obtained for the following six criteria:

- TAC: total annualized cost for the region (equation 11.13).
- INV: total investment cost for the region (equation 11.11).
- OMC: total operation and maintenance cost for the region (equation 11.12).
- DO: minimum DO concentration at any monitoring point (equation 11.5).
- BOD: maximum CBOD concentration at any monitoring point (equation 11.6).
- NH_4: maximum NH_4 concentration at any monitoring point (equation 11.7).

All the selected criteria are minimized apart from the DO criterion, which is maximized. However, a user could easily examine the problem using other criteria.

After the selection of criteria, the multi-criteria model generator computes a pay-off table, shown in *Table 11.2*. First, six selfish optimal solutions are computed for each criterion consecutively. The row labeled Utopia summarizes the criteria values obtained from selfish solutions, for which each criterion is optimized in successive single-criterion optimization runs. For example, the first selfish optimization run minimizes the TAC with no regard to the other criteria. Thus, a low value of TAC is obtained (US$1.55 million represents the cost of existing treatment plants), but poor values of the water quality criteria are obtained. After all components of the utopia point are computed, six auxiliary problems are solved

Table 11.2. Pay-off table for the six-criteria problem.

Criterion	Criteria value					
optimized	TAC	INV	OMC	DO	BOD	NH_4
TAC	1.55	0.0	1.55	0.14	25.80	4.71
INV	6.06	0.0	6.06	3.60	11.60	3.84
OMC	1.55	0.0	1.55	0.14	25.80	4.71
DO	14.40	34.3	8.45	5.38	9.81	1.70
BOD	14.40	32.7	8.45	5.38	9.81	1.70
NH_4	14.40	29.9	8.45	5.38	9.81	1.70
Utopia	1.55	0.0	1.55	5.38	9.81	1.70
Nadir	14.40	50.3	8.69	0.14	25.80	4.71
Nadir*	14.40	34.3	8.45	0.14	25.80	4.71

for computing a first approximation of a nadir point. The nadir point cannot be computed by selfish optimization (see Makowski, 1994c, for details). Therefore, the row labeled Nadir summarizes the worst values of criteria that were obtained in any part of the analysis. This is a better approximation of the nadir point than the values provided in the row labeled Nadir*, which is composed of the worst values obtained during the selfish optimizations only.

The pay-off table is useful as a guide for the evaluation of trade-offs among the criteria and provides good initial information about the ranges of criteria values that can be expected from any rational decision. A typical starting point is to choose criteria values at the utopia point as aspiration levels, and values at the nadir point as reservation levels. The resulting neutral compromise solution is shown in *Table 11.3* as solution No. 1. This solution is computed before the interactive analysis starts, with trade-offs between the criteria automatically calculated based on the utopia point and on the current approximation of the nadir point. Therefore, the neutral compromise solution has a low value of DO due to the very low nadir value for DO. Since the DO level is an important water quality indicator (a desirable value is about 5 mg/l), this solution is far from acceptable.

At this point, an interactive multi-criteria analysis may be implemented to search for acceptable solutions. The analysis is typically composed of several sessions, during which a number of solutions are generated and analyzed. A cycle (discussed in this section) composed of analysis of previous solutions, selection of new aspiration and reservation levels, and optimization is conventionally called an iteration: one iteration for this problem takes about 1–2 minutes on a Sun Sparc-10 workstation. Usually, several iterations are needed to explore one region of the Pareto-optimal solution space.

Although a total of six criteria were considered, at different stages some criteria were examined more closely than others. For example, at the initial stage the user

Table 11.3. Clustered selected solutions from multi-criteria optimization. Solutions marked by A contain solutions Nos. 22, 23, 25, and 26. Solutions marked by B contain solutions Nos. 27–49, excluding solutions Nos. 32, 40, 45, and 46 (for abbreviations refer to text).

Nos.	TAC	INV	OMC	DO	BOD	NH_4
1	6.6	13.5	5.1	2.8	17.5	3.2
6	7.5	21.5	5.0	3.4	16.8	3.1
20	8.7	28.5	5.3	4.8	17.3	3.0
32	8.7	22.5	6.0	5.0	16.5	3.0
40	10.0	23.1	7.3	5.1	10.2	2.4
45	10.9	26.3	7.9	5.2	10.1	2.2
46	10.3	28.0	7.0	5.2	10.1	1.8
2–5	3.6–4.2	1.0–2.8	3.5–4.1	2.9–3.1	19.5–21.8	4.0–4.3
8–11	5.4–5.8	1.2–2.6	5.3–5.5	3.7–4.0	17.7–17.8	3.7–3.8
12–14	6.2–6.6	1.0–2.3	6.1–6.4	4.0–4.2	11.2–11.5	3.7
15–17	7.7–7.9	6.7–8.0	6.9–7.0	4.5	11.1	3.6
18–19	7.7–7.9	10.0–12.0	6.5	4.7	10.4–10.6	3.2–3.3
21	7.5	17.0	5.5	4.8	20.2	3.8
A	7.9–8.2	10.5–12.5	6.6–6.8	4.8–4.9	10.2–10.6	3.1–3.3
24	8.1	16.0	6.2	4.9	10.7	3.3
B	8.2–9.6	12.6–17.6	6.5–7.5	4.9–5.2	10.1–10.6	2.9–3.3
50, 51	10.8–11.8	29.9–32.9	7.3–8.0	5.3	9.9–10.1	1.7
52, 53	14.1–14.4	50.3	8.2–8.5	5.4	9.8	1.7

was motivated by the results of the neutral compromise solution to examine more closely the relation between DO and INV. Therefore, the aspiration and reservation values for those criteria were changed more carefully, while those values for the other four criteria were set relatively loosely (i.e., the reservation level was close to the nadir point, and the aspiration level was near the corresponding value of a previous solution). The values of INV and DO were then stabilized (by setting relatively narrow ranges between the respective aspiration and reservation levels) and the range between aspiration and reservation levels for the other criteria narrowed. This is why several sets of solutions (see *Table 11.3*, which will be discussed in more detail in Section 11.5) with similar values of criteria were obtained.

11.5 Discussion of Results

Before presenting general results of the multi-criteria analysis, it is useful to outline the present water quality situation (for details and longitudinal profiles of concentrations, see Somlyódy *et al.*, 1994a, 1994b). The DO level under low flow conditions in August is around 2–3 mg/l at the middle stretch of the river, while CBOD and NH_4 may exceed 15 mg/l and 4 mg/l, respectively. Overall, these values

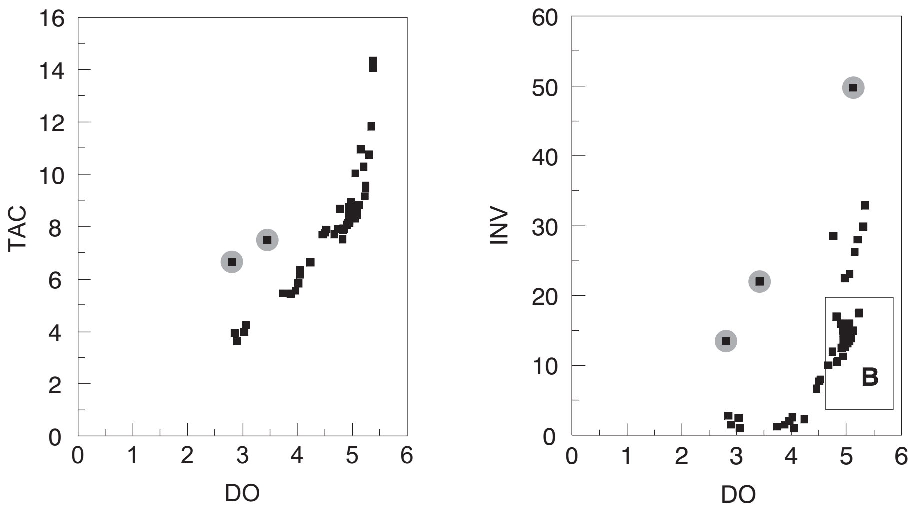


Figure 11.2. TAC and INV (both in million US$) versus DO (mg/l). The set of solutions marked by B contains solutions Nos. 27–49, excluding solutions Nos. 32, 40, 45, and 46.

indicate generally poor water quality, perhaps Class IV in a five-class evaluation system.

Selected solutions for the six-criteria model analysis are summarized in *Table 11.3*. A complete set of solutions can be found in Makowski *et al*. (1995). All presented solutions have been sorted according to increasing values of DO and are labeled by their rank (subsequently referred to as "solution No."). Thus, solution No. 1 has the lowest value of DO, while solution No. 53 has the highest. However, the solutions in *Table 11.3* are grouped by TAC in order to illustrate the following characteristics of the decision problem.

The results can be evaluated with a special focus on the trade-offs between investment cost and operating cost, as well as those among the three water quality constituents:

1. Considering all of the solutions, *Figure 11.2* shows an "exponential" increase of TAC as a function of the DO level. The relation between INV and DO shows that this increase is even faster for the investment costs. The reasons for this are two-fold: the design scenario includes a high temperature and thus low DO saturation levels, and a large number of small emissions are considered in the model as noncontrollable discharges. The shapes of the dependencies TAC(DO) and INV(DO) indicate that DO is an important water quality indicator to be used for management purposes, since the cost of implementing a policy will be highly sensitive to the level of DO desired. However, it is evident

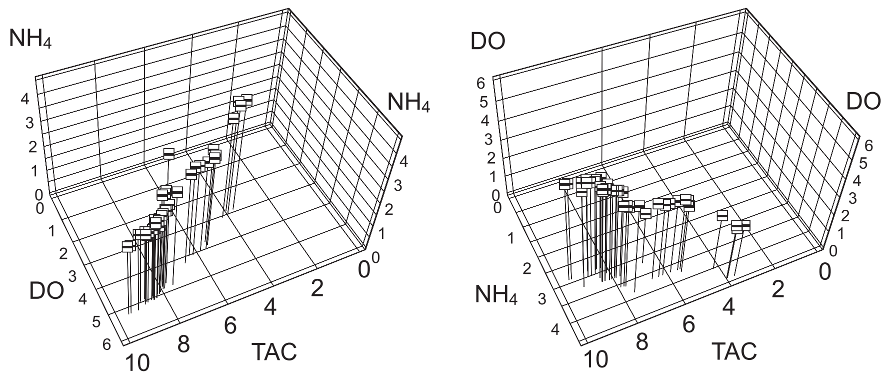


Figure 11.3. TAC (million US$) versus NH_4 and DO (mg/l).

that the aspiration levels for the other criteria also have a significant role. In particular, one can observe three outlying solutions in *Figure 11.2* (enclosed in circles; in the figure on the left-hand side two solutions coincide). Those solutions (Nos. 1, 6, and 53 in *Table 11.3*) have substantially higher TAC and INV values than others with similar DO values. This is because these solutions include capital-intensive technologies (either upgrading or the construction of new plants) for the reduction of NH_4 which are not "cost-effective" from the viewpoint of DO improvement.

2. *Figure 11.3* shows in the (TAC, NH_4, DO) coordinate system clusters of "DO-driven" solutions, along with some with different aspiration levels for NH_4. The same solutions are shown in each half of the figure, only the DO and NH_4 axes have been switched to provide another view. The appearance of three groups of solutions (for DO levels below 3, around 4, and around 5 mg/l) clearly demonstrates the trade-off between the two water quality criteria (DO and NH_4), as well as the cost consequences. The figure also illustrates the strong correlation between improvement of the DO and NH_4 levels, due to the fact that the two water quality variables are interrelated in both the wastewater technologies and in biochemical processes in the receiving waters (see the following).
3. In many cases, TAC is used as a measure to compare different project alternatives. However, if a budget for investments is limited or nonexistent (which is the current situation in many Central and Eastern European countries), INV could be a preferred criterion, and there is a trade-off between the two. This is illustrated by solutions Nos. 22–26, in which investments between US$10.5 and 16 million result in practically the same ambient water quality (see *Table 11.3*). Note that solution No. 24 requires much higher INV and substantially lower OMC than the other four solutions in this set. This clearly indicates the attractiveness of multi-criteria analysis for the reported application case.

One should point out that the water quality indicators and costs used as criteria are correlated. For instance, the oxidation of CBOD and NH_4-N reduces the DO level, and thus a single DO criterion will automatically lead to changes in both BOD and NH_4-N. The coupling of the pollutants takes place not only through biochemical processes in the river but also at the treatment plants. For example, the control of NH_4-N requires nitrification which can be done only after carbon (CBOD) removal is done. Similarly, on the cost side, it is obvious that TAC is uniquely determined by INV and OMC, if the discount rate and the project's life-time are known. Thus, trade-offs between the three economic and the three environmental criteria exist, but they may be weaker than in many other decision problems, for which there are weaker dependencies between criteria.

11.6 Advantages and Disadvantages of Multi-Criteria Analysis

From this analysis of the Nitra River case, a number of important differences between the multi-criteria approach of MCMA and the single-objective optimization approach become apparent. We do not claim that MCMA is necessarily a better approach than single-objective optimization or decision analysis (SODA), but the possible advantages and disadvantages of each approach should be discussed.

The first difference between MCMA and SODA results from the (obvious) fact that in single-criterion optimization only one criterion is specified (usually a cost-related criterion), and the other criteria (such as water quality indices) are treated as constraints. In such analyses, for some constraint values (a priori unknown), one can expect large changes in the value of the selected objective (criterion) to result from relatively small changes in constraint values. For illustration, consider the results presented in *Table 11.4*, showing a number of solutions with similar values of DO and quite different costs (with the aspiration level for DO set to 5.0 mg/l). The right-hand side of the table shows the number (a higher number corresponding to a more expensive technology) of the technology selected at each MWWTP. Two letter abbreviations correspond to a location of a MWWTP (see *Figure 11.1* for treatment plants; Section 11.3 for explanations of alternative technologies; and Somlyódy *et al.*, 1994b, for details).

For example, a significantly lower cost might be obtained (e.g., for solution No. 1 or No. 2) with the minimum DO value set to 4.9 mg/l instead of a round value like 5.0 mg/l, which is a common constraint value selected when considering national water quality goals and classification systems (and which would force selection of one of the much more expensive solutions No. 3, No. 4, or No. 5, resulting in practically the same level of DO). This is especially true for mixed integer programming (MIP) problems such as this one, in which changes in the treatment

Table 11.4. Solutions from multi-criteria optimization illustrating the sensitivity of the problem.

No.	INV	OMC	DO	BOD	NH_4	Ha	Le	Pr	Pa	Ba	To	Ni	Zl	Vr	Su	No
1	11.3	6.8	4.93	10.6	3.29	1	3	2	1	3	6	1	2	3	2	1
2	12.6	6.9	4.98	10.6	3.26	2	1	2	1	3	6	1	3	2	2	1
3	14.5	6.6	5.05	10.6	3.26	2	1	2	1	3	6	1	2	0	0	2
4	14.8	6.7	5.08	10.6	3.26	3	1	2	1	3	6	1	2	0	0	2
5	13.1	6.9	5.10	10.2	3.10	2	1	2	1	3	6	2	1	2	1	1

technologies at various locations lead to discontinuous changes in the investment cost. One should also note that the marginal (dual) values of the constraints are usually not useful for sensitivity analysis of a MIP problem (e.g., Williams, 1990).

Thus, MCMA provides a natural way for dealing with multi-criteria problems and a better approach to the analysis of solutions that nearly meet the constraints imposed in a single-criterion sensitivity analysis. Sensitivity analysis in SODA has a number of both theoretical and practical limitations, which can be very time-consuming and tedious for the user (Makowski, 1994c). It is likely that interesting solutions found by MCMA would be overlooked in a SODA sensitivity analysis. Moreover, the applied MCMA approach can provide a number of solutions that are not available from the SODA, if SODA uses a weighted sum of criteria as a goal function (Makowski, 1994c).

The MCMA approach developed is interactive and very fast, so that the development of a few dozen alternatives does not require more than perhaps two hours for an experienced user with a good understanding of the problem. As shown, a user can easily find a number of Pareto-optimal solutions which are most interesting to her/him. However, the evaluation of these solutions can be more problematic and much more time consuming than the evaluation of solutions from SODA. For instance, solutions that are very close to each other except for values of one of the criteria can appear strange at first, perhaps confusing the decision maker.

Some users may have difficulties in evaluating six criteria visually and quickly. Although multi-criteria model analysis techniques allow for the use of a much larger number of criteria, (e.g., Nakayama, 1994), our experience shows that it is sometimes easier to consider only two or three criteria at a time, while either stabilizing or giving a broad range of aspiration/reservation levels to the other criteria.

Of course, from a purely methodological point of view, SODA is just one particular case of MCMA (for one criterion only), and the MCMA technique developed provides much more functionality without a substantial increase in computational complexity. In our case, the numbers of additional rows and columns generated because of the multi-objective formulation were smaller than 1% of the respective dimensions of the core model. Nonetheless, SODA is more trusted, because it is a classical, well known, and established methodology. SODA is also simpler, especially for inexperienced users, because it does not require knowledge of the methodological background of the relatively new MCMA. Because confidence in the methodology plays a key role in real applications, and developers of a DSS often overestimate the technical background of the DM and the time available for the problem analysis, the potential benefits of MCMA may not be realized. In such a case the application of SODA, which allows the user to specify a well-defined criterion and set of constraints, could prove to be more useful.

Before concluding, a caveat is needed regarding the reliability of the optimal solutions obtained from either SODA or MCMA. Within any cluster of solutions, the treatment configuration shows a certain amount of variability. This variability appears primarily for smaller emissions, while the technologies for larger ones such as Topolcany (To), Nove Zamky (No) and Nitra (Ni) remain rather robust. If we consider the significantly different solutions of *Tables 11.3* and *11.4,* the large number of potential policies is evident. However, a significant portion of them may not be feasible in practice for reasons external to the model. For example, in practice it is crucial to introduce reliable and easily implementable strategies; such concepts play an important role in formulating legislation but are not incorporated into the model. As was shown in Somlyódy *et al*. (1994b), "cheap" alternatives in *Table 11.3* may be too vulnerable if the assumed design scenario is not realized. Also, minimum treatment levels would most likely be specified at all plants. Thus, a decision maker would select a solution with DO about 5 mg/l, requiring INV equal to US$13–25 million. Although the lack of a reliability criterion in the MCMA could be considered a serious weakness, it should be stressed that no optimization methodologies are meant to lead directly to actual decisions. Instead, they are tools which, if used properly, can greatly help the decision maker to understand the issues and identify rational alternatives.

11.7 Conclusions

The results from the Nitra River basin study demonstrate the value of multi-criteria model analysis. Significant trade-offs between the water quality criteria (DO and NH_4 concentrations) and between the cost criteria (investment and total annualized costs) have been identified and evaluated. MCMA helps decision makers to understand these trade-offs when formulating water quality goals, setting budgets, and implementing treatment strategies for the region.

The results highlighted here, found from the application of MCMA, are in close agreement with results found from the sequential application of SODA. Both methodologies serve primarily to provide information to the decision maker and identify alternatives. Thus, a careful analyst would be most likely to apply both methods of optimization, as well as simulation for a more detailed analysis of given alternatives. Nonetheless, a number of advantages and disadvantages of the MCMA approach have been identified.

The main advantage of MCMA is the flexibility provided in model examination. Using the MCMA method, one can easily analyze different "regions" of Pareto-efficient solutions. The selection of these regions depends on the preferences of the user expressed in terms of objective (criterion) values, and can be easily changed during the model analysis upon learning about possible efficient solutions. Because the constraints imposed on objectives in SODA are replaced by the selection of aspiration and reservation levels, the risk of infeasibility in model analysis is eliminated, many problems related to sensitivity analysis are reduced, and a larger number of interesting solutions can easily be found. Conversely, the use of MCMA requires an additional methodological background, some experience in interacting with a computer, and more time for the detailed analysis of a larger set of solutions worthy of examination.

Applications of multi-criteria optimization have been limited mainly by a lack of modular tools that facilitate multi-criteria model analysis. The methodology and software tools developed for this application case make it easier to apply advanced decision support methods to other case studies of regional water quality management. However, several methodological questions are open and provide challenging tasks for further research. These primarily involve the consideration of a reliability-based criterion. Furthermore, the decision maker's understanding and acceptance of MCMA methods should be considered in any application.

Acknowledgments

The development of decision support tools for the Nitra River basin has been been the subject of collaborative research among the Water Resources and the Methodology of Decision Analysis Projects of IIASA, the Water Resources Institute (Bratislava, Slovakia), and the Vah River Basin Authority. The specific work reported here, supported in part by the 1993 IIASA Young Scientists' Summer Program, relied heavily on data and results from this collaboration.

This chapter is based to a large extent on the paper by Makowski *et al*. (1996). In particular, the authors are grateful to David Watkins for his substantial contribution to the research reported in this paper, as well as to the American Water Resources Association for granting permission to reuse part of the material published in this paper.

Chapter 12

Land Use Planning

Günther Fischer and Marek Makowski

12.1 Introduction

The increasing human population in developing countries is exerting pressure on finite land resources, frequently causing overexploitation and land degradation. Sectoral and single-objective approaches to planning for the alleviation of this situation have often not been effective, and an integrated approach is required that involves all stakeholders, accommodates the qualities and limitations of each land unit, and produces viable land use options (FAO, 1995).

Current land use issues in the rural and peri-urban spheres frequently involve environmental versus developmental conflicts. For instance, whether it is preferable to use scarce resources to rehabilitate degraded land or to improve prime agricultural land; whether smallholder settlements will be able to support the expanding population better than large-scale mechanized farming; the threat from encroachment of urban development on to high quality agricultural land, and the most appropriate use of scarce water resources.

In its 1998 revision, the UN medium variant population projection indicates an increase of the world's population to about 8.9 billion by the year 2050 (United Nations, 1999), with a possible range between 7.3 and 10.7 billion. Most experts agree that with full and adequate application of modern agricultural technology, the world's land resources could provide sufficient food, fiber, animal feed, biofuel, and timber for such a population increase. In practice, however, there will most likely be acute land shortages in many countries, especially in developing ones (Fischer and Heilig, 1996).

Land has many functions besides the production of food and fiber. The principal functions include (FAO, 1995):

- Production (crop agriculture, animal husbandry, aquaculture, and inland fishery).
- Ecological (i.e., as the basis for terrestrial biodiversity).
- Climate regulation (source and sink of greenhouse gases, heat balance, and global hydrological cycle).
- Hydrologic (to regulate the flow of surface and groundwater resources).
- Storage (a storehouse for minerals and raw materials for human use).
- Waste and pollution control (to buffer and transform hazardous compounds).
- Living space (to provide the physical basis for human settlements, etc.).
- Archival or heritage (to store and protect the evidence of the cultural history of mankind).
- Connective space (to provide space for transport of produce and movement of people, animals, and plants).

The suitability and capability of land to provide these functions and services varies greatly over the world. Also, the importance attributed to these functions in the land use planning processes of individual countries varies with the socio-economic conditions and cultural traditions. The capability of land to provide for one or more of these functions is by no means static. It may be improved through human influences, but more often than not the land has been degraded by human action.

At the global level, land use and land cover change have been identified as being significant to a range of themes and issues central to the study of global environmental change. Alterations in the earth's surface hold major implications for the global radiation balance and energy fluxes, contribute to changes in biogeochemical cycles, alter hydrological cycles, and influence biodiversity and ecological complexity (Turner *et al*., 1995). Through these environmental impacts at local, regional, and global levels, land use and land cover changes driven by human activity have the potential to significantly affect food security and the sustainability of the world agricultural and forest product supply systems (Fischer *et al*., 1996). These diverse roles of land use have been recognized in a large number of research publications and international conferences, symposia, and workshops devoted to the subject over the past few years, as well as the United Nations' Agenda 21 (Keating and Shah, 1993).

In most developing countries, the socioeconomic needs of rapidly increasing populations are the main driving force in the allocation of land resources to various kinds of uses, with food production as the primary land use. Heavy population pressure and the resulting increased competition by different types of land users have emphasized the need for more effective land use planning and policies. Rational and sustainable land use is an issue of great concern to governments and to land users interested in preserving the land resources for the benefit of present and future populations.

Policymakers and land users face two basic challenges: (i) the need to reverse trends of land degradation in already cultivated areas by improving conditions and re-establishing their level of fertility; and (ii) to prevent the degradation of land resources in new development areas through appropriate and just allocation and use of these resources to maintain productivity and minimize soil erosion. In both cases an integrated approach to planning and management of land resources is a key factor to implementing solutions that will ensure that land is allocated to uses providing the greatest sustainable benefit, a principle anchored in Chapter 10 of the UNCED Agenda 21 (Keating and Shah, 1993).

Thus, the Food and Agriculture Organization of the United Nations (FAO) pointed out the need for comprehensive new approaches in land use and development planning (FAO, 1990):

> How people or nations use their land depends on complex, interrelated factors which include the characteristics of the land itself, economic factors, social, legal, and political constraints, and the needs and objectives of the land users. In order to make rational decisions, it is necessary to:
>
> - collect the right information about the physical, social, and economic aspects of the land area in question; and
> - assess the land's relative suitability for different uses in the light of the needs and objectives of the land user and the community.

Starting in 1976, the FAO, with the collaboration of the International Institute for Applied Systems Analysis (IIASA), has developed and applied the Agro-Ecological Zone (AEZ) methodology, supporting databases and software packages. The FAO has assisted member countries in finding rational solutions to various problems of land resources appraisal for planning sustainable agricultural development. This involves linking land use options with other development goals in such areas as food production, food self-sufficiency, cash crop requirements, population supporting capacity, issues of soil fertility constraints, soil erosion risks, and land degradation. The AEZ approach was first applied in a global study of Land Resources for Populations of the Future (FAO/IIASA/UNFPA, 1983), which focused on the determination of the ecological potential of land resources for food production and the appropriate policies for their management. Subsequently, the

AEZ methodology has been extended, refined, and utilized in national and subnational assessments of land productivity and population supporting capacity in various countries, such as Bangladesh, China, Mozambique, Nepal, Nigeria, the Philippines, and Thailand.

To further enhance the AEZ methodology for national and subnational applications, the FAO, together with the concurrence of the Kenyan Government and the participation of IIASA, undertook an AEZ case study of Kenya (FAO/IIASA, 1993). The Kenya study included various advances in AEZ procedures, such as: the linkage to Geographic Information Systems (GIS); various methodological refinements; and the development of socioeconomic and land resources databases. These enhancements are discussed in this chapter. They encompass a representation of multiple cropping systems, livestock and crop sector integration, assessment of fuelwood productivity, and recently, multi-criteria optimization of land allocation. The AEZ model analysis facilitates the integration of ecological, technological, social, demographic, and economic considerations for the appraisal of land resources of specific districts in Kenya for policy formulation and development planning.

12.2 Land Use Planning

Historical records, by reporting many failures at conflict resolution over land use and land tenure rights, provide ample evidence that land use planning is as old as humankind (Sombroek and Eger, 1996). The aim of land use planning is to create the preconditions for achieving a sustainable and environmentally sound, socially desirable, and economically appropriate form of land use.

In response to these articulated needs, the FAO developed a framework of principles and procedures that provide guidance to land use planners, and that can be adapted to local needs (FAO, 1993). The Guidelines provide an overview of the planning process in ten steps. These are not seen as a recipe for the planning process but rather as a flexible guide and starting point for more detailed local or national procedures. The implementation of these ten steps, although presented here in sequence, are considered to result in an iterative process:

1. Establish goals and ground rules for planning.
2. Prepare an outline of the plan.
3. Identify and structure the problems and opportunities.
4. Identify and design alternative and promising land use types.
5. Evaluate land suitability.
6. Appraise – for alternative land use types – the environmental, economic, and social impacts.
7. Choose the best achievable land use.

8. Plan for change; draw up a final version of the land use plan.
9. Implement the plan (e.g., by using agricultural consulting and tax incentives).
10. Review and revise the plan in view of implementation experience, stakeholders response, or new goals.

As will become clear in later sections, the powerful software tools discussed in this chapter are intended to support planners in the core tasks of planning, i.e., in the execution of steps 5 to 7.

In the FAO document, efficiency, equity, and sustainability of land use decisions are emphasized. It is well recognized that sustainability is a politically powerful concept (IIASA, 1992) with various, sometimes controversial, definitions put forward. One such definition, published by the National Research Council, postulates that sustainable development "should include management of the use of a resource so it can meet human demands of the present generation without decreasing opportunities for future generations" (NRC, 1991). Although intuitively clear in its meaning, the application of such a definition in the selection of concrete development strategies requires quantitative approaches and is difficult because of the complexity and numerous interactions – in space and time – in managed and unmanaged ecosystems.

As stated by the FAO (FAO, 1990), principles to be adopted for promoting sustainable land use include:

- Managing the land in order to maintain or improve its productive capacity (i.e., to protect against erosion, to maintain the nutrient status by adequate use of manure or fertilizer, and to safely dispose of toxic wastes).
- Assessing and preparing for predictable hazards – natural hazards such as drought, floods, and landslides cannot be completely avoided but the risks can be reduced, e.g., by water supply and flood protection schemes and by avoiding risky sites.
- Minimizing the loss of productive land: prime farmland and land of unique value (for any desirable land use) should be conserved.

Change, be it for demographic, socioeconomic, or environmental reasons, creates conflicts between competing uses of resources, in particular land, and between the interests of individual land users and the common good. For instance, development of new farmland often competes with the preservation of wetlands or forest ecosystems; urbanization and industrialization may irreversibly convert valuable agricultural land.

The process of land use planning and its implementation hinges on three elements (FAO, 1995):

- The stakeholders involved in or affected by managing the land units.
- The qualities or limitations of each component of the land units being planned.
- The viable land use options considered and available in the area.

In a more technical sense, the factors of planning are:

- The amount of land available and its tenure; the quality, potential productivity, and suitability of the land.
- The level of technology used to exploit the land resources.
- The demographic conditions, and the needs and standards of living of the people.

Each of these factors interacts with the others.

Decision making in agriculture, at the local level by farmers and by policy makers at the regional/national level, starts from an assessment of production options. Assessment of agricultural production has traditionally been classified along two lines. One approach is followed in the biological and physical sciences and is concerned with the production potential of land, usually without too much concern regarding the socioeconomic constraints and dynamic aspects of realizing this potential. The second approach is supported by social and economic sciences and concentrates on behavioral aspects of farmers or authorities. Resources such as land or water are briefly described, with only a very elementary understanding of their physical production potential. Both approaches have their merits, but each gives only partial explanations of current agricultural practices or policy implications.

There is an urgent need for formalized methods that will bring together these disciplines; evidence suggests that increasing pressure on land has resulted in the implementation of agricultural production systems that go far beyond the land carrying capacity. Recommendations of alternative sustainable land uses that imply changes in actual agricultural production systems should prevent land degradation or environmental damage and also be an improvement in terms of the land user's economic viability and performance (Antoine *et al*., 1997; Fischer *et al*., 1998).

Basic principles to be observed in sound land evaluation were first put forward in the 1970s. In 1976, the FAO published *A Framework for Land Evaluation* (FAO, 1976). The Framework defines land units in terms of their characteristics (measurable factors such as slope, soil texture, rainfall, etc.), and qualities (effects such as temperature regime and moisture availability, which result from a combination of characteristics), matches them with potential uses in terms of the requirements of such uses, and then rates the land in terms of suitability for the use. A use cannot be rated as suitable unless it is sustainable. The Framework, and a number of subsequent publications, provide fairly exhaustive lists of land characteristics and land qualities (FAO, 1984, 1985, 1990).

These lists were the basis of further development of the AEZ methodology. This methodology was applied to assess the food production potential of lands in the developing world (FAO/IIASA/UNFPA, 1983). While this assessment was severely limited by the availability and quality of data and the rather poor capacity, at that time, to store and process spatial data sets by computer, the importance of such work for development was recognized by the 1983 FAO Conference. Consequently, the approach was further developed in a case study concerned with the development and implementation of a national level methodology for the determination of land use potentials in Kenya, as a tool for policy formulation and development planning (FAO/IIASA, 1993). The specific role of this policy tool may be defined as assisting in the planning of sectors and regions, bridging the gap between conventional macro-planning and specific project planning.

The AEZ methodology uses a land resources inventory to assess, for a given level of input, all feasible agricultural land use options as well as expected production of relevant and agro-ecologically feasible cropping activities. On the basis of this agronomic evaluation, and using specific socioeconomic data to specify constraints, targets, and production options, the attainment of spatial resource allocation objectives can be optimized. The optimization results provide perspectives on the capability of Kenya's land resources, technology, and policy, etc., to improve as well as sustain agricultural production. These perspectives are intended to provide a useful guide to national planning.

Sustaining food and cash-crop production is a major agricultural development policy of the Kenyan government as set out in various government documents (see e.g., Republic of Kenya, 1986, 1994a, 1994b). This policy recognizes the importance of the agricultural sector, which in 1992 accounted for 27% of Kenya's total gross domestic product (GDP), 81% of total employment in the economy, and also earned a substantial amount of foreign exchange. The agricultural development policy of Kenya defines specific objectives and targets that include, but are not limited to (Onyeji *et al*., 1996):

- Growth of agricultural GDP.
- Increasing foreign exchange earnings.
- Enhancing food security and nutritional status.
- Increasing employment and revenue generation.
- Combating rural poverty and ensuring regional equity.
- Increasing farm incomes.
- Improving resource conservation.

While some of these objectives are synergetic, several of them are intrinsically antagonistic. Therefore, it is important to understand what level of attainment of the key objectives can be reached, and what the trade-off between them is. As task manager for Chapter 10 of UNCED Agenda 21, the FAO promotes the integrated planning and management of land resources in cooperation with regional institutions and individual countries as well as land users. Reaching a consensus on land use should be a major objective in the conceptualization of decision support systems (DSSs) for sustainable land use.

Feasible "real world" solutions are compromise solutions, resulting from a trade-off between various conflicting objectives, thus not maximizing single objectives, but finding an efficient and acceptable balance between the requirements of the stakeholders in the land and resources availability. Different kinds of objectives may need to be included, expressing not only economic values of land products but also addressing goals that cannot always be expressed in monetary terms – such as biodiversity, people's preferences, equity, or minimizing risk and uncertainty.

Decision making in land use also involves the consideration of a number of goals that cannot be aggregated into a single criterion to be used as a performance measure for ranking alternatives. Usually models may have to be run a number of times in order to identify a best, or even acceptable, solution. The elements of a solution are not fixed to a specific value, but are variable within certain ranges determined by resource availability and socioeconomic realities. Many options need to be examined in order to generate the information and knowledge required for these decisions and to quantify and display the trade-offs between conflicting objectives.

12.3 A Summary of the AEZ Methodology

Based on the concepts and needs articulated in the previous sections, FAO has developed a methodological framework for assessments of land productivity designed for use in agricultural development planning and natural resources management. AEZ involves the inventory and classification of the land resources in a way meaningful for assessments of the potential of agricultural production systems. This characterization of land resources includes components of climate, soils, and landform, basic for the supply of water, energy, nutrients, and physical support to plants.

Crops require heat, light, and water in varying amounts. Temperature, water, and solar radiation are key climatic parameters that condition the geographic distribution and net photosynthesis and allow crops to accumulate dry matter. Crops have specific temperature requirements for their growth and development, and prevailing temperatures set the limits of crop performance when moisture (and radiation) requirements are met. Vice versa, when temperature requirements are met, the growth of a crop is largely dependent on how well the length of its growth cycle matches the period when water is available. Crop performance depends additionally on the availability of nutrients in the soil, the capacity to store water, and mechanical support for crops. Therefore, agro-ecological zoning also includes an inventory of relevant soil and landform characteristics, and an edaphic assessment in relation to the crops studied.

