Global Optimization at Work

Promotoren: dr. P. van Beek
Hoogleraar in de Operationele Analyse

ir. A.J.M. Beulens
Hoogleraar in de Toegepaste Informatiekunde

[illegible]

Global Optimization at Work

Eligius Maria Theodorus Hendrix

Proefschrift
ter verkrijging van de graad van doctor
op gezag van de rector magnificus
van de Landbouwuniversiteit Wageningen,
dr. C.M. Karssen,
in het openbaar te verdedigen
op dinsdag 23 juni 1998
des namiddags te 13.30 uur in de Aula.

[illegible]

ISBN 90-5485-874-5
©1998, E.M.T. Hendrix
Cover: Paulien C. Wijnker

NN08201, 2462

Stellingen

1. Als wiskunde een wetenschap is die niet de waarneembare werkelijkheid betreft, dan is discussie over wiskundige resultaten overbodig.

2. De filosofie of de wetenschap in het algemeen heeft niet tot taak de wereld te veranderen, maar haar te analyseren.
(*Antithese op Marx*)

3. Een poffertjespan is niet alleen een culinair instrument, maar ook een handig hulpmiddel om te demonstreren wat Globale Optimalisering is. Het ontbreken ervan in de Angelsaksische cultuur is een tekortkoming.

4. Het is onwaarschijnlijk dat stochastische globale optimaliseringsmethoden zullen worden bedacht, waarvan de verwachte rekentijd polynomiaal toeneemt in het aantal variabelen van het op te lossen probleem.
(*Dit proefschrift*)

5. Bij de stochastische globale optimaliseringsmethoden bestaat er een karakteristieke functie die voldoende informatie bevat om optimaal te kiezen tussen globaal en lokaal zoeken.
(*Dit proefschrift*)

6. Zoeken naar waarheid is een black-box globaal optimaliseringsprobleem.
(*Karl Popper* $\cap$ *Global Optimization*)

7. Gebruik van de uitdrukking "we analyse an algorithm" in plaats van "we propose an algorithm" in wetenschappelijke artikelen over mathematische besliskunde bevordert objectieve wetenschapsbeoefening.

8. Socialisme is een overwinning op de menselijke natuur.

9. De rijkdom van de westerse wereld is gebaseerd op mondiale uitbuiting.

10. Het feit dat de cycloop uit de Odyssee de woorden οὐδεις (niemand) en 'Οδυσσευς (Odysseus) verwart, kan er op duiden dat cyclopen hardhorend zijn.
(*Odyssee, Homerus*)

11. Preferentie van carnaval boven 11-steden door Brabanders is een voorbeeld van risicomijdend gedrag.

E.M.T. Hendrix
Global Optimization at Work
Wageningen, 23 juni 1998

Preface

The study on Global Optimization which finally resulted in this book, started in 1989 and was inspired by the work of Janos Pintér on methods and by Peter Janssen who showed me the relevance for environmental models. Many people stimulated and encouraged me since. Researchers from companies presented their interesting practical problems; among others Quaker Chemical, Unilever Research, Stork Veco, Philips CFT and ABC fodders contributed in pleasant cooperation. Many people from research institutes did not get tired of my stories and provided me with more practical puzzles. Among these were RIZA, PR, IBN-DLO, (former) GLW, RIVM, IMAG-DLO and ATO-DLO. Many researchers within Wageningen Agricultural University delivered interesting optimization problems. With Sipko Mous, Huub Scholten, Olivier Klepper and Jacqueline Bloemhof I worked on common papers.

Students contributed; I kept them busy with Global Optimization problems and they kept me thinking by posing new questions. Carmen Mecking and Henriette de Blank contributed to papers. Jaap Roosma did most of the programming work and was a stimulating friend. Colleagues at other universities, Delft and Groningen were backup for difficult mathematical questions. I am grateful to people from the Global Optimization community and Walter Stortelder with whom I discussed the results and who read parts of the work. I thank the crew of the Department of Mathematics in Trier for their hospitality.

The (former) Department of Mathematics in Wageningen provided me with a pleasant working environment. My nearest neighbours lend me their ear and their blackboard to have it covered with ellipses and spheres. The ladies at the secretariat translated pencil written notes into readable formulae and Frits Claassen took over my work when I was consuming time on the book.

In the last years, Paul van Beek, Edwin Romeijn, Adrie Beulens and Theo Hendriks took the role of supervisor. During the last months, Bill Baritompa, Guus Boender and Gerard Bot went through the complete final text. Paulien C. Wijnker designed the cover.

Finally I thank my family, friends and relatives for their patience (also during the running exercises). They had to cope with my local optima, both up and down.

Contents

Chapter 1. *Position of this study*

1.1. *Introduction*

In this work we study the position of global optimization (GLOP) methods as a Decision Support tool for complex practical problems. Global optimization methods are algorithms with the purpose to find the global optimum of a real valued continuous function over a feasible set, in situations where there exist several so-called local (not global) optima. Until recently most literature on global optimization focused on theoretical properties of problems and methods. The question in this book is, what global optimization can offer to a group of potential users. At one side there exists literature on global optimization, which mainly focuses on theoretical achievements of the methods, see Handbook on Global Optimization (Horst and Pardalos, 1995). At the other side there are potential users. This book, GLOP at Work, should help the potential user in the direction of actually applying Global Optimization methods. The target group of this study uses mathematical modelling for research or practical decision problems, though it does not consist of experts in optimization. In this study, stimulated by experience at the Agricultural University, cases were used of the following not mutually exclusive categories of modellers (and their typical models) and potential users of global optimization methods. The categories are elaborated in Section 1.2:
- Researchers in agricultural and environmental studies
- Designers using mathematical models to describe their designs
- OR decision scientists of environmental and agricultural planning problems.

The categories do not cover the group of all potential users, but have in common that all the users apply mathematical modelling and optimization to get a better understanding of a practical problem, an object system.

The main question is:

Given a potential user with an arbitrary global optimization problem, which route can be taken in the GLOP forest to find solutions of the problem?

This book intends to bridge the gap between the potential users and literature on the theoretical achievements of global optimization algorithms. The distance between a modeller and the literature on global optimization is, in general, large. The purpose of this book is to be helpful in looking for solution methods when one tries to solve practical global optimization problems.

A side question is:

How can the user influence the search process of solution methods given the knowledge of the user of the underlying problem and which information becoming available during the search is useful for steering the search process?

From these first questions we will follow the idea on science according to the great 20th century philosopher Karl Popper (1902-1994); "we start with a problem". By further analyzing one problem, we get better insight and arrive at the next problem. New challenging questions are found when we proceed along the path of finding appropriate methods for potential users.

Figure 1.1 outlines the object of study; the interaction between modeller and global optimization algorithmic toolbox. The arrows in Figure1.1 do not intend to represent consecutive steps which are undertaken, but distinguish the information streams which may occur when solving an optimization problem.

Arrow a. Central element in Figure 1.1 is the modeller. He has *formulated* a mathematical model of a part of the observable world of interests to him, his object system. The scope of our study is indicated by the dotted box in Figure 1.1. The art of mathematical modelling is a very interesting subject which is mainly left outside the scope of our study. Let us only remark that in general a mathematical model is considered 'good', when it describes the image of the object system in the head of the model well, it 'fits reality'. Furthermore, it is common in operations research that a modeller has already particular solution methods in mind when formulating the model, so that mathematical structures are put into the model. We abstract from this effect and assume the model to be given and predefined by the modeller. As in the general loop in the methodology of OR (see e.g. Winston, 1994), the modeller may want to revise his model after having studied the optimization problem. We will restrict ourselves to the process of looking for solution methods for a given so called derived optimization problem only.

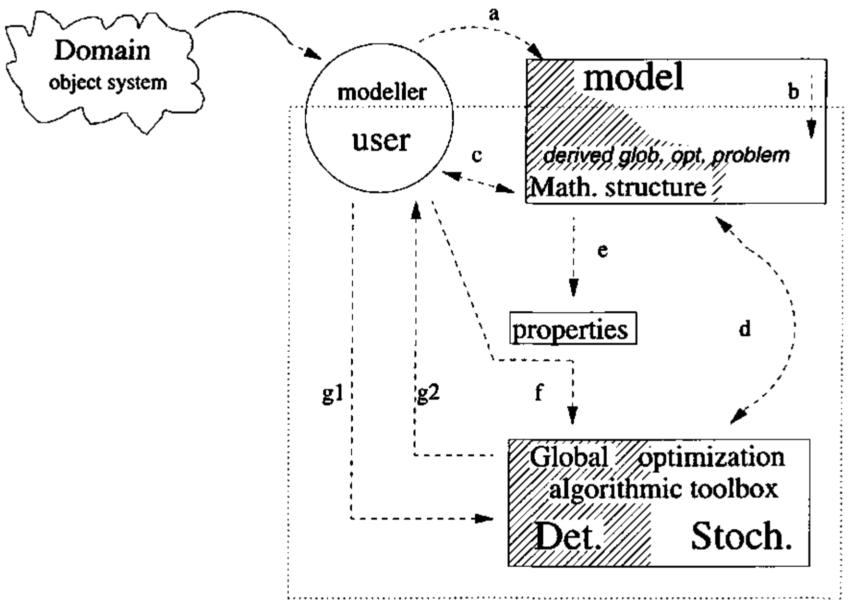

Figure 1.1: View on elements in this study

Arrow b. represents the idea that the model which is primarily constructed to describe and analyze reality is considered as an optimization problem. In Section 1.3 this topic is discussed in more detail. Depending on the derived optimization problem, the modeller has become a potential *user* of global optimization methods. The generic global optimization problem formulation which we consider is:

$$\min f(x),\ x \in X \subset \mathbb{R}^n, \tag{1.1}$$

in which $f(x)$ is a real valued continuous function and x varies in a continuous way in a feasible set X. One can call the problem a general Nonlinear Programming formulation. However, methods in nonlinear optimization in general aim at finding a (local) optimum giving a starting point. Global optimization aims at finding the global optimum in a so called multi-extremal problem: there exist several local (not global) optima. The relation with problems where x takes integer values, integer programming, is discussed in Chapter 3 and may help the reader to get a better understanding of global optimization.

The next step is to look for solution procedures for the derived optimization problem. Ignorant of the existence of multiple optima in general one tries to generate (local) solutions with nonlinear programming software. Often the occurrence of various optima, discovered when trying various starting points, is considered a mistake.

Arrow c. When several optima are discovered, the *recognition* of certain mathematical structures *explains* the occurrence of multiple optima. This is illustrated in Chapters 2 and 4 by several cases. The study on the mathematical structures in the optimization problem (possibly inherited from the model) may also take place before looking for an appropriate solution method. When general purpose nonlinear programming software is easily available, it is a good idea to start generating optima in order to scan the feasible space and to get a better insight in the behaviour of the model.

Arrow d. represents the interaction between the optimization problem and the algorithms. First of all one should get the optimization problem in the format the implemented algorithm requires, or the other way around, implement an algorithm which fits the way the model has been implemented. All algorithms require the *evaluation of the objective function* and possibly constraint functions. The number of function evaluations is a common performance indicator to assess the efficiency of algorithms.

Arrow e. The traditional literature on global optimization in general focuses on arrow e, the derivation of properties given assumptions about the mathematical problem structure. Therefore terminology referring to the structure occurs such as bilinear programming, concave programming, Lipschitz optimization etc. (Chapter 2). The results in the literature typically consist of theorems and experiments with test problems. To quote the mathematician P. Erdös, "A mathematician is a device for turning coffee into theorems". Specific algorithms are derived for the toolbox and are studied further given assumptions on the problems, which leads to mathematical properties on the behaviour of the algorithms.

Arrow f. Given the recognized mathematical structure, the user may go into the analysis of the problem and the properties, study literature, and *exploit* the structure to derive or *select* specific optimization algorithms to solve his problem. The choice for algorithms does not only depend on the most refined exploitation of the structure, but is often driven by the availability of methods in the environment of the user. The appearance of electronic means such as e-mail, public domain software and the world wide web has enlarged the direct environment of a researcher, leading to a larger availability of algorithms. See Pintér (1996b) for an overview of existing implementations of methods in global optimization software.

In our work a global classification of methods is used, elaborated in Section 1.5 and sketched in Figure 1.1. In this view, the methods which require a certain mathematical structure are called deterministic and the other methods which are often based on random generation of feasible points and nonlinear local optimization routines are called stochastic methods. This global classification is used to divide this work into two parts. One part (Chapters 2, 3 and 4) mainly discusses how information on the mathematical structures can be used by deterministic methods and the second part (Chapters 5 and 6) illustrates optimization problems which typically only can be handled by stochastic methods and discusses particular specific problems for this class. The concluding Chapter 7 provides guidelines for the target user groups, derived from the results and illustrations in the remainder of the book.