Technical specifications (including management) within a socioeconomic setting under which a specific crop is grown have been defined as land utilization types (LUT). Crop suitability assessments, in essence, are based on matching of crop specific adaptability characteristics and crop/LUT ecological requirements with attributes of the components of individual land units.

The AEZ suitability assessment is foremost an environmental approach that provides an explicit geographic dimension for establishing spatial inventories and databases on land resources and crop production potential. The data requirements are sufficiently limited and it uses readily available data to the maximum. Moreover, it is comprehensive in terms of coverage of factors affecting agricultural production. The approach has also proven to be relevant for assessments of potential agricultural responses to scenarios of climate change (Fischer and van Velthuizen, 1996).

12.3.1 The land resources database

The land resources inventory brings together several layers of information on physical environmental resources and allows the creation of unique ecological land units (termed agro-ecological cells) within which landform, soil, and climate conditions are quantified and considered nearly homogeneous.

The climatic resource inventory of Kenya records both temperature and soil moisture conditions. The quantification of temperature attributes has been achieved by defining reference thermal zones. As temperature seasonality effects of latitude are minor, temperature zones are closely correlated to altitude ranges (Braun, 1982). To cater for differences in temperature adaptability of crops, pasture, and fuelwood species, nine thermal zones have been identified, based on ranges of 2.5°C.

Quantification of moisture conditions was achieved through the concept of reference length of growing period (LGP). Reference LGP is defined as duration (in days) of the period when temperature permits crop growth and soil moisture supply exceeds half the potential evapotranspiration; it includes the time required to evapotranspire up to 100 mm of soil moisture storage. The moisture period regime has been inventoried by means of three complementary attributes:

- Number of separate LGPs within a year, summarized as a historical profile of pattern of length of growing periods per year (LGP-pattern).
- The mean total dominant length of growing period, i.e., the sum of the mean dominant and associated length of growing periods occurring during the year.
- Year-to-year variability of moisture conditions.

The original AEZ soil and terrain resources inventory for Kenya (FAO/IIASA, 1991) was based on the 1:1 million scale *Exploratory Soil Map of Kenya* (Sombroek *et al*., 1982). This information, in particular the soil association composition database, has been updated at the Kenyan Agricultural Research Institute (KARI) in the form of the KENSOTER project (Kenya Soil Survey, 1995). The soil classification follows the revised legend of the FAO soil map of the world (FAO, 1988). About 400 different soil map units are distinguished, describing soil associations or soil complexes that are composed of dominant soils, associated soils, and inclusions, or related to water bodies and major urban areas. A mapping unit composition table contains the percentage allocation of the map units by soil type, slope class, soil texture, and soil phases. It also contains information derived from the legend of the soil map regarding landform and geology/parent material.

In addition to the soil and climate layers, seven other layers of information have been incorporated in the land resources database, providing information on cash-crop zones, forest zones, parkland areas, location of irrigation schemes, tse-tse infestation areas, administrative boundaries, and a digital elevation model. The individual map layers were digitized and stored in a grid (raster) format of 1085 rows and 900 columns, each grid cell representing an area of one square kilometer. The land resources inventory combines both geo-referenced information as provided in the different map overlays and statistical information (percentage distribution) as contained in the soil mapping unit composition and slope composition tables. The compilation of the resource inventory includes two main steps: (i) overlaying of map layers and creation of a GIS database file; and (ii) application of soil mapping unit composition and slope composition tables.

The Kenyan land resources database has been used to assess land productivity and crop production potential of the country. Analysis has been carried out for a large number of land utilization types,[1] including crops, pastures, and fuelwood species, and separately for three specified levels of technology (FAO/IIASA, 1993). A total of 25 crop species are included in the assessment, divided into 64 crop types

[1]Practically all food crops shown in the official statistics to be of importance in Kenyan agriculture and several major cash crops are included in the assessment.

to account for differences in ecotype adaptation and growth cycle within each crop species. In addition, pastures – comprising some 30 grass and legume species – and two classes of fuelwood species – with and without nitrogen fixing ability – are accounted for in the resource appraisal. Three steps were involved in the estimation of production potential: (i) crop suitability assessment; (ii) land productivity assessment; and (iii) estimation of extents of land with cultivation potential and determination of its production potential.

Databases generated during steps (i) to (iii) provide important information for tackling more complex district planning tasks involving food production targets, crop and livestock interaction, cash-crop production objectives, fuelwood requirements, and the need to preserve forest and wildlife resources, etc.

12.3.2 Crop suitability assessment

Crops have climatic requirements for photosynthesis and phenology, both of which bear a relationship to yield. In the AEZ methodology, crops are classified into five climatic adaptability groups according to distinct photosynthetic characteristics.

For example, barley, oat, wheat, phaseolus beans, and white potato have a C3 photosynthesis pathway and belong to adaptability group I. They are adapted to operate under cool conditions in locations with less than 20°C annual mean daily temperature, i.e., an altitude above about 1,500 meters under Kenyan conditions. Annual mean daily temperatures below 12.5°C are assumed to pose a risk of frost damage too great for successful cultivation of these crops.

Thermal requirements for each of the 64 crop types, for pastures, and fuelwood species have been rated in relation to inventoried thermal zones in Kenya. Five suitability classes are employed, i.e., classes S1 to S4 and N. A rating of S1 indicates that the temperature conditions for growth and yield physiology and phenological development are optimal and that thermal conditions do not hinder achievement of maximum yield potential.[2] Ratings of S2, S3, and S4 indicate that temperature conditions for crop growth and development are suboptimal, resulting in the suppression of attainable yields by some 25, 50, and 75%, respectively. A rating of N indicates that prevailing temperatures are not suitable for the cultivation of that crop.

Potential yields for crops, pastures, and fuelwood species by LGP zone were then adjusted considering moisture-related constraints of water stress, weeds, pests and diseases, and workability. Yields refer to single crops that act as building blocks in the formulation of annual cropping patterns and crop rotations. Annual crops are

[2]Agronomically attainable yield potential from an agro-climatic viewpoint, i.e., on suitable soils and terrain in suitable thermal zones. Maximum yield potential was determined with a simple crop biomass production model, see Appendix 1 in Fischer and van Velthuizen (1996).

matched to the individual component length of growing periods (in the case of multiple LGPs per year), and perennial crops are evaluated with respect to the average total length of growing period. LGP-pattern evaluation also takes into account the probability weights associated with the historical pattern of occurrence of length of growing periods. This provides a measure of the variability of yields, quantifying potential yields under average, above-average, and below-average moisture conditions.

Many soil characteristics, e.g., natural fertility, salinity, pH, and gypsum content, can be defined in a range that is optimal for a given crop, a range that is critical, and a range that is unsuitable under present technology. These relationships (FAO, 1981) have been applied in the matching of the inventoried soil units with the soil requirements of crops. The soil unit rating relates soil properties to the degree to which crop requirements can be met under a given management practice. Where appropriate, the soil unit rating is modified according to limitations implied by texture evaluation, stoniness, and soil phase evaluation.

Limitations imposed by slope are taken into account in a threefold manner:

- Each mapping unit is split up with regard to the inventoried slope class into uniform slope ranges according to a slope composition table. Seven slope ranges have been considered as appropriate for LUT corresponding to: 0–2%, 2–5%, 5–8%, 8–16%, 16–30%, 30–45%, and $>$45%.
- A slope-cultivation association screen defines those slope ranges that are permissible for cultivation in relation to crop type and management level.
- Land productivity assessment also involves the calculation of potential topsoil loss due to water erosion, as well as relating topsoil loss to productivity loss.

In each land unit, topsoil loss is estimated for all relevant LUTs, i.e., all applicable crop combinations, pastures and fuelwood species, and levels of inputs, using a modified[3] Universal Soil Loss Equation, or USLE (Wischmeier and Smith, 1978). In the formulation of the USLE, length and steepness of the slope are important explanatory variables in estimating topsoil loss. Yield loss is related to the estimated topsoil loss through a set of linear relationships accounting for soil properties (susceptibility to erosion), regeneration capacity of topsoil (depending on climatic conditions), and level of management.

[3]A combined factor, as used in the original formulation of the USLE, accounting for the effects of vegetation cover and management techniques, was split into two separate terms, a crop cover factor and a management factor.

12.3.3 Land productivity assessment

For estimating land productivity potential, further consideration has to be given to including multiple cropping, fallow requirements to maintain soil fertility, and production security constraints to reflect risk aversion of subsistence farming under climatic uncertainty.[4] Multiple cropping refers to the intensification of arable land use, both in time and space. The principles of yield increases resulting from a better use of time with crops in sequence is complementary to increases arising from a more efficient use of space with crops in mixture.

Sequential cropping is possible in areas where climatic conditions, temperature, and moisture supply permit crop growth beyond the duration of one crop. This mostly applies to areas in Kenya with an average total length of growing period above 210 days. The algorithms implemented for land productivity assessment explicitly construct and evaluate all feasible sequential crop combinations, both monoculture and multiculture, by matching individual crop cycle requirements to the relevant component length of growing periods as implied by the LGP pattern.

Intercropping increments, i.e., yield advantages from practices like strip, alley, and relay cropping, are assumed to increase with length of growing period. No yield advantage is adjudged in LGPs of less than 120 days (Kassam, 1980). Also, it is assumed that advantages would be largest with crop mixtures where the individual component crops are rated as very suitable. The relative contribution of intercropping is assumed to be most pronounced under a low level of inputs. No increment is applied under a high level of inputs, as mechanization requirements associated with this management level are generally regarded as being incompatible with intercropping practices. Also, intercropping increments were not applied to some crops, such as wetland rice, sugarcane, banana, oil palm, pastures, and fuelwood species.

AEZ assessment is concerned with sustainable agricultural practices. In their natural state, many soils cannot be continuously cultivated without experiencing degradation, manifesting itself in decreased crop yields and deterioration of physico-chemical soil properties. The assumptions on fallow requirements incorporated into the model are formulated for four main groups of crops – cereals, legumes, roots and tubers, banana and sugarcane – and relate to soil unit, thermal zone, and moisture regime (FAO/IIASA, 1993).

[4]For instance, it is stipulated that subsistence farmers select LUTs having a high probability of succeeding also in "bad" years, even at the expense of lowering average expected production.

12.4 The AEZ Land Use Allocation Model

A detailed specification of the AEZ core model for land use allocation is far beyond the scope of this book. Therefore only an outline of the essential features of the model is provided here (for more information on the model specification, refer to Fischer and Antoine, 1994a).

As outlined in Section 12.3, each agro-ecological cell was assessed in terms of all feasible agricultural land use options of interest in the analysis. At a given level of input, the productivity assessment records expected production of relevant and agro-ecologically feasible cropping activities, in terms of main produce as well as various by-products (e.g., crop residues and by-products), extents by suitability class, input requirements, and degradation hazard (i.e., potential soil and productivity loss due to water erosion). An application of the AEZ framework constructs an inventory of potential agricultural land use options with relevant quantitative information. Such an inventory is essential for devising optimal land use patterns that take into account physical, socioeconomic, technological, and political objectives and constraints.

With the implementation of multi-criteria decision support tools, searching for optimal land use is not limited to optimization of a single-objective goal function over a set of constraints. Instead, a user of the model can examine various trade-offs between several objectives within the given set of constraints. In the Kenya study, the criteria are selected out of the set of outcome variables (cf. Section 12.4.3). The constraints defining the core model are linear (see Section 12.4.1). Therefore the resulting optimization problem is linear and a reliable and fast interior point method solver makes it possible to analyze large-scale problems that can arise in this context, with several thousand decision variables and constraints.

The land allocation model has been developed for integrating livestock, crop, and fuelwood production sectors within the AEZ framework. As with any model of this kind, the formulation is revised and improved as new insights, needs, or new quantified information becomes available. The strength of the approach lies in its extensive and consistent use of spatial information for assessing agricultural land use options within the context of district development planning. This assessment considers simultaneously several objectives such as maximizing revenues from crop and livestock production, maximizing food output, maximizing district self-reliance in agricultural production, and minimizing environmental damages from erosion.

12.4.1 The core model constraint set

A formal model specification is not presented here owing to space limitations. Instead, various sets of constraints are summarized below in a descriptive manner.

A realistic assessment requires a thorough description of relevant constraints to be considered in the selection of optimal land use. These can relate to technological conditions, physical limitations, social, institutional and economic constraints, and political targets. In the following, we briefly discuss the set of constraints that has been implemented in the Kenya study. Not all the constraints need to be activated in every scenario, but can be included when appropriate and relevant.

Demand targets by aggregate commodity group. Lower and/or upper bounds or equality constraints on food availability, specified by broad commodity groups (e.g., cereals, pulses, roots, meats), can be used to satisfy food demand targets from domestic production and imports. The user can either supply absolute levels of target demand or have demand targets constructed by the core model generator from per capita demand targets and demographic information.

Commodity production targets. Lower and/or upper bounds or equality constraints on individual commodity production (e.g., wheat, white potato, beef) can be selected to achieve appropriate commodity bundles in the production plan. This, for instance, could be an appropriate device to enforce sufficient production of cash crops in food maximizing scenarios.

Limits on harvested area. The harvested area defined by a broad commodity group (e.g., cereals, pulses, roots) can be controlled by means of lower and/or upper bounds and equality constraints implemented at district level. This can be useful to ensure a desired allocation of land to cash crops or fuelwood production.

Crop-specific land use constraints. Lower and/or upper bounds and equality constraints have been implemented to limit crop-specific use of arable land resources.

Total arable land use constraint. Lower and/or upper bounds or equality constraints on total arable land use by broad climatic zone and/or district serve to reflect considerations regarding land use other than for agricultural production purposes (e.g., forest areas, specific nonagricultural uses).

Production input requirements. These constraints are associated with the quantification of production inputs required according to the specified level of technology. Input requirements are derived from a technology matrix by interpolation; i.e., from a set of tabular functions that relate, for each crop and livestock system, different yield levels to input requirements in terms of seed (traditional and/or improved), fertilizer (N, P, and K), power, and plant protection/veterinary inputs. In addition, labor required for soil conservation measures is quantified. This set of constraints can be applied to ensure that input requirements for crop and livestock production fall within the limits of the available resources in terms of relevant input categories (e.g., labor, capital, fertilizer, power). Negative input–output coefficients are used for activities that generate resources (e.g., power from animals).

Crop-mix constraints. A set of constraints, to be specified optionally either by broad climatic zones or by agro-ecological zone, can be used to exercise control over cropping patterns by enforcing limitations on shares (minimum and maximum levels) of arable land use to be occupied by individual crop groups. The level of enforcement for this set of constraints is controlled by the selection of scenario parameters.

Human calorie/protein ratio requirements. These constraints ensure that, by broad climatic zones, the crop production plan is such that the ratio of calories to protein obtained from food products stays within nutritionally acceptable ranges.

Distribution of livestock population over livestock zones. The concept of livestock zones relates the climatic information contained in the resource inventory to broader climatic zones relevant to describing and delineating different livestock systems and formulating their integration with the crop production plans of the respective agro-ecological zones. Sixteen livestock zones are distinguished in the Kenya study. The livestock population distribution constraints allow lower and/or upper bounds to be imposed or equality constraints on shares in total livestock populations (herd tropical livestock units, or TLUs) to be considered in each of the livestock zones.

Distribution of livestock systems. This set of constraints affects the composition of the supported livestock population within each livestock zone in terms of different livestock systems. This is done by imposing lower and/or upper bounds or equality constraints on the shares of individual livestock systems in the total number of livestock units supported in the zone. In the Kenya study, up to 10 livestock systems, out of a total of some 30, at traditional, intermediate, and improved management levels, have been considered in each livestock zone: this includes pastoral production systems of camel, cattle, and sheep and goats, and sedentary production systems of cattle, sheep and goats, pigs and poultry.

Constraints on number of animals. Lower and upper bounds on the number of TLUs per livestock system can be specified to guide the selection and allocation of livestock systems.

Livestock feed requirement constraints. When setting up feed demand–supply balance constraints, it is important to include relevant aspects of quality and quantity of feed supplies in time and space. In the Kenya study, livestock feed balance constraints are formulated by individual livestock zones. The required feed supply to support livestock populations has to be provided from feed sources within each livestock zone, i.e., crop by-products and residues, pastures and browse, fallow grazing, browse from fuelwood trees, and – in some scenarios – primary products. Each set of constraints, by zone, is formulated in terms of four items: minimum and maximum daily dry matter intake, digestible protein of feed ration, and metabolizable energy.

Since the seasonal variation in quality and quantity of feed supplies often plays a critical role for livestock breeding in pastoral areas, two feeding periods within the year have been distinguished: the wet season and the dry season. The length of each period in a particular agro-ecological cell varies according to the climatic information in the land resources inventory. It is assumed that the length of the wet season equals the site-specific length of growing period.

Zone level production risk constraint. The AEZ land resources inventory of Kenya includes information on the variability of rainfall, and hence, the varying length and type of the growing period. This allows for assessing production options in terms of good, average, and bad years. While valuations used in the objective function usually refer to average productivity, zone level risk constraints are implemented to ensure that the resulting land allocation emphasizes the stability of the production plan also in bad years (i.e., in vulnerable areas to give preference to crop combinations that will produce also in bad years, even at the expense of lower average output).

Cell use consistency constraint. It is necessary to explicitly impose that the sum of shares allocated to different crop production activities in each land unit does not exceed 100%, i.e., that each piece of land can only be used and allocated once (this does not preclude sequential multi-cropping). Unlike the constraints described above that are independent of the number of records in the land inventory, the cell use consistency constraint has to be imposed whenever more than one cropping activity is feasible in a particular agro-ecological cell. As a consequence, the number of rows in the constraints matrix might become large.

Crop rotation constraints. Continued monocropping over time is not considered a sustainable agricultural practice under most circumstances as it exhausts soil fertility more easily and may cause pest and disease problems. Although the AEZ land use allocation model is essentially static, not explicitly considering crop rotations over time, this element has been captured by imposing upper limits on the share of each cell that can be occupied by an individual crop activity. For example, imposing a limit of 70% as the maximum share for maize in a particular cell can be interpreted as requiring that maize cannot be grown in more than 7 out of 10 cropping years, in addition to fallow requirements.

The monocropping restrictions are controlled through scenario parameters and are implemented as simple lower and upper bounds on cropping activities. They are not enforced in a cell when no alternative cropping options exist. Also, cassava and perennial crops like banana, oil-palm or sugar cane, or environmentally less demanding land uses, such as crop combinations including legumes, or pastures and fuelwood species, are not restricted by monocropping constraints.

Cell level production risk constraints. As outlined above for the zone level, crop production risk constraints are also implemented at the cell level to ensure that the resulting land allocation emphasizes the stability of the production plan in bad years. The constraint is specified such that land use options selected in the optimal solution should provide output levels in “bad” years that do not fall below a user-specified threshold level in comparison to the best possible output obtainable in bad years among all viable cropping options.

Environmental impact constraints. These constraints are included to ensure that the optimal production plans are also environmentally compatible, demanding that the environmental impacts in each cell must not exceed tolerable limits. At this stage, only soil degradation from water erosion is quantified. Tolerable soil loss is dealt with by filtering out unacceptable crop combinations rather than imposing inequalities in the constraints matrix.

12.4.2 Decision variables

The core model contains three groups of decision variables that determine optimal agricultural land use, livestock numbers supported, and optimal allocation of feed supplies to different livestock systems, respectively. These three groups are as follows:

- The land use shares, i.e., the share of agro-ecological cell j allocated to a cropping, grassland, or fuelwood activity k.
- The number of animal units of livestock system s kept in zone z.
- The feed ration of feed item h from crop i allocated to livestock system s in period t in zone z.

These variables form the columns of the constraint matrix, the core model activity set. Values of these variables are provided by the solver as the result of solving a parametric optimization problem that is automatically generated in order to compute a Pareto efficient solution corresponding to preferences that are interactively specified by a user.

Values of decision variables and of criteria can be inspected by the user and are then used for generating various reports by the software developed for the AEZ model. It is important to note that the linkage to the GIS is maintained while generating a scenario (see Section 12.5.1) as well as during the analysis of scenario results.

12.4.3 Outcome variables

Typically, six to eight variables are interactively selected from the set of outcome variables (defined in the core model outlined in Section 12.4.1) to serve as criteria in multi-criteria analysis of the AEZ model. The following outcome variables are defined in the Kenya study:

1. Maximize food output (weighted sum of food energy and protein available for human beings after conversion and processing into food commodities).
2. Maximize net revenue.
3. Minimize production costs.
4. Maximize gross value of output.
5. Minimize weighted sum of arable land use (weight of 1 assigned to crops and fuelwood species and 0.1 to grassland).
6. Minimize area harvested.
7. Maximize food output in bad years (weighted sum of food energy and protein available for human consumption as in outcome 1 above, but evaluated for climatic conditions typical for years with low precipitation levels).
8. Minimize total erosion (total soil loss over all land units).
9. Maximize district self-reliance (defined as minimum of the individual commodity group self-sufficiency ratios, i.e., target production over demand achieved).
10. Minimize erosion at the level of agro-ecological cells (largest soil loss per ha occurring in any used land unit).

The last criterion provides an example of an objective that reflects the spatial detail of the GIS resource database. Other examples of criteria where the spatial content of the information is important could, for instance, express crop diversification or equity of expected farm incomes.

The above-listed ten most frequently used outcome variables are predefined in a configuration file in order to allow for easy selection of criteria by the user (refer

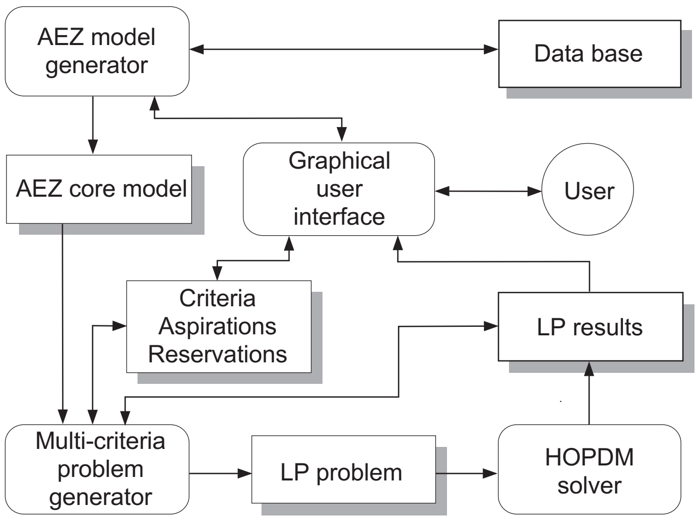


Figure 12.1. The structure of a DSS for land resources assessment and agricultural development planning.

to *Figure 12.9*). However, a more experienced user can interactively select other variables as criteria (refer to *Figure 12.8* for an example of this).

12.5 DSS for Land Resources Assessment

This section describes the structure of the DSS used for land resource assessment and the functionality of its components.

The functional structure of the AEZWIN DSS is illustrated in *Figure 12.1*. AEZWIN is an acronym for AEZ for Windows, where AEZ is traditionally used for the applied methodology of land resources assessment outlined in previous sections and described in more detail by Fischer and Antoine (1994a). Note that the structure of AEZWIN DSS illustrated in *Figure 12.1* is the same as the structure of the DSS for the water quality management discussed earlier in Chapters 3 and 10 and later in more detail in Chapter 11. Apart from some technical issues (analysis of the AEZ model requires the solution of LP problems, whereas the water model is of a mixed integer programming (MIP) type, therefore different solvers are used) the only and obvious difference between these two DSSs is due to their respective core models. This is a strong argument for using a modular structure of a DSS, as discussed in Chapter 3.

In order to provide the user with a uniform interface for the generation and analysis of a scenario of the AEZ model, a specialized application, called AEZWIN, has been developed. Two modules of AEZWIN are presented in the following subsections, namely the AEZ model generator and several illustrative elements of the user interface. The remaining parts of the AEZWIN DSS illustrated in *Figure 12.1* are hidden from a user. They include the following steps:

- The multi-criteria problem is generated and is converted into a single-criterion parametric problem (see Makowski, 1994c, for details).
- The corresponding single-criterion is generated in the LP_DIT format (see Makowski, 1994b and forthcoming, for background and documentation, respectively).
- An efficient LP solver is provided for computing Pareto efficient solutions. The solver, called HOPDM, based on the Interior Point Method (see Chapter 7 and paper by Gondzio and Makowski, 1995a, for details) makes it possible to interactively solve medium-scale LP problems on a PC.

Hence, from the user's point of view, an instance of a multi-criteria optimization problem (that provides as an optimal solution a Pareto efficient solution corresponding to the preferences specified by the user) is generated and solved automatically.

12.5.1 AEZ model generator

Various approaches might be applied for the generation of diverse models for decision support. Some models (e.g., the RWQM model used in Chapter 11) are generated by specialized software that does not require any interaction with the user. However, such interaction is required for the generation of a scenario for the AEZ model. Additionally, the generation of a scenario for the AEZ model requires a selection of various options and processing of data that are pertinent to selected options and that are needed for a corresponding instance of the AEZ model. A sophisticated system of programs (Fischer and Antoine, 1994b) has been developed for this purpose. This software served originally for the application of linear optimization techniques for analyzing land use scenarios with regard to single-objective functions, such as maximizing agricultural production or minimizing the cost of production under specific physical environmental and socioeconomic conditions and constraints. However, this software is equally useful for specifying scenarios in the multi-objective case.

A detailed presentation of all functions of AEZWIN is beyond the scope of this book. *Figure 12.2* gives a general overview of the flow and integration of information implemented in the AEZ, whose methodology is summarized in Section 12.3. In order to illustrate the functionality of this DSS, we present only a list of items available from the main menu of AEZWIN and an example of several dialogs. Detailed documentation of AEZ is given by Fischer *et al*. (1998).

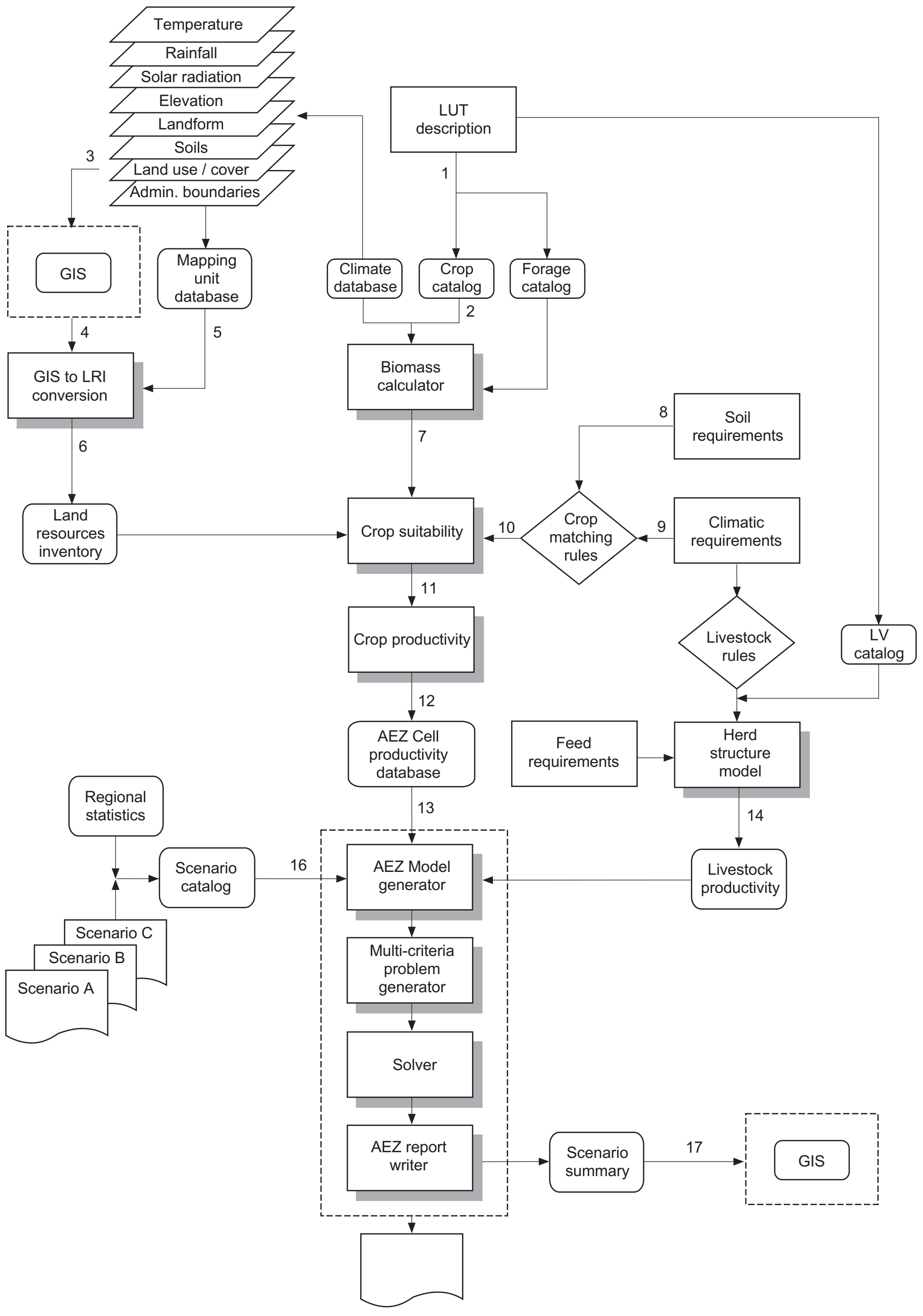
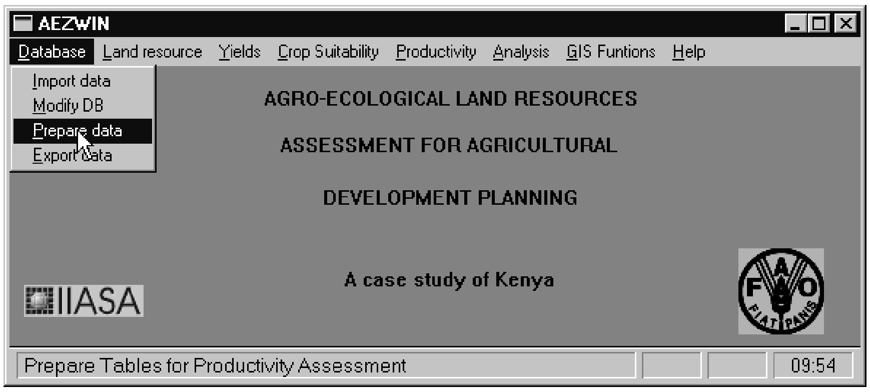


Figure 12.2. AEZ information flow and integration.



AGRO-ECOLOGICAL LAND RESOURCES

ASSESSMENT FOR AGRICULTURAL

DEVELOPMENT PLANNING

A case study of Kenya

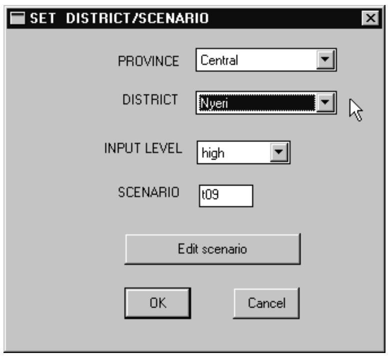
Prepare Tables for Productivity Assessment

Figure 12.3. Main menu of AEZWIN.

The main menu of AEZWIN (see *Figure 12.3*) contains the following eight menu items:

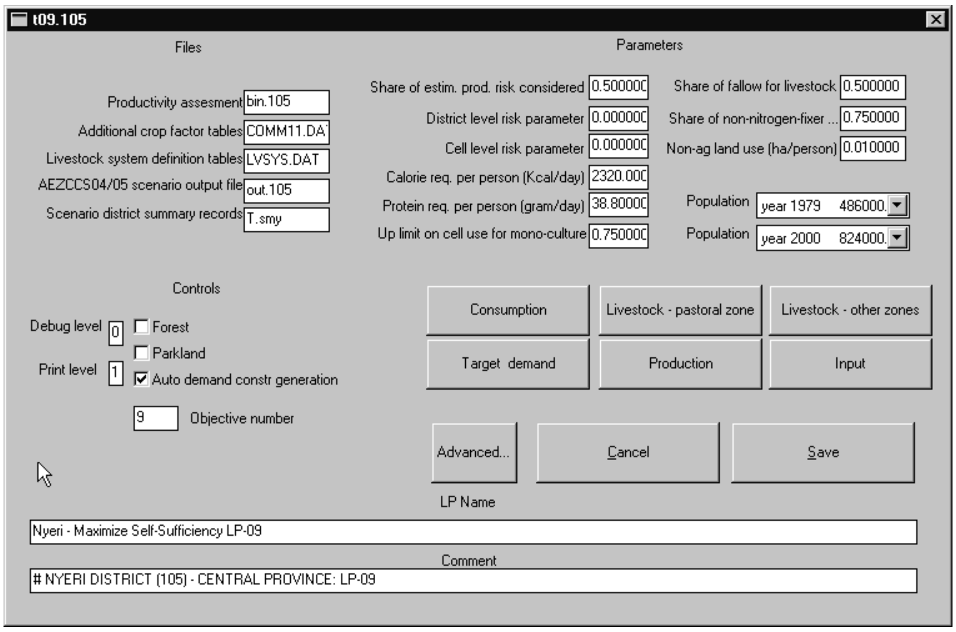
1. *Database*: To import, export, or modify records in the AEZ database, other than a land inventory.
2. *Land resources*: To view inventory files and to calculate, view, and print various statistics from the land resources inventory.
3. *Yields*: To generate maximum attainable yields by agro-climatic zone.
4. *Crop suitability*: To run a single crop suitability assessment and determine the potential arable land for a selected district.
5. *Productivity*: To construct for each agro-ecological cell feasible multiple crop combinations, evaluate crop production options, and filter out the best alternatives for later consideration in district analysis.
6. *Analysis*: To select a district for analysis, generate the AEZ core model specification file, call the LP-solver or perform the multi-criteria model analysis by the MCMA, create reports of district planning scenarios, generate an LP_DIT file (see Chapter 10 and the next section for a more detailed description of the MCMA).
7. *GIS functions*: To display compressed raster maps and to (optionally) transfer control to a GIS system.
8. *Help*: To activate on-line documentation of AEZWIN (which also includes a detailed tutorial on using AEZWIN).

The Analysis menu provides the possibility to select a scenario through a dialog illustrated in *Figure 12.4*. Such a scenario may be defined by experienced users in a traditional way by editing configuration files. However, an interactive module for editing scenarios is available in AEZWIN. The corresponding dialog (that can be



PROVINCE Central
DISTRICT Nyeri
INPUT LEVEL high
SCENARIO t09

Figure 12.4. Dialog for selecting a district and a scenario.