Arrows g. The arrows g1 and g2 represent the information between user and global optimization method during the execution of a search algorithm. On one hand the user, when coinciding with the modeller, may have knowledge (g1) about promising areas to find optima, the number of optima, bounds on function values and on decision variables. This information can be used to speed up the search process. On the other hand, the algorithms generate information (g2) about function values, local optima of the optimization problem and on the success of the algorithm itself. This information is useful to interact in the algorithm e.g. by changing bounds or changing parameters of the selected (by arrow f) algorithm. The information in the arrows g is typically value information, i.e. it depends on the instance. In the discussion of cases and examples within this book, it will be studied which information is useful to speed up the algorithms.

In Chapter 2 the mathematical structures, recognised in literature to be useful, are given and examples show, how these structures may be recognized (arrow c) and explain the multi-extremality of the corresponding optimization problems.

In Chapter 3 the relation between traditional global optimization (continuous variables) and integer programming is discussed. It is not only interesting to see how problem formulations (arrow b) can be translated into the other class, but also the solution approaches and the way of analyzing properties (arrow e) is similar.

Chapter 4 is devoted to three larger cases where we elaborate the whole route of problem formulation, deriving properties and constructing specific algorithms (arrows b, c, e and f). The cases serve as examples for potential users attempting to solve similar problems. Moreover, some simple examples are given as in introduction to the branch-and-bound concept.

Chapter 5 starts with practical examples of models for which the mathematical structure of the corresponding optimization problem can be considered hidden. Each function evaluation implies running a larger model using numerical integration or Monte Carlo simulation. In the discussion how to solve these problems using stochastic approaches, focus is on the information which is useful to speed up the search process (arrows g1 and g2). Most literature on stochastic methods aims at the globality property; in the long run (in limit) the optimum is reached. In a practical situation the user requires answers in a finite period. Therefore, the concept of rules to control the search in a finite amount of time is discussed in Chapter 5.

A separate Chapter 6 is devoted to a specific subclass of parameter estimation problems and possible useful information generated by the stochastic methods for this class.

The remainder of Chapter 1 is used to elaborate on various elements of Figure 1.1. In Section 1.2 focus is on the target user groups. In Section 1.3 we elaborate a general view on the use of modelling and optimization by the user groups. In Section 1.4 the question on the interaction between users and global optimization methods is studied. In Section 1.5 the global division of the methods in the two classes is discussed.

1.2. *Target user groups*

The group of potential users of global optimization algorithms under study here, is characterised by persons applying mathematical models for research. The cases in this book are based on experience of modellers from the following three typical groups. The way of approaching optimization problems described in this book is of course also applicable in other scientific directions.

A. Researcher in agricultural and environmental sciences
In general, the researcher has knowledge of an object system such as vegetation growth, pig growth, river pollution, microbiological contamination of food etc. To get a better understanding of the object system, mathematical models from chemistry, biology and physics are used. These models range from statistical models to complex models based on a set of differential equations. Tools from statistics, continuous simulation and optimization are available in packages and libraries to analyze a model. Typically, a model run implies numerical integration and examination of several output variables. Optimization in this context can be used to discover extremes of the model, to estimate parameters and to find out experimental designs for measurements. The researcher is in general not an expert in optimization. In this book some typical problems which may be crossed with respect to optimization by this group are mentioned and some remedies are given.

B. Designers using mathematical models
In design problems there are design parameters which can be controlled on the one hand and properties, criteria describing the quality of a design on the other.
Mathematical models are used to describe the relations between those two elements. Again, models can consist of simple physical equations from literature or based on simple regression. They can also be more complex based on differential equations to describe, for example, the resistance of a ship. The models are applied to what we will call the *evaluation step*; given a design, the properties are calculated. Finding the 'best' design often leads to a multicriteria problem; the score on all properties of the design should be high. Nonlinear programming in general and global optimization in particular may be applied to find optima of multicriteria functions.

C. OR practitioner in environmental and agricultural planning
The OR practitioner studies management problems; calculation models are developed to support decisions in for instance agrologistics, reservoir management and farm management. Numerous publications exist on the path from practical problems to building models (Williams, 1990 and Bloemhof et al., 1995). We will not go into detail on the modelling itself. Optimization is very popular for this group and linear

programming and combinatorial optimization have been applied successfully leading to further development of decision support systems. Global optimization has not yet reached this level. A separate chapter (Chapter 3) is devoted to the relation between the common techniques of linear and combinatorial optimization on the one hand and nonlinear and global optimization on the other.

A major question in this study is how and where the user may discover a global optimization problem and what the global optimization toolbox has to offer. This study is more directed towards the mathematical structure of the models which leads to choices of global optimization methods than towards the target user groups. We believe that the typical model structure does not depend so much on the target user group, as will become clear from examples and cases throughout this study. In Section 1.3 a view on the use of optimization by the modellers of the various target user groups is presented.

The modeller uses his domain knowledge to derive by abstraction a mathematical model of an object system. The mapping from object to model is not unique.

- The modeller can choose between several types of models.
- Implicitly a choice is made of that part of the domain knowledge which is taken into the model.

In the next stage a modeller will use simulation and sensitivity analysis, or possibly optimization to confront outcomes of the model with the domain knowledge in his head, his view on reality, thus getting a better understanding of the object system. We will not go into detail on the art of mathematical modelling in general. Let us only remark that in general a mathematical model is considered 'good', when it describes the image of the object system in the head of the model well, it 'fits reality' related to the problem at hand.

The central issue is to have a closer look at possibilities of global optimization in this context. A view is presented on several elements which always can be found in models, independent of their complexity or form. The derivation of an optimization problem (arrow b in Figure 1.1) is discussed.

- The first element is the **structure** of the model which has been selected and developed by the modeller. As mentioned before, the model can be more or less complex and often linearity is an important aspect.
- Input of the model consists of **technical parameters** such as the Bernoulli constant, ability of the soil to contain water, and **data** such as rainfall, sales, location of potential depots. The data may be varying, may have a stochastic nature.
- Via calculation the model, possibly with the aid of some internal variables, arrives at values for the **output variables** z such as predicted discharge of a river, properties of a product, total amount of labour needed.

A first step in the direction of an optimization problem appears (arrow b) when the modeller decides which inputs of the model are seen as controllable, **decision variables** x. That are the variables he wants to vary such as parameters to be estimated, design parameters, number of hectares etc, in order to study the behaviour of the model given by the values of the output variables.

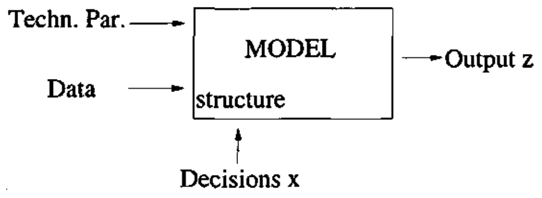

Figure 1.2: View on elements in a mathematical model

The output of the model is confronted with targets such as data from measurements on the output variables, requirements on the properties, amount of labour available etc. This confrontation can be formalised by building a criterion, objective function or formulating restrictions in which the modeller states what is 'good' and what is 'bad'.