Productivity assesment bin.105
Additional crop factor tables COMM11.DA
Livestock system definition tables LVSYS.DAT
AEZCCS04/05 scenario output file out.105
Scenario district summary records T.smy

Share of estim. prod. risk considered 0.50000C
District level risk parameter 0.00000C
Cell level risk parameter 0.00000C
Calorie req. per person (Kcal/day) 2320.00C
Protein req. per person (gram/day) 38.8000C
Up limit on cell use for mono-culture 0.75000C

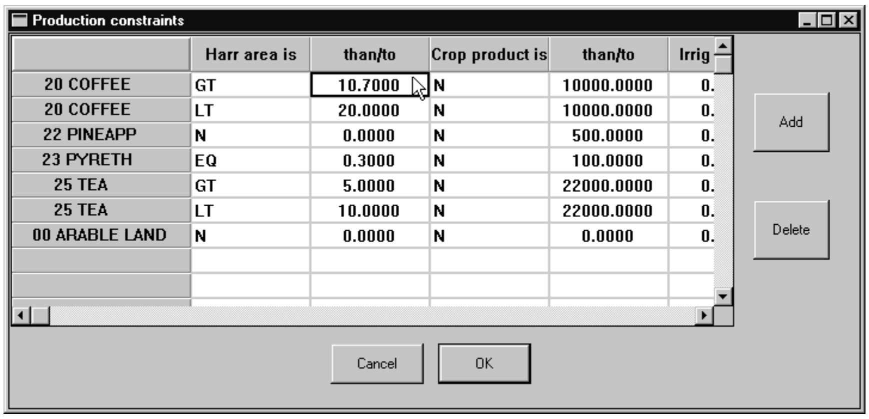
Share of fallow for livestock 0.500000
Share of non-nitrogen-fixer ... 0.750000
Non-ag land use (ha/person) 0.010000
Population year 1979 486000.
Population year 2000 824000.

Debug level 0
Print level 1
☐ Forest
☐ Parkland
☑ Auto demand constr generation
9 Objective number

LP Name
Nyeri - Maximize Self-Sufficiency LP-09
Comment
NYERI DISTRICT (105) - CENTRAL PROVINCE: LP-09

Figure 12.5. Edit scenario dialog.

activated by pressing the Edit scenario button in the dialog shown in *Figure 12.4*) is illustrated in *Figure 12.5* and provides a summary of selections that the user can use to define a particular scenario. *Figure 12.6* shows an example for entering (or modifying) production targets that is available after selecting the Production button in the Editing scenario dialog. Note that production targets can be specified for either or both output level and acreage.



	Harr area is	than/to	Crop product is	than/to	Irrig
20 COFFEE	GT	10.7000	N	10000.0000	0.
20 COFFEE	LT	20.0000	N	10000.0000	0.
22 PINEAPP	N	0.0000	N	500.0000	0.
23 PYRETH	EQ	0.3000	N	100.0000	0.
25 TEA	GT	5.0000	N	22000.0000	0.
25 TEA	LT	10.0000	N	22000.0000	0.
00 ARABLE LAND	N	0.0000	N	0.0000	0.

Figure 12.6. Dialog for modification of production targets.

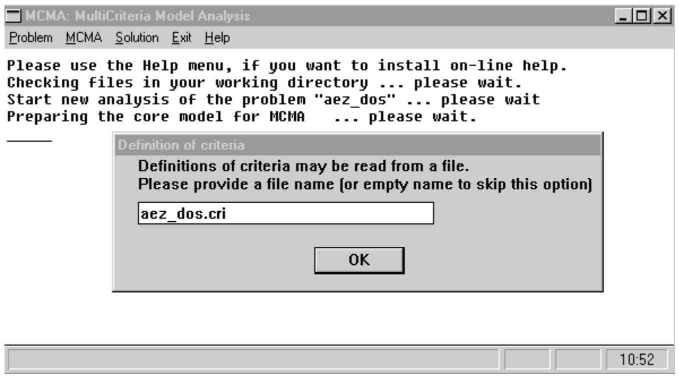
Selection and possible modification of a scenario concludes the definition of an instance of the AEZ core model, which can be generated and analyzed in a way outlined in Section 12.5.2. The results of multi-criteria analysis can be processed further in order to either review the results with the help of a GIS, or to produce printed reports.

12.5.2 Multi-criteria analysis of the AEZ model

An instance of the AEZ core model is generated in the LP_DIT format as the result of selecting a scenario, as discussed in the previous section. Such a core model conforms to the requirements specified in Section 6.6 and therefore can be analyzed according to the methodology presented in Part I of this book.

Multi-criteria model analysis of the AEZ model is implemented as a modular tool called MCMA (which is available on the Web, see the Appendix for details). A tutorial session with MCMA using the AEZ model is documented by Granat and Makowski (1998). This chapter provides only a short outline of such an analysis.

Multi-criteria model analysis requires the completion of a preparatory stage and starts with the interactive selection of a core model definition file. Next, the user interactively selects from a list of outcome variables of the core model those that will be used for criteria. A predefined selection of outcome variables and corresponding criteria definitions to be used in MCMA can optionally be provided in a file (cf. *Figure 12.7*). The set of predefined outcome variables is defined in Section 12.4.3. From this set the user typically selects a subset of outcome variables to be considered as criteria (this is done by selecting the ignore type of criterion for the other variables). Thereafter, the dialog for selecting variables shown in *Figure 12.8* can be used for selecting additional outcome variables. This dialog includes a Help

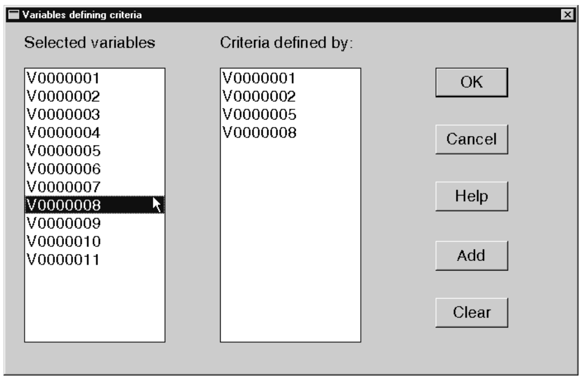

MCMA: MultiCriteria Model Analysis
Problem MCMA Solution Exit Help
Definition of criteria
OK
10:52

```
Please use the Help menu, if you want to install on-line help.
Checking files in your working directory ... please wait.
Start new analysis of the problem "aez_dos" ... please wait
Preparing the core model for MCMA   ... please wait.
```

Definitions of criteria may be read from a file.
Please provide a file name (or empty name to skip this option)

aez_dos.cri

Figure 12.7. Default selection of a file containing predefined criteria for AEZ core model.

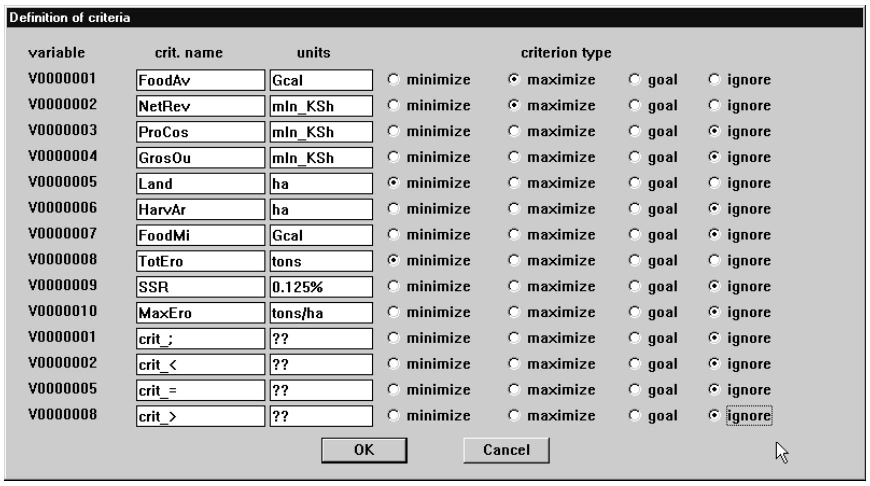


Selected variables	Criteria defined by:
V0000001	V0000001
V0000002	V0000002
V0000003	V0000005
V0000004	V0000008
V0000005	
V0000006	
V0000007	
V0000008	
V0000009	
V0000010	
V0000011	

Figure 12.8. Dialog for interactive selection of outcome variables from variables of the core model.

button that provides additional information. In order to avoid scrolling a long list of all variables of the core model, one can define a mask (the first characters of names of variables) for variables that are displayed for selection. Several masks can be defined sequentially for providing a list composed of names that start with different characters. The user can also define, for smaller models, an empty mask that will result in including all the variables into the selection list.

Next, the user selects for each variable a criterion name and its type using the dialog shown in *Figure 12.9*. One of the following three types of criteria has to be selected: minimized, maximized, and goal. The meaning of the first two types is clear, but the goal type requires an explanation. The goal type of criterion should be used only if the meaning of this criterion is such that it should neither be minimized nor maximized, but it should attain a given target value. The dialog shown in *Figure 12.9* also provides the possibility to select only several criteria from the



Definition of criteria

variable	crit. name	units	criterion type			
V0000001	FoodAv	Gcal	minimize	[✓ maximize]	goal	ignore
V0000002	NetRev	mln_KSh	minimize	[✓ maximize]	goal	ignore
V0000003	ProCos	mln_KSh	minimize	maximize	goal	[✓ ignore]
V0000004	GrosOu	mln_KSh	minimize	maximize	goal	[✓ ignore]
V0000005	Land	ha	[✓ minimize]	maximize	goal	ignore
V0000006	HarvAr	ha	minimize	maximize	goal	[✓ ignore]
V0000007	FoodMi	Gcal	minimize	maximize	goal	[✓ ignore]
V0000008	TotEro	tons	[✓ minimize]	maximize	goal	ignore
V0000009	SSR	0.125%	minimize	maximize	goal	[✓ ignore]
V0000010	MaxEro	tons/ha	minimize	maximize	goal	[✓ ignore]
V0000001	crit_;	??	minimize	maximize	goal	[✓ ignore]
V0000002	crit_<	??	minimize	maximize	goal	[✓ ignore]
V0000005	crit_=	??	minimize	maximize	goal	[✓ ignore]
V0000008	crit_>	??	minimize	maximize	goal	[✓ ignore]

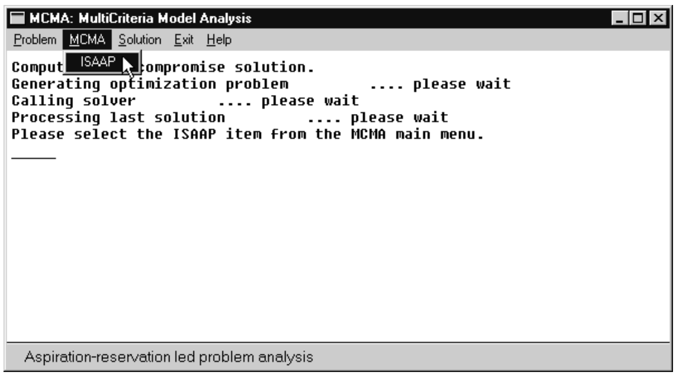
OK Cancel

Figure 12.9. Dialog for definition of criteria.

larger set of predefined criteria (in this example only four criteria were selected and the other six were ignored).

The definition of criteria finishes the preparatory interaction phase. Then the MCMA starts a series of optimizations in order to compute the utopia point, an approximation of the nadir point, and a neutral compromise Pareto-optimal solution. If there are n selected criteria this will require solving $(2n + 1)$ LP problems. The user can follow the information about the generation of parametric single-criterion optimization problems and update values of the utopia point and approximation of the nadir point that will be displayed in the main MCMA window. Once the computation of the neutral compromise solution is completed (which took about one minute for this example on a Pentium-180 PC), the information about the definition of criteria and the pay-off table is stored by MCMA in a file that can be used later for a continuation of analysis. The computation of the neutral compromise solution is reported by a message shown in *Figure 12.10*.

After computation of the neutral compromise solution, interactive multi-criteria model analysis can be continued with the help of the interactive specification of aspiration-based preferences (ISAAP) tool, which is a modular tool included in the MCMA (see Chapter 10 for details). ISAAP can be started by selecting the ISAAP item from the MCMA main menu. The neutral compromise solution (e.g., as presented in *Figure 12.11*) is typically not acceptable, because the aspiration and reservation components are set by MCMA to be equal to the corresponding components of the utopia and the nadir point, respectively. However, such a solution is a good starting point for further analysis of the model.


MCMA: MultiCriteria Model Analysis
Problem MCMA Solution Exit Help
ISAAP
Aspiration-reservation led problem analysis

```
Comput      compromise solution.
Generating optimization problem         .... please wait
Calling solver          .... please wait
Processing last solution         .... please wait
Please select the ISAAP item from the MCMA main menu.
```

Figure 12.10. Information about computation of the pay-off table and of the neutral compromise solution.

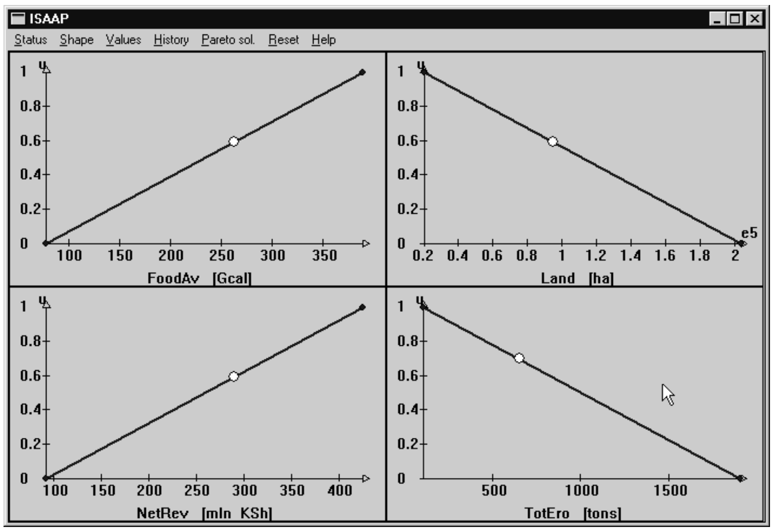


Figure 12.11. First interaction: neutral compromise solution.

Suppose that the user primarily wants to improve the level of food production and therefore attempts to achieve this by increasing the reservation level for the corresponding criterion to about 300 (from less than 100, which was the nadir value for this criterion), without changing aspiration and reservation levels for other criteria (see *Figure 12.12*). This can be achieved by clicking the mouse to points corresponding to a new reservation level. In this case we would like to move the reservation level of the criterion Food, therefore we should first click to a value around 200, then to around 250, and finally to 300, which is easy when handling the mouse.

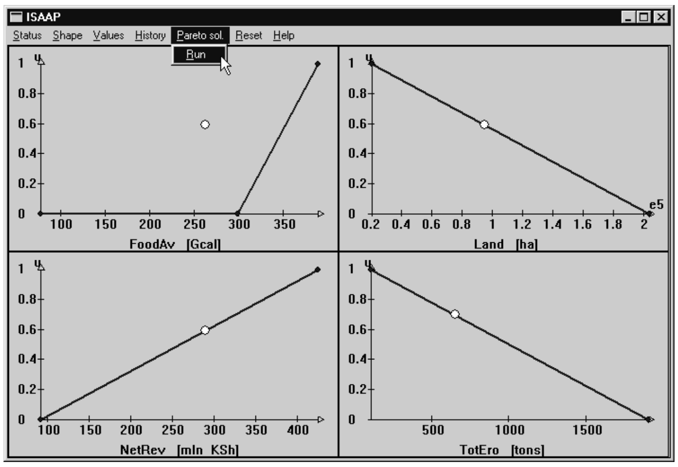


Figure 12.12. Increasing aspiration level for the criterion Food.

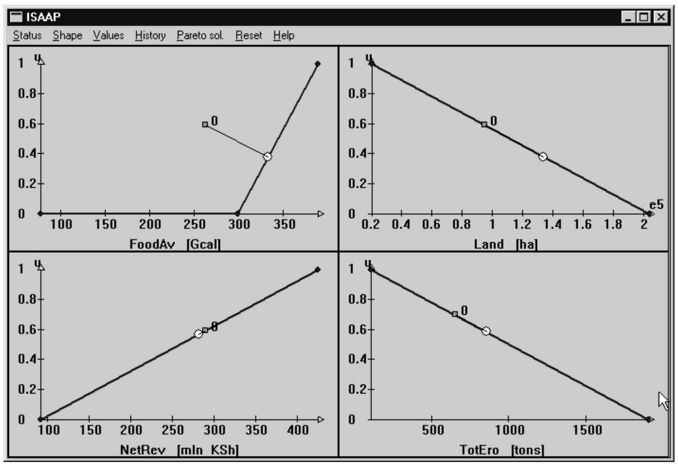


Figure 12.13. Solution for preferences set in *Figure 12.12*.

The Pareto-optimal solution that corresponds to the revised preferences is illustrated in *Figure 12.13*. The user indeed succeeded to considerably improve the food production level. However, this also resulted in substantial increases of land use and of erosion. Note that the level of net revenue has not changed substantially, but the satisfaction level (measured by the value of the membership function scaled between zero and one) has somewhat decreased. This decrease corresponds to the fact that the current solution is more distant from the aspiration level than the compromise solution.

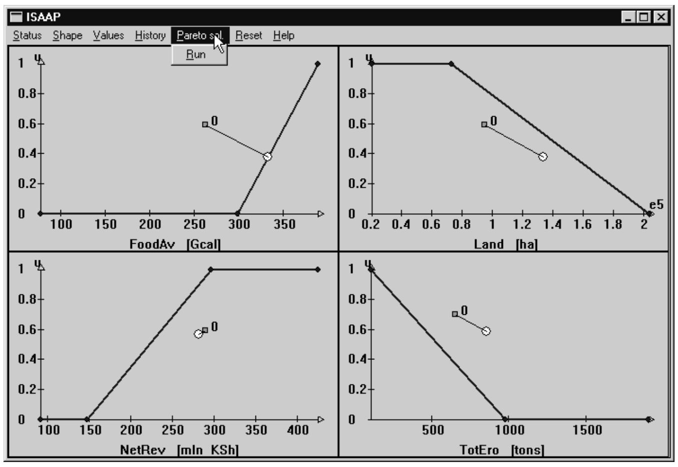


Figure 12.14. Setting of preferences in the second iteration.

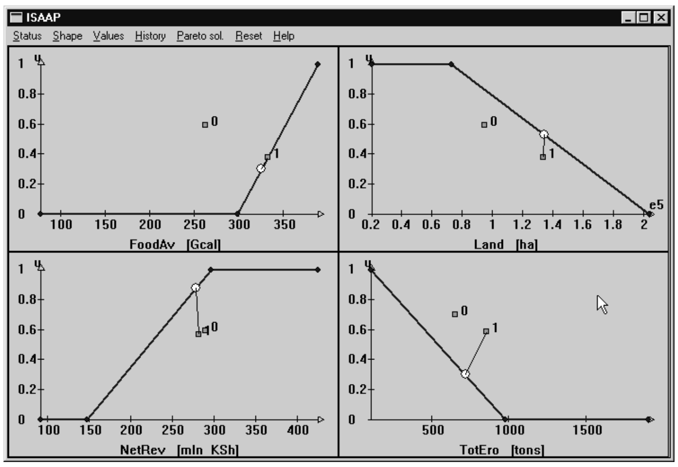


Figure 12.15. Results of the second iteration.

After the first attempt, suppose that the user would now like to keep the aspiration and reservation level for the Food criterion without changes, but would like to decrease the level of erosion. In order to make these goals more realistic, the user is willing to accept an increased use of arable land, and would be satisfied with the currently achieved level of revenue (but would not like to let revenue drop below 150). *Figure 12.14* illustrates setting of the aspiration and reservation levels that may represent such preferences. The corresponding Pareto-optimal solution is presented in *Figure 12.15*. We can observe that there is only a modest improvement for the TotEro criterion without many changes in the values of the other three criteria. The decrease of the satisfaction level for the TotEro criterion might be surprising because the value of this criterion has improved (the erosion has decreased). However, an explanation for this phenomenon is easy, if one remembers the interpretation of the membership function that represents the satisfaction level. In the first (neutral compromise) solution the satisfaction for this criterion was much higher because the reservation level was much higher, therefore the value of this criterion in the neutral compromise solution is meeting our expectations quite well. However, in the second iteration we increased our expectations considerably (by moving the reservation level to the value of 1,000), therefore the value of this criterion for the new solution is much less satisfactory (taking into account the new preferences) than in the neutral compromise solution.

In a typical session such an interactive process of changing preferences in response to the Pareto-efficient solutions presented will continue until a satisfactory solution is reached or room for flexibility is exhausted.

For a more complete description of further possible interaction steps, the reader is advised to consult the documentation and the tutorial for this DSS by Fischer *et al*. (1998). This paper is also available from the Web (see the Appendix for details). Other possibilities offered by ISAAP for the interactive multi-criteria model analysis are presented in Chapter 10.

12.6 Conclusions

The approach presented in this chapter illustrates linking multi-criteria methods of model analysis with GIS land resources databases, which results in a powerful DSS tool in land use decision-making support. A cumbersome approach of converting all but one criteria into constraints and applying a sensitivity analysis of a sequence of single-criterion solutions, as it is necessary in single-criterion optimization, is replaced by an interactive specification of the preferences of the decision maker. Moreover, the simplicity and flexibility of the approach assist the user, during the process of decision-making, to better understand the decision situation. The approach presented is interactive and fast, hence the development of some dozen solutions does not require more than a few hours for a user with a good understanding of the problem.

The user does not need to be experienced in sensitivity analysis and scenario generation techniques, which are necessary for the analysis based on a single-criterion approach. On the one hand, the detailed evaluation of a large number of solutions obtained during multi-criteria model analysis can be much more time-consuming than the evaluation of a smaller number of solutions typically analyzed by single-criterion optimization. On the other hand, the analysis of a large number of solutions corresponding to different areas of the Pareto-efficient set provides a more complete understanding of the problem. Solutions that are close to each other, as obtained in the previous example, can appear confusing at first to the decision maker. The ISAAP tool provides an option for analyzing the history of solutions that helps in selecting solutions. Some users also have difficulty when evaluating more than three criteria visually. Special techniques are provided by ISAAP to facilitate such an evaluation. This can be done by a sequential selection of groups of criteria that are investigated more closely while the remaining criteria are either inactive or their values are stabilized around a target value desired by the user (as selected for each criterion).

The multi-criteria model analysis method can also be used for a more detailed model analysis in two ways that have not been applied so far in the case study reported in this chapter. The first one is called soft simulation. This is an extension of the traditional simulation allowing a combination of multi-criteria analysis with (soft) setting of values of selected variables. Second, it allows for treatment of a group of constraints as so-called soft constraints, i.e., constraints that can be violated up to a certain bound interactively controlled by the user. Both techniques are discussed in more detail in Part I of this book. It is planned to apply these techniques in analysis with the AEZWIN as soon as the software will be enhanced to provide these options.

To avoid a misleading conclusion (which, however, frequently arises), namely that the usage of such a DSS package might replace a real decision maker, it should be stressed again that the system is designed to help a decision maker to concentrate on real decision making, while the program takes care of various cumbersome computations involved in the analysis of scenarios and provides information that furthers the analysis of the consequences of different options and alternatives. The user needs to define the various scenarios of interest, changing his/her preferences and priorities when learning interactively about the consequences of possible decisions. Röling (1994) has explored the limitation of focusing exclusively on building scenarios on the basis of interactive multi-criteria model analysis, without paying attention to human decision making in developing and applying those scenarios.

For a successful application of DSSs in land use planning in developing countries, it is necessary to overcome various limitations and difficulties. In many of these countries, lack of data and poor data quality constitute serious limitations to an application of computer-based systems of land resources management. Lack of trained personnel to apply these systems in solving practical problems is another constraint, which often causes the available systems to be underutilized and sometimes not used at all.

Acknowledgments

The authors of this chapter would like to thank Robert Brinkman and Jacques Antoine of FAO for their long-term collaboration in the development of the AEZ methodology and for their role in promoting the collaboration between FAO and IIASA. Thanks are also due to Wolfgang Prante of FAO for his detailed comments on both the software and its documentation. His suggestions have helped remarkably in improving the documentation, user interface, and organization of the distributable version of the software.

The authors acknowledge and appreciate the collaboration with Janusz Granat of the Warsaw University of Technology, who has developed components of MCMA and the GUI of the AEZWIN. The collaboration of Paweł Białoń, of the Warsaw University of Technology, in the development of the AEZWIN prototype is also acknowledged.

Chapter 13

Effect-Focused Air Quality Management

Markus Amann and Marek Makowski

13.1 Introduction

There is substantial concern about the environmental impacts of air pollution at the local, regional, and global scale. It has been shown that observed levels of various air pollutants can threaten human health, vegetation, wildlife, and cause damage to physical structures. In order to limit the negative effects of air pollution, measures to reduce emissions from a variety of sources have been initiated.

Once emitted, many air pollutants remain in the atmosphere for some time before they are finally deposited on the ground. During this time, they are transported with the air mass over long distances, often crossing national borders. As a consequence, at a given site, the concentration of pollutants and their deposition on the ground is influenced by a large number of emission sources, frequently in many different countries. Thus, actions to efficiently abate air pollution problems have to be coordinated internationally.

Several international agreements have been reached over the last decade in Europe to reduce emissions in a harmonized way. Protocols under the Convention on Long-Range Transboundary Air Pollution (LRTAP) focus on reducing emissions of sulfur dioxide (SO_2), nitrogen oxides (NO_x), and volatile organic compounds (VOC). Several directives of the European Union prescribe emission standards for large combustion plants, for mobile sources, and limit the sulfur content in liquid fuels.

Most of the current agreements determine required abatement measures solely in relation to technical and economic characteristics of the sources of emissions, such as available abatement technologies, costs, historic emission levels, etc. No relation is established with the actual environmental impacts of emissions. For achieving overall cost effectiveness of strategies, however, the justification of potential measures in relation to their environmental benefits must also be taken into account. Recently, progress has been made in quantifying the environmental sensitivities of various ecosystems. Critical loads and critical levels have been established reflecting the maximum exposure of ecosystems to one or several pollutants not leading to environmental damage in the long run. Such threshold values have been determined on a European scale, focusing on acidification and eutrophication as well as on vegetation damage caused by tropospheric ozone.

It is generally expected that the current policies on emission reductions will greatly reduce the levels of tropospheric ozone. However, the measures will not be sufficient to eliminate the problem everywhere in Europe. To meet the environmental long-term targets aiming at the protection of human health and vegetation, as they are currently discussed in the context of the European Commission's ozone strategy, further measures will be necessary. Since most of the low-cost options for abating emissions have already been adopted in the current strategies, further action aiming at the sustainability of Europe's ecosystems will have to embark on more costly measures. Cost effectiveness will be an important argument for gaining acceptance of proposed policies.

The recent progress in quantifying the sensitivities of ecosystems adds an important feature to the analysis and development of cost-effective strategies to achieve and maintain emission levels that do not endanger the sustainability of ecosystems. Such an analysis is based on integrated assessment models that combine information and databases on the economic, physical, and environmental aspects relevant for strategy development.

Optimization tools have been used to systematically analyze the relevant pieces of information with the aim to identify cost-effective solutions for the air pollution problem.

The remainder of this chapter introduces the basic concept of a prominent integrated assessment model, the Regional Air Pollution Information and Simulation (RAINS) model developed at the International Institute for Applied Systems Analysis (IIASA), Laxenburg, Austria. Section 13.3 provides a description of the mathematical model followed by Section 13.4, which discusses selected aspects of its practical implementation. Section 13.5 presents practical examples of policy applications of the RAINS model.

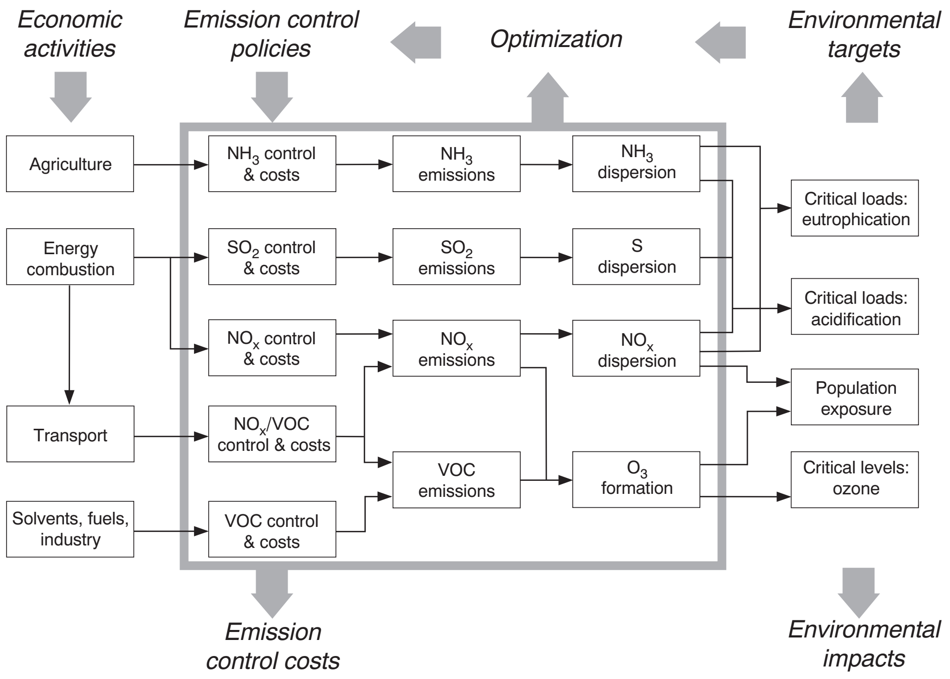


Figure 13.1. Schematic flowchart of the RAINS model.

13.2 Methodological Approach

The RAINS model developed at IIASA provides a consistent framework for the analysis of emission reduction strategies, focusing on acidification, eutrophication, and tropospheric ozone. RAINS comprises modules for emission generation (with databases on current and future economic activities, energy consumption levels, fuel characteristics, etc.), for emission control options and costs, for atmospheric dispersion of pollutants, and for environmental sensitivities (i.e., databases on critical loads). In order to create a consistent and comprehensive picture of the options for simultaneously addressing the three environmental problems (acidification, eutrophication, and tropospheric ozone), the model considers emissions of SO_2, NO_x, ammonia (NH_3), and VOC. A schematic diagram of the RAINS model is displayed in *Figure 13.1*. A detailed description of the RAINS model can be found in Alcamo *et al.* (1990).

The RAINS model can be operated in the "scenario analysis" mode, i.e., following the pathways of the emissions from their sources to their environmental impacts. In this case the model provides estimates of regional costs and environmental benefits of alternative emission control strategies. Alternatively, an "optimization mode” is available to identify cost-optimal allocations of emission reductions in order to achieve specified environmental targets.

13.2.1 Energy and emissions estimates

The European implementation of the RAINS model incorporates databases for 38 regions in Europe. The time horizon extends from the year 1990 up to the year 2010.

The RAINS model incorporates databases on critical loads and critical levels compiled at the Coordination Center for Effects (CCE) at the National Institute for Public Health and Environmental Protection (RIVM) in the Netherlands (Posch *et al*., 1997).

Inputs to the RAINS model include projections of future energy consumption on a national scale up to the year 2010. The model stores this information as energy balances for selected future years, distinguishing fuel production, conversion, and consumption for 22 fuel types in 6 economic sectors. These energy balances are complemented by additional information relevant for emission projections, such as boiler types (e.g., dry bottom versus wet bottom boilers), size distribution of plants, age structures, fleet composition of the vehicle stock, etc.

Agricultural activities are a major source of ammonia emissions, which in turn contribute to the acidification problem. Next to specific measures directed at limiting the emissions from livestock farming, the development of the animal stock is an important determinant of future emissions. The projections of future agricultural activities currently implemented in the RAINS model have been compiled from a variety of national and international studies on the likely development of the agricultural system in Europe.

The forecast of the future development of VOC emission generating activities is linked to other information on general economic development. About half of the anthropogenic emissions of VOC originates from combustion, extraction, and distribution of fossil fuels. Therefore, the information on projected levels of fuel consumption is used to estimate future emissions of VOC from relevant sources (i.e., traffic, stationary combustion, extraction, and distribution of fuels).

Although there is a large variety of options to control emissions, an integrated assessment model focusing on the pan-European scale has to restrict itself to a manageable number of typical abatement options in order to estimate future emission control potentials and costs. Consequently, the RAINS model identifies a limited list of characteristic emission control options for each of its application areas (i.e., emission source categories considered in the model). For each of these measures, the model extrapolates the current operating experience to future years, taking into account the most important country- and situation-specific circumstances modifying the applicability and costs of the techniques.

For each of the available options of emission control, RAINS estimates the specific costs of reductions, taking into account investment-related and operating costs. Investments are annualized over the technical lifetime of the pollution control equipment, using a discount factor of 4%. The technical performance as well as investments, maintenance, and material consumption are considered to be technology-specific and thereby, for a given technology, equal for all European countries. Fuel characteristics, boiler sizes, capacity utilization, labor, and material costs (and stable sizes and applicability rates of abatement options for ammonia) are important country-specific factors influencing the actual costs of emission reduction under given conditions. Detailed descriptions of the methodology adopted to estimate emission control costs can be found in Amann (1990), Klaassen (1991), Cofala *et al*. (1997), and Klimont *et al*. (1998).

The databases on emission control costs have been constructed on the basis of the actual operating experience of various emission control options documented in a number of national studies (e.g., Schörer, 1993) as well as in reports of international organizations (e.g., Rentz *et al*., 1987; OECD, 1993; Takeshita, 1995). Country-specific information has been extracted from relevant national and international statistics (UN/ECE, 1996) The list of control options for NO_x, NH_3, and VOC, and the country-specific data used for the cost calculations were reviewed by the negotiating parties of the LRTAP Convention.

13.2.2 Modeling the dispersion of sulfur and nitrogen compounds in the atmosphere

The RAINS model estimates deposition of sulfur and nitrogen compounds due to the emissions in each country, and then sums the contributions from each country with a background contribution to compute total deposition at any grid location. These calculations are based on source-receptor matrices derived from a Lagrangian model of long-range transport of air pollutants in Europe, developed by the Cooperative Programme for Monitoring and Evaluation of the Long-Range Transmission of Air Pollutants in Europe (EMEP).

The EMEP model is a receptor-oriented, single-layer air parcel trajectory model, in which air parcels follow two-dimensional trajectories calculated from the wind field at an altitude that represents a dominating transport type within the atmospheric boundary layer. Balances of chemical development within the air parcels are described by ordinary first-order differential equations integrated in time along the trajectories as they follow atmospheric motion. The transport equations take into account emissions from the underlying grid with a mesh size of 50 km, chemical processes in the air, and wet and dry deposition to the ground surface. Model calculations are based on six-hourly input data of the actual meteorological conditions for specific years.

In order to capture the inter-annual meteorological variability, model runs have been performed for 11 years (1985–1995) by Barret and Sandnes (1996). For each of these years, balances of sources (aggregated to entire countries) and sinks (in a regular grid mesh with a size of 150×150 km) of pollutants have been calculated. These annual source-receptor balances have been averaged over 11 years and rescaled to provide the spatial distribution of one unit of emissions. The resulting atmospheric transfer matrices are then used as input data in the RAINS model.

13.2.3 Modeling ozone formation

The RAINS model describes changes in long-term ozone levels at a given receptor site (over a six-month period) as a function of the precursor emissions (NO_x and VOC).