We assume an objective function f to be minimized has been derived from the model leading to an optimization problem in which the decision variables can be varied in a continuous way. Optimization is nothing more than varying the decision variables such that the desired values of the output variables formalised by a criterion function and constraints are finally reached. This idea applies for the parameter estimation problem, the design problem as well as for the management problem as far as mathematical models are used.

The researcher (group A) modelling the discharge (output) of a river uses data on rainfall and wind velocity. For calibration purposes he may want to find good values for the resistance parameters of the bottom, such that the predicted discharge fits well measured values.
A designer (group B) of a ship is confronted with technical specifications of the ship such as the volume of various compartments. Numerous technical parameters describe the strength of various parts of the ship. Outcomes may be the stability and resistance of the ship, which the designer wants to optimize.
The decision scientist (group C) in charge of finding a good plan for a location allocation problem is confronted with data on the location of customers and their demand for facilities from a distribution centre and the possible locations of these centra. A technical number may be the costs of one kilometre of transportation. The outcome (z) consists of the costs and number of centra implied by the location-allocation plan x.

A derived optimization problem asks for optimization algorithms which can be found in mathematical programming and optimization literature. The choice of the optimization algorithm depends on the structure of the problem, as far as this is known. When there is a linear input-output relation between z and x the logical choice is to apply linear programming, which has been done since the first applications in 1947. When the choice given by the decision variables has to be made from discrete alternatives, combinatorial optimization will be applied. In this work focus is on nonlinear programming, i.e. the x variables can be varied continuously between lower and upper bounds and the output can be seen as possibly nonlinear functions, $z_i=g_i(x)$, on the decision variables. Some of the functions appear in restriction form, some are combined in an objective function f. In many situations no analytical expression is available for the nonlinear functions g_i, as the result z_i of the model calculation is derived by numerical integration or even Monte Carlo simulation (Chapter 5).

1.4. *Interaction between users and GLOP methods*

The user follows a path from object system via modelling to methods for analysis of the mathematical model. In this context we view GLOP methods as search methods rather than solution methods. GLOP methods try to find the global optimum of the mathematical optimization problem in an acceptable amount of calculation time. General nonlinear programming techniques search for a (local) optimum given a starting point. The search process done by the GLOP methods returns much more information which can be used to gain a better understanding of the practical problem. Moreover, this information might be used by the modeller, to interfere in the search process interactively. At the other side, the user has a better understanding of the practical problem, object system, and mathematical model which can be used to influence, improve and speed up the search process done by the GLOP method. This information from his domain knowledge can consist of

g1) value information, e.g. promising areas of the feasible set, bounds on the function values. In optimization literature this is called **instance dependent information.**

c) information on the special mathematical structure of the model, e.g. the mathematical model is a concave quadratic programming problem (Chapter 2). This is called **structure dependent information.**

The information generated by running a GLOP method (g2) is typically value information. This can consist of graphical information and numerical indicators which are presented in Chapter 5. The user can combine both types of information to get a better understanding of the practical problem and to influence the search process by

- selection of the GLOP method used,
- interfering interactively during running a method, for instance change or set bounds on criterion value or decision variables.

The question is, which information is useful for which intersection of the set of mathematical problems versus the set of GLOP methods. We do not intend to give a complete elaboration of this question. Instead ideas and views will be derived from practical examples on this topic. Nevertheless following the literature on global optimization, a global division is made between methods based on deterministic methods, which in general require a lot of information on the structure of a model, and the group of methods which is based on random search and local (nonlinear) optimization, which requires much less information from the user for the search process.

1.5. *Global Classification of methods and problems*

In this Section the global division which is used throughout this study is discussed. Starting point for the optimization problem is the model developed by the user and his further domain knowledge. In general the user might not be aware of the multi-extremal character of the derived optimization problem. For finding optima, nonlinear programming algorithms, implemented in routines such as GAMS/MINOS, GRG and Lancelot for problems with constraints are used. For an overview on optimization software see e.g. Moré and Wright (1993). Often routines from libraries, numerical recipes, but also solvers provided in spreadsheet programs are used when only bounds on the variables exist in the model. The appearance of standard solvers in spreadsheet programs makes nonlinear programming very easily accessible for modellers. Such routines can be called local optimization routines; given a starting point a local optimum is returned. The analysis to find out whether there are multiple optima is done in general by trying various starting points. In this way the user can discover that there are multiple optima. The methods were already divided into two classes:

- **Deterministic** methods require a certain mathematical structure,
- **Stochastic** methods are based on the random generation of feasible points and nonlinear local optimization routines and require no specific structure.

A more profound discussion on the classification of methods may be found in Törn and Žilinskas (1989). The question in this work is simply where someone with a practical problem should start looking given his information on the problem to be solved. The stochastic methods are considered more general purpose, they require no further information and will work whenever an appropriate implementation is available. The deterministic methods aim at a guarantee to find the global optimum in a finite number of steps. It is not only the question if they should be used, but also whether they can be used given the possibility to obtain the information on the structure of the problem by analysis (arrow c in Figure 1.1).

Information on the special structure gives that the model may potentially have multiple optima. Value information such as negative eigenvalues of the Hessean[1] indicate that there exists more than one optimum. In order to do this analysis the user needs insight in the structure or values of the model. At least the analytical expressions of object function and constraints should be available. This may not always be the case. At this point the global division in problems can be made.

I. Analytical expressions available. Special structure can be obtained.

II. Oracle structure: No analytical expressions available.

[1] Many people write Hessian, but in our opinion this does not honour the German mathematician Ludwig Otto Hesse (1811-1874) and it is not consequent (compare e.g. Boolean).

I. Analytical expressions available

Concerning models of type I, we typically think of practical problems such as high dimensional planning problems applied by OR decision scientists, with several restrictions of the same type. The resulting problems are similar to Linear Programming problems such as quadratic or bilinear programming problems. Besides those problems, we can think of lower dimensional technical design problems applied by a designer. In Chapters 2 and 4 several examples are given.