Most of the available models for ozone formation are process-oriented and contain a considerable amount of detail of the chemical mechanisms and meteorological factors relevant for ozone formation. Since their computational complexity makes it impossible to use them directly within the framework of an integrated assessment model, the reaction of more complex "reference" models had to be summarized into a "reduced-form" model. Using statistical methods, the reduced-form models were derived from a large sample of scenario results developed with the EMEP photo-oxidants model in the form:

$$aot_{lj} = \sum_{i \in I} (a_{lij} v_i + b_{lij} n_i + \gamma_{lij} n_i^2) + \alpha_{lj} en_{lj}^2 + \beta_{lj} en_{lj} nlv_{lj} + k_{lj} \qquad (13.1)$$

where aot_{lj} stands for the long-term ozone exposure of l-th type at j-th receptor. The coefficients could be interpreted in the following way relating to the physical and chemical processes that determine ozone formation in the atmosphere:

- k_{lj} includes the effects of background concentrations of O_3 and its precursors, and natural VOC emissions.
- $a_{lij} v_i$ provides the linear country-to-grid contribution from VOC emissions in country i, allowing for meteorological effects.
- $b_{lij} n_i$ provides the linear country-to-grid contribution from NO_x emissions in country i, allowing for meteorological effects.
- $\alpha_{lj} en_{lj}^2$ takes account of the average nonlinearity (in the O_3/NO_x relationship) experienced along trajectories arriving at receptor j and any nonlinear effects local to that receptor.
- $\beta_{lj} en_{lj} nlv_{lj}$ allows interactions between NO_x and VOC along the trajectories.

The coefficients a_{ij} and b_{ij} may also be regarded as a composite source-receptor matrix.

For each receptor site and ozone exposure type, the coefficients of model (13.1) were derived by linear regression analysis.

13.2.4 Optimization

The optimization mode of integrated assessment models can be a powerful tool in the search for cost-effective solutions to combat air pollution problems. In the RAINS-acidification model, optimization techniques have been used to identify the cost-minimal allocation of resources in order to reduce the gap between current sulfur deposition and the ultimate targets of full critical loads achievement.

In the case of tropospheric ozone, a systematic search for cost effectiveness appears even more attractive. The fact that several pollutants (NO_x and VOC emissions) are involved, and that important nonlinearities between precursor emissions and ozone levels have been recognized, diminish the likelihood that some good intuitive ad hoc solutions can be identified using the classical scenario analysis methods.

For the particular air quality management task, there are three nonstandard aspects of using optimization in searching for cost-effective solutions:

- Hard constraints on the air quality targets in each grid area typically result in high costs of emissions reduction. Therefore, requirements for the air quality indices are optionally represented as soft constraints expressed either by additional penalties for violating the air quality targets or as additional objectives in a multi-objective formulation.
- The model (like most large optimization models) may have many suboptimal solutions, i.e., structurally different solutions for which the corresponding goal function values are similar. Therefore, a regularization option for the optimization problem has been implemented in order to prioritize suboptimal solutions close to a specific reference point defined by given values of emissions.
- For some analytical tasks it is desirable to examine trade-offs between two criteria: the costs and the distance from given levels of emissions. This can be achieved by a specialized implementation of the softly constrained inverse simulation described in Chapter 5.

Implementations of these aspects and their advantages for the problem analysis are discussed in more detail in Sections 13.3.5 and 13.5, respectively.

13.3 Model Definition

13.3.1 Notation

A distinction should be made between a set I of sources of various types of air pollution, and a set J of areas for which various quality indicators are assessed. Conventionally, the names emitter and receptor are used for elements of such sets.

The model definition requires the use of the following indices:

- Index $i \in I$ corresponds to emitters. The numbers of elements in I correspond to the number of countries (about 40).
- Index $is \in S_i$ corresponds to a sector that emits either NO_x or VOC or a linear combination of them; S_i is a set of sectors in the i-th country. Sets S_i may have up to five elements.
- Index $j \in J$ corresponds to receptors. The numbers of elements in J corresponds to numbers of grids (about 600).
- Index $l \in L$ corresponds to a combination of ozone thresholds and a year. The set L may have up to six elements.
- Index $m \in M$ corresponds to a set of receptors for which balancing of violations and surpluses of targets is defined.

Emission Sectors

Emissions are analyzed for sets of emitters that are located in a certain area, which is typically a country. However, for some types of analysis, sets of NO_x and VOC emitters may be further subdivided into subsets called sectors, in order to account for measures that can be applied to emitters that belong to a sector. In such a case emitters that belong to a particular sector emit either NO_x or VOC or a linear combination of them. A sector is defined by a quadruple:

$$\{is, i, \lambda, \mu\} \tag{13.2}$$

where is is an integer number that uniquely identifies a sector, an index i corresponds to a country in which emitters belonging to this sector are located, and the following convention is used for the λ member of this quadruple:

$$\lambda = \begin{cases} 0, & \text{if only NO}_\text{x} \text{ is emitted} \\ \text{a negative number}, & \text{if only VOC is emitted} \\ \text{a positive number}, & \text{if both NO}_\text{x} \text{ and VOC are emitted.} \end{cases} \tag{13.3}$$

Moreover, a (positive) value of λ defines the following relation between the amount of VOC emission corresponding to a unit emission of NO_x:

$$v_{is} = \lambda_{is} n_{is} + \mu_{is} \tag{13.4}$$

13.3.2 Decision variables

The main decision variables are the annual emissions of the following four types of primary air pollutants from either sectors or countries: (i) n_{is}, annual emission of NO_x from sector is; (ii) v_{is}, annual emission of nonmethane VOC from sector is; (iii) a_i, annual emission of NH_3 from country i; and (iv) s_i, annual emission of SO_2 from country i.

Additionally, optional decision variables are considered for scenarios that allow limited violations of air quality targets. For such scenarios variables corresponding to each type of the considered air quality targets are defined for each receptor. Each variable represents a violation of a given environmental standard. Optionally, violations of targets can be balanced with surpluses (understood as a difference between a target and a corresponding actual concentration/deposition). For reasons of efficiency, one variable is used for both violations of targets and surpluses (positive values represent violations while negative values correspond to a part of a surplus that is used to balance violations of targets).

Therefore, the following decision variables are optionally defined: (i) y_{lj}, violation of ozone exposure targets (surplus, if $y_{lj} < 0$); (ii) ya_j, violation of acidification targets (surplus, if $ya_j < 0$); and (iii) ye_j, violation of eutrophication targets (surplus, if $ye_j < 0$).

13.3.3 Outcome variables

The consequences of applications of computed values of the decision variables are evaluated by the values of outcome variables. However, several auxiliary variables needed for the definitions of outcome variables have to be specified first.

Auxiliary Variables

The annual emission of NO_x (n_i) is defined by:

$$n_i = \sum_{is \in S_i} n_{is}. \tag{13.5}$$

The annual emission of VOC (v_i) is defined by:

$$v_i = \sum_{is \in S_i} v_{is}. \tag{13.6}$$

The mean effective emissions of NO_x, including natural sources, experienced at the j-th receptor (en_{lj}) is given by:

$$en_{lj} = \sum_{i \in I} e_{lij} n_i + enn_{lj} \tag{13.7}$$

where enn_{lj} are given effective natural emissions of NO_x.

The representation of another nonlinear term (nlv_{lj}) defining ozone exposure at the j-th receptor is defined by:

$$nlv_{lj} = \sum_{i \in I} d_{lij} v_i. \tag{13.8}$$

The coefficients e_{ij}, d_{ij} depend on the meteorology and are obtained from EMEP model calculations.

Definition of Outcome Variables

One outcome variable represents the sum of costs of reductions of emissions; four sets of additional outcome variables correspond to various indices of air quality.

Annual costs related to the reduction of a corresponding emission to a certain level are given by a piece-wise linear (PWL) function for each type of emission and for each emitter. Each PWL function is composed of segments defined by pairs of points; each point is composed of the emission level and the corresponding cost of reducing emission to this level.[1] Therefore, a PWL function is defined for each member of sets of NO_x and/or VOC emitters declared by (13.2) and for NH_3 and SO_2 emitters (the latter for each country). Formally, the following PWL functions define the annual cost related to reducing the level of emission to a level given by argument(s) of the function: $ca_i(a_i)$ for a_i, $cs_i(s_i)$ for s_i, and $c_i(n_i, v_i)$ for n_i and v_i. The term $c_i(n_i, v_i)$ is defined by:

$$c_i(n_i, v_i) = \sum_{s \in S_i} c_s(\cdot) \tag{13.9}$$

where $c_s(\cdot)$ is a cost function for NO_x or for VOC or for joint NO_x and VOC reduction, depending on $type\_def$ defined by (13.3).

Each cost function defines its domain by specifying lower and upper bounds for its argument(s). This implicitly defines lower and upper bounds for all emissions that are used as bounds defined in Section 13.3.4. Those bounds may be made tighter by an optional specification of bounds for emissions from countries or sectors.

For the sake of brevity, the sum of costs is defined by:

$$cost = \sum_{i \in I} (ca_i(a_i) + cs_i(s_i) + c_i(n_i, v_i)). \tag{13.10}$$

Such a function is continuous and convex but it is not smooth. Therefore, it has to be represented by another function that meets typical requirements of nonlinear solvers. Such a modification is presented in Section 13.4.3.

[1]The precise definition of PWL functions is given by equation (13.34) on page 384.

For each receptor, the following four outcome variables correspond to various indices of air quality:

aot_{lj}, the long-term ozone exposure of the l-th type, is assumed to be a function of the nonmethane VOC and NO_x emissions, v_i and n_i, respectively, from each emitter country i, and the mean "effective" emissions (of NO_x and VOC), en_j, and nlv_j, experienced at the receptor over the period in question. The general model formulation adopted is:

$$aot_{lj} = \sum_{i \in I}(a_{lij}v_i + b_{lij}n_i + \gamma_{lij}n_i^2) + \alpha_{lj}en_{lj}^2 + \beta_{lj}en_{lj}nlv_{lj} + k_{lj}. \quad (13.11)$$

$ac1_j$, acid deposition of Type 1:

$$ac1_j = tns_j(\sum_{i \in I} tn_{ij}n_i + \sum_{i \in I} ta_{ij}a_i + kn_j) + \sum_{i \in I} ts_{ij}s_i + ks_j. \quad (13.12)$$

$ac2_j$, acid deposition of Type 2:

$$ac2_j = \sum_{i \in I} tn_{ij}n_i + \sum_{i \in I} ta_{ij}a_i + tss_j(\sum_{i \in I} ts_{ij}s_i + ks_j) + kn_j \quad (13.13)$$

eu_j, eutrophication:

$$eu_j = \sum_{i \in I} tn_{ij}n_i + \sum_{i \in I} ta_{ij}a_i + kn_j \quad (13.14)$$

where tn_{ij}, ta_{ij}, ts_{ij} are transfer coefficients for NO_x, NH_3, and SO_2, respectively; kn_j and ks_j are constants for nitrogen and sulphur background depositions; tns_{ij}, tss_{ij} are scaling coefficients that convert acidification coefficients of one type into acidification coefficients of another type, for NO_x and NH_3, and for SO_2, respectively.

Environmental effects caused by acid deposition, excess nitrogen deposition (described by a two-element linear critical loads function), and by eutrophication are evaluated at each receptor by a PWL function that represents an accumulated excess over the threshold of the environmental long-term target:

$aac1_j$, accumulated excess of acidification of Type 1:

$$aac1_j = PWL_j^{ac1}(ac1_j). \quad (13.15)$$

$aac2_j$, accumulated excess of acidification of Type 2:

$$aac2_j = PWL_j^{ac2}(ac2_j). \quad (13.16)$$

aeu_j, accumulated excess of eutrophication:

$$aeu_j = PWL_j^{eu}(eu_j). \quad (13.17)$$

13.3.4 Constraints

Each of the decision variables defined in Section 13.3.2 for $i \in I$ or for $is \in S_i$ is implicitly bounded by a corresponding definition of the domain of the corresponding cost function, which represents costs associated with the reduction of emission (see Section 13.3.3). For some scenarios this domain may be further restricted by a specification of optional bounds.

Violations of targets are constrained at each receptor by corresponding lower and upper limits specified for each target type and for each grid:

$$y_{lj}^{min} \leq y_{lj} \leq y_{lj}^{max}, \tag{13.18}$$

$$ya_j^{min} \leq ya_j \leq ya_j^{max}, \tag{13.19}$$

$$ye_j^{min} \leq ye_j \leq ye_j^{max}. \tag{13.20}$$

The long-term ozone exposure of the l-th type is constrained at each receptor by:

$$aot_{lj} - y_{lj} \leq aot_{lj}^{max} \tag{13.21}$$

where aot_{lj} is defined by (13.11) and aot_{lj}^{max} is a given maximum ozone exposure for the l-th threshold at the j-th receptor.

Constraint (13.21) without the term $-y_{lj}$ represents a so-called hard constraint for long-term ozone exposure and such a formulation is typically used in traditional formulations of optimization problems. It can also be used in the presented model by selecting an option that does not allow for generation of variables y_{lj}. However, assuming hard constraints for air quality targets results in more expensive solutions, which are caused by constraints that are active in only one or two receptors. Introduction of the term $-y_{lj}$ converts a hard constraint into a so-called soft constraint. This allows a violation of a target air quality. Such a violation is: (i) constrained by upper bounds on variables y_{lj}; (ii) compensated by surpluses (i.e., differences between actual exposure and the corresponding target) in other receptors belonging to the same set of receptors (e.g., located in the same country or region); and (iii) controllable by a trade-off between violation of targets and corresponding costs of reducing emissions (see Section 13.3.5 for details).

Constraints for accumulated excess of the two types of acidification and of eutrophication are defined in a similar way:

$$aac1_j - ya_j \leq aac_j^{max}, \tag{13.22}$$

$$aac2_j - ya_j \leq aac_j^{max}, \tag{13.23}$$

$$aeu_j - ye_j \leq aeu_j^{max}. \tag{13.24}$$

Optionally, violations of targets can be balanced with surpluses of targets over sets of receptors denoted in the following constraints by J_m, where $m \in M$ is the index of a set of receptors. Obviously, lower bounds in conditions (13.18), (13.19), and (13.20) have to be negative in such a case. The balances are represented by the following constraints:

$$\sum_{j \in J_m} wo_{lmj} y_{lj} \leq tbo_{lm} \qquad l = 0, \tag{13.25}$$

$$\sum_{l=1}^{L} \sum_{j \in J_m} wo_{lmj} y_{lj} \leq \sum_{l=1}^{L} tbo_{lm}, \tag{13.26}$$

$$\sum_{j \in J_m} wa_{mj} ya_j \leq tba_m, \tag{13.27}$$

$$\sum_{j \in J_m} we_{mj} ye_j \leq tbe_m, \tag{13.28}$$

where $wo_{lmj}, wa_{mj}, we_{mj}$ are the given weighting coefficients, $J_m, m \in M$ are sets of receptors, and tbo_{lm}, tba_m, tbe_m are target balances for the m-th set of receptors for the l-th type of ozone exposure, for acidification, and for eutrophication, respectively. The sets J_m are defined implicitly by nonzero elements of sparse matrices wo_l, wa, and we, respectively.

13.3.5 Goal function

As already discussed in Parts I and II of this book, optimization is merely a tool that helps to analyze complex models. For large nonlinear models, the typical tools of interactive multi-criteria model analysis cannot be applied effectively because of the time required for one optimization run. Therefore, single-criterion optimization is used, but with some modifications typical for multi-criteria analysis. Namely, it is applied for scenario analysis aimed at finding and analyzing alternative policy options that correspond to various trade-offs between the following three goals:

- Minimization of total costs of emissions reduction.
- Minimization of violations of environmental standards (with optional compensations of violations by surpluses).
- Stabilization of solutions.

Since the first two components have already been discussed, only the last one requires further elaboration.

A typical problem in applying optimization techniques for decision support is caused by essentially different solutions (with almost the same value of the original goal function) for instances of a mathematical programming problem that differ very little. From the optimization point of view the quality of a solution is assessed primarily through the value of the goal function. Solutions of slightly perturbed problems may differ substantially. However, from an application point of view, an equally important indication of solution robustness is how close perturbed solutions are. Consider, for example, two instances of the RAINS model that differ very little. The values of the goal functions for such solutions will be almost the same. However, often the optimal solution of the first instance has a high reduction of emission in country A and a low reduction in country B, whereas the optimal solution for the second instance has a low reduction in country A and a high reduction in country B. In order to avoid this problem, a technique called regularization (see Section 7.6.1 for a more detailed discussion) can be applied.

The following composite goal function is applied for scenario analysis using single-criterion optimization:

$$goal\_function = cost + \Theta + \Psi \tag{13.29}$$

where the *cost* term corresponds to the sum of the costs of emission reductions and it is defined by (13.10); the regularizing term Θ and the penalty term Ψ (which also plays a role in regularization) are defined as follows.

The regularizing term Θ is defined by:

$$\Theta = \epsilon \|z - \bar{z}\|^2 \tag{13.30}$$

where ϵ is a given positive (not necessarily small) number, z denotes a vector composed of decision variables that correspond to emissions, and $\bar{z}$ is a given vector composed of desired (reference) values of emissions. This term for small values of the parameter ϵ plays the role of a regularizing term, i.e., it assures that the optimal solution (for a problem that does not have a unique local optimal solution) will be the optimal solution closest to the point defined by the given reference vector $\bar{z}$. Its role is twofold. First, it helps to avoid large variations of solutions (with almost the same value of the original goal function) for problems that differ very little. Second, it substantially improves the numerical stability of the optimization problem.

Additionally, for large values of the parameter ϵ, the term Θ can be used for the technique known as softly constrained inverse simulation. Thus, it is possible to analyze trade-offs between minimization of costs and solutions that correspond closely to various given patterns of emissions defined by $\bar{z}$.

The role of the term Ψ is twofold. First, it optionally serves as a penalty term for variables y, ya, and ye. Second, it provides regularization for these decision variables, which are not covered by the Θ term. The term Ψ is defined by:

$$\Psi = \sum_{l \in L} \sum_{j \in J} \psi(y_{lj}, \rho_o, \sigma_o) + \sum_{j \in J} \psi(ya_j, \rho_a, \sigma_a) + \sum_{j \in J} \psi(ye_j, \rho_e, \sigma_e) \quad (13.31)$$

where $\rho_o, \rho_a, \rho_e, \sigma_o, \sigma_a, \sigma_e$ are given positive parameters, and the function $\psi(\cdot)$ is defined by:

$$\psi(x, \rho, \sigma) = \begin{cases} -\rho\sigma x - \rho\sigma^2/2 & \text{for } x < -\sigma \\ \rho/2x^2 & \text{for } |x| \leq \sigma \\ \rho\sigma x - \rho\sigma^2/2 & \text{for } x > \sigma. \end{cases} \quad (13.32)$$

Note that $\psi(x, \rho, \sigma)$ is a smooth function that, depending on the parameters ρ and σ, can be used for both purposes corresponding to the role of the term Ψ outlined above. For large values of the parameters ρ and σ, it plays a role of a classical quadratic penalty function. Such a function can be used for examination of trade-offs between violations of the air quality standards and minimization of costs. Additionally, for some scenarios in which the balances between violations of environmental targets and surpluses – given by equations (13.25) to (13.28) – are accounted for, it may not be desirable to apply any penalty function. However, in such cases it is still necessary to apply regularization in order to deal correctly with the soft constraints defined by the introduction of variables y_{lj}, ya_j, ye_j into constraints (13.21) to (13.24). A quadratic function is not suitable for this purpose because often violations and surpluses take small values in some grids and large values in others. Therefore, it is not possible to find a value of the parameter ρ such that it would allow for large values of violations/surpluses in some grids, while serving as a regularizing term for other grids where violations/surpluses may be three orders of magnitude smaller. Consequently, when used for regularization purposes alone, the function ψ is defined using small values of both parameters ρ, σ, which implies using a flat piece-wise linear function with a small quadratic segment needed to make such a function smooth.

We summarize the discussion on the form of the goal function (13.29) by stressing the fact that a properly defined goal function is the key element for achieving two goals. These goals are, namely, providing a tool for a comprehensive problem analysis and assuring possibly good numerical properties of the optimization problem. The specific form of this model (in particular, the penalty terms for soft constraint violations – the regularizing terms) makes it very similar to a multi-objective formulation. However, due to the size of the model, an application of typical interactive tools for multi-criteria model analysis is not possible. Therefore, the multi-objective methodology is used implicitly: a goal function should be suitable to allow for generation of various parametric single-objective optimization problems that facilitate analysis of trade-offs between conflicting criteria. Finally, a goal function formulation should improve numerical properties of the optimization problem. This role of the goal function is often forgotten because it does not have an obvious interpretation in terms of the problem that is being analyzed. However, this element is very important for obtaining solutions that are not only correct from the theoretical (optimization) point of view, but which also have properties important for their substantive interpretation. For the latter, a regularization of the optimization problem is typically a very desirable feature.

13.4 Model Generation and Analysis

The generation and analysis of a mathematical programming problem that corresponds to the model defined in Section 13.3 is a challenging task from the operations research point of view. The problem is a large-scale nonlinear optimization problem with a large linear component. The following methodological and technical issues are of broader interest and are discussed in subsequent sections:

- The generation and solution of a nonlinear optimization problem.
- Data handling.
- The representation of a PWL function by a smooth function.
- Preprocessing of a nonlinear optimization problem.
- Scaling.

A more detailed discussion of various modeling paradigms applied to the generation and analysis of the RAINS model and a more technical description of one cycle of its analysis can be found in Makowski (2000).

13.4.1 Generation and solution of a nonlinear optimization problem

The model defined in Section 13.3 is a large nonlinear model. A commonly accepted "rule of thumb" is to try various solvers for optimization of a nontrivial nonlinear model. Therefore three solvers, namely CFSQP (Lawrence *et al*., 1996), CONOPT (Drud, 1996), and MINOS (Murtagh and Saunders, 1987) are used for solving the resulting optimization problem. For reasons that have already been discussed in Section 6.5.1, a problem-specific model generator has been implemented in C++ for the RAINS model.

The implementation of software that uses several solvers is interesting from a software–engineering perspective. Each solver has a different interface (i.e., the way of formulating an optimization problem). However, the majority of the software components are common to all solvers. An object-oriented programming approach is therefore a natural choice because it greatly simplifies the software development task by handling common parts in base classes and by providing solver-specific interfaces through inherited classes. The approach is conceptually very simple: each of the solvers is available as a library of Fortran subroutines. The generator is composed of C++ classes that are specific for each solver. These classes are inherited from base classes, which handle a common part (for all solvers) of functions that has to be provided by the generator. A problem-specific report writer processes the results into an easily interpretable form. Another class supports a portable interface (between Unix and NT) between C++ and Fortran. Hence, three versions of executables can be easily produced, and each is composed of the generator, the report writer (postprocessor), and one of the solvers. This allows the user easy handling of computations of series of scenarios using various solvers. The code is operational on Unix (Solaris) at the time of writing. However, the code is portable, therefore it should be easy to implement its version for NT.

A nonlinear solver requires routines that compute values and the Jacobian of the constraints and goal function. CONOPT and MINOS require such functions only for a nonlinear part, and CFSQP also requires it for a linear part of the optimization problem. A substantial part of total computation time is used for execution of these functions; therefore, the efficiency of their implementation is important. The code for the Jacobian of the nonlinear part has been generated by Mathematica (Wolfram, 1996), with prior use of the FullSimplify operator that substantially simplifies the formulas. The formulas obtained from Mathematica have been modified (by an appropriate handling of common expressions) in order to optimize computation time. This is an efficient way to generate a bug-free and efficient code, which is required by all nonlinear solvers.

Finally, one should notice that the dimensions of the model are not fixed. For some scenarios a part of the constraints and/or variables do not need to be generated. Moreover, the dimensions of matrices and vectors used in the model definition vary substantially for various types of analysis. Fortunately, constructors of C++ template classes handle such problems in a natural and efficient way.

13.4.2 Data handling

The data required for the specification of one instance of the model can be logically divided into three sets:

1. A small number of control parameters that either specify file names or select various options available for model generation and analysis. These data are provided in a small text file.
2. Parameter values that are specific for a given scenario (e.g., bounds for emissions, and air quality targets).
3. All other parameter values used for the model specification.

The model has a large number of parameters, but this in itself would not be a problem. The challenge comes from the fact that various subsets of the parameter values are provided as a result of data processing that is performed on numerous computers. Data handling for the model has to meet the following requirements:

- Efficient handling of large amounts of data.
- Binary compatibility, at least for Unix and NT.
- Easy handling of basic data structures (sparse and dense matrices having elements of basic types).
- No royalty fees.

The hierarchical data format (HDF) public domain software (Koziol and Matzke, 1998) is used for handling data in the model. The basic data structures are handled by a collection of well tested C++ template classes; this collection is also used for the LP-DIT tool described later (see Section 13.4.4). In order to use the HDF library efficiently, an interface C++ template class has been implemented for supporting easy handling by HDF of the used data structures. A prototype implementation of the use of HDF with the model is documented by Haagsma (1996).

13.4.3 Representation of a PWL function by a smooth function

Costs of emission reductions are given (cf. Section 13.3.3) as PWL functions of the emission level. PWL functions are not smooth, therefore, the most efficient nonlinear solvers (which require smooth functions) cannot be used directly for solving a problem having PWL functions.

This section provides an outline of the approach applied for representing a PWL function by a smooth function (SF). For the sake of brevity we will use notation that is local to this section.

Assume that a PWL function $f(x)$ is composed of K segments defined by $K+1$ pairs:

$$\{p_k, y_k\} \qquad k \in [0, K] \tag{13.33}$$

where p_k are values of the argument of this function and y_k are the corresponding function values. Therefore, $f(x)$ can be defined as:

$$f(x) = \alpha_k x + \beta_k \qquad x \in [p_{k-1}, p_k] \qquad k \in [1, K] \tag{13.34}$$

where

$$\alpha_k = \frac{y_k - y_{k-1}}{p_k - p_{k-1}} \tag{13.35}$$

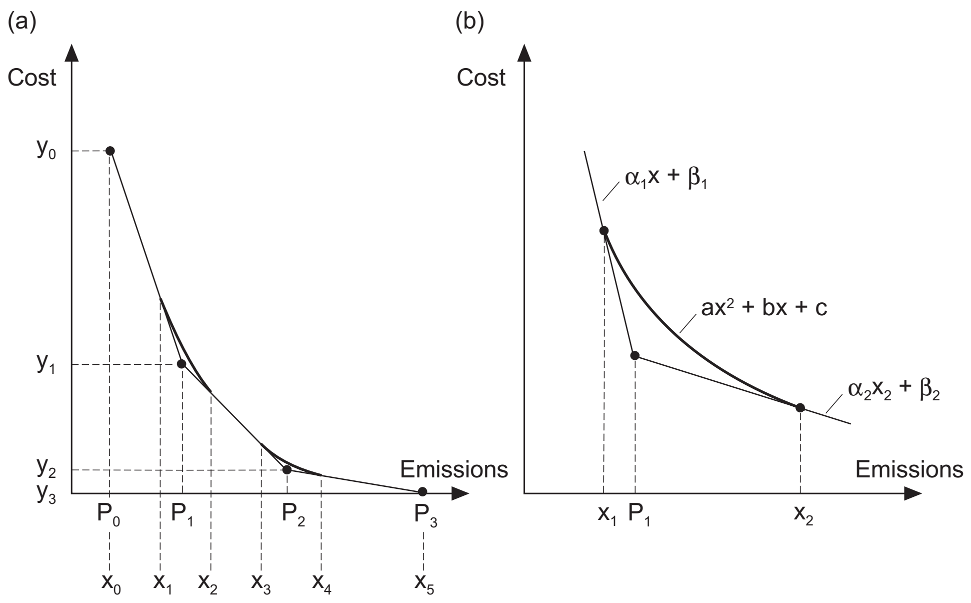


Figure 13.2. Illustration of a smooth function representing a piece-wise linear function; (a) shows a function that approximates a PWL function composed of three segments and (b) shows an enlargement of this function in the neighborhood of the vertex p_1.

$$\beta_k = y_k - \alpha_k p_k. \tag{13.36}$$

The function $f(x)$ is supposed to be strictly convex, therefore

$$\alpha_{k-1} < \alpha_k \qquad k \in [1, K]. \tag{13.37}$$

The following piece-wise quadratic smooth function $sf(x)$ composed of I segments ($I = 2K - 1$) is used as a representation of the corresponding PWL function illustrated in *Figure 13.2*:

$$sf(x) = a_i x^2 + b_i x + c_i \qquad x \in [x_{i-1}, x_i], \qquad i \in [1, I]. \tag{13.38}$$

Such a function is composed of large parts of linear segments that correspond to the original PWL function and short quadratic parts, which replace the pieces of the original segments in the neighborhoods of vertices.

A detailed analysis of how the parameters of the function $sf(x)$ are defined is beyond the scope of this book. However, Makowski *et al.* (1998) address this issue. The approach outlined here guarantees good approximations of the non-smooth cost functions by smooth functions that also have acceptable values of second derivatives.

13.4.4 Preprocessing a nonlinear optimization problem

It is commonly known that preprocessing a large optimization problem can dramatically reduce computation time. Preprocessing of LP problems is a standard feature of any good LP solver. However, preprocessing of nonlinear models is a much more difficult task, which cannot be done efficiently by a general purpose nonlinear solver (see Drud, 1997, and also Chapter 7). Therefore, preprocessing the optimization problem in a problem generator is a much more efficient way than an attempt to preprocess a nonlinear problem within a nonlinear solver.

Preprocessing in the generator is divided into three parts:

- Outcome variables defined by equations (13.9) through (13.14) are not generated. Instead of generating outcome variables, the affected constraints are reformulated to equivalent forms without use of the outcome variables; auxiliary functions are implemented to provide values of outcome variables for the report writer.
- All linear constraints are combined into LP-DIT format (see Makowski, 1998), and the preprocessing implemented in LP-DIT is applied to these constraints. Unfortunately, only part of the LP preprocessing methods can be applied, namely the part based on the analysis of the primal problem. Nevertheless, for many types of scenarios, even a majority of linear constraints can be removed from the optimization problem that is passed to the nonlinear solver.
- Preprocessing of the nonlinear part is now (at the time of writing) limited to the reduction of variables en_j and nlv_j and elimination of equations (13.7) and (13.8).

Preprocessing substantially reduces the computation time and memory requirements needed for solving an instance of the RAINS model. This confirms theoretical expectations. However, while preprocessing of linear models is done by LP solvers (and therefore a model developer need not worry about this), preprocessing of nonlinear models still requires a substantial amount of consideration and time for a problem-specific implementation. Nevertheless, the experiences from application of preprocessing encourage further research aimed at more general solutions as well as at implementations of problem-specific preprocessing to other nonlinear models for which solution time is an important factor.

13.4.5 Scaling

Scaling of a model is closely related to model specification, which is discussed in more detail in Section 6.6. Good scaling of a linear model that contains non-substantive coefficients might be impossible (see Section 6.6 for an explanation).

LP models having only substantive coefficients are typically well scaled by any LP solver. For nonlinear models, however, the inclusion of only substantive elements into a specification is a necessary, albeit not a sufficient, condition for its good scaling. This is a reason why scaling is typically not implemented in nonlinear solvers and it is the responsibility of a model developer to provide a well-scaled optimization problem into a nonlinear solver.

All parameters of the RAINS model are verified in order to include only substantive coefficients (see Section 13.2.3). Filtering out small coefficients (in absolute terms) of the model is the first and necessary condition for a good scaling of a model. However, a nonlinear model requires a problem-specific scaling procedure.

A detailed discussion of scaling implemented in the model is far beyond the scope of this book. Therefore we only mention that each instance of the optimization problem is scaled in the generator to ensure two things. First, absolute values of the elements of the Jacobian and of the Hessian are smaller than 10^5. Second, an attempt is also made to ensure that the smallest (nonzero) absolute values of the Jacobian are "not too small" (in the current implementation greater than 10^{-5}). It is impossible to achieve it for the cost functions that have "steep" and "flat" parts (for low and high emissions, respectively); in such cases the elements of the Jacobian corresponding to the "flat" part are small (which in turn often results in a suboptimal solution).

Scaling of nonlinear models is an important element of model specification. Our experiences from early stages of model development show that a badly scaled model creates numerical problems for all three solvers used.[2] Scaling of LP models is efficiently done by LP solvers. However, scaling of nonlinear models cannot be automatized and therefore it remains the responsibility of the model developer.

13.4.6 Additional remarks on analyzing large-scale nonlinear models

A general discussion on model specification and the solution of nonlinear optimization problems is provided in Chapters 6 and 7, respectively. A more specific discussion on the generation and solution of optimization problems that corresponds to an analysis of various scenarios of the RAINS model is provided in Section 13.4.1. In this section we briefly summarize some of our other experiences that are of wider interest for practitioners who work with large-scale nonlinear models.

We start with a summary of the problem size. The size of the optimization problem depends on the selection of various options. For scenarios that result in large

[2]CONOPT provides a very useful diagnosis by making an analysis at a starting point and refusing to solve a badly scaled problem.

optimization problems, such problems have about 14,000 variables (12,000 nonlinear,[3] 2,000 linear), 18,000 constraints (3,500 nonlinear, 14,500 linear), and about 240,000 elements of the Jacobian (60,000 nonlinear, 180,000 linear). Note that these sizes would be substantially larger (e.g., the number of nonlinear constraints doubled, and the number of other constraints and variables also considerably increased), if a traditional approach to the model specification would be adopted and outcome variables (or separate variables for violations and surpluses) generated. This observation alone stresses the importance of making a careful distinction between the model formulation for a user and the corresponding specification of the model, which is generated and solved.

The CPU time (on Sun Sparc Ultra-4, 300 MHz) needed for solving a large problem ranges from 2 to 70 hours. In our experience a typical round of analysis requires optimization of dozens of scenarios. Therefore, avoiding long computation time is always one of the top priorities. Unfortunately, there is no guaranteed recipe to ensure that computation time is kept at a minimum. However, we summarize the main measures that have helped to reduce the CPU time substantially. Such measures can be grouped into the following categories: (i) problem specification; (ii) preprocessing, including scaling; (iii) regularization; and (iv) sequence of scenarios and advanced starting points.

The first two have already been discussed but are included to stress their primary importance.

The importance and interpretation of regularization has been discussed in Section 13.3.5. However, an additional comment on a selection of parameters of the function $\psi(x, \rho, \sigma)$, defined by (13.32), can help to ensure an efficient use of such a function in other applications. Obviously, for large enough values of the parameter σ, the function $\psi(\cdot)$ is equivalent to the classical quadratic penalty function. However, an application of such a function in a regularizing term for a problem for which both small values of the parameter ρ should be set and small values of some variables are expected, is practically ineffective. This is because the corresponding terms will be smaller than tolerances used in optimization. As a result, the corresponding optimization problem has bad numerical properties that often result in a substantial increase of optimization time and/or in problems with identifying a solution as an optimal solution. Therefore in such situations it is much better to apply the function $\psi(\cdot)$ with small values of both parameters ρ and σ. This effectively results in a flat smooth and symmetric function composed of a small quadratic segment (the easiest way to get a smooth and symmetric function) and of two linear

[3]A variable is considered nonlinear if it enters a nonlinear constraint; a constraint is considered nonlinear if it has at least one nonlinear element of the Jacobian. In our model the nonlinear constraints are defined by (13.21), therefore each of them has mostly nonlinear elements.

segments. Such a function provides good regularization for all corresponding variables while allowing some of the variables to assume relatively large values. One should note that an application of a quadratic penalty/regularizing function with different penalty parameters for various variables (which is an idea appealing to many practitioners) is not practical in our case (because it is not possible to identify rational penalty parameters for each variable).

Planning a sequence of scenarios that can be run in a row, each (excluding the first) using an advanced starting point, is a problem-specific issue. However, its implementation can provide substantial savings in CPU time, therefore it is worthy of further exploration. The only general advice we can offer is to start with a problem that has the smallest feasible set (i.e., the tightest constraints, or does not have soft constraints), and rank subsequent scenarios according to growing feasible sets. From our experience, scenarios with larger penalty coefficients ρ for the variables representing violations of original constraints (variables y in our soft constraints) and with smaller regularizing coefficients ϵ should be computed prior to scenarios with smaller ρ and larger ϵ.

A small technical detail of practical value for optimization problems that take several days to solve is periodic dumping of information. This information can be used later for a specification of an advanced starting point (with an additional bonus of providing the possibility of continuation of computation without too many losses in case of hardware problems). Fortunately, nonlinear solvers have to communicate with functions that are submitted by users for computation of nonlinear constraints and nonlinear elements of the Jacobian. Therefore an advanced user can take this opportunity to store information when one of the user-supplied functions is executed. We have implemented a timer class that is used within the function to compute the value of the objective function. This allows for storing the current values of all decision variables at user-specified CPU time intervals at negligible cost.

Finally, we would like to point out that an application of even the best nonlinear solvers by practitioners who have a reasonable understanding of optimization techniques often requires consultations with authors of a solver. Putting aside problems that are documented (which sometimes even require additional explanations), there are more subtle issues (starting with alternative problem formulations, to the adjustment of dozens of optimization parameters, and ending with various versions of a solver) where consulting authors of solvers is an invaluable alternative to time-consuming experimenting that may not necessarily result in success. Our experience from contacting authors of nonlinear solvers is excellent: we always receive prompt and useful advice, and also an updated version of a solver when necessary. Therefore, we suggest that practitioners do not hesitate to contact authors of solvers whenever they are uncertain and/or the computation time is too long.

13.5 Analytical Support for Negotiations

So far the discussion has dealt with a new, nonlinear part of the RAINS model related to the management of tropospheric ozone. Earlier versions of the RAINS model (which was a linear model used for the analysis of acidification and eutrophication) were used extensively, particularly for analytical support of international negotiations. In the following, we present lessons derived from these applications and then discuss some of the recent results, which illustrate selected elements of the analysis of the nonlinear model.

The RAINS model was used during the last decade to provide analytical support for the negotiations that led to several important international agreements on emission control in Europe.

The Second Sulphur Protocol of the UN/ECE LRTAP Convention

In 1994, the Parties to the UN/ECE LRTAP Convention signed the Second Sulphur Protocol, which establishes country-specific obligations for reductions of SO_2 emissions ranging from 0% to 86%. These national emission ceilings were the outcome of a negotiation process, which was heavily reliant on a series of scenario and optimization analyzes performed with the RAINS model. The rationale for the nonuniform allocation of emission reductions was cost-minimization, driven by the environmental objective of reducing excess sulfur deposition above the critical loads by at least 60%. With only a few exceptions, the negotiated and accepted national obligations did not deviate by more than ± 3 percentage points from the underlying optimization result produced by the RAINS model.

The Second NO_x Protocol of the UN/ECE LRTAP Convention

The Parties of the Convention are presently negotiating the Second NO_x Protocol, aiming at a multipollutant (emissions of SO_2, NO_x, NH_3, and VOC)/multieffect (acidification, eutrophication, ground-level ozone, and human health) approach. IIASA's RAINS model is the main tool for analyzing the cost effectiveness of alternative emission control scenarios, and the basic model concept closely follows the approach described above. Results of IIASA's analysis are regularly reviewed by the UN/ECE Task Force on Integrated Assessment Modelling and the Task Force for Economic Aspects of Abatement Strategies, and are presented to the negotiating body of the Convention, i.e., the Working Group on Strategies.

The Approach to Air Quality Control of the European Union

A few years ago the Commission of the European Union embarked on an effect-focused rationale for guiding further emission controls. A basic driving force was the decision laid down in the Fifth Environmental Action Plan to establish the full achievement of critical loads and critical levels as a long-term objective for the EU environmental policy. When designing the practical strategy of the air quality management, the Commission adopted a two-step approach. In an initial step, the Commission proposed environmental targets for acidification and ground-level ozone and attempted to reach agreement on them with the member states and the European Parliament. The cost-effective allocation of emission controls for achieving these targets would subsequently provide the basis for a proposal of a directive on national emission ceilings.

The interim environmental targets proposed by the Commission were motivated by the finding of integrated assessment models; these showed that until 2010 the full achievement of critical loads and critical levels would be extremely difficult, if not impossible. The RAINS model has been used to explore ranges of interim targets for acidification and ground-level ozone and to explore the implied distribution of costs and environmental benefits, if these targets were to be attained in a cost-effective way.

In 1997, the Commission proposed an "Acidification Strategy", which established, inter alia, the target to reduce the area of unprotected ecosystems by at least 50% (with acid deposition above their critical loads) by the year 2010. This proposal was subsequently discussed by the ministerial council and by the European Parliament.

At present, work is progressing on developing the EU "Ozone Strategy", again with the objective to agree on environmental interim targets for ground-level ozone. A series of interim reports, based on analysis conducted with the RAINS model, explores the implications of alternative target setting rules, the implied costs and environmental benefits for the individual member states, and the robustness of the optimization results against changes in important input assumptions (e.g., pre/post Kyoto agreement energy scenarios, emission controls in non-EU countries). Particular attention is devoted to the appropriate treatment of extreme situations, i.e., whether an interim strategy should allocate major resources to avoid extreme (and perhaps rather rare) events, or prioritize the improvement of "typical" situations, where the overall benefits might be higher. Another important aspect is the dependence between emission controls targeted at ground-level ozone, and acidification-related objectives, with the aim to rely as much as possible on existing synergisms while avoiding situations where clear trade-offs emerge. This dependence is a practical concern for the reduction of NO_x emissions, which always result in environmental benefits for acidification, but may, under certain conditions, deteriorate ozone levels.

Table 13.1. Summary of the environmental targets for the optimized scenario.

	Targets
Acidification	
Reduction ("gap closure") of accumulated excess acidity	95%
Maximum excess deposition for 2% of the most sensitive ecosystems	850 eq/ha
Health-related ozone	
Reduction ("gap closure") of the health relevant exposure criterion (AOT60)	67%
Maximum AOT60, to be achieved in 4 out of 5 years	2.9 ppm·h
Vegetation-related ozone	
Reduction ("gap closure") of the health vegetation exposure criterion (AOT40)	33%
Maximum excess AOT40, mean over 5 years	10 ppm·h

Abbreviations: eq = acid equivalents; ha = hectare; ppm·h = parts per million per hour.

13.5.1 An optimized scenario for meeting environmental interim targets in Europe

Earlier it was shown that cost effectiveness implies differentiated requirements for emission reductions, taking into account regional differences in environmental sensitivities, differences in the potential and the costs for further emission reductions, and in meteorological conditions (Amann *et al*., 1998). The observed variations of these factors in Europe lead to the observation, however, that the burden for additional emission control measures imposed by cost-optimized strategies on individual European countries might also be differentiated.

As an example of a cost-optimized scenario presented in the course of European negotiations, optimized reductions of SO_2, NO_x, NH_3, and VOC emissions for meeting the environmental targets listed in *Table 13.1* are provided in *Tables 13.2* and *13.3*. It can be seen from these tables that optimized emission reductions impose differentiated requirements on the individual countries.

13.5.2 A flat-rate emission control scenario (FR)

In order to facilitate political acceptance, many conventional international environmental agreements attempt to pursue a concept of equity among the participating parties. One notion of equity frequently used in such environmental accords is to limit total national emissions in relation to the levels of a common base year. Uniform cutbacks of emissions apply to all countries.

While such uniform reduction schemes have their merits in facilitating political agreements, usually they do not result in cost-effective solutions, as is clearly

Table 13.2. Emissions for the optimized scenario (OPT) compared to the reference case (REF), the starting point for the optimization. Percentage changes relate to the year 1990.

	SO_2				NO_x				VOC				NH_3			
	REF		OPT		REF		OPT		REF		OPT		REF		OPT	
	kt	Change	kt	Change	kt	Change	kt	Change	kt	Change	kt	Change	kt	Change	kt	Change
Austria	42	–55	42	–55	113	–41	94	–51	208	–41	133	–62	67	–13	67	–13
Belgium	208	–38	64	–81	207	–41	111	–68	212	–47	103	–74	96	–1	60	–38
Denmark	90	–51	48	–74	136	–50	136	–50	86	–47	86	–47	72	–6	70	–9
Finland	116	–50	116	–50	162	–41	162	–41	112	–47	112	–47	31	–23	31	–23
France	489	–61	256	–80	1,044	–44	757	–59	1,242	–48	866	–64	798	–1	727	–10
Germany	608	–88	453	–91	1,263	–53	1,062	–60	1,137	–63	947	–69	571	–25	396	–48
Greece	562	12	562	12	344	0	338	–2	205	–39	202	–40	74	–8	74	–8
Ireland	70	–61	32	–82	81	–28	66	–42	46	–59	46	–59	126	–1	122	–4
Italy	593	–65	593	–65	1,186	–42	879	–57	1,176	–43	935	–54	416	–10	416	–10
Luxembourg	4	–71	4	–71	10	–55	5	–77	8	–58	5	–74	7	0	7	0
Netherlands	74	–63	53	–74	312	–42	261	–52	241	–51	154	–69	136	–42	106	–55
Portugal	146	–49	146	–49	197	–5	112	–46	144	–34	127	–41	67	–6	67	–6
Spain	793	–64	759	–65	892	–23	822	–29	669	–36	669	–36	353	0	353	0
Sweden	67	–44	67	–44	200	–41	181	–46	287	–42	235	–52	48	–21	48	–21
UK	980	–74	537	–86	1,186	–58	1,186	–58	1,351	–49	1,032	–61	297	–10	264	–20
EU-15	4,842	–70	3,731	–77	7,333	–45	6,171	–53	7,123	–49	5,651	–60	3,159	–12	2,807	–22

Abbreviation: kt = kilotons; OPT = optimized scenario.

Table 13.3. Emission control costs for the optimized scenario (OPT) compared to the reference case (REF). Control costs are in million ECU/year.

	SO_2			NO_x/VOC			NH_3			Total		
	REF	OPT	Total	REF	OPT	Total	REF	OPT	Total	REF	OPT	Total
Austria	174	0	174	784	137	921	0	0	0	958	137	1,095
Belgium	341	193	534	1,000	856	1,856	0	312	312	1,341	1,361	2,702
Denmark	115	21	136	383	0	383	0	1	1	498	22	520
Finland	204	0	204	525	0	525	0	0	0	729	0	729
France	1,004	123	1,127	6,180	1,265	7,445	0	65	65	7,184	1,453	8,637
Germany	2,146	554	2,700	8,704	1,369	10,073	0	1,258	1,258	10,850	3,181	14,031
Greece	331	0	331	933	19	952	0	0	0	1,264	19	1,283
Ireland	108	19	127	410	10	420	9	49	58	527	77	604
Italy	1,577	0	1,577	6,881	640	7,521	12	0	12	8,470	640	9,110
Luxembourg	9	0	9	60	45	105	15	0	15	84	46	130
Netherlands	306	17	323	1,486	246	1,732	237	680	917	2,029	944	2,973
Portugal	152	0	152	1,092	284	1,376	0	0	0	1,244	284	1,528
Spain	678	13	691	4,793	17	4,810	28	0	28	5,499	30	5,529
Sweden	293	0	293	976	40	1,016	113	0	113	1,382	40	1,422
UK	1,148	238	1,386	5,934	643	6,577	0	23	23	7,082	904	7,986
EU-15	8,586	1,178	9,764	40,140	5,572	45,712	413	2,389	2,802	49,139	9,139	58,278

shown by the following example. In this example, the average reduction percentages of the optimized[4] scenario (for four individual pollutants) presented above is applied to all countries. This implies that all countries would reduce their SO_2 emissions by 77% in relation to 1990, their NO_x emissions by 53%, their VOC emissions by 60%, and their NH_3 emissions by 22%. In practice these reductions imply different stringencies (due to different rates in economic growth, etc.), therefore, the overall costs of such a scenario are about 10% higher than in the optimized case.

Despite the increased costs, detailed analysis demonstrates that such a uniform reduction scenario would result in a lower environmental improvement – for the EU-15 as a whole – than the optimized scenario. For acidification, the uniform scenario would achieve full protection for 18% fewer ecosystems than the optimized scenario. For health-related ground-level ozone, the exposure index of this scenario is 19% higher than for the optimized case.

A graphical comparison of the changes in the environmental indicators in relation to emission control costs is provided in *Figures 13.3* and *13.4*. These figures show clearly that, for the EU-15 as a whole, flat-rate emission reductions of the FR scenario result in a significantly lower cost effectiveness for all three environmental problems considered (acidification, health-, and vegetation-related ozone exposure).

13.5.3 Scenarios gradually moving toward uniform emission reductions

Another series of scenarios was developed with the aim of keeping emission reductions as uniform as possible within the EU-15 countries but at the same time ensuring that the environmental targets of the optimized scenario would be achieved.

In practice, this was achieved by including a regularization term Θ, defined by equation (13.30), in the goal function of the corresponding mathematical programming problem. This term for large values of the parameter ϵ can be interpreted as a penalty on each deviation of an optimized emission reduction level from an exogenously specified "target" emission level. Therefore the vector z used in the definition of Θ is composed of the decision variables (emissions relative to 1990) and the vector $\bar{z}$ is defined by the uniform target emission levels (relative to 1990).

Depending on the values of the coefficients ϵ, various trade-offs between the deviations from these target levels and the overall emission control costs can be examined. For a sufficiently small value of ϵ the term Θ is just a regularizing term; therefore the optimization ends up with the emission levels of the original OPT scenario. Increasing the value of ϵ ultimately pushes all emission reductions to the

[4]Assuming the objective to be the sum of the emission reduction costs for all countries.

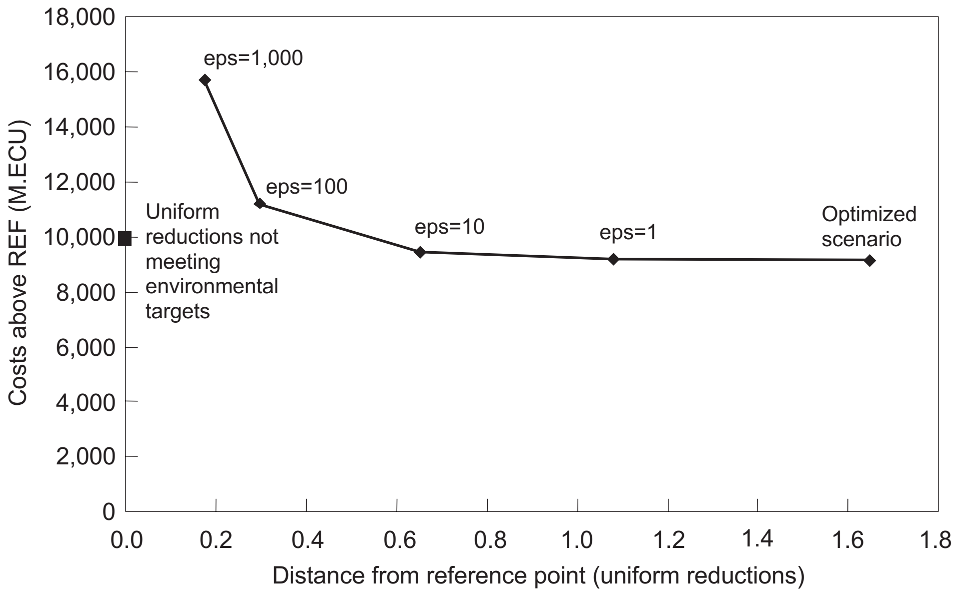


Figure 13.3. Emission control costs (above REF) of the uniform reduction scenario and the sensitivity runs with different regularization coefficients ϵ compared to those of the optimized scenarios.

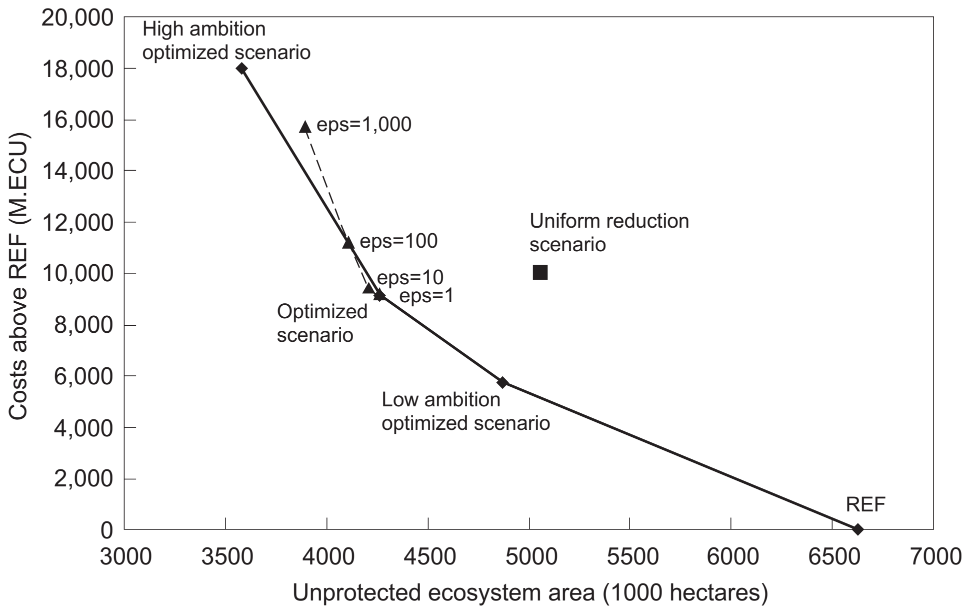


Figure 13.4. Cost effectiveness in terms of the ecosystems protection (acidification) of the uniform reduction scenario and the sensitivity runs with different regularization coefficients ϵ compared to those of the optimized scenarios.

target levels of the FR scenario (if these achieved the OPT targets). The latter corresponds to the softly constrained inverse simulation as discussed in Section 13.3.5.

To this end, four scenarios were carried out with values for regularization coefficients ϵ equal to 1, 10, 100, and 1,000, respectively. *Figure 13.3* shows the changes in national emission control costs for these four scenarios. Maintaining the environmental targets of the optimized scenario, a reduced variability of national emission reductions will lead to increased costs (up to 60% for the most stringent scenario analyzed). As an example, *Figure 13.4* compares the cost effectiveness of the optimized and nonoptimized scenarios in relation to the area of ecosystems protected against acidification. Again, it is clear that there is a certain price to be paid for lower variability in national emission reduction obligations.

13.6 Conclusions

This chapter illustrates methodological and software engineering issues pertinent to the generation and analysis of complex and large nonlinear models as discussed in Part II of this book, but illustrated here by their applications to a complex, real-world problem.

Identification and examination of various cost-effective policy options aimed at improving the air quality in Europe is based on a large-scale complex model, which has been developed over several years by the Transboundary Air Pollution (TAP) project team at IIASA. The team is composed of specialists in various fields who have been collaborating closely with several other groups affiliated at various European institutes. The specifications of the model have been modified over these years in order to better fit the requirements of the users. This has also resulted in implementation of extensions of traditional OR methods that greatly enhance the usefulness of various optimization techniques for policy analysis. Due to the size of this nonlinear model no typical interactive multi-criteria model analysis can be applied to its analysis. However, it has been shown how various formulations of single-criterion optimization problems, using some methodological concepts related to multi-objective model analysis, can provide similar advantages for a more complete model analysis.

The presented model also serves as a good illustration of the methodological and software engineering issues discussed in Chapters 3 and 6. It shows that the specification, generation, analysis, and maintenance of such a large, complex, nonlinear model require knowledge, experience, and collaboration of interdisciplinary teams. It is also an example of a problem for which problem-specific software had to be developed because no general purpose modeling system was suitable for this application because of its complexity, requirements for efficiency, and distribution. However, although the software is problem-specific, it is built using a number of modular tools that can be easily adapted for other problems.

Acknowledgments

The RAINS model has been developed over several years by the TAP (Transboundary Air Pollution) project[5] at IIASA in collaboration with several European institutions. The authors of this chapter (the first leading the TAP project and the second working with methodological projects at IIASA) have been collaborating for several years in the development of methodology and software for generation and optimization-based analysis of the RAINS model. The authors acknowledge all members of the TAP project for their participation in this long-term activity. In particular, the authors thank Dr. Chris Heyes and Dr. Wolfgang Schöpp for their contributions to the formulation and development of the RAINS model.

The authors thank Dr. Arne Drud of ARKI Consulting and Development A/S, Denmark, for continuously providing the latest versions of the CONOPT libraries (Drud, 1996), and for his consultations on advanced topics related to formulation and solving nonlinear optimization problems.

Finally, the authors acknowledge and appreciate the contribution of three participants of the IIASA's Young Scientists Summer Program, namely Piotr Zawicki and Paweł Białoń, both of Warsaw University of Technology, who have developed the prototypes (cf. Zawicki, 1995, and Białoń, 1996, respectively) of model generators for the previous versions of the RAINS model, and Ijsbrand Haagsma of Delft University of Technology, who has implemented a prototype for data handling through HDF (a platform independent software for data storage; see Haagsma, 1996).

[5]http://www.iiasa.ac.at/∼rains/.

Chapter 14

Energy Planning

Sabine Messner, Manfred Strubegger, and Andrzej P. Wierzbicki

14.1 Decision Support for Energy Planning

Energy planning is a general term that is applied to a variety of issues. It can address, for example, the design of energy supply and utilization in new buildings; it can also address municipal planning of district heat supply and the structure of heating systems. In national energy planning, the focus is on political targets such as a diversification of energy sources or environmental targets such as a reduction in acidification of soil and lakes. International bodies – e.g., the World Energy Council (WEC), the International Energy Agency (IEA), and the Intergovernmental Panel on Climate Change (IPCC) – investigate the future of our energy system on a multinational or even global level (e.g., IPCC, 1990; Houghton and Callander, 1992; or WEC, 1992). Many research institutions and scientists focus their research on the long-term future of the global energy system (see e.g., Häfele, 1981; Darmstadter and Landsberg, 1983; Guilmot *et al.*, 1986; Goldemberg *et al.*, 1988; Manne and Richels, 1992).

14.1.1 Objectives in energy planning

The targets of such investigations differ depending on the scope of the problem and the decision makers involved. Industrial bodies and energy utilities strive for strategies with minimal costs. In this case other objectives, such as environmental or social aspects, are typically viewed as constraints. As an example, fuel use in a power plant can be constrained due to site-specific regulatory emission limits. We see, however, that such constraints are actually soft, not hard, which can result in various difficulties when they are treated in a classical way (compare Chapters 5, 6, and 11). Soft constraints should instead be modeled by additional objectives in a multi-objective formulation.

Three types of issues are addressed by industrial or utility decision makers:

- Long-term planning (e.g., supporting decision makers for periods of 15–20 years), especially for investment decisions.
- Medium-term planning for one or two years, generally addressing resource allocation, contract and plant management.
- Short-term operation planning for the day or week, deciding on plant schedules and unit commitment.

National or international bodies, politicians, and scientific investigators focus directly on the political, social, and environmental aspects of the problem. Such investigations generally emphasize the longer term, focusing on 20 to 50 years, while medium-term analyses in this field have a time frame of some 5 to 10 years. Recently, the issue of global warming – with the very long time constants involved – has triggered a multitude of energy-related studies with a time horizon up to 2100 or even beyond. Such investigations are not directly targeted toward investment or operation decisions, they rather provide a foundation for long-term policy decisions, legal acts and international treaties, and provide a basis for focusing R&D funds on specific problems.

14.1.2 Models for energy planning

A wide variety of models have been employed in energy modeling. Four types have found most widespread use, and are listed below.

- *Model 1.* Simulation models that mimic the technical features of the modeled system. Their results are directly determined by the input. Such models are useful to evaluate fully defined systems with no degrees of freedom. Open questions have to be investigated by the user prior to model application. These models are used for technology-oriented analyses, e.g., to determine the size of the heating system and the degree of insulation for different housing types (Hirst, 1980; Schmits *et al.*, 1981; Fiala, 1982; Vogt and Mickle, 1983).
- *Model 2.* Econometric models expanding past behavior into the future. Econometric estimates of time series are the basis for models explaining one variable, e.g., energy use, by some driving force, such as income or prices. These models

are mainly used for evaluating demand and investigating consumer responsiveness to prices (see Taylor *et al.*, 1982; Bohi and Zimmermann, 1984; Moroney, 1984; Bhatia, 1990).

- *Model 3.* Economy models viewing the energy sector as part of the overall economy. Various types of these models exist, such as general equilibrium models or neoclassical growth models (see Hudson and Jorgenson, 1974, 1979; Houthakker and Kennedy, 1979; Rogner, 1982; Kydes *et al.*, 1995). Their common feature is that the energy system is viewed from the outside, with the main focus on the interrelations with the rest of the economy. These models are mainly used to evaluate energy policies with respect to the consequences on the economy or vice versa.
- *Model 4.* Optimization models deriving optimal investment strategies or operation plans for specific utilities or municipalities. These models are used in national energy planning or international investigations for analyzing the future of an energy system. Such models derive optimal strategies assuming optimal behavior of all acting agents under given constraints.

Among the optimization models, the most commonly used approach is linear programming (LP, see e.g., Dantzig, 1963; Williams, 1990). LP models have found widespread application in refinery scheduling, national energy planning,[1] and technology related long-term energy research (see Nakićenović and Messner, 1983; Häfele *et al.*, 1986; Okken *et al.*, 1992; Hamilton, 1994; Ybema *et al.*, 1995). There is a growing tendency to base municipal energy plans on formalized modeling tools, mostly with LP models (examples are described in TEMAPLAN GmbH, 1982; Sundström *et al.*, 1983; Pfaff *et al.*, 1991; and Joseffson *et al.*, 1994).

Unit commitment problems, with decisions on the start-up or shut-down of thermal units, and expansion planning, deciding on which type and size of power plant to build, involve discrete decisions that can be modeled by yes or no (0 or 1) decision variables. This feature complicates linear models, but can be handled by mixed integer linear formulation or MIP (Mixed Integer Programming; refer to Chapter 7).

Alternative modeling approaches include dynamic programming, which gives fast solutions on 0–1 decision problems with low complexity. Smooth nonlinear optimization problems are often solved using Lagrange relaxation methods. Recently, some modern techniques such as genetic algorithms and simulated annealing (see Vilkko and Lautala, 1990), have penetrated the decision support market in the energy utility sector. For more complex problems, various combinations of modeling approaches are applied to solve the problem in a hierarchical or iterative

[1]The energy systems model MARKAL (Fishbone *et al.*, 1983) is applied by IEA member countries for concerted analyses in special topics. The current focus is on the global warming issue.

manner (Gollmer and Wutke, 1994; Scholten, 1994). All these approaches (partly described in Chapter 7) have been used in analyzing energy models.

The accuracy of depicting the system, the choice of modeling technique, and decision variables all depend on the type of system, the time horizon, and the questions asked.

It should be stressed that a fifth type of modeling approach – multi-objective optimization and modeling as presented in earlier chapters of this book, particularly the approach that uses multi-objective optimization as a tool of initial model analysis – is not yet widely used in energy modeling and planning. However, early attempts to use such an approach were made by Grauer *et al.* (1985a, 1985b), and some further applications of this type will be described in this chapter. With the growing computational power of the average computer and the broadening access to good optimization and decision support software, more applications of this type might be expected. However, a decisive factor here is also education: specialists often tend to use concepts that they learned at universities, where the classical paradigm of simulation and optimization is presented.

14.1.3 General requirements of decision support

Energy-related DSSs generally fall into two categories and the two types of users and decision makers:

- Systems applied by specialists (in industrial, institutional, or scientific environments), where the evaluated results are presented to the decision maker. This type of decision support is generally applied if high-ranking persons are involved or if the decisions have far-reaching consequences.
- Systems continuously used for short- to medium-term planning. Such systems are mostly applied by the decision makers themselves. As an example, scheduling and resource planning for the production of an electric utility is decided by the load dispatcher, who is also using the decision support software.

The most important requirement for DSSs involving high-ranking decision makers is flexibility and the ability to answer any question that might arise. Strategic investigations often call for changes in existing models. The requirements on the man-machine-interface (MMI) are moderate, because the system is applied by specialists with specific interest in the issue. The decision makers, who usually do not apply the DSSs themselves, require relevant strategic information of the what-is-when type. They need to understand the implications of certain decisions and the interrelations of the components in the system investigated.

Generally, decisions with far-reaching consequences are not taken according to the solutions proposed by a model. They are evaluated on the basis of model results, but incorporating the specific experience, skills, and judgment (see Chapter 2) of the persons involved. Because of the needed flexibility and the necessity to account for intuition, the methodology of multi-objective modeling and decision support as described in earlier chapters of this book might be particularly appropriate for this type of situation; we shall illustrate this later with examples.

DSSs of the second type, used for example for scheduling the operation of power plants and energy exchange contracts for a utility, which are applied by the load dispatcher with some special training, have quite different requirements. They must be linked to the data base of the utility in order to always reflect the current situation; they have to be of high quality (error-free) and give defined results under all circumstances; and the MMI must be easy to operate and understand and cover all possible manipulations including transfer of the results to any consecutive process. In operation planning, the decision maker (load dispatcher) analyzes the proposed solution in view of her/his experiences and objectives and either accepts it or disagrees with one or more of the decisions involved. As a consequence, the DSS has to have the flexibility to start from the last solution proposed and incorporate additional constraints or other settings given by the load dispatcher. Most DSSs of that sort also offer the option of simulating a solution defined completely by the operator of the power system.

A major problem faced by load dispatchers of energy utilities involves uncertainties in basic parameters influencing the decision process. Uncertainties in short-term decision problems usually concern the demand given by the load curve over the day or the week and the uncertainties in the supplies from some energy sources (e.g., inflow into a water reservoir, water freight of rivers). In countries like Austria this concerns hydropower generation, which is highly dependent on rainfall in the water catchment areas. Various types of forecasting systems can expand the range of fairly exact results to a period of up to eight hours. Forecasts exceeding this time frame tend to be of a more speculative nature. Consequently, another requirement of online DSSs is the ability to start a new model run at any point in time from the current situation, revising the plans in operation according to changes in forecast data. Failures of power plants can also lead to new emergency plans.

Maintaining the consistency of the data base and the model is of the utmost importance in continuous model application. Changes in exogenous conditions, internal structure, or objectives of the system should be continuously monitored.

In this second type of decision support situation, the methodology used for decision support is of secondary significance; of primary significance is reliability. However, flexibility is also an important feature – as exemplified by the requirement to start any model run and any decision process anew from the current situation.

14.2 Examples of Decision Support in Energy Planning

The following examples describe several applications that have been developed to support decisions in the area of energy planning. All of them are based on LP or MIP optimization models. They cover various applications, different decision makers, and diverse time frames. Because of limited space, their description will be necessarily short, with only general methodological comments.

14.2.1 Short- and medium-term planning for utilities

Planning the operation of a utility power system (e.g., fuel storage, hydroreservoirs, contracts) involves a hierarchical set of time frames. Fuel stockage and use, annual hydrostorage utilization and contractual arrangements are managed in annual resource allocation plans. Actual scheduling is done on a weekly or daily basis, while sudden changes in the system's parameters have to be covered by emergency plans established within a few minutes.

Optimization tools to provide support for these decisions and plans have been based on various types of methods, such as dynamic programming, nonlinear Lagrangian optimization, and MIP (see Carpentier and Merlin, 1982; Vilkko and Lautala, 1990; Hanselmann *et al.*, 1992; SVOR/ASRO, 1993; Verstege and Maubach, 1994). Mixed integer models have proven to be the most reliable tools to represent complex systems and to provide easy parametrization for various kinds of applications (as they can be applied with a standardized software tool).

Systems that integrate the planning horizons from short-term plan revisions for the next few hours up to planning for the next fiscal year begin to penetrate the market. These time horizons are linked conceptually by using results from the longer time horizon to generate constraints for the shorter one or by integrating models with different time horizons. Model characteristics for annual resource allocation problems and the daily/weekly operation model differ in terms of technical detail, but the basic underlying model is the same.

A model for annual resource allocation problems is applied in the energy planning division of utilities and used to determine how contracts should be negotiated, what quantities of fuel to order, and how seasonal hydrostorage reservoirs should be used. It can also be used for strategic planning problems and what-if questions, e.g., to investigate the impacts of building a new power plant or some assumed failures in system components. The model is used by one or more utility-based specialists that operate the model when analyses are required.

In its applications for daily and weekly operation planning the model is used by the load dispatcher. It runs at least once a day to generate the plan for the next day, or more often in cases of major changes in the parameters of the power system. The main requirements of the DSS for such applications are:

- On-line data exchange with other programs and use of process information from the power plants.
- Computational speed.
- Easy management.
- The possibility to override the results and incorporate the load scheduler's decisions into the actual plans.

An example of such a system is described in Harhammer (1986) and the IBM report (1989) on the system JARO. This system has a very high degree of flexibility in its mathematical kernel. The current version, XOPT, is used by four utilities for short-term scheduling or annual resource allocation (see Kümmerling, 1990; or Steinbauer, 1992). It has a modular structure with the following major elements: thermal power plant; hydropower plant; contract; and balance of supply and customer demand.

Additional modules allow various complex formulations such as logical relations between units, reserve conditions, nonconvex efficiency curves, and various types of constraints based on legal, contractual, or technical conditions. The model makes it possible to set up a new application by parametrizing the energy system in terms of input data. This modeling effort requires an in-depth knowledge of the utility energy system and experience in model setup. The objective of this process is to generate a model that is accurate enough to satisfy all operational requirements of the utility and produce results that meet the required standards in terms of quality and accuracy. However, adequate model formulation has to enable fast computations (Nemhauser and Wolsey, 1988). Presently, model sizes range from 5,000 to 20,000 variables, with CPU times between 3 and 30 minutes for scheduling and resource allocation models, respectively.

Figure 14.1 provides an overview of the interfaces of XOPT, as documented in Landis & Gyr (1995). The optimization package has six external interfaces: to the user (MMI); to the power plant processes (to get information about the operational state of the power plants and give information on planned changes); to an additional optimization package (e.g., for a cascaded hydropower system with a highly nonlinear model, see Landis & Gyr, 1993) as a subsystem with iterative linkage via quantities and shadow prices; the link to the load and hydraulic forecast modules; to external computational equipment such as a printer or plotter; and to the central archive of the utility via databank routines.