For such problems the user can continue treating the model with nonlinear programming codes, local searches, such as GAMS/MINOS using several starting values. When he adds more information on the special mathematical structure, global optimization methods are available (Horst and Tuy, 1990) based on branch-and-bound and approximation which contain a **guarantee** that the global optimum is delivered in a 'finite number of steps'. The special structures are discussed in Chapter 2 and contain, among others, the concave, fractional, quadratic, bilinear and Lipschitzian structure. Even the analytical expressions can be used directly by interval methods (Hansen (1992) or Kearfott (1997)) in a branch-and-bound framework. For the user as a practical problem solver however, 'finite number of steps' guarantee may imply thousands of days and gigabytes of computer memory; the guarantee **may not be reached for a given practical problem**. The user can put in specific value information such as bounds on the function value, on the Lipschitz constant or on the second derivative. Some methods even require those data. In Chapter 2 and 4 properties of applying such methods are given and illustrated.

The Branch-and-Bound approaches in global optimization are similar to approaches used in combinatorial optimization, which are in general much better known to our target groups. Therefore, in Chapter 3 relations between nonlinear and linear and between integer programming and global optimization are discussed from a theoretical as well as practical viewpoint.

Chapter 6 specifically deals with another type of practical problems. It concerns parameter estimation in nonlinear regression models applied by target user group A, the researcher in agricultural and environmental sciences. The least squares problems of those model calibration problems have a particular structure for which specific local optimization algorithms have been developed (e.g. the DUD method and Marquardt method), which are included into statistical packages such as SAS and SPSS and can be found in algorithmic libraries. For the application of global optimization the structure of the problems can in general not be used. When analytical expressions are available at least interval arithmetic methods can be applied (Csendes, 1988). The local optima are a source of information to the model builder. One optimum may correspond to a very good fit to the data for one group of output variables, whereas another optimum may represent a good fit of another group of output variables. Furthermore the size of so called level sets is directly related to the idea of confidence regions for estimated parameter values in a statistical sense. Those ideas are developed further in Chapter 6.

II. Oracle structure

In many practical parameter estimation problems, the calculation of the 'goodness of fit' criterion for the calibration of descriptive physical, chemical, biological, ecological models implies the solution of a set of differential equations. This means there is no analytical (closed form) expression of the criterion function to be minimized. To evaluate the function means giving parameter values to a subroutine and after some time (seconds or even minutes) receiving the function value. This is the meaning of type II of problems, the Oracle structure. Another type of practical problems with this structure originate from design problems, which imply the calculation of a set of differential equations at each function evaluation. It is not only numerical integration which blows up the calculation time of one model run. Often the simulation over time periods in a difference equation context requires computing time. Chapter 5 contains an example of a decision problem on water reservoir management, decomposed in such a way, that every function evaluation implies the calculation of the consequence of a decision rule over years given hydraulic data. Other practical examples are given in Chapter 5.

These problems of type II have in common that for the problems solved, in general the dimension of decision vector x is less than 10. The deterministic GLOP methods cannot be used for this type of problems, because the possibly useful special structure of the model is hidden. On the other hand, the user may have value information on, for example, promising areas and the roughness of the criterion function due to his understanding of the practical problem. This information and the information revealed by the search process can be of help to control the search process and choosing between methods based on local optimization and global searches based on random search techniques. In Chapter 5 and 6 properties of applying such methods to practical problems are given.

Resuming, the remaining contents of this work is built up as follows.
Chapter 2 outlines and illustrates mathematical structures which can be recognised and then used by deterministic global optimization algorithms.
Chapter 3 gives the theoretical and practical relation between integer programming and global optimization starting with a view on the relation between nonlinear and linear programming.
Chapter 4 elaborates several cases. It discusses and illustrates the information which can be used to choose or develop specific deterministic methods and shows the information returned by the methods.
Chapter 5 gives several small examples, cases which are appropriate to be treated by stochastic global optimization methods. Some specific problems from the viewpoint of the methods are discussed.
Chapter 6 focuses on the topic of parameter estimation from the viewpoint of stochastic methods.
Chapter 7 results in guidelines which are derived from the six previous chapters.

Chapter 2. *Model structures required by deterministic GLOP methods*

2.1. *Introduction*

Chapter 1 and Figure 2.1 show us that when we start looking for deterministic solution methods, roughly two sides can be distinguished. One side is the toolbox of algorithms and the other side the user with his derived optimization problem. The algorithms in the toolbox of deterministic methods demand a certain mathematical structure. At the other side, a certain structure in the model is given or has to be discovered (arrow c).

First, in Section 2.2 a review is given on the terminology about the information on mathematical structures which can be exploited by deterministic methods. In the following Sections the other side is highlighted by showing how examples derived from practical problems reveal their mathematical structure. In Chapter 4 some cases are elaborated which demonstrate how deterministic methods can be used to solve practical problems.

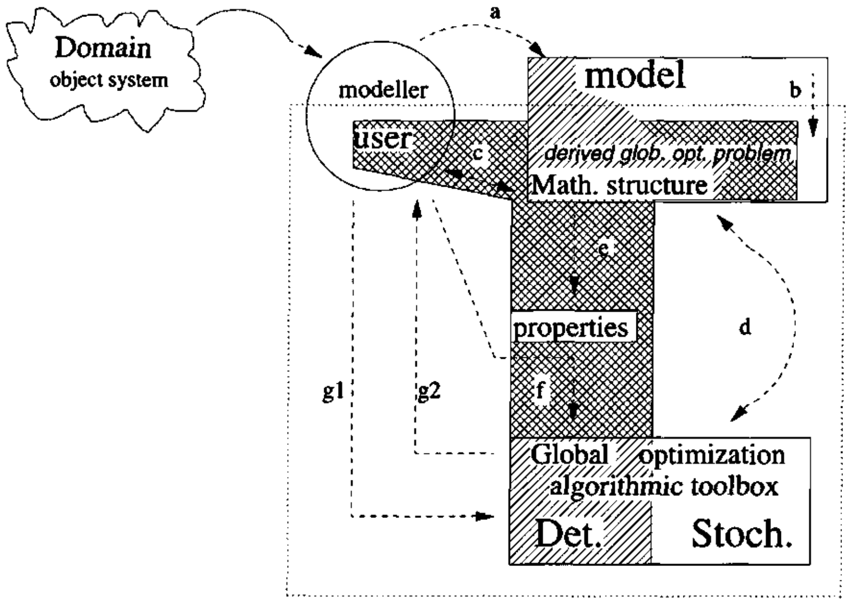

Figure 2.1: Focus on the mathematical structure and useful properties

At the toolbox side, from the early times of mathematical programming, the problem of finding the global optimum of a particular mathematical programming problem has been a challenge. Special structures (to be introduced in Section 2.2), such as quadratic, concave, bilinear structures, have been used for mathematical analysis and for the construction of specific algorithms. In 1990 the book 'Global

Optimization, deterministic approaches' (Horst and Tuy, 1990) appeared, which gives an overview with many references of the developments achieved. In 1991 the Journal of Global Optimization appeared, which gives the opportunity to follow further developments of specific algorithms. It goes too far to mention all literature which appeared in the field. Nevertheless, it is worthwhile to mention the appearance of another summary in 1995, 'the Handbook on Global Optimization' (ed. Horst and Pardalos, 1995), which includes more than deterministic approaches only. Further monographs are appearing in the series 'Nonconvex Optimization and its Applications' of Kluwer. All these works give the opportunity to follow the side of the theoretical achievements. In Section 2.2 we only give a brief summary of the main topics and further refer to these works.

A user may be attracted by the **guarantee** character of deteministic methods. How should the modeller with his derived practical optimization problem formulation start looking (**recognition**) for a useful mathematical structure? The multi-extremality of the problem is discovered and he is looking for approaches to get good solutions in reasonable calculation time. If he consults an expert or theoretic literature on global optimization we may arrive at the story of the hammer and the nail. When someone has a hammer (methodology) he is inclined to see nails everywhere. The problem for the owner (modeller) of a screw (specific optimization problem) however, is to start looking for a screwdriver, which is most appropriate to handle the screw. The modeller does not immediately start with the knowledge of the specific mathematical structure (screw in this analogy) of his problem when he is ignorant of the relevance of the difference between structures (screw and nail).

The eyes of a quadratic programming thinker will be inclined to see quadratic optimization problems everywhere. A specific optimization problem which can be written in quadratic form, often also can be formulated in other ways, such as a bilinear or fractional programming problem. The starting model may be formulated as a screw. By seeing it as a nail, a hammer can do a good job, but might not be as efficient as the driver. This means that in the process of selecting an algorithm and interacting during the execution of an algorithm, one can make more or less use of all the information which is in the mathematical optimization problem. Section 2.2 gives a global overview on the available technology, toolbox, and corresponding mathematical structure required by it.

We first give a summary of the terminology and most important properties of a mathematical structure referring for further information to the references given. In Sections 2.3, 2.4 and 2.5 examples follow of cases which show a structure which is useful for deterministic approaches. In Chapter 4 the use of those structures is elaborated. In Section 2.6 an excursion to a subtopic is made. In our experience we came across modellers who thought that their model was multi-extremal due to the structure of the model. It appeared that the existence of multiple optima was due to a symmetry in the translation from model to object system (arrow a in Figure 1.1); there is more than one solution of the model representing the same object. Some examples are given in Section 2.6 to help modellers to recognise this pitfall.

2.2. *Some mathematical structures in multi-extremal problems*

When the modeller starts analyzing the model and possibly finds multiple optima, typical mathematical structures might appear which explain the multi-extremality of the problem. A brief summary is given here of the terminology without extensive use of references and mathematical details. A complete mathematical description can be found in the handbook on Global Optimization (ed. Horst and Pardalos, 1995) and in Horst and Tuy (1990).

The problem to be solved by the modeller is assumed to consist of an objective function f to be minimized and possibly some (less than or equal) inequality constraints. The constraints and the objective function can be exchanged and/or transformed. This is not relevant here. The generic global optimization problem formulation which we consider is:

$$\min f(x),\ x \in X \subset \mathbb{R}^n, \tag{1.1}$$

in which $f(x)$ is a real valued continuous function and x varies in a continuous way in a feasible set X. In a practical situation the set X is often compact, i.e. bounded and closed. In the traditional literature on Global Optimization, the mathematical structures are enumerated and properties derived. We are especially interested in how the user may recognise (arrow c) the structure from his problem formulation. In order to recognise a useful structure the problem should be of type I., analytical expressions available. Problems of type II., Oracle structure, cannot be used to derive useful mathematical structures for deterministic GLOP methods. We furthermore make a distinction which cannot be found in literature on global optimization. For the user it is important how to recognise the structures. Therefore the structures are divided here into two groups. Given the explicit mathematical expression of the problem formulation, the groups are:

A: structure can only be derived by analysis of the mathematical expressions
B: structure is recognisable directly from the mathematical expressions.

A. Analysis necessary to reveal the mathematical structure
The following structures may be found, after an analysis of the mathematical expressions in the derived optimization problem, .

Concave functions
A popular expression in global optimization is *nonconvex optimization*. This refers directly to an important property of convexity:

- If f is a convex function and X is a convex set, there is only (at most) one local and global minimum.

The most common structure of multi-extremal problems is therefore nonconvexity. The other way around; minimizing a nonconvex objective function f, does not necessarily imply multi-extremality (the existence of multiple optima) but may explain the occurrence. One could call concavity an extreme form of nonconvexity. A property used by deterministic methods is the following.

- If f is a concave function and X is compact, the local minimum points coincide with extreme points of X.

Figure 2.2 illustrates this property for the concave function $f(x) = 4-x^2$ on the feasible set $X= [-1,2]$. The extreme points of the interval are the minimum points. When in general the feasible set is a polytope, then in a worst case situation, every vertex may correspond to a local minimum. Algorithms exist to perform an efficient so called *vertex enumeration*. The problem of minimizing a concave objective function on a closed convex feasible set is called *concave programming*. Figure 2.2 also shows the possibility to construct a so-called affine underestimating function $\varphi(x)$, based on the definition of concave functions. Given two iterates x_k and x_l, the function value for every convex combination of the iterates, $x = \lambda x_k + (1-\lambda)x_l$ is underestimated by

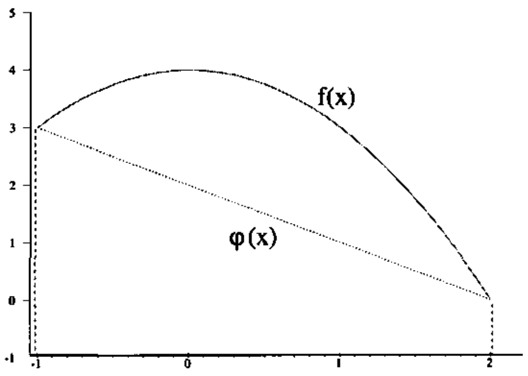

Figure 2.2: Concave function $f(x)$ and affine minorant $\varphi(x)$

$$f(x) = f(\lambda x_k + (1-\lambda)x_l) \geq \lambda f_k + (1-\lambda)f_l = \varphi(x),\ 0 \leq \lambda \leq 1. \qquad (2.1)$$

For the example of Figure 2.2 this can be derived as follows.
Let $x_k = 2$ and $x_l = -1$, so an arbitrary point x in the interval is a convex combination of the extreme points: $x = \lambda x_k + (1-\lambda)x_l = 2\lambda - (1-\lambda) \Rightarrow \lambda=(x+1)/3$.
Now the affine function $\varphi(x) = \lambda f(2) + (1-\lambda)f(-1) = 3(1-\lambda) = 2-x$ underestimates $f(x)$ on the interval $[-1,2]$.