During the initial learning phase of model configuration and application, the planner and the load dispatcher sometimes encounter situations where model

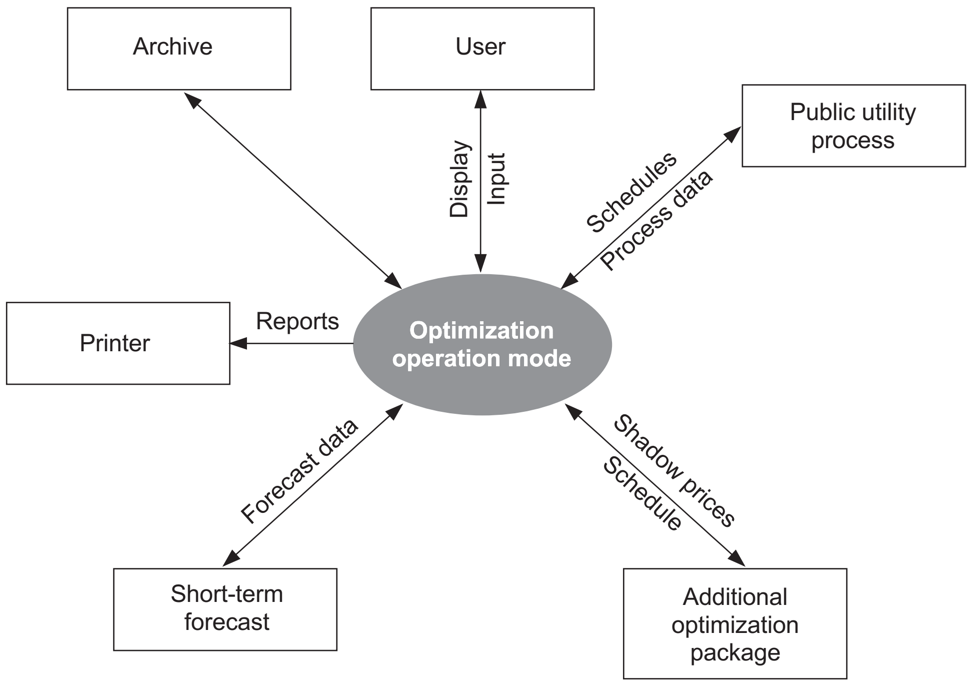


Figure 14.1. Environment of annual energy planning package, XOPT.

outcomes suggest schedules that are not commonly applied. These schedules, which are innovative in the sense of applying existing machines in a different way to reduce expenditures, deepen the understanding of the power generation system. In the wider sense, this innovation process could be grouped into the new innovation procedures as suggested by Cowan and Foray (1995), which are triggered by good simulation or optimization models.

This example of application shows that even single-criteria optimization might be useful for decision support in well-defined situations.

14.2.2 Planning a metropolitan energy system

The public utility of Vienna used the dynamic linear programming model MESSAGE (Messner and Strubegger, 1995) for its most recent energy plan (Wiener Stadtwerke, 1991) to evaluate the future development of the municipal energy system, especially with respect to the coordinated expansion of the gas and district heat grids. The organizational structure is such that electricity, district heat, and natural gas are handled by three players in the energy market of Vienna. In order to help establish the 1991 energy plan, they delivered the input data for formulating an energy model for Vienna and monitored the development of this model. The resulting model subdivides Vienna into five regions with clear priorities for the future expansion of the gas and district heat grids. Pfaff *et al.* (1991) give an overview of this effort.

As mentioned before, a multi-objective approach is natural in more long-term planning problems of energy systems. The model for Vienna has been set up as a multi-objective optimization model ensuring that three objectives are minimized: system costs; energy imports; and pollutant emissions (using an aggregated index for various pollutants, which is generally applied in Vienna). A simple aggregation of these objectives into one is not possible; for example, although energy imports can be counted as costs, the dependence on imported energy is a separate issue and it is difficult to convert this objective into monetary units; the same concerns pollutant emissions.

The approach used to formalize the minimization of multiple objectives is based on the reference point optimization method as described in earlier chapters (see also Grauer *et al.*, 1985a, 1985b). This method allows the user to formalize each objective in its natural units. In an application, each objective is formulated as a target trajectory over time, hence it is actually an example of multi-objective trajectory optimization (see Wierzbicki, 1980a).

The modeling and optimization system tries to find decisions that result in a uniform achievement of the target trajectories. If they are not attainable, the target trajectories are approached uniformly close, and if they are attainable, they are uniformly improved. Internally, these target trajectories are evaluated in relation to so-called utopia and nadir trajectories. The single points for the utopia trajectories are derived by optimizing each single time-step for every objective. The utopia trajectories are established by putting all these single optimal points together; they are utopic in the sense that, although each single point can be reached, there is no feasible solution that achieves all of them. The nadir points are computed by putting together the worst solution for each single time-step and objective and thus, as explained in Chapter 4, these points are only an approximation for an actual nadir point.

Figure 14.2 shows the nadir and utopia points for an example with two objectives (in which case, exceptionally, the nadir can be estimated exactly). It also shows how an optimal solution is reached from two given examples of reference points (denoted by small circles) by driving the solution toward the Pareto-optimal frontier in the objective outcome space. If the reference point lies inside the feasible region, then an improvement with respect to both objectives is reached; however, in the case of too high expectations, when the reference point is outside the feasible region, the optimum is worse than the reference point for both objectives.

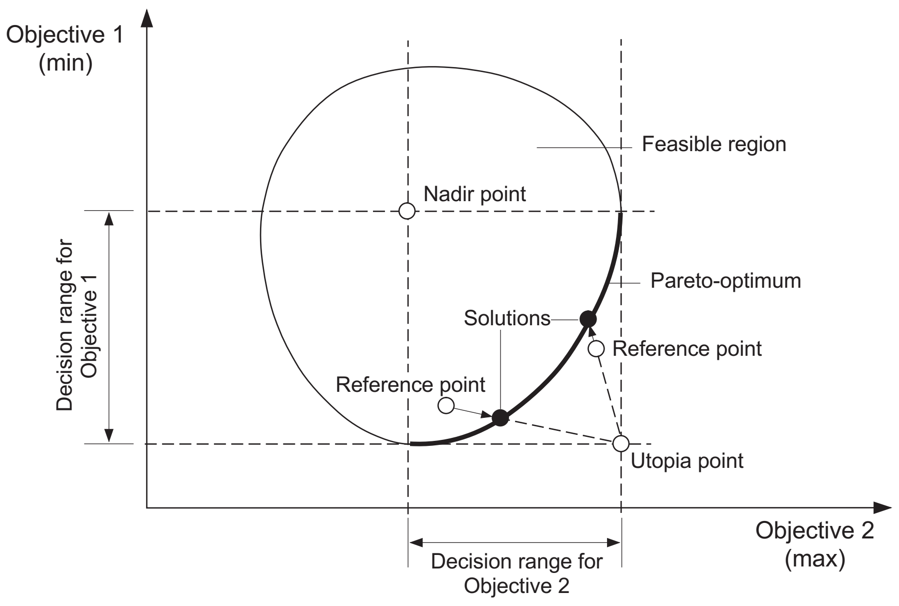


Figure 14.2. Two-dimensional solution space for reference point method.

In addition to supporting a multi-objective solution of complex systems, the approach helps to investigate the shape of the Pareto-optimal frontier in the objective space (the thick line in *Figure 14.2*). Clearly, in the case of multi-objective optimization of three objective trajectories over many time periods, the actual number of objectives is rather high and the Pareto frontier is much more complicated than shown in *Figure 14.2*. However, this indicates precisely the strength of a reference point or reference trajectory approach: even for the most complicated Pareto frontiers, we can find points on them and see how they change if the reference point is altered.

Figure 14.3 shows the optimal energy mix for Vienna in the year 2015 for each of the single objectives, as well as the compromise solution reached with the reference point method. Cost minimization yields a considerable share of oil, while energy minimization (due to the higher efficiency) and emission minimization (due to the lower emissions) both result in higher shares of natural gas. In the case of energy minimization, investments in insulation reduce heating demands. The compromise solution also gives considerable demand reductions, and a fuel mix similar to energy minimization, but with a slightly higher share of natural gas.

As an example, the NO_x emissions related to the four results are shown in *Figure 14.4*. These emissions are highest for the cost minimization scenario with 85% of the 1990 level, while energy and emission minimization both result in a reduction

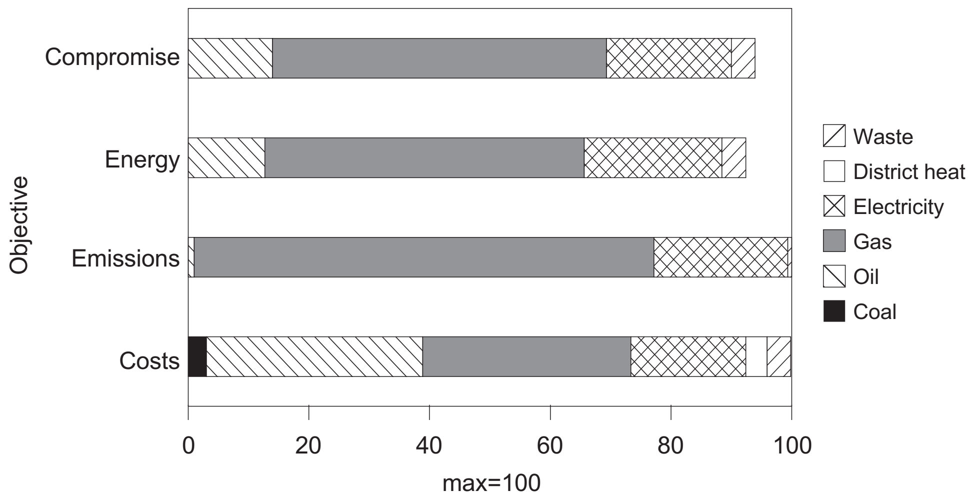


Figure 14.3. Vienna: Patterns of final energy consumption per energy carrier in 2015, depending on the optimized objective (costs, emissions, energy), and a compromise solution.

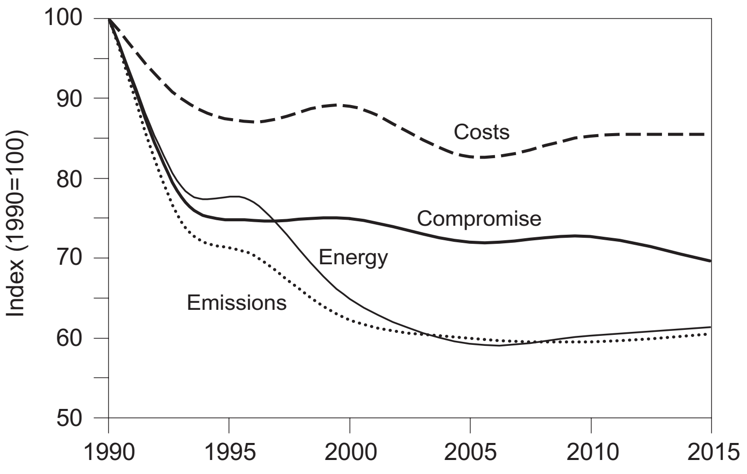


Figure 14.4. Vienna: Trajectories of NO_x emissions in three separate cases of objective minimizations (costs, energy, emissions) and a compromise case, 1990–2015.

to 60% already by 2005. The compromise solution reduces NO_x emissions to 70% and lies between the two outcomes.

The model was optimized on the IBM mainframe of the Vienna municipal utility, with a MINOS solver (see Murtagh and Sanders, 1983); CPU times were in the range of two to three hours.

This example shows that a multi-objective optimization approach based on reference trajectories for a dynamic, multiperiod model can produce realistic scenarios. Limits on attaining various goals are given when optimizing separate objectives; the energy planner can thus learn what is attainable and what is not. A compromise scenario represents both intertemporal and intercriteria compromises. The multi-objective reference trajectory approach is thus a flexible tool of inverse scenario analysis (see Chapter 5).

14.2.3 Expansion planning for district heat supply

Historically, the Austrian federal state of Lower Austria did not have its own capital. All relevant infrastructure and legislative and executive bodies were located in Vienna, which is situated within Lower Austria, but is a federal state in itself. In a referendum in 1992 the population of Lower Austria decided to select St. Pölten, a city with 40,000 inhabitants, as capital of the state. St. Pölten had to provide a new government district. Clearly, the utility of St. Pölten had the task to supply this new district with energy in a cost-effective and environmentally benign manner. As St. Pölten already has a district heat system with two production sites, the prime choice is to expand this system to also supply the new government buildings. Several alternative options are available for the future supply of district heat, and construction can take place at each of the two existing sites (commonly labeled North and South) and at a potential new site called East, which would be in the vicinity of the new buildings.

In order to depict all these investment options, a MIP model was built (71 technological options were represented with integer variables for the investment decisions for 11 time-steps).[2] Additionally, district heat trunk lines connecting the subareas and the distribution system were also modeled. Reserve in the supply system was guaranteed by a constraint requiring an adequate generation capacity for each period. Electricity from co-generation can be delivered to the utility of Lower Austria at a contractually fixed price, while district heat is delivered to the customers at a set price.

The mixed integer model generated for this problem consists of 6,500 constraints, and 6,300 variables with 300 integer variables. The solution of the full MIP problem requires several CPU hours on an IBM workstation (RS/6000 model 375) using the IBM Optimization Subroutine Library (IBM, 1992).

The technical director of the utility had planned to use the integrated model of all supply options in the system to check the various alternatives of investment measures that had been evaluated so far. The model was used to simulate these

[2]For this analysis, an extension of the MESSAGE model (see Messner and Strubegger, 1995) was used while expanding its capability by modeling investment variables as integer decision variables.

Table 14.1. St. Pölten: Expenditure and income as a percentage of the total.

	Discounted expenditures (income) as % of total						
	Invest-ments	Fuel cost	O&M costs	Personnel costs	Elec-tricity	District heat sales	Income
A	21.74	37.71	25.79	14.76	–22.47	–80.50	(2.98)
B	24.68	37.43	24.03	13.86	–26.88	–75.58	(2.46)
C	25.47	37.27	23.75	13.52	–26.76	–73.74	(0.49)
D	20.18	37.49	26.93	15.40	–18.35	–83.96	(2.32)

Abbreviations: O&M, operating and maintenance costs. For abbreviations A–D refer to the text.

decisions, that is, with fixed 0–1 variables for the major decisions. Using the model in a free mode, leaving all decision variables open, showed that the optimization could give a better solution than that derived without a formalized model. Additional sensitivity analyses were then performed on four technical issues, listed below.

- A: A free optimal solution.
- B: Simple hot water boilers are not allowed to enter the solution.
- C: Existing steam turbines fall out of operation earlier.
- D: Gas turbines are not permitted in the southern site.

Table 14.1 shows the expenditures and incomes related to these variants. All figures are based on the total discounted costs (scrap values were not subtracted) and expressed as a percentage of the overall expenditure (investments; fuel cost; operating and maintenance costs; personnel costs, excluding the income from electricity; electricity and district heat sales). The last column (income) shows the discounted income generated over the optimization horizon of 17 years as a percentage of overall cost.

In *Figure 14.5* the discounted net income is compared to the overall discounted system cost, and both are in relation to the highest value. The highest discounted income can clearly be generated with variant A, which has the least constraints. Variant D, with no gas turbines allowed in the south, has the lowest overall expenditures of these variants. If the existing turbines go out of operation earlier than planned (variant C), expenditures are highest and the lowest income is generated (less than 20%, as opposed to 100% in variant A). In this case some investments would have to be made earlier, with a considerable negative effect on profitability.

In this example, the single-objective optimization methodology was employed; however, the essence of this example lies in the use of the optimization approach not as a goal, but as a tool of analyzing and comparing various variants of possible strategic decisions.

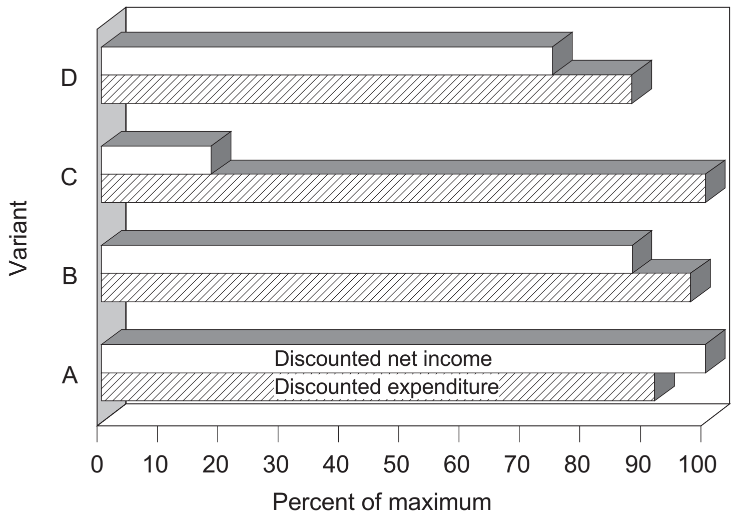


Figure 14.5. St. Pölten: Expenditure and income as a percentage of the total.

14.2.4 Long-term policy issues

The Environmentally Compatible Energy Strategies Project (ECS) at IIASA has, together with the World Energy Council (WEC), developed a set of long-term scenarios to investigate the energy implications of global economic development and its environmental impacts, including the issue of global warming (WEC, 1992). In comparison with the previous examples, this effort is a more scientific one, and has no direct interaction with potential decision makers in the modeling process. Rather, the results of the study and the model outcomes are targeted to give decision makers from various fields, such as industry, policy, and R&D, background information on long-term energy and environmental issues.

The modeling approach used is similar to long-term planning of utilities but, in fact, it is an example of multi-objective modeling used for clarifying complex issues. The dynamic optimization model MESSAGE III (see Messner and Strubegger, 1995), is applied at a global level, with the world divided into 11 regions. The regional models are interlinked by international energy trade of all major energy commodities. The overall model size is in the order of 35,000 variables and 50,000 constraints. Optimization was performed with a special solver HOPDM (see Gondzio and Makowski, 1995a) prepared by another IIASA project – the Methodology of Decision Analysis (MDA, see the description of this solver in Chapter 7). This solver used an interior point algorithm and thus reduced the time needed to solve very large LP problems; the time actually needed was approximately one hour CPU time on a SUN Sparcstation 1000 with two CPUs.

Background research required for the model analyses included knowing the energy resource base available, the future technological options, and the dynamics of technological parameters over time. Three conceptually different types of scenarios were developed, covering: A, high-growth; B, middle-course; and C, ecologically driven types of expectations concerning the future economy and energy system.

For this comprehensive effort, modeling tools were required in addition to MESSAGE III. The two major additional components of the model set applied are the scenario generator (SG), a framework to evaluate the future development of the economy and energy requirements on the basis of historical time series. The SG applies an approach of path dependence, where economic development can be achieved on different development paths (e.g., an American versus a Japanese type of development model). The third model applied is the macroeconomic model 11R (see Schrattenholzer and Schäfer, 1996), which is based on the well-known Global 2100 model (see Manne and Richels, 1992). It is used to investigate the economy-wide consequences of the energy development scenarios.

For the longer-term focus of the study, which is up to 2100, the issue of global warming is of importance. Here, the model provides information on annual and cumulative CO_2 emissions for all scenarios. Using the carbon cycle and climate model developed by Wigley *et al.* (1994), these carbon emissions are translated into the corresponding atmospheric concentrations of CO_2 and potential temperature changes.

The information provided by the IIASA–WEC analysis focuses on the needs of decision makers in three ways. First, it gives an overview of features of the future energy system common to all scenarios and demonstrates how differences in short-term economic and technology policies translate into long-term divergences of the structure of energy systems. Second, it investigates the investment requirements over the coming 25 years, especially in the developing world. Third, it analyses the resource requirements in terms of fossil and nonfossil sources of energy. Specific industry issues are addressed in special sections of the full report.

Figure 14.6 gives a sample study result: an overview of the oil export quantities and related income for Middle East and North African countries in the scenarios, including historical data starting in 1963. The figure clearly shows that per annum export quantities of US$240 billion (in constant 1990 prices) that were reached in the early seventies were not achieved easily again. As present-day prices are also not considerably higher than the US$44 per barrel achieved in 1980, annual revenues through 2050 will remain below the 1980 level of US$320 billion. However, the 1990 income level of US$140 billion seems to be a lower bound for potential future revenues from oil exports of the region.[3]

[3]All prices referred to are in constant 1990 US$.

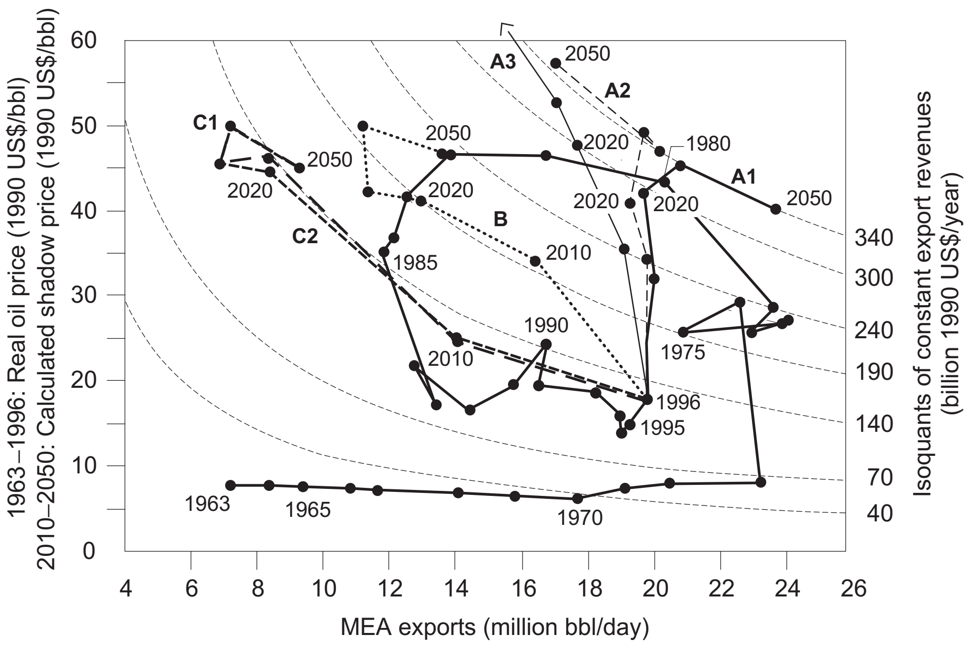


Figure 14.6. Oil export quantities (in barrels, bbl) and revenues for the Middle East and North Africa (MEA), historical development from 1963–1996 and in scenarios to 2050.

The study outcomes were presented at the 17th Congress of the World Energy Council in Houston, Texas, in September 1998. Results have also been published in Nakićenović *et al*. (1998).

From a methodological point of view, the results of the study show the usefulness of a multi-objective modeling approach. Such an approach treats multi-objective optimization (including dynamic multi-objective optimization) as a tool of flexible model analysis, necessary for addressing such complex issues.

Part VI

Conclusions

Epilogue

Jaap Wessels and Andrzej P. Wierzbicki

This book is a compilation of several years of experience in developing and using model-based decision support systems (DSSs). As discussed in the introduction, there are many settings for decision making. The abundance of possible settings often confuses the discussion on decision support. Therefore, we have restricted our attention in this book to a particular class of decision situations or decision-making settings, corresponding mostly to diverse environmental applications. In this way we hope to increase the verifiability of our assumptions and the consistency of our approach.

This restriction to a particular class of decision situations or settings does not mean that the approach or its parts are not relevant for other situations or settings. In fact, several of the concepts and tools presented in this book have been inspired by completely different examples and decision settings. In addition, most of the authors have also been active in other areas of applications and types of decision settings. Nevertheless, the approach advocated in this book is primarily recommended for environmental problems of a strategic nature.

One of the things that we have learned is that it is indeed possible to develop an approach to decision support which is suitable for a particular class of decision settings or situations. This is promising, coming after much debate concerning the possibility of developing a generic approach. The choice was often between a general approach for all decision-making problems and a specific approach for every special case. A general approach might be appealing for its elegance, but it is always rejected by practitioners, either because of its vagueness or for not fitting to particular cases in reality. A specific approach for every special case, however, may be effective, but is not a very efficient way of developing decision support systems. For several classes of operational decision-making settings, it has been shown that common approaches are possible, even if these classes might be relatively narrow in scope. In this book we indicate that another common approach may be also possible for some classes of strategic decision-making problems.

In this Epilogue, we review the main issues of this book and draw some conclusions from our experiences.

Strategic Environmental Decision Making

Environmental decision-making situations are typically characterized by some natural processes that can be considered as the kernel of the decision setting. Such natural processes might be physical, chemical, or biological; often they also have economic dimensions and consequences. These processes are generated by natural phenomena as well as by human activities. If we speak about environmental decision making, then we mean primarily influencing human activities, in some cases also influencing natural processes. The aim of environmental decision making is always to decrease the harmful consequences of these activities and processes.

Because of these features of environmental decision-making situations, the core of the decision support approach becomes a model that classifies the consequences of proposed ways of influencing these processes and activities. For this type of problem there is no alternative to model building, except, perhaps, experimenting. Indeed, in some cases, small-scale experiments might provide an alternative to mathematical modeling; however, for more costly cases of strategic environmental decision making, large-scale experiments become too expensive and risky, thus modeling becomes necessary. Typically, the forms of models used in environmental decision making depend strongly on the natural processes involved and on the required level of detail. Therefore, one might exploit the same model types for subclasses only: air transportation models may be used for various types of air pollution problems, and river stream models may be used for various problems concerning river basin quality. Nonetheless, models related to environmental decision making are typically complex, and an essential problem of decision support in such cases is a multi-objective analysis of decision outcomes predicted by complex models.

Strategic environmental problems also have other features that are common to all strategic decision problems: diverse groups of decision makers and people concerned in decision outcomes, multiple goals and objectives, and many constraints of a soft rather than a hard type. Because of the diversity of the decision makers, the approach taken in this book does not focus on direct support to high-level political decision makers, but rather on supporting the needs of decision analysts who are responsible for a careful examination of the consequences of various possible decisions. Specific methods of dealing with multiple objectives and soft constraints, based mostly on the reference point approach in multi-objective optimization and model analysis, have shown their usefulness for different classes of problems, examples of which are analyzed in this book.

Tools

Specific model types might be used for particular subclasses of decision problems only. Nevertheless, model building is time consuming and expensive; thus, it is important to build reusable models. This point is even more valid when it comes to tools for handling and investigating the models; such tools may be developed in a way that they are applicable to a large variety of cases, as discussed in the chapters on applications. The use of such tools is not restricted to strategic environmental decision making; they might be applied to any decision setting involving complicated mathematical models.

Various types of tools have been described in this book, predominantly in the second part. Modeling and simulation tools, optimization tools, and tools supporting choice have been described in detail. However, one lesson from our experiences is particularly important: the effectiveness of computerized decision support depends strongly on a careful and innovative preparation of tools for building user interfaces.

The Use of Decision Support Systems

In strategic environmental decision making it is unlikely that the final decision makers will use the DSSs themselves. Therefore, DSSs are typically used by other persons, e.g., advisors, experts, analysts, who provide expertise and advice to actual decision makers. Moreover, the decision-making process usually consists of several rounds, quite often involving negotiations. These can be formally recognized and organized, but they can also be based on an informal balance of power. In such decision processes, model-based decision support approaches and tools may play various roles outlined below.

Participant's back office tool. Each participant of the decision process – in particular, of a negotiation process – may strengthen her/his position by a careful preparation, using decision support, or asking staff members for a careful analysis.

Insight improvement tool. The participants in a negotiation process, or groups of them, might gain insights into the possibilities, difficulties, and sensitivities by attending training sessions with a DSS, or by reading reports from such sessions.

Common knowledge tool. Negotiations are brought to a higher level if the participants accept model studies as a proxy reality, summarizing common knowledge about the problem. Real consequences for various proposals might be predicted and studied by using the accepted model, e.g., if a participant of a negotiations process comes up with a proposal advertised as a great sacrifice for the proposer and a large gain for other parties, this claim can be verified by using the accepted model.

General negotiation tool. A DSS can be used as a flexible tool in negotiations, e.g., for preparing negotiation positions and rationalizing them; and for studying sequences of negotiation proposals and anticipating the next ones.

The Development of Decision Support Systems

The experiences reported in this book indicate that the development of DSSs for strategic environmental decision making should be a joint effort involving experts in the subject area, modelers, and decision support experts. Some of these roles might be combined, for example, if modeling is well developed in a specific subject field, then the roles of a subject expert and a modeler might coincide. However, it is important that the developers of a DSS remain in close contact with the real decision makers in order to ensure that their concerns are expressed in the system under development and that the final system will be accepted by them.

For the other experiences discussed in this book, we stress the importance of good data bases, and good libraries of tools. One of the most important requirements is a modular structure of a DSS that enhances the reusability of system modules. In such modular structures, user interfaces play an important role.

Final Remarks

As indicated in the introduction, this book is neither a textbook nor a handbook of tested recipes. It is more a monograph reporting many years of experience. The experience presented here will be of value to researchers in various fields: decision analysis and decision support, but also environmental analysis, mathematical model building and analysis, etc. We hope that the issues discussed in this book will gain in importance with the development of the new era of the information society, where information, knowledge, and ways of processing them become a decisive part of human activities.

The editors welcome comments and/or corrections, particularly in the formulae (e-mail: marek@iiasa.ac.at).

Appendix: Software Description

Marek Makowski

This Appendix provides information about the availability of software developed as a result of scientific cooperation between the Methodology of Decision Analysis Project at the International Institute for Applied Systems Analysis (IIASA)[1] and various research institutions in Poland.

Conditions of Use

The software described in this Appendix is available, free of charge, from http://www.iiasa.ac.at/∼marek for users who accept the license agreement. At the time of writing, the following conditions applied to the license agreement:

- The software is only used for research or educational purposes provided that the use is noncommercial. For commercial use a separate written agreement with the author(s) of the software is necessary.
- Redistribution of the software requires written permission from IIASA or from the authors. No part of the software can be modified or incorporated into other software without written permission from the authors.
- The user agrees to cite the software, if used, by referencing this book and by mentioning that the software has been developed in cooperation with IIASA. A copy of an article or a report with such a citation should be sent to the address specified on the software distribution page.

[1]Detailed information about IIASA can be found on the Web at http://www.iiasa.ac.at.

- In no event shall IIASA or the authors of the software or the institutions that employee the authors be liable for any damages whatsoever (including, without limitation, loss of profits, business interruption, loss of information, or other pecuniary loss) arising out of the use of or inability to use the distributed software.
- The software is provided on the "as it is basis". The user may contact the authors directly in case of any problems with the software or if the software requires modification for her/his application. However, the authors of the software are not obliged to provide any assistance in the use of the software.

The users, who accept the license agreement posted on the Web, will be asked to complete a short registration form, and will then be granted access. The data provided in the registration form are available only to the authors of the software. The registered users will receive e-mail information whenever a new version of the software becomes available.

Short Software Description

This section provides a brief description of the software items that are available at the time of writing. More detailed and periodically updated information is available on the Web. This information contains links to documentation (most of which is available on-line) and provides contacts with most of the authors of the software.

The following software items are available for two operating systems, namely SunSolaris 2.6 (or higher) and MS-Windows 95/NT.

- *MCMA*: This is a tool for multi-criteria model analysis as described in Chapter 10. The MCMA, which contains ISAAP (modular tool for interactive specification and analysis of aspiration-based preferences), is documented in Granat and Makowski (1998). Full documentation of MCMA (which also contains a tutorial guide to MCMA) is available on-line from the same URL. The users of MCMA also have access to the documentation while running this package. MCMA can be used for multi-criteria model analysis of any LP or MIP core model. MCMA's version for nonlinear models will be available in autumn 1999.
- *HOPDM (Higher Order Primal–Dual Method)*: This is a robust and efficient LP code that compares favorably with the up-to-date commercial solvers, especially for large-scale LP problems. It has all the usual features of an LP solver, such as sparse basis factorization and updates, problem scaling, limited presolving, and a heuristic search for an advanced initial solution. The algorithm

implemented in HOPDM is a new variant (Gondzio, 1997) of a primal–dual logarithmic barrier method that uses multiple correctors of centrality. The version of HOPDM available from IIASA is composed of a C++ part that handles data and memory management and the Fortran part of the original solver, and is documented in Gondzio (1995). HOPDM is used as a default solver for LP type core models that are analyzed with the help of MCMA.

- *HYBRID*: This is a mathematical programming package that includes all the functions necessary for the solution of LP problems and includes the standard features of an LP solver listed in the description of HOPDM. HYBRID uses a nonsimplex method that combines the proximal multiplier method and an active set algorithm. This method differs from the ordinary multiplier method by adding a quadratic form to the augmented Lagrangian function and making the function strongly convex with respect to primal variables. This regularizing term also improves the numerical stability of problems that have an almost nonunique solution (which is typical for most large problems) and, from many optimal solutions, enables the one that is closest to a given reference point to be selected. In the proximal multiplier method, the LP problem is solved by minimizing a sequence of piecewise quadratic, strongly convex functions subject to simple constraints (lower and upper bounds). For the sparse Cholesky factorization a part of the code of the HOPDM solver is used. The simple constraints (lower and upper bounds) for variables are not violated during optimization and the resulting sequence of multipliers is feasible for the dual problem. Constraints other then those defined as simple constraints may be violated, however, and therefore the algorithm can be started from any point that satisfies the simple constraints.
- *SALP*: This is a modern implementation of the modified primal simplex algorithm (Dantzig, 1963) for large-scale linear programming and includes the standard features of an LP solver in the description of HOPDM. Moreover, it employs both the standard minimum reduced cost pricing algorithm and the exact steepest edge procedure. In addition to these features, SALP employs a number of nonstandard implementation tricks, e.g., an efficient approximation of the steepest edge procedure, the exact linear penalty method with a bounded penalty coefficient and guaranteed interpretation of results, an efficient restart procedure after arbitrary simultaneous changes to the right hand side and objective vectors, duplicate representation of the constraint matrix (both row-wise and column-wise) speeds up the pricing algorithms. SALP is documented in Świętanowski (1995). It has been successfully used as a front end to a mixed integer branch-and-bound program MOMIP. The numerical procedures of SALP have also been incorporated in a decomposition-based optimizer for two-stage stochastic programming.

- *MOMIP (Modular Optimizer for Mixed Integer Programming)*: This is a branch-and-bound solver for middle-sized mixed integer linear programming problems. MOMIP is essentially an experimental solver. It proved to be very reliable and efficient on the medium-scale models we analyzed (up to 2,000 continuous variables/constraints and a few hundred integer variables). Therefore, we have made it available for wider use. However, MOMIP does not aspire to compete with commercial solvers in its capability to solve large-scale or complex integer problems. MOMIP was originally designed as part of a wider linear programming modular library but only the customized solver is available from IIASA. MOMIP is used as a default solver for MIP-type core models that are analyzed with the help of MCMA.
- *LP_DIT (Linear Programming Data Interchange Tool)*: This is a C++ template class library for handling LP/MIP models documented by Makowski (1999). The library can be used for problem specific generators of core models. LP_DIT is also available in executable forms that can be used as a utility for conversion of LP/MIP models available in the MPS format to the LP_DIT format and for handling solutions and model definitions in the LP_DIT format. The LP_DIT format is used in all the software listed above. Moreover, it can be used for a formal diagnostics of a model specification.
- *DIDAS-N*: This is a system that supports interactive multicriteria analysis of nonlinear substantive (core) models. Models of this type include two classes of variables: input variables and outcome variables. Input variables can be subdivided into decision variables and parametric variables (model parameters that are kept constant during multiobjective analysis but might be changed during parametric or sensitivity analysis). An essential part of a nonlinear model definition are model equations, that is, nonlinear functions that define the dependence of all outcome variables on input variables. The DIDAS-N system helps in the definition, edition, initial analysis, and verification of multiobjective decision analysis of nonlinear models. An important feature of the system is that it supports also automatic calculations of all derivatives of nonlinear model functions.

Additionally, two core models (described in Chapters 11 and 12, respectively) are available as tutorial examples for MCMA.

The list of available software items and operating systems may change in the future; therefore, for current information, the reader is advised to check the Web at http://www.iiasa.ac.at/∼marek. Registered users of the software described in this Appendix will receive an automatically generated e-mail whenever an updated version of a particular item becomes available.