The minorant $\varphi(x)$ can be used to derive **lower bounds** of the minimum objective function value on a bounded set. An example of an algorithm exploiting this concavity property is given in Section 4.4.

Concavity of the objective function from a given practical model formulation may be hard to identify. Concavity occurs for instance in situations of economies of scale. In cases where f is two times differentiable it can be checked whether the eigenvalues of the Hessean are all non-positive.

- If the eigenvalues of the Hessean of f are all non-positive on X, f is a concave function on X.

This property may be hard to check. The eigenvalues, representing the second derivatives, give a measure how concave the function is. Notice that the affine underestimator $\varphi(x)$ does not require the value information of the eigenvalues. This is a strong point of the structure. Notice furthermore that the underestimation becomes worse, less tight, when f is more concave, the second order derivatives are more negative. Concavity is a basis for many algorithms and other structures.

Differentiable convex functions (d.c.-functions)

When the function f can be written as the *difference* of two convex functions, $f(x) = f_1(x) - f_2(x)$, it can be called a d.c.-function. For many people the word 'differentiable' is confusing as it has nothing to do with differentiation of the function. Figure 2.3 shows the function $f(x) = 3/(6+4x) - x^2$, which has two minima on the interval [-1,1]. Many discussions of this structure in the literature on global optimization start with the statement "almost every function can be expressed as the difference of two convex functions". Splitting the function in a difference of two convex functions is called a *d.c.-decomposition.* For the example function a logical choice is to consider $f(x)$ as the difference of $f_1(x) = 3/(6+4x)$ and $f_2(x) = x^2$ (decomposition DC_1). The construction of a convex underestimating function of f proceeds as follows. The concave part, $-f_2(x)$, is underestimated by an affine underestimating function $\varphi_2(x)$ based on (2.1) and added to the convex part f_1. In this way a convex underestimating function $f_1 + \varphi_2$ appears, which can be used to derive lower bounds of the objective function on bounded sets. For the example function now $\varphi_2(x)$=-1 underestimates $-f_2(x) = -x^2$ resulting in the convex minorant of decomposition DC_1 in Figure 2.3. The decomposition is not unique. Let us imagine that the user did not recognise the obvious decomposition of the example function, but only discovered that it is not convex and the second derivative is bounded below by a value of -8. Now a decomposition can be constructed by adding a convex function with a second derivative of 8 and subtracting it again: $f_1(x) = f(x) + 4x^2$ and $f_2(x) = 4x^2$. The resulting convex minorant $f(x) + 4x^2$ -4 depicted in Figure 2.3 as minorant DC_2, is much worse, less tight than the first one.

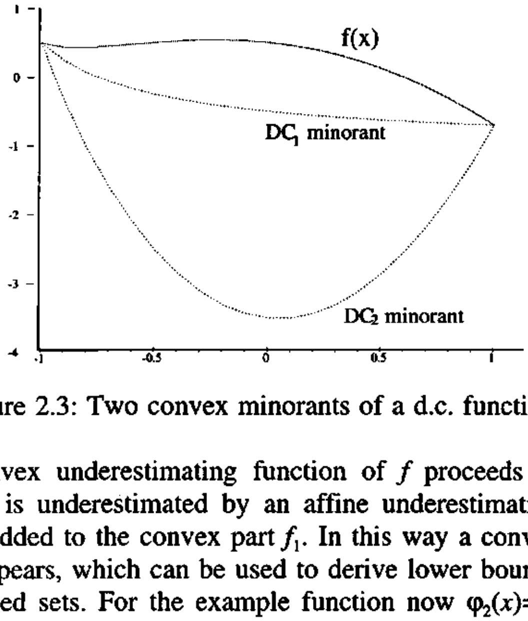

Figure 2.3: Two convex minorants of a d.c. function

This exercise teaches us several things. Indeed, nearly every function can be written as a d.c. function by adding and subtracting a strong convex function. The condition that the second derivative is bounded below is sufficient. For practical algorithmic development this statement is not useful. At first sight the minorant construction uses no value information. However, first a d.c.-decomposition has to be discovered. If the lower bound on the second derivative is used, value information is necessary.

Another related structure is the so called concept of *reverse convex programming*, minimizing a convex function on a convex set intersected by one reverse convex constraint i.e. the constraint defines the complement of a convex set. A d.c.-function on a convex set X can be transformed to a reverse convex structure by defining the problem:

$$\min z + f_1(x),\ z \geq -f_2(x),\ \ x \in X.$$

The dimension of the problem increases by one as one variable, z, is added. For the example of Figure 2.3 (and the first decomposition) the transformation leads to reverse convex program:

$$\min z + 3/(6+4x),\ z \geq -x^2,\ \ x \in [-1,1]$$

Both structures require the same type of solution approaches.
For further theoretical results on d.c.-programming consult the overview by Tuy (1995).

Lipschitz continuous functions
A function f is called a Lipschitz continuous function on X when the slope of the function is bounded. More formally, there exists a scalar L such that

$$|f(x_1) - f(x_2)| \leq L\ \|x_1 - x_2\| \quad \forall\ x_1, x_2 \in X. \tag{2.2}$$

We will discuss this structure in more detail as it is used in Chapter 4, to derive a specific algorithm.
The determination of the Lipschitz continuity of a given function f, in contrast to concavity, is not very hard. As long as discontinuities, or 'infinite derivatives' do not occur, e.g. when f is smooth on X, the function is also Lipschitz continuous.
The relation with derivatives (slopes) is given by

$$L \geq \frac{|f(x_1)\ -\ f(x_2)|}{\|x_1\ -\ x_2\|} \tag{2.3}$$

so that L can be estimated by the maximum of $\|\nabla f(x)\|$. The surveys on Lipschitz optimization, see Hansen and Jaumard (1995) and Pintér (1996a), stress the generality of the Lipschitz continuity assumption; it applies for nearly every practical problem.

The weak point is that in order to exploit the structure, value information is necessary; the value of the Lipschitz constant is required for algorithmic use. Notice that (2.2) also applies for any overestimate of the Lipschitz constant L. Finding such an overestimate is in general as difficult as the original optimization problem. In the test functions illustrating the performance of Lipschitz optimization algorithms, trigonometric functions are often used so that estimates of the Lipschitz constant can be derived easily. As will be shown, the structure provides a guarantee that the global optimum cannot be missed; it will be found in the end. We emphasise here that this guarantee does not apply when an overestimate of L is not available such as for Oracle functions (section 1.5), in contrast to what is suggested in Pintér (1996a).