References

Alcamo, J., Shaw, R., and Hordijk, L. (eds), 1990, *The RAINS Model of Acidification*, Kluwer Academic Publishers, Dordrecht, Netherlands.

Allais, M., 1953, Le comportement de l'homme rationel devant le risque: Critique des postulates et axiomes de l'ecole Americaine, *Econometrica*, **21**:503–546.

Amann, M., 1990, Energy use, emissions and abatement costs, in J. Alcamo, R. Shaw, and L. Hordijk (eds), *The RAINS Model of Acidification*, Kluwer Academic Publishers, Dordrecht, Netherlands.

Amann, M., Bertok, I., Cofała, J., Gyarfas, F., Heyes, C., Klimont, Z., Makowski, M., Schöpp, W., and Syri, S., 1998, Cost-effective Control of Acidification and Ground-level Ozone, Fourth Interim Report, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Andriole, S., 1989, *Handbook of Decision Support Systems*, TAB Professional and Reference Books, Blue Ridge Summit, PA, USA.

Angehrn, A., and Lüthi, H.-J., 1990, Intelligent decision support systems: A visual interactive approach, *Interfaces*, **20**(6):17–28.

Antoine, J., Fischer, G., and Makowski, M., 1997, Multiple criteria land use analysis, *Applied Mathematics and Computation*, **83**(2–3):195–215.

Arrow, K., 1951, *Social Choice and Individual Values*, Yale University Press, New Haven, CT, USA.

Averick, B., Carter, R., Moré, J., and Xue, G.-L., 1992, The MINPACK-2 test problem collection, *Technical Memorandum ANL/MCS-TM-153*, Argonne National Laboratory, Argonne, WI, USA.

Axelrod, R., 1984, *The Evolution of Cooperation*, Basic Books, New York, NY, USA.

Balas, E., Ceria, S., and Cornuejols, G., 1993, A lift-and-project cutting plane algorithm for mixed $0-1$ programs, *Mathematical Programming*, **58**:295–324.

Bangemann, M. (ed.), 1994, *Europe and the Global Information Society*, Report for the Council of Europe, Brussels, Belgium.

Barret, K., and Sandnes, H., 1996, Transboundary acidifying air pollution calculated transport and exchange across Europe, 1985–1995, in K. Barret and E. Berge (eds), *Transboundary Air Pollution in Europe*, MSC-W Status Report, Meteorological Sythesizing Centre - West, Norwegian Meteorological Institute, Oslo, Norway.

Beale, E.M.L., 1979, Branch and bound methods for mathematical programming systems, in P.L. Hammer, E.L. Johnson, and B.H. Korte (eds), *Annals of Discrete Mathematics*, **5**:201–219, *Discrete Optimization*, North-Holland, Amsterdam, Netherlands.

Beale, E.M.L., and Tomlin, J.A., 1970, Special facilities in a general mathematical programming system for nonconvex problems using ordered sets of variables, in J. Lawrence (ed.), *Proceedings of the 5th IFORS Conference*, Tavistock, London, UK, pp. 447–454.

Belton, V., and Vickers, S., 1988, VISA–VIM for MCDA, *Proceedings of the VIIIth International Conference on MCDM*, Manchester, UK.

Belton, V., and Vickers, S., 1992, VIDEA integrated DEA and MCDA: A visual interactive approach and modification of Karmarkar's algorithm to multiobjective linear programming problems, *Proceedings of the Tenth International Conference on Multiple Criteria Decision Making*, National Chiao Tung University, Taipei, Taiwan, July.

Bentley, J., 1988, *More Programming Pearls: Confessions of a Coder*, Addison-Wesley, New York, NY, USA.

Bergson, H., 1903, *An Introduction to Metaphysics* (English translation, T. Hulme, 1912, G.P. Putnam & Sons, New York, London).

Bertalanffy, L., 1968, *General Systems Theory: Foundations, Development, Applications*, Braziller, New York, NY, USA.

Bertsekas, D.P., 1982, *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, New York, NY, USA.

Bertz, M., Bischof, C., Corliss, G., and Griewank, A. (eds), 1996, *Computational Differentiation: Techniques, Applications, and Tools*, SIAM, Philadelphia, PA, USA.

Bhargava, H., Krishnan, R., and Mueller, R., 1996, Decision Support on Demand: Emerging Electronic Markets for Decision Technologies, Working Paper, Naval Postgraduate School, Systems Management Department, Monterey, CA, USA.

Bhargava, H., Krishnan, R., and Mueller, R., 1997, Decision support on demand: On emerging electronic markets for decision technologies, *Decision Support Systems*, **19**(3):193–214.

Bhatia, R., 1990, Econometric methods for energy planning and policy, in H.-H. Rogner, A.M. Khan, and G. Furlan (eds), *Economics, Modeling, Planning and Management of Energy*, World Scientific Publishing Co., Singapore.

Białoń, P., 1996, Optimization-based Analysis of a Simplified Ozone Model, WP-96-134, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Bischof, C., and Griewank, A., 1996, Tools for the automatic differentiation of computer programs, in G. Alefeld, O. Mahrenholtz, and R. Mennicken (eds), *ICIAM/GAMM 95: Numerical Analysis, Scientific Computing, Computer Science*, Special Issue of Zeitschrift für Angewandte Mathematik und Mechanik (ZAMM), pp. 267–272.

Bischof, C., Carle, A., Khademi, P., and Mauer, A., 1994, The ADIFOR 2.0 system for the automatic differentiation of Fortran 77 programs, *Argonne Preprint ANL/MCS-P481-1194*, Argonne National Laboratory, Argonne, WI, USA.

Bischof, C., Carle, A., Khademi, P., and Mauer, A., 1996, The ADIFOR 2.0 system for the automatic differentiation of Fortran 77 programs, *IEEE Computational Science & Engineering*, **3**:18–32.

Bisschop, J., and Entriken, R., 1993, *AIMMS, The Modeling System*, Paragon Decision Technology, Haarlem, Netherlands.

Bisschop, J., and Fourer, R., 1996, New constructs for the description of combinatorial optimization problems in algebraic modeling languages, *Computational Optimization and Applications*, **6**(1):83–116.

Bisschop, J., and Kuip, C., 1993, Hierarchical sets in mathematical programming, *Computational Optimization and Applications*, **1**(4):415–438.

Bodily, S., 1991, The Influence Diagram: A Modern Graphical Tool for Decision Modeling, CP-91-17, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Bohi, D.R., and Zimmermann, M.B., 1984, An update on econometric studies of energy demand, *Annual Review of Energy*, **9**:105–145.

Bonczek, R., Holsapple, C., and Whinston, A., 1981, *Foundations of Decision Support Systems*, Academic Press, New York, NY, USA.

Bostrom, R., and Watson, R.T. (eds), 1992, *Computer Augmented Teamwork: A Guided Tour*, Van Nostrand Reinhold, New York, NY, USA.

Botkin, J.W., Elmandjra, M., and Malitza, M., 1979, *No Limits to Learning: Bridging the Human Gap: A Report to the Club of Rome*, Pergamon Press, Oxford, UK.

Brans, J.P., and Vincke, Ph., 1985, A preference ranking organization method, *Management Science*, **31**:647–656.

Braun, H., 1982, Temperatures in Kenya (relationships with attitude; monthly and daily), *Kenya Soil Survey M18*, Ministry of Agriculture, Republic of Kenya, Nairobi, Kenya.

Brooke A., Kendrick, D., and Meeraus, A., 1988, *GAMS, A User's Guide*, The Scientific Press, Redwood City, CA, USA.

Brooke, A., Kendrick, D., and Meeraus, A., 1992, *GAMS, A User's Guide, Release 2.25*, The Scientific Press, Redwood City, CA, USA.

Bruaset, A., and Langtangen, H., 1997, Comprehensive set of tools for solving partial differential equations: Diffpack, in M. Dæhlen and A. Tveito (eds), *Numerical Methods and Software Tools in Industrial Mathematics*, Birkhäuser, Boston, MA, USA.

Carpentier, J., and Merlin, A., 1982, Optimization methods in planning and operation, *Electrical Power & Energy Systems*, **4**(1).

Changkong, V., and Haimes, Y., 1983, *Multiple Objective Decision Making: Theory and Methods*, North-Holland, Amsterdam, Netherlands.

Char, B., Geddes, K., Gonnet, G., Leong, B., Monagan, M., and Watt, S., 1992, *First Leaves: A Tutorial Introduction to Maple V*, Springer-Verlag, Berlin, Germany.

Charnes, A., and Cooper, W., 1967, *Management Models and Industrial Applications of Linear Programming*, John Wiley & Sons Inc., New York, NY, USA.

Charnes, A., and Cooper, W.W., 1977, Goal programming and multiple objective optimization, *Journal of Operations Research Society*, **1**:39–54.

Choi, Y., and Kim, S., 1992, A multiple criteria decision support system MC-VISA, *Proceedings of the Tenth International Conference on Multiple Criteria Decision Making*, National Chiao Tung University, Taipei, Taiwan, July.

Cleveland, W., and McGill, R., 1984, Graphical perception: Theory, experimentation, and application to the development of graphical methods, *Journal of the American Statistical Association*, **79**(387):531–554.

Clímaco, J., 1987, The TRIMAP package as powerful teaching tool, *Multiple Criteria Decision Making and Risk Analysis Using Microcomputers*, Vol. 56 of *NATO ASI Series, Computer and System Sciences*, Springer-Verlag, Berlin, Germany.

Clímaco, J., and Antunes, C., 1992, Man-machine interfacing in MCDA, *Proceedings of the Tenth International Conference on Multiple Criteria Decision Making*, National Chiao Tung University, Taipei, Taiwan, July.

Code, E., 1970, A relational model for large shared data banks, *Communications of the Association for Computing Machinery*, **13**(6):377–387.

Cofała, J., Kurz, R., and Amann, M., 1997, Application of the Current EU Air Emissions Standards to the Central and Eastern European Countries: An Integrated Assessment of the Environmental Effects, Draft Final Report to the European Environmental Agency (EEA), International Institute for Applied Systems Analysis, Laxenburg, Austria.

Cohon, J., 1978, *Multiobjective Programming and Planning*, Academic Press, San Diego, CA, USA.

Cohon, J., and Marks, D., 1975, A review and evaluation of multiobjective programming techniques, *Water Resources Research*, **11**(2):208–219.

Cohon, J., and Marks, D., 1977, Reply, *Water Resources Research*, **13**(3):693–694.

Conn, A., Gould, N., and Toint, P., 1992, *LANCELOT, A Fortran Package for Large-Scale Nonlinear Optimization*, Vol. 17 of Series in Computational Mathematics, Springer-Verlag, Berlin, Germany.

Cowan, R., and Foray, D., 1995, The Changing Economics of Technological Learning, WP-95-39, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Dantzig, G.B., 1963, *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ, USA.

Darmstadter, J., and Landsberg, H., 1983, *Energy, Today and Tomorrow: Living with Uncertainty*, Resources for the Future, Washington, DC, USA.

Debreu, G., 1959, *Theory of Value: An Axiomatic Analysis of General Equilibrium*, Yale University Press, New Haven, CT, USA.

DeMarchi, C., Jolma, A., Masliev, I., Perera, B., Smith, M., and Somlyódy, L., 1996, A Spreadsheet Tool for River Environment Assessment, Management and Planning, STREAMPLAN, Users Manual, International Institute for Applied Systems Analysis, Laxenburg, Austria.

DeMarchi, C., Ivanov, P., Jolma, A., Masliev, I., Smith, M., and Somlyódy, L., Innovative tools for water quality management: DESERT and STREAMPLAN, *Water Science and Technology* (forthcoming).

DeSanctis, G., 1984, Computer graphics as decision aids: Directions for research, *Decision Science*, **15**:463–487.

Dinkelbach, W., and Isermann, H., 1973, On decision making under multiple criteria and under incomplete information, in J.L. Cochrane and M. Zeleny (eds), *Multiple Criteria Decision Making*, University of South Carolina Press, Columbia, SC, USA.

Dolk, D., 1988, Model management and structured modeling: The role of an information resource dictionary system, *Communications of the Association for Computing Machinery*, **31**(6):704–718.

Dreyfus, H., and Dreyfus, S., 1986, *Mind over Machine: The Role of Human Intuition and Expertise in the Era of Computers*, Free Press, New York, NY, USA.

Drud, A.S., 1992, CONOPT: A large scale GRG code, *ORSA Journal on Computing*, **6**(2):207–218.

Drud, A., 1996, CONOPT: A system for large scale nonlinear optimization, *Reference Manual for CONOPT Subroutine Library*, ARKI Consulting and Development A/S, Bagsvaerd, Denmark.

Drud, A., 1997, Interactions between nonlinear programming and modeling systems, *Mathematical Programming*, **79**(1–3):99–123.

Eick, S., and Wills, G., 1995, High interaction graphics, *European Journal of Operational Research*, **81**:445–459.

Emery, J., 1987, *Management Information Systems: The Critical Strategic Resource*, Oxford University Press, New York, NY, USA.

Ermoliev, Yu., and Wets, R.J-B., 1988, *Numerical Techniques for Stochastic Optimization*, Springer-Verlag, Berlin, Germany.

Exp, 1995, *Expert Choice – Decision Support Software, Tutorial*, Version 9.0 edition, Expert Choice Inc., Pittsburgh, PA, USA.

FAO, 1976, A Framework for Land Evaluation, Technical Report, Food and Agriculture Organization of the United Nations, Rome, Italy.

FAO, 1981, Report on the Agro-ecological Zones Project: Methodology and Results for Africa, World Soil Resources Report 48/1, Food and Agriculture Organization of the United Nations, Rome, Italy.

FAO, 1984, Guidelines: Land Evaluation for Rainfed Agriculture, Technical Report, Food and Agriculture Organization of the United Nations, Rome, Italy.

FAO, 1985, Guidelines: Land Evaluation for Rainfed Agriculture, Technical Report, Food and Agriculture Organization of the United Nations, Rome, Italy.

FAO, 1988, FAO-UNESCO Soil Map of the World: Revised Legend, World Soil Resources Report 60, Food and Agriculture Organization of the United Nations, Rome, Italy.

FAO, 1990, Land Evaluation for Development, Technical Report, Food and Agriculture Organization of the United Nations, Rome, Italy.

FAO, 1993, Guidelines for Land Use Planning, FAO Development Series 1, Food and Agriculture Organization of the United Nations, Rome, Italy.

FAO, 1995, Planning for Sustainable Use of Land Resources: Towards a New Approach, Land and Water Bulletin 2, Food and Agriculture Organization of the United Nations, Rome, Italy.

FAO/IIASA, 1991, Agro-ecological Land Resources Assessment for Agricultural Development Planning: A Case Study of Kenya, World Soil Resources Reports and Technical Annex 1-8 71/8–71/8, International Institute for Applied Systems Analysis, Laxenburg, Austria and Food and Agriculture Organization of the United Nations, Rome, Italy.

FAO/IIASA, 1993, Agro-ecological Assessment for National Planning: The Example of Kenya, FAO Soils Bulletin 67, Food and Agriculture Organization of the United Nations, Rome, Italy.

FAO/IIASA/UNFPA, 1983, Potential Population Supporting Capacities of Land in the Developing World, Technical Report of Project Land Resources for Populations of the Future FPA/INT/13, Food and Agriculture Organization of the United Nations and United Nations Fund for Population Activities, Rome, Italy, and International Institute for Applied System Analysis, Laxenburg, Austria.

Feldbaum, A.A., 1966, *Optimal Control Systems,* Academic Press, New York, NY, USA.

Fiala, W., 1982, Raumwärmebedarf: Modell zur Bewertung der wesentlichen Einflußfaktoren, *Energiepfade für Österreich,* Schriftenreihe der TU Wien, Metrica, Vienna, Austria.

Findeisen, W., Bailey, F.N., Brdyś, M., Malinowski, K., Tatjewski, P., and Woźniak, A., 1980, *Control and Coordination in Hierarchical Systems*, John Wiley & Sons Ltd, Chichester, UK.

Fischer, G., and Antoine, J., 1994a, Agro-ecological Land Resources Assessment for Agricultural Development Planning: A Case Study of Kenya – Making Land Use Choices for District Planning, Report 71/9, Food and Agriculture Organization of the United Nations and United Nations Fund for Population Activities, Rome, Italy, and International Institute for Applied System Analysis, Laxenburg, Austria.

Fischer, G., and Antoine, J., 1994b, Agro-ecological Land Resources Assessment for Agricultural Development Planning: A Case Study of Kenya – Making Land Uses Choices for District Planning, User Manual and Software, International Institute for Applied Systems Analysis, Laxenburg, Austria, and Food and Agriculture Organization of the United Nations, Rome, Italy.

Fischer, G., and Heilig, G., 1996, Population Momentum and Demand on Land and Water Resources, WP-96-149, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Fischer, G., and van Velthuizen, H., 1996, Climate Change and Global Agricultural Potential: A Case Study of Kenya, WP-96-071, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Fischer, G., Ermoliev, Y., Keyzer, M., and Rosenzweig, C., 1996, Simulating the Socio-economic and Bio-geophysical Driving Forces of Land-use and Land-cover Change: The IIASA Land-use Change Model for Multistage Stochastic Programs, WP-96-010, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Fischer, G., Granat, J., and Makowski, M., 1998, AEZWIN: An Interactive Multiple-criteria Analysis Tool for Land Resources Appraisal, IR-98-051, International Institute for Applied Systems Analysis, Laxenburg, Austria.
Available on-line from http://www.iiasa.ac.at/∼marek/pubs.

Fishbone, L.G., Giesen, G.A., Goldstein, H.A., Hymen, H.A., *et al.*, 1983, User's Guide for MARKAL (BNL/KFA Version 2.0), BNL-46319, Brookhaven National Laboratory, Upton, New York, NY, USA.

Fletcher, R., 1987, *Practical Methods of Optimization*, John Wiley & Sons Inc., New York, NY, USA.

Fodor, J., and Roubens, M., 1994, *Fuzzy Preference Modeling and Multicriteria Decision Support*, Kluwer Academic Publishers, Dordrecht, Netherlands.

Fourer, R., 1983, Modeling languages versus matrix generators for linear programming, *ACM Trans. Math. Software*, **9**(2):143–183.

Fourer, R., Gay, D., and Kernighan, B., 1993, *AMPL, A Modeling Language for Mathematical Programming*, The Scientific Press, San Francisco, CA, USA.

Fourer, R., Gay, D., and Kernighan, B., 1996, *Using AMPL Plus*, Compass Modeling Solutions Inc., San Francisco, CA, USA.

Fragnière, E., Gondzio, J., Sarkissian, R., and Vial, J., 1997, Structure Exploiting Tool in Algebraic Modeling Languages, Technical Report 1997.2, Department of Management Studies, University of Geneva, Geneva, Switzerland.
Available on-line from http://www.ecolu-info.unige.ch/∼logilab.

French, S., 1986, *Decision Theory: An Introduction to the Mathematics of Rationality*, John Wiley & Sons Inc., New York, NY, USA.

Gabriel, K., 1971, The biplot graphic display of matrices with application to principal component analysis, *Biometrica*, **58**(3):453–467.

Gal, T., 1977, A general method of determining the set of all efficient solutions to a linear vectormaximum problem, *European Journal of Operational Research*, **1**(5): 307–322.

Gardner, H., 1984, *The Mind's New Science: A History of the Cognitive Revolution*, Basic Books, New York, NY, USA.

Gay, D., 1991, Automatic differentiation of nonlinear AMPL models, in A. Griewank and G. Corliss (eds), *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, SIAM, Philadelphia, PA, USA.

Gay, D., 1996, More AD of nonlinear AMPL models, in M. Bertz, C. Bischof, G. Corliss, and A. Griewank (eds), *Computational Differentiation: Techniques, Applications, and Tools*, SIAM, Philadelphia, PA, USA.

Geoffrion, A., 1987a, An introduction to structured modeling, *Operations Research*, **37**:30–51.

Geoffrion, A., 1987b, An introduction to structured modeling, *Management Science*, **33**(5):547–588.

Geoffrion, A., 1989a, Computer-based modeling environments, *European Journal of Operational Research*, **41**(1):33–43.

Geoffrion, A., 1989b, The formal aspects of structured modeling, *Operations Research*, **37**(1):30–51.

Geoffrion, A., 1992a, The SML language for structured modeling: Levels 1 and 2, *Operations Research*, **40**(1):38–57.

Geoffrion, A., 1992b, The SML language for structured modeling: Levels 3 and 4, *Operations Research*, **40**(1):58–75.

Geoffrion, A., 1994, Structured modeling: Survey and future research directions, *ORSA CTS Newsletter*, **15**(1):10–20.

Geoffrion, A., Dyer, J.S., and Feinberg, A., 1972, An interactive approach for multicriterion optimization, with an application to the operation of an academic department, *Management Science*, **19**:357–368.

Gill, Ph.E., Murray, W., and Wright, M.H., 1981, *Practical Optimization*, Academic Press, London, UK.

Gill, P.E., Murray, W., Murtagh, B.A., Saunders, M.A., and Wright, M.H., 1992, GAMS/MINOS, in A. Brooke, D. Kendrick, and A. Meeraus (eds), *GAMS – Release 2.25*, The Scientific Press, San Francisco, CA, USA.

Glushkov, V.M., 1972, Basic principles of automation in organizational management systems, *Upravlayushcheye Sistemy i Mashiny*, Vol. 1 [in Russian].

Goldemberg, J., Johansson, T.B., Reddy, A.K.N., and Williams, R.H., 1988, *Energy for a Sustainable World*, John Wiley & Sons Inc., New York, NY, USA.

Gollmer, J., and Wutke, B., 1994, Hierarchische Entscheidungsfindung in komplexen Planungsprozessen der Energieversorgung: Strategien, Modelle, Beispiele, Optimierung in der Energieversorgung, *VDI-Berichte 1140*, VDI-Verlag, Düsseldorf, Germany.

Gondzio, J., 1995, HOPDM (Version 2.12): A fast LP solver based on a primal-dual interior point method, *European Journal of Operational Research*, **85**:221–225.

Gondzio, J., 1996a, Multiple centrality corrections in a primal-dual method for linear programming, *Computational Optimization and Applications*, **6**:137–156.

Gondzio, J., 1996b, Warm Start of the Primal-Dual Method Applied in the Cutting Plane Scheme, Technical Report 96.3, Department of Management Studies, University of Geneva, Geneva, Switzerland.
Available on-line from http://www.ecolu-info.unige.ch/∼logilab.

Gondzio, J., 1997, Presolve analysis of linear programs prior to applying an interior point method, *INFORMS Journal on Computing*, **9**(1):73–91.

Gondzio, J., and Makowski, M., 1995a, HOPDM: Modular Solver for LP Problems, User's Guide to Version 2.12, WP-95-50, International Institute for Applied Systems Analysis, Laxenburg, Austria.
Available on-line from http://www.iiasa.ac.at/∼marek/pubs.

Gondzio, J., and Makowski, M., 1995b, Solving a class of LP problems with primal-dual logarithmic barrier method, *European Journal of Operational Research*, **80**(1):184–192.

Granat, J., 1993, Parametric programming approaches to local approximation of the efficient frontier, in J. Wessels and A.P. Wierzbicki (eds), *User-Oriented Methodology and Techniques of Decision Analysis and Support*, Vol. 397 of Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, Berlin, Germany.

Granat, J., and Makowski, M., 1998, ISAAP – Interactive Specification and Analysis of Aspiration-based Preferences, IR-98-052, International Institute for Applied Systems Analysis, Laxenburg, Austria.
Available on-line from http://www.iiasa.ac.at/∼marek/pubs.

Granat, J., Kręglewski, T., Paczyński, J., and Stachurski, A., 1994, IAC-DIDAS-N++ Modular Modeling and Optimization System, Part I: Theoretical Foundations, Part II: Users Guide, Report of the Institute of Automatic Control, Warsaw University of Technology, Warsaw, Poland.

Granat, J., Kręglewski, T., Paczyński, J., Stachurski, A., and Wierzbicki, A.P., 1996, A modular system of software tools for multicriteria model analysis, in J. Doležal and J. Fidler (eds), *System Modelling and Optimization*, Chapman & Hall, London, UK, pp. 363–370.

Grauer, M., Messner, S., and Strubegger, M., 1985a, Interactive Decision Analysis in Energy Planing and Policy Assessment, WP-85-59, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Grauer, M., Messner, S., and Strubegger, M., 1985b, An integrated programming package for multiple-criteria decision analysis, in M. Grauer, M. Thompson, and A.P. Wierzbicki (eds), *Plural Rationality and Interactive Decision Processes*, Vol. 248 of Lectures in Economics and Mathematical Systems, Springer-Verlag, Berlin, Germany.

Grauer, M., Thompson, M., and Wierzbicki, A.P. (eds), 1985c, *Plural Rationality and Interactive Decision Processes*, Vol. 248 of Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, Berlin, Germany.

Griewank, A., 1994, Computational differentiation and optimization, in J. Birge and K. Murty (eds), *Mathematical Programming: State of the Art*, University of Michigan Press, Ann Abor, MI, USA, pp. 102–131.

Griewank, A., and Corliss, G. (eds), 1993, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, SIAM, Philadelphia, PA, USA.

Grigg, N., 1996, *Water Resources Management: Principles, Regulations, and Cases*, McGraw-Hill, New York, NY, USA.

Gross, B.M., 1966, Space-time and post-industrial society, *1965 Seminar of the American Society of Public Administration*, Syracuse University, Syracuse, NY, USA.

Güler, O., den Hertog, D., Roos, C., Terlaky, T., and Tsuchiya, T., 1993, Degeneracy in interior point methods for linear programming: A survey, *Annals of Operations Research*, **46**:107–138.

Guilmot, J-F., McGlue, D., Valette, P., and Waeterloos, G., 1986, *Energy 2000*, Commission of the European Communities, Brussels, Belgium.

Haagsma, I., 1996, Platform Independent Storage of Data: An Application of HDF for Simplified Ozone Model, WP-96-133, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Hacking, I. (ed.), 1981, *Scientific Revolutions*, Oxford University Press, Oxford, UK.

Häfele, W. (ed.), 1981, *Energy in a Finite World*, Ballinger Publishing Company, Cambridge, MA, USA.

Häfele, W., Barnett, H., Messner, S., Strubegger, M., and Anderer, J., 1986, Novel integrated energy systems: The case of zero emissions, in W.C. Clark and R.E. Munn (eds), *Sustainable Development of the Biosphere*, Cambridge University Press, Cambridge, UK.

Haimes, Y., and Hall, W., 1974, Multiobjectives in water resource systems analysis: The surrogate trade-off method, *Water Resources Research*, **10**:615–624.

Hall, W., and Dracup, J., 1970, *Water Resources Systems Engineering*, McGraw-Hill, New York, NY, USA.

Hamilton, L.D., 1994, New York MARKAL: An evaluation of carbon dioxide emission control in New York state, *International Journal of Global Energy Issues*, **6**(1/2).

Hanselmann, M., Friedrich, R., Hönes, R., and Schaal, D., 1992, Tageseinsatzoptimierung mit dem Programmsystem Profako, *Elektrizitätswirtschaft*, **91**(9).

Harhammer, P.G., 1986, Entscheidungsunterstützendes System Energie Management, *Der Wirtschaftsingenieur*, **18**(2):36–38.

Healy, W.C., 1964, Multiple choice programming, *Operations Research*, **12**:122–138.

Hemming, T., and Troutt, M., 1992, Go-ii, a graphically oriented system for discrete interactive multi-criterion decisions, *Proceedings of the Tenth International Conference on Multiple Criteria Decision Making*, National Chiao Tung University, Taipei, Taiwan, July.

Henig, M., 1982, Proper efficiency with respect to cones, *Journal of Optimization Theory and Applications*, **22**:618–630.

Hestenes, M.R., 1980, *Conjugate Direction Methods in Optimization*, Springer-Verlag, Berlin, Germany.

Hipel, K., 1992, Multiple objective decision making in water resources, *Water Resources Bulletin*, **28**(1):3–22.

Hirst, E., 1980, A model of household fuel uses, in *Simulation of Energy Systems, Part 1*, Simulation Council Inc., La Jolla, CA, USA.

Houghton, J.T., and Callander, B.A. (eds), 1992, *The Supplementary Report to the IPCC Scientific Assessment*, Cambridge University Press, Cambridge, UK.

Houthakker, H.S., and Kennedy, M., 1979, A long-run model of world energy demands, supplies and prices, in *Directions in Energy Policy: A Comprehensive Approach to Energy Resource Decision-Making*, Ballinger Publishing Company, Cambridge, MA, USA.

Hudson, E.A., and Jorgenson, D.W., 1974, U.S. energy policy and economic growth, 1975–2000, *Bell Journal of Economics and Management Science*, **5**(2).

Hudson, E.A., and Jorgenson, D.W., 1979, The economic impact of policies to reduce U.S. energy growth, in *Directions in Energy Policy: A Comprehensive Approach to Energy Resource Decision-Making*, Ballinger Publishing Company, Cambridge, MA, USA.

Huntley, I., and James, D. (eds), 1990, *Mathematical Modelling: A Source Book of Case Studies*, Oxford University Press, Oxford, UK.

IBM, 1989, *Entscheidungsunterstützendes System Energy Management, Programmsystem JARO, Bedienerhandbuch*, IBM, Vienna, Austria.

IBM, 1992, *Optimization Subroutine Library, Guide and Reference, Release 2,* IBM, Kingston, NY, USA.

IIASA, 1992, Science and sustainability, *Selected Papers on IIASA's 20th Anniversary*, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Ikebuchi, S., and Kojiri, T., 1992, Multi-objective reservoir operation including turbidity control, *Water Resources Bulletin*, **28**(1):223–232.

Ingarden, R., 1972, What is new in Husserl's 'Crisis', in A. Tymieniecka (ed.), *The Later Husserl and the Idea of Phenomenology,* D. Reidel Publishing Co., Dordrecht, Netherlands.

IPCC (Intergovernmental Panel for Climate Change), 1990, *Climate Change: The IPCC Scientific Assessment,* Cambridge University Press, Cambridge, UK.

Ivanov, P., Masliev, I., DeMarchi, C., and Somlyódy, L., 1996, Decision Support System for Evaluating River Basins Strategies DESERT, Users Manual, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Jansen, B., de Jong, J., Ross, C., and Terlaky, T., 1993, Sensitivity Analysis in Linear Programming: Just Be Careful! Technical Report AMER.93.022, Shell Internationale Research Maatschappij B.V., The Hague, Netherlands.

Janssen, R., 1992, *Multiobjective Decision Support for Environmental Management,* Kluwer Academic Publishers, Dordrecht, Netherlands.

Jarvenpaa, S., and Dickson, G., 1988, Graphics and managerial decision making: Research based guidelines, *Communications of the Association for Computing Machinery,* **31**(6):764–773.

Jaszkiewicz, A., and Słowiński, R., 1992a, Cone contraction method with visual interaction for multiple-objective non-linear programs, *Journal of Multi-Criteria Decision Analysis,* **1**:29–46.

Jaszkiewicz, A., and Słowiński, R., 1992b, The light beam search over non-dominated surface of a multiple-objective programming problem, *Proceedings of the Tenth International Conference on Multiple Criteria Decision Making*, National Chiao Tung University, Taipei, Taiwan, July.

Jaszkiewicz, A., and Słowiński, R., 1994, The LBS Package: A Microcomputer Implementation of the Light Beam Search Method for Multi-objective Nonlinear Mathematical Programming, WP-94-07, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Johansen, R., 1988, *Groupware: Computer Support for Business Firms,* The Free Press, New York, NY, USA.

Jolma, A., DeMarchi, C., Smith, M., Perera, B.J.C., and Somlyódy, L., 1997, StreamPlan: A support system for water quality management on a river basin scale, *Environmental Modelling and Software,* **12**(4):275–284.

Jones, C., 1996, *Visualization and Optimization*, Kluwer Academic Publishers, Dordrecht, Netherlands.

Joseffson, A., Johnsson, J., and Wene, C.O., 1994, Community Based Regional Energy-Environmental Planning, WP-78, Fondazione Eni Enrico Mattei, Milano, Italy.

Kahneman, D., and Tversky, A., 1982, The psychology of preferences, *Scientific American*, **246**:160–173.

Kaliszewski, I., 1994, *Quantitative Pareto Analysis by Cone Separation Techniques*, Kluwer Academic Publishers, Dordrecht, Netherlands.

Kallio, M., Lewandowski, A., and Orchard-Hays, W., 1980, An Implementation of the Reference Point Approach for Multi-Objective Optimization, WP-80-35, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Kang, M., Wright, G., Chandrasekharan, R., Mookerjee, R., and Worobetz, N., 1997, The design and implementation of OR/SM: A prototype integrated modeling environment, *Annals of Operations Research*, **72**:211–240.

Kassam, A., 1980, Multiple Cropping and Rainfed Crop Productivity in Africa, Report on the Second FAO/UNFPA Expert Consultation on Land Resources for Populations of the Future, Food and Agriculture Organization of the United Nations, Rome, Italy.

Keating, M., and Shah, M., 1993, *The Earth Summit's Agenda for Change: A Plain Language Version of Agenda-21 and the Other Rio Agreements*, Centre for Our Common Future, Geneva, Switzerland.

Keeney, R.L., 1992, *Value-Focused Thinking: A Path to Creative Decisionmaking*, Harvard University Press, Cambridge, MA, USA.

Keeney, R.L., and Raiffa, H., 1976, *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*, John Wiley & Sons Inc., New York, NY, USA.

Kenya Soil Survey, 1995, 1:1 m Soils and Terrain Database of Kenya (KENSOTER), Technical Report, Agricultural Research Institute, Nairobi, Kenya.

Kiwiel, K.C., 1985, *Methods of Descent in Nondifferentiable Optimization*, Springer-Verlag, Berlin, Germany.

Klaassen, G., 1991, Costs of Controlling Ammonia Emissions in Europe, SR-91-02, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Klimont, Z., Amann, M., and Cofała, J., 1998, Estimating Costs for Controlling Emissions of Volatile Organic Compounds (VOC) from Stationary Sources in Europe, Technical Report, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Kok, M., and Lootsma, F.A., 1985, Pairwise comparison methods in multiple objective programming, *European Journal of Operational Research*, **22**:44–55.

Korhonen, P., 1986, Solving discrete multiple criteria problems by using visual interaction, in G. Fandel, M. Grauer, A. Kurzhanski, and A.P. Wierzbicki (eds), *Large Scale Modelling and Interactive Decision Analysis*, Vol. 273 of Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, Berlin, Germany.

Korhonen, P., 1991, Using harmonious houses for visual pairwise comparison of multiple criteria alternatives, *Decision Support Systems*, **7**:47–54.

Korhonen, P., and Laakso, J., 1984, A visual interactive method for solving the multiple-criteria problem, in M. Grauer and A.P. Wierzbicki (eds), *Interactive Decision Analysis*, Vol. 229 of Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, Berlin, Germany.

Korhonen, P., and Laakso, J., 1986, A visual interactive method for solving the multiple criteria problem, *European Journal of Operational Research*, **24**:277–287.

Korsan, R., 1996, *Decision Support Systems in Mathematica*, Springer-Verlag, Berlin, Germany.

Kosslyn, S.M., 1980, *Image and Mind*, Harvard University Press, Cambridge, MA, USA.

Koziol, Q., and Matzke, R., 1998, HDF5: A New Generation of HDF, Reference Manual and User Guide, National Center for Supercomputing Applications, Champaign, IL, USA. Available on-line from http://hdf.ncsa.uiuc.edu/nra/HDF5/.

Kręglewski, T., Paczyński, J., Granat, J., and Wierzbicki, A.P., 1988, IAC-DIDAS-N: A Dynamic Interactive Decision Analysis and Support System for Multicriteria Analysis of Nonlinear Models with Nonlinear Model Generator Supporting Model Analysis, CP-88-112, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Kręglewski, T., Granat, J., and Wierzbicki, A.P., 1991, IAC-DIDAS-N: A Dynamic Interactive Decision Analysis and Support System for Multicriteria Analysis of Nonlinear Models, CP-91-010, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Krzysztoforowicz, R., Castano, E., and Fike, R., 1977, Comment on "a review and evaluation of multiobjective programming techniques", *Water Resources Research*, **13**(3):690–692.

Kümmerling, H., 1990, JARO/KWOPT: Ein Programmsystem zur Ermittlung der optimalen Strombeschaffung für EVU, *Elektrizitätswirtschaft*, **89**(13).

Kuhn, T.S., 1964, A function of thought experiments, in I. Hacking (ed.), *Scientific Revolutions*, Oxford University Press, Oxford, UK (originally published in *L'aventure de la science, Melanges Alexandre Koyre*, **2**:307–334, Hermann, Paris, 1964).

Kuhn, T.S., 1970, *The Structure of Scientific Revolutions*, 2nd edition, Chicago University Press, Chicago, IL, USA.

Kuip, C., 1993, Algebraic languages for mathematical programming, *European Journal of Operational Research*, **67**(1):25–51.

Kydes, A.S., Shaw, S.H., and McDonald, D.F., 1995, Beyond the horizon: Recent directions in long-term energy modeling, *Energy*, **20**(2).

Laabs, H., and Schultz, G., 1992, Reservoir management rules derived with the aid of multiple objective decision making techniques, *Water Resources Bulletin*, **28**(1):211–222.

Land, A.H., and Powell, S., 1979, Computer codes for problems of integer programming, in P.L. Hammer, E.L. Johnson, and B.H. Korte (eds), *Annals of Discrete Mathematics*, **5**:221–269, *Discrete Optimization*, North-Holland, Amsterdam, Netherlands.

Landis & Gyr, 1993, *Kraftwerkseinsatzplanung, Kraftwerkseinsatzoptimierung EKW/ ÖDK, Realisierungspflichtenheft*, Vienna, Austria.

Landis & Gyr, 1995, *Energieeinsatzoptimierung, Anforderungspflichtenheft*, Vienna, Austria.

Lawrence, C., Zhou, J., and Tits, A., 1996, User's Guide for CFSQP Version 2.4: A C Code for Solving (Large Scale) Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Inequality Constraints, Technical Report, Institute for Systems Research, University of Maryland, College Park, MD, USA.

Lewandowski, A., and Granat, J., 1991, Dynamic BIPLOT as the interaction interface for aspiration based decision support systems, in P. Korhonen, A. Lewandowski, and J. Wallenius (eds), *Multiple Criteria Decision Support*, Vol. 356 of Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, Berlin, Germany.

Lewandowski, A., and Wierzbicki, A.P. (eds), 1989, *Aspiration Based Decision Support Systems*, Vol. 331 of Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, Berlin, Germany.

Lewandowski, A., Rogowski, T., and Kręglewski, T., 1985, A trajectory oriented extension of DIDAS and its applications, in M. Grauer, M. Thompson, and A.P. Wierzbicki (eds), *Plural Rationality and Interactive Decision Processes*, Vol. 248 of Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, Berlin, Germany.

Liu, R.R.-S., and Hou, Y.-W.C., 1992, A hypertext system implemented on X window, *Proceedings of the Tenth International Conference on Multiple Criteria Decision Making*, National Chiao Tung University, Taipei, Taiwan, July.

Lootsma, F.A., 1993, Scale sensitivity in multiplicative AHP and SMART, *Journal of Multi-Criteria Decision Analysis*, **2**(2):87–110.

Lorenz, K., 1965, *Evolution and Modification of Behavior: A Critical Examination of the Concepts of the "Learned" and the "Innate" Elements of Behavior*, The University of Chicago Press, Chicago, IL, USA.

Loucks, D., Stedinger, J., and Haith, D., 1981, *Water Resource Systems Planning and Analysis*, Prentice-Hall, Englewood Cliffs, NJ, USA.

Lucas, H., 1981, An experimental investigation of the use of computer-based graphics in decision making, *Management Science*, **27**(7):757–768.

Luenberger, D.G., 1973, *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, Reading, MA, USA.

Luenberger, D.G., 1984, *Linear and Nonlinear Programming*, 2nd edition, Addison-Wesley, Reading, MA, USA.

Machina, M.J., 1983, The economic theory of individual behavior toward risk: Theory, evidence and new directions, *Technical Report No. 433*, Center for Research on Organizational Efficiency, Stanford University, Stanford, CA, USA.

Makowski, M., 1994a, Design and Implementation of Model-based Decision Support Systems, WP-94-86, International Institute for Applied Systems Analysis, Laxenburg, Austria. Available on-line from http://www.iiasa.ac.at/∼marek/pubs/.

Makowski, M., 1994b, LP-DIT, Data Interchange Tool for Linear Programming Problems, (Version 1.20), WP-94-36, International Institute for Applied Systems Analysis, Laxenburg, Austria. Available on-line from http://www.iiasa.ac.at/∼marek/pubs/.

Makowski, M., 1994c, Methodology and a Modular Tool for Multiple Criteria Analysis of LP Models, WP-94-102, International Institute for Applied Systems Analysis, Laxenburg, Austria.
Available on-line from http://www.iiasa.ac.at/∼marek/pubs/.

Makowski, M., 1996, Methodology and modular tools for aspiration-led analysis of LP models, in J. Doležal and J. Fidler (eds), *System Modelling and Optimization*, Chapman & Hall, London, UK, pp. 371–378.

Makowski, M., LP-DIT++, C++ class library for Data Interchange Tool for Linear Programming Problems (Version 2.06), International Institute for Applied Systems Analysis, Laxenburg, Austria (forthcoming).

Makowski, M., 2000, Modeling paradigms applied to the analysis of European air quality, *European Journal of Operations Research*, **122**(2):23.

Makowski, M., and Sosnowski, J., 1989, Mathematical programming package HYBRID, in A. Lewandowski and A.P. Wierzbicki (eds), *Aspiration Based Decision Support: Theory, Software and Applications*, Vol. 331 of Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, Berlin, Germany.

Makowski, M., and Sosnowski, J., 1993a, HYBRID: Multicriteria Linear Programming System for Computers under DOS and UNIX, in J. Wessels and A.P. Wierzbicki (eds), *User-Oriented Methodology and Techniques for Decision Analysis and Support*, Vol. 397 of Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, Berlin, Germany.

Makowski, M., and Sosnowski, J.S., 1993b, HYBRID-FMS: An Element of DSS for Designing Flexible Manufacturing Systems, in P. Korhonen, A. Lewandowski, and J. Wallenius (eds), *Multiple Criteria Decision Support*, Vol. 356 of Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, Berlin, Germany.

Makowski, M., Heyes, C., and Schöpp, W., 1998, Specification of Mathematical Model for Cost-effective Control of Acidification and Ground Level Ozone, Technical Report, International Institute for Applied Systems Analysis, Laxenburg, Austria.
Available on-line from http://www.iiasa.ac.at/∼marek/pubs/prepub.html.

Makowski, M., Somlyódy, L., and Watkins, D., 1995, Multiple Criteria Analysis for Regional Water Quality Management: The Nitra River Case, WP-95-22, International Institute for Applied Systems Analysis, Laxenburg, Austria.
Available on-line from http://www.iiasa.ac.at/∼marek/pubs.

Makowski, M., Somlyódy, L., and Watkins, D., 1996, Multiple criteria analysis for water quality management in the Nitra basin, *Water Resources Bulletin*, **32**(5):937–951.

Mangasarian, O.L., 1981, Iterative solution of linear programs. *SIAM Journal for Numerical Analysis*, **18**(4):606–614.

Manne, A.S., and Richels, R.G., 1992, *Buying Greenhouse Insurance: The Economic Costs of CO_2 Emission Limits*, The MIT Press, Cambridge, MA, USA.

March, J., and Simon, H., 1958, *Organizations*, John Wiley & Sons Inc., New York, NY, USA.

Mareschal, B., and Brans, J.-P., 1988, Geometrical representations for MCDA, *European Journal of Operational Research*, **34**:69–77.

Mehrotra, S., 1992, On the implementation of a primal–dual interior point method, *SIAM Journal on Optimization*, **2**:575–601.

Messner, S., and Strubegger, M., 1995, User's Guide for MESSAGE III, WP-95-69, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Michalevich, M.V., 1986, Stochastic Approaches to Interactive Multicriteria Optimization Problems, WP-86-10, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Monagan, M., 1996, Automatic differentiation in computer algebra systems: An implementation of the forward and reverse mode in Maple, in M. Bertz, C. Bischof, G. Corliss, and A. Griewank (eds), *Computational Differentiation: Techniques, Applications, and Tools*, SIAM, Philadelphia, PA, USA.

Moré, J., and Wright, S. (eds), 1993, *Optimization Software Guide*, SIAM, Philadelphia, PA, USA.

Moroney, J.R. (ed.), 1984, Econometric models of the demand for energy, *Advances in Economics of Energy and Resources*, **5**, Jai Press Inc., Greenwich, UK.

Murtagh, B.A., and Saunders, M.A., 1983, MINOS 5.0 user's guide, *Technical Report SOL 83-20*, Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, CA, USA.

Murtagh, B.A., and Saunders, M.A., 1987, MINOS 5.1 user's guide, *Technical Report SOL 83-20R*, Stanford University, Stanford, CA, USA.

Muser, D., and Saini, A., 1995, *STL Tutorial and Reference Guide, C++ Programming with the Standard Template Library*, Addison-Wesley, Reading, UK.

Nakayama, H., 1994, Aspiration Level Approach to Interactive Multi-objective Programming and Its Applications, WP-94-112, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Nakićenović, N., and Messner, S., 1983, Die zukünftige Nutzung der Sonnenenergie in Westeuropa, Forschungsbericht T83-001, Bundesministerium für Forschung und Technologie, Bonn, Germany.

Nakićenović, N., Grübler, A., and McDonald, A. (eds), 1998, *Global Energy Perspectives*, Cambridge University Press, Cambridge, UK.

Nemhauser, G.L., and Wolsey, L.A., 1988, *Integer and Combinatorial Optimization*, John Wiley & Sons Inc., New York, NY, USA.

Ng, W.Y., 1991, An interactive descriptive graphical approach to data analysis for trade-off decisions in multi-objective programming, *Information and Decision Technologies*, **17**:133–149.

Nielsen, S., 1995, A C++ class library for mathematical programming, in A. Sofer and S. Nash (eds), *The Impact of Emerging Technologies on Computer Science and Operations Research*, Kluwer, Dordrecht, Netherlands.

NRC (National Research Council), 1991, Managing Global Genetic Resources: Forest Trees, Technical Report, National Research Council, Board on Agriculture, National Academy Press, Washington, DC, USA.

Nunamaker, J., 1990, Electronic meeting system to support group work: Theory and practice in Arizona, Working Paper, College of Business, University of Arizona, Tucson, AZ, USA.

OECD (Organisation for Economic Co-operation and Development), 1993, Advanced emission controls for power plants, OECD documents, OECD, Paris, France.

Ogryczak, W., 1996, A note on modeling multiple choice requirements for simple mixed integer programming solvers, *Computers and Operations Research*, **23**:199–205.

Ogryczak, W., and Lahoda, S., 1992, Aspiration/reservation-based decision support: A step beyond goal programming, *Journal of Multi-Criteria Decision Analysis*, **1**(2): 101–117.

Ogryczak, W., and Zorychta, K., 1993, Modular Optimizer for Mixed Integer Programming – MOMIP Version 1.1, WP-93-055, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Ogryczak, W., and Zorychta, K., 1996, Modular Optimizer for Mixed Integer Programming – MOMIP Version 2.3, WP-96-106, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Ogryczak, W., Studziński, K., and Zorychta, K., 1989, A generalized reference point approach to multiobjective transshipment problem with facility location, in A. Lewandowski and A.P. Wierzbicki (eds), *Aspiration Based Decision Support Systems* Vol. 331 of Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, Berlin, Germany.

Okken, P.A., Lako, P., Gerbers, D., Kram, T., and Ybema, J.R., 1992, CO_2 removal in comparison with other options for reduced CO_2 emissions, in *Proceedings of the first International Conference on Carbon Dioxide Removal*, Pergamon Press, Oxford, UK.

Onyeji, S., Fischer, G., and Kamau, W., 1996, Agro-ecological Assessment for National Planning in Kenya: Database Structure for District Analysis, WP-96-073, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Osyczka, A., 1994, C Version of Computer Aided Multicriterion Optimization System (CAMOS), Workshop on Advances in Methodology and Software in DSS, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Part-Enander, E., Sjöberg, A., Melin, B., and Osaksson, P., 1996, *The Matlab Handbook*, Addison-Wesley, Reading, MA, USA.

Pawlak, Z., 1991, *Rough Sets: Theoretical Aspects of Reasoning about Data*, Kluwer, Dordrecht, Netherlands.

Pfaff, R., Loibl, O., Messner, S., and Strubegger, M., 1991, Energy model for the Vienna energy plan, *Österreichische Zeitschrift für Elektrizitätswirtschaft*, May, Vienna, Austria.

Piela, P., McKelvey, R., and Westerberg, A., 1993, An introduction to the ASCEND modeling system: Its language and interactive environment, *Journal of Management Information Systems*, **9**(3):91–121.

Plauger, P., 1995, *The Draft Standard C++ Library*, Prentice-Hall, London, UK.

Polak, E., 1976, On the approximation of solutions to multiple criteria decision making problems, in M. Zeleny (ed.), *Multiple Criteria Decision Making*, Springer-Verlag, Berlin, Germany.

Polyak, B.T., and Tretiyakov, N.V., 1972, An iterative method for linear programming and its economic interpretation, *Economic and Mathematical Methods*, **8**:740–751 [in Russian].

Popper, K.R., 1945, *The Open Society and Its Enemies*, Routledge, London, UK.

Popper, K.R., 1959, *The Logic of Scientific Discovery*, Hutchinson, London, UK.

Popper, K.R., 1975, The rationality of scientific revolutions, in R. Harre (ed.), *Problems of Scientific Revolution*, Oxford University Press, Oxford, UK, pp. 72–101.

Popper, K.R., 1983, *Realism and the Aim of Science*, Hutchinson, London, UK.

Posch, M., Hettelingh, J., de Smet, P., and Downing, R., 1997, Calculation and Mapping of Critical Thresholds in Europe, Status Report, Coordination Center for Effects, RIVM, Bilthoven, Netherlands.

Pospelov, G.S., and Irikov, V.I., 1976, *Program- and Goal-Oriented Planning and Management*, Sovietskoye Radio, Moscow, Russia [in Russian].

Powell, M.D.J., 1969, A method for nonlinear constraints in minimization problems, in R. Fletcher (ed.), *Optimization*, Academic Press, London, UK.

Powell, M.D.J. (ed.), 1981, *Nonlinear Optimization 1981*, Academic Press, London, UK.

Powell, S., 1985, Software, in M. O'hEigertaigh, J.K. Lenstra and A.H.G. Rinnooy Kan (eds), *Combinatorial Optimization: Annotated Bibliographies*, John Wiley & Sons Inc., New York, NY, USA, pp. 190–194.

Pylyshyn, Z.W., 1984, *Computation and Cognition: Towards a Foundation of Cognitive Science*, MIT Press, Cambridge, MA, USA.

Raiffa, H., 1980, *The Art and Science of Negotiations*, Harvard University Press, Cambridge, MA, USA.

Rapoport, A., 1989, *Decision Theory and Decision Behavior*, Kluwer Academic Publishers, Dordrecht, Netherlands.

Rentz, O., Remmers, J., and Plinke, E. (eds), 1987, *Proceedings of the Workshop on Emission Control Costs*, 28 September – 1 October 1987, Esslingen am Neckar, Germany, Executive Body for the Convention on Long-range Transboundary Air Pollution, Institute for Industrial Production (IIP), University of Karlsruhe, Karlsruhe, Germany.

Republic of Kenya, 1986, Economic Survey 1986, Technical Report, Central Bureau of Statistics, Ministry of Planning and National Development, Nairobi, Kenya.

Republic of Kenya, 1994a, On national food policy, *Sessional Paper 2*, Nairobi, Kenya.

Republic of Kenya, 1994b, On recovery and sustainable development to the year 2010, *Sessional Paper 1*, Nairobi, Kenya.

Rios, S. (ed.), 1994, *Decision Theory and Decision Analysis: Trends and Challenges*, Kluwer Academic Publishers, Dordrecht, Netherlands.

Rockafellar, R.T., 1976, Augmented Lagrangians and applications of the proximal point algorithm in convex programming, *Mathematics of Operations Research*, **1**:97–116.

Röling, N., 1994, Platforms for decision-making about ecosystems, in L. Fresco, L. Stroosnijders, J. Bouma, and H. van Keulen (eds), *The Future of the Land: Mobilizing and Integrating Knowledge for Land Use Options*, John Wiley & Sons Inc., New York, NY, USA.

Rogner, H-H., 1982, A Long-Term Macroeconomic Equilibrium Model for the European Community, RR-82-13, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Roy, B., 1977, A conceptual framework for a prescriptive theory of decision-aid, *TIMS Studies in Management Sciences*, No. 6, North-Holland, Amsterdam, Netherlands.

Roy, B., and Vincke, Ph., 1981, Multicriteria analysis: Survey and new directions, *European Journal of Operational Research*, **8**:207–218.

Ruszczyński, A., 1993, Parallel decomposition of multistage stochastic mathematical programming problems, *Mathematical Programming*, **58**:201–228.

Ruszczyński, A., and Świętanowski, A., 1996, On the Regularized Decomposition Method for Two-stage Stochastic Linear Problems, WP-96-014, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Saaty, T., 1980, *The Analytical Hierarchy Process*, McGraw-Hill, New York, NY, USA.

Saltzman, M.J., 1994, Survey: Mixed-Integer Programming, *OR/MS Today*, April:42–51.

Sawaragi, Y., and Nakamori, Y., 1992, Shinayakana Systems Approach in Modelling and Decision Support, *Proceedings of the 10th International Conference on Multiple Criteria Decision Making*, Vol. I, National Chiao Tung University, Taipei, Taiwan, July, pp. 77–86.

Sawaragi, Y., Nakayama, H., and Tanino, T., 1985, *Theory of Multiobjective Optimization*, Academic Press, New York, NY, USA.

Schmits, K., Terhorst, W., and Voss, A., 1981, Simulation techniques in energy analysis, *Mathematical Modeling of Energy Systems*, Sijthoff, The Hague, Netherlands.

Schörer, B., 1993, Technologies to clean up power plants: Experience with a 21 billion DM FGD and SCR retrofit program in Germany, Parts 1 and 2, *Staub – Reinhaltung der Luft*, **53**:87–92, 157–160.

Scholten, J., 1994, Lösung komplexer Optimierungsprobleme am Beispiel der Energieeinsatzplanung, *Elektrizitätswirtschaft*, **93**(20).

Schrattenholzer, L., and Schäfer, A., 1996, World Regional Scenarios, Described with the 11R Model of Energy-Economy-Environment Interactions, WP-96-108, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Seo, F., and Sakawa, M., 1988, *Multiple Criteria Decision Analysis in Regional Planning*, Reidel Publishing Company, Dordrecht, Netherlands.

Shafike, N., Duckstein, L., and Maddock III, T., 1992, Multicriterion analysis of groundwater contamination management, *Water Resources Bulletin*, **28**(1):33–44.

Simon, H.A., 1955, A behavioral model of rational choice, *Quarterly Journal of Economics*, **69**:99–118.

Simon, H.A., 1957, *Models of Man*, Macmillan, New York, NY, USA.

Śmierzchalski, R., and Lisowski, J., 1994, Computer simulation of safe and optimal trajectory avoiding collision at sea, Joint Proceedings, WSM Gdynia, Poland, and Hochschule Bremerhaven, Germany.

Sobczyk, J., and Wierzbicki, A.P., 1994, Pulsar Algorithms: A Class of Coarse-grain Parallel Nonlinear Optimization Algorithms, WP-94-53, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Sombroek, W.G., and Eger, H., 1996, What do we understand by land use planning: A state-of-the-art report, *Entwicklung+ländlicher Raum*, **30**(2):3–7.

Sombroek, W.G., Braun, H.M.H., and van der Pouw, B.J.A., 1982, Exploratory soil map and agro-climatic zone map of Kenya, 1980, scale 1:1 million, *Report e1, Kenya Soil Survey*, Ministry of Agriculture, Republic of Kenya, Nairobi, Kenya.

Somlyódy, L., 1994, Quo vadis water quality management in Central and Eastern Europe? *Water Science and Technology*, **30**(5):1–14.

Somlyódy, L., 1996, Challenges of water quality management in Central and Eastern Europe, *World Meteorological Organization Bulletin*, **45**(4):342–346.

Somlyódy, L., Kularathna, M., and Masliev, I., 1994a, Development of least-cost water quality control policies for the Nitra River Basin in Slovakia, *Water Science and Technology*, **30**(5):69–78.

Somlyódy, L., Masliev, I., Petrovic, P., and Kularathna, M., 1994b, Water quality management in the Nitra River Basin, CP-94-02, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Sosnowski, J.S., 1981, Linear programming via augmented Lagrangian and conjugate gradient methods, in S. Walukiewicz and A.P. Wierzbicki (eds), *Methods of Mathematical Programming,* Proceedings of a 1977 Conference in Zakopane. Polish Scientific Publishers, Warsaw, Poland.

Sosnowski, J.S, 1990, A Method for Minimization of Piecewise Quadratic Functions with Lower and Upper Bounds, CP-90-003, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Sprague, R., 1983, A framework for the development of decision support systems, *Decision Support Systems: A Data Based, Model-Oriented, User-Developed Discipline*, Petrocelli, Princeton, NJ, USA.

Sprague, R., and Watson, H. (eds), 1993, *Decision Support Systems: Putting Theory into Practice,* Prentice-Hall, Englewood Cliffs, NJ, USA.

Springer, S., and Deutsch, G., 1981, *Left Brain – Right Brain,* Freeman, San Francisco, CA, USA.

Steinbauer, E., 1992, Erfahrungen des STEWEAG-Lastverteilers mit einem neuen Software-Produkt für die Kraftwerkseinsatzoptimierung, *Österreichische Zeitschrift für Elektrizitätswirtschaft,* **43**(2).

Steuer, R.E., 1986, *Multiple Criteria Optimization: Theory, Computation, and Application,* John Wiley & Sons Inc., New York, NY, USA.

Steuer, R.E., and Choo, E.V., 1983, An interactive weighted Tchebycheff procedure for multiple objective programming, *Mathematical Programming*, **26**:326–344.

Stroustrup, B., 1997, *C++ Programming Language*, 3rd edition, Addison-Wesley, Reading, MA, USA.

Sundström, B., Snickars, F., and Pellijeff, M., 1983, A Model for Long Term Energy Planning, Case Study Sundsvall, IEA Annex VII, Subtask C, Stockholm, Sweden.

SVOR/ASRO, 1993, *Optimization in Planning and Operation of Electric Power Systems,* Lecture notes of the SVOR/ASRO tutorial, Schweizerische Vereinigung für Operations Research, Thun, Switzerland, 14–16 October 1992, Physica-Verlag, Heidelberg, Germany.

Świętanowski, A., 1995, A Penalty Based Simplex Method for Linear Programming, WP-95-005, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Takeshita, M., 1995, Air Pollution Control Costs for Coal-fired Power Stations, Technical Report IEAPER/17, IEA Coal Research, London, UK.

Tan, K., and Benbasat, I., 1993, The effectiveness of graphical presentation for information extraction: Accumulative experimental approach, *Decision Sciences,* **24**:167–191.

Taylor, L.D., Blattenberger, G.R., and Rennhack, R.K., 1982, Some problems and issues in building econometric models of energy demand, in *Perspectives on Resource Policy*

Modeling: Energy and Minerals, Ballinger Publishing Company, Cambridge, MA, USA.

TEMAPLAN GMBH, 1982, *Long-Term Energy Supply Strategies for Stockholm County*, Böblingen, Germany.

Thiessen, E., and Loucks, D., 1992, Computer assisted negotiation of multiobjective water resources conflicts, *Water Resources Bulletin*, **28**(1):163–178.

Thomann, R., and Mueller, J., 1987, *Principles of Surface Water Quality Modeling and Control*, Harper and Row, New York, NY, USA.

Toffler, A., 1980, *The Third Wave,* W. Morrow, New York, NY, USA.

Tomlin, J.A., 1970, Branch and bound methods for integer and nonconvex programming, in J. Abadie (ed.), *Integer and Nonlinear Programming*, North-Holland, Amsterdam, Netherlands, pp. 437–450.

Tomlin, J.A., and Welch, J.S., 1993, Mathematical programming systems, in E. Coffman and J.K. Lenstra (eds), *Handbook of Operations Research and Management Science: Computation,* North-Holland, Amsterdam, Netherlands.

Tukey, J., 1977, *Exploratory Data Analysis*, John Wiley & Sons Inc., New York, NY, USA.

Turner, B., Skole, D., Sanderson, S., Fischer, G., Fresco, L., and Leemans, R., 1995, Land-use and Land-cover Change Science/Research Plan, Report 35, IGBP, Stockholm, Sweden and Geneva, Switzerland.

UN/ECE, 1996, Energy Balances for Europe and North America 1992, 1993–2010, Technical Report, United Nations Economic Commission for Europe, Geneva, Switzerland.

United Nations, 1999, World Population Prospects: The 1998 Revision, Technical Report, UN Population Division, New York, NY, USA.

van Gigch, J.P., 1991, *System Design Modeling and Metamodeling*, Plenum Press, New York, NY, USA.

Verstege, J., and Maubach, K-D., 1994, Mittelfristige Energieeinsatzoptimierung in der Energieversorgung, *VDI-Berichte*, **1140**, VDI-Verlag, Düsseldorf, Germany.

Vessey, I., and Galletta, D., 1991, Cognitive fit: An empirical study of information acquisition, *Information Systems Research*, **2**(1):63–84.

Vetschera, R., 1989, A preference-preserving projection technique for MCDM, *Diskussionsbeiträge Serie I – 256,* Fakultät für Wirtschaftswissenschaften und Statistik, Konstanz, Germany.

Vilkko, N., and Lautala, P., 1990, Short-term electric power production scheduling using simulated annealing algorithm, in *Control ’90, Proceedings of the IASTED International Symposium,* ACTA Press, Anaheim, CA, USA.

Vincke, Ph., 1992, *Multicriteria Decision Aid,* John Wiley & Sons Ltd, Chichester, UK.

Vogt, W.G., and Mickle, M.H. (eds), 1983, Modeling and Simulation, Vol. 14, in *Proceedings of the Fourteenth Annual Pittsburgh Conference,* University of Pittsburgh, Pittsburgh, PA, USA.

Vollmer, G., 1984, Mesocosm and objective knowledge, in F.M. Wuketits (ed.), *Concepts and Approaches in Evolutionary Epistemology,* D. Reidel Publishing Co., Dordrecht, Netherlands.

von Neumann, J., and Morgenstern, O., 1944, *Theory of Games and Economic Behavior,* Princeton University Press, Princeton, NJ, USA.

Wall, K.D., 1993, A model of decision making under bounded rationality, *The Journal of Economic Behavior and Organization,* **20**(3):331–352.

WEC (World Energy Council), 1992, *Energy for Tomorrow's World: The Realities, the Real Options and the Agenda for Achievements,* Kogan Page, London, UK.

Wessels, J., and Wierzbicki, A.P. (eds), 1993, *User-Oriented Methodology and Techniques of Decision Analysis and Support*, Vol. 397 of Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, Berlin, Germany.

Wiener Stadtwerke, 1991, *Neue Wege für Wiens Energie: Energiekonzept der Stadt Wien, 2. Fortschreibung,* Bohmann Druck und Verlag, Vienna, Austria.

Wierzbicki, A.P., 1971, A penalty function shifting method in constrained static optimization and its convergence properties, English paper in a Polish journal: *Archiwum Automatyki i Telemechaniki*, **16**(4).

Wierzbicki, A.P., 1980a, Multiobjective Trajectory Optimization and Model Semiregularization, WP-80-181, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Wierzbicki, A.P., 1980b, The use of reference objectives in multiobjective optimization, in G. Fandel and T. Gal (eds), *Multiple Criteria Decision Making: Theory and Applications*, Vol. 177 of Lecture Notes in Economic and Mathematical Systems, Springer-Verlag, Berlin, Germany.

Wierzbicki, A.P., 1982a, A mathematical basis for satisficing decision making, *Mathematical Modelling,* **3**(5):391–405.

Wierzbicki, A.P., 1982b, Note on the equivalence of Kuhn-Tucker complementarity conditions to an equation, *Journal of Optimization Theory and Applications,* **37**(3).

Wierzbicki, A.P., 1984, *Models and Sensitivity of Control Systems*, Elsevier-WNT, Amsterdam, Netherlands.

Wierzbicki, A.P., 1986, On the completeness and constructiveness of parametric characterizations to vector optimization problems, *OR-Spektrum,* **8**:73–87.

Wierzbicki, A.P., 1988a, Education for the cultural era of informed reason, in J.G. Richardson (ed.), *Windows on Creativity and Invention*, Lomond Publications, New York, NY, USA.

Wierzbicki, A.P., 1988b, Dynamic Aspects of Multi-Objective Optimization, in A. Lewandowski and V. Volkovich (eds), *Multiobjective Problems of Mathematical Programming,* Vol. 351 of Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, Berlin, Germany.

Wierzbicki, A.P., 1992a, The Role of Intuition and Creativity in Decision Making, WP-92-078, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Wierzbicki, A.P., 1992b, Multi-objective Modeling and Simulation for Decision Support, WP-92-80, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Wierzbicki, A.P., 1992c, Multiple Criteria Games: Theory and Applications, WP-92-079, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Wierzbicki, A.P., 1997a, Convergence of Interactive Procedures of Multi-Objective Optimization and Decision Support, in M.H. Karwan, J. Spronk, and J. Wallenius (eds), *Essays in Decision Making*, Springer-Verlag, Berlin, Germany.

Wierzbicki, A.P., 1997b, On the role of intuition in decision making and some ways of multicriteria aid of intuition, *Journal of Multi-Criteria Decision Analysis*, **6**:65–76.

Wierzbicki, A.P., and Makowski, M., 1992, Multi-objective Optimization in Negotiation Support, WP-92-07, International Institute for Applied Systems Analysis, Laxenburg, Austria. Available on-line from http://www.iiasa.ac.at/∼marek/pubs.

Wigley, T.M.L., Salmon, M., and Raper, S.C.B., 1994, Model for the Assessment of Greenhouse-gas Induced Climate Change, Version 1.2, Climate Research Unit, University of East Anglia, Norwich, UK.

Williams, H.P., 1990, *Model Building in Mathematical Programming*, John Wiley & Sons Ltd, Chichester, UK.

Williams, H.P., 1991, *Model Building in Mathematical Programming (Third edition)*, John Wiley & Sons Inc., New York, NY, USA.

Williams, H.P., 1993, *Model Solving in Mathematical Programming*, John Wiley & Sons Ltd, Chichester, UK.

Wischmeier, W., and Smith, D., 1978, Predicting Rainfall Erosion Losses, Handbook 536, United States Department of Agriculture, Washington, DC, USA.

Wittgenstein, L., 1922, *Tractatus Logico-Philosophicus*, Harcourt Brace, New York, NY, USA.

Wolfram, S., 1996, *The Mathematica Book*, 3rd edition, Mathematica Version 3, Cambridge University Press, Cambridge, UK.

Wright, G., Worobetz, N., Kang, M., Mookerjee, R., and Chandrasekharan, R., 1997, OR/SM: A prototype integrated modeling environment based on structured modeling, *INFORMS Journal on Computing*, **9**(2):134–153.

Wright, G., Chaturvedi, A., Mookerjee, R., and Garrod, S., 1998, Intergated modeling environments in organizations: An empirical study, *Information Systems Research*, **9**(1):64–84.

Wuketits, F.M., 1984, Evolutionary epistemology: A challenge to science and philosophy, in F.M. Wuketits (ed.), *Concepts and Approaches in Evolutionary Epistemology*, D. Reidel Publishing Co., Dordrecht, Netherlands.

Ybema, J.R., Lako, P., Gielen, D.J., Oosterheert, R.J., and Kram, T., 1995, Prospects for Energy Technology in the Netherlands, Netherlands Energy Research Foundation ECN-C-95-002, Petten, Netherlands.

Young, L.F., 1983, Computer support for creative decision-making: Right-brained DSS, in H.G. Sol (ed.), *Processes and Tools for Decision Support*, North-Holland, Amsterdam, Netherlands.

Yu, P.L., 1985, *Multiple-Criteria Decision Making: Concepts, Techniques, and Extensions*, Plenum Press, New York, NY, USA.

Yu, P.L., 1990, *Forming Winning Strategies: An Integrated Theory of Habitual Domains*, Springer-Verlag, Berlin, Germany.

Yu, P.L., 1995, *Habitual Domains: Freeing Yourself from the Limits on Your Life,* Highwater Editions, Shawnee Mission, KS, USA.

Zadeh, L.A., 1978, Fuzzy sets as a basis for a theory of possibility, *Fuzzy Sets and Systems*, **1**:3–28.

Zakrzewski, R., 1989, Analysis of Linear Mathematical Models with the Help of MENTAT-L System, MSc Thesis, Institute of Automatic Control, Warsaw University of Technology, Warsaw, Poland [in Polish].

Zawicki, P., 1995, Software Tools for Generation, Simulation and Optimization of the Simplified Ozone Model, WP-95-107, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Zeleny, M., 1974, A concept of compromise solutions and the method of the displaced ideal, *Computers and Operations Research*, **1**:479–496.

Zimmermann, H.J., 1987, *Fuzzy Sets, Decision Making, and Expert Systems,* Kluwer Academic Publishers, Dordrecht, Netherlands.

Zimmermann, H.J., 1997, A fresh perspective on uncertainty modeling: Uncertainty versus uncertainty modeling, in B. Ayyub and M. Gupta (eds), *Uncertainty Analysis in Engineering and Sciences: Fuzzy Logic, Statistics and Neural Networks Approach*, Kluwer Academic Publishers, Dordrecht, Netherlands, pp. 353–364.

Zimmermann, H.J., and Sebastian, H.-J., 1994, Fuzzy Design – Integration of Fuzzy Theory with Knowledge-Based System Design, in *Fuzzy-IEEE 94 Proceedings*, Pergamon Press, Orlando, FL, USA.

Zionts, S., and Wallenius, J., 1976, An interactive programming method for solving the multicriteria problem, *Management Science*, **22**:653–663.

Zionts, S., and Wallenius, J., 1983, An interactive multiple objective linear programming method for a class of underlying nonlinear utility functions, *Management Science*, **29**:519–529.

Zorychta, K., and Ogryczak, W., 1981, *Linear and Integer Programming* WNT, Warsaw, Poland [in Polish].

Subject Index

Author Index