Consider the case where one tries to find the global minimum f^* with a predefined accuracy ε. In this case one can construct a grid in X, such that for points x_1 and x_2 in the grid $\|x_1 - x_2\| \leq \varepsilon/L$. By evaluating all points in the grid a best point is found of which the function value deviates less than the accuracy ε from the optimum f^*. A large impact on the study on Lipschitz optimization is perhaps the formulation of an algorithm by Danilin and Piyavskii (1967) and (independently) Shubert (1972), also called the saw tooth cover algorithm. Although it was formulated to minimize univariate functions, it has stimulated multidimensional elaborations, as will be illustrated in Chapter 4. Given former iterations x_k, f_k, the core of the algorithm is based on the relation

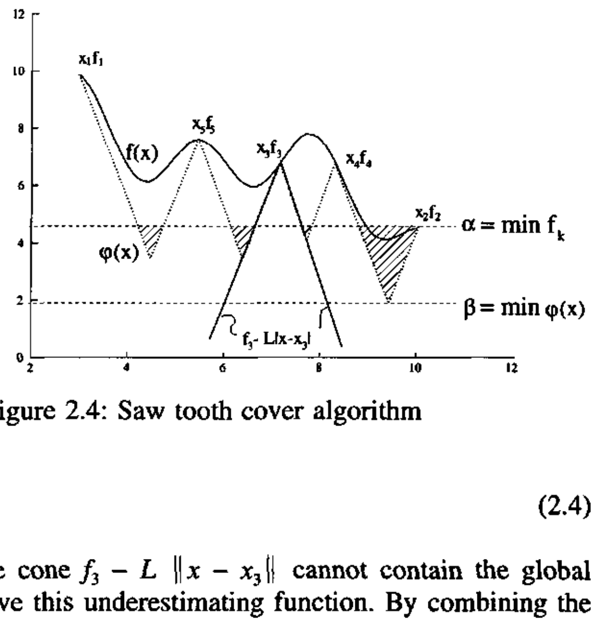

Figure 2.4: Saw tooth cover algorithm

$$f(x) \geq f_k - L\ \|x - x_k\|. \tag{2.4}$$

In Figure 2.4 the area under the cone $f_3 - L\ \|x - x_3\|$ cannot contain the global minimum, as the function is above this underestimating function. By combining the cones of all iterates x_k, f_k, intersecting the area above the cones, the saw tooth minorant $\varphi(x)$ appears (the dotted line in Figure 2.4),

$$\varphi(x) = \max_k \{f_k - L\ \|x - x_k\|\}. \tag{2.5}$$

The function $f(x)$ lies everywhere above the minorant $\varphi(x)$, so that the global minimum of $f(x)$ is above the minimum β of $\varphi(x)$. Moreover the global minimum lies below the best function value α which has been found. By taking as the new iterate a minimum point of $\varphi(x)$, the algorithm converges to the global minimum f^* which is enclosed by an upper bound $\alpha = \min_k f_k$ and a lower bound $\beta = \min_x \varphi(x)$.

The algorithm can be considered from a branch-and-bound point of view; the area of the epigraph of $\varphi(x)$ above level α can be cut away, as it cannot contain the global minimum. The area below the graph of $\varphi(x)$ has already been excluded due to (2.4). Only the shaded areas in Figure 2.4 can contain the global minimum. As the global optimum is not cut away, the available value of an overestimate of the Lipschitz constant gives the guarantee that (in the end) the global minimum is approximated.

Breiman and Cutler (1989) use the same type of underestimation for the case where a bound K on the second derivative is available; so $K \geq |f''(x)|\ x \in X$ or more general (in higher dimensions) overestimates the largest absolute eigenvalue of the Hessean. The analogy of (2.4) is given by

$$f(x) \geq f_k + f_k'(x - x_k) - \tfrac{1}{2} K(x - x_k)^2$$

for a function of one variable and in general

$$f(x) \geq f_k + \nabla f_k^T(x - x_k) - \tfrac{1}{2} K \, \|x - x_k\|^2 \ . \tag{2.6}$$

Similar to (2.4) the underestimating function $\varphi(x)$ can be taken as the maximum over k of the parabolas (2.6). The determination of lower bound β and upper bound α can be done similar to the Danilin-Piyavskii-Shubert algorithm. In a multidimensional situation the Breiman-Cutler algorithm has some advantageous geometric properties, which go too far to discuss here.

The determination of an upper bound K on the second derivative has the same drawback as finding an overestimate of the Lipschitz constant in a practical situation; it requires analysis to obtain an overestimate from the mathematical expressions in the problem formulation. Notice that larger overestimates of either L or K, lead to less tight lower bounds on the function to be minimized and therefore influences the efficiency of algorithms negatively.

It is worthwhile to discuss some geometric observations due to Baritompa and Cutler (1994) and Baritompa (1994). Consider the point of view that the area under the cones (figure 2.3) or parabolas of (2.6) can be thrown away as it cannot contain the global minimum. Baritompa shows that it is not necessary to have global overestimates of either Lipschitz constant L or second derivative K. If one would know the local behaviour around the global minimum point x^*, in general better information is available to cut away larger areas. If a value K^* would be available such that

$$f(x) \le f^* + \tfrac{1}{2}K^*\|x - x^*\|^2 \ , \ x \in X \tag{2.7}$$

(the gradient in x^* is the zero vector), then the area under the collection of parabolas

$$\varphi(x) = \max_k \ (f_k - \frac{1}{2} K^*\|x - x_k\|^2) \tag{2.8}$$

cannot contain the global minimum (see Baritompa, 1994). Notice that $\varphi(x)$ in (2.8) is not necessarily an underestimating function of f. A similar idea would be to run the Danilin-Piyavskii-Shubert algorithm not with the Lipschitz constant L defined by the maximum of (2.2) over $x_1, x_2 \in X$, but by using the parameter (not necessarily the Lipschitz constant)

$$L^* = \max_{x \in X} \frac{f(x) - f^*}{\|x - x^*\|} \le L. \tag{2.9}$$

The saw tooth cover with slope L^* of Figure 2.3 in this case is not necessarily an underestimating function everywhere, but also does not cut away the global minimum.

The saw-tooth-cover idea illustrated the elegance of deterministic methods; the global optimum is approximated in a finite number of steps. The practical consequence of the methods outlined here is that it is necessary to have value information on the slopes or second derivative either globally or locally around the global optimum (Baritompa). Concave programming does not require this value information as such, but may require more analysis to discover concavity or, in case of d.c.-programming, to construct a d.c.-decomposition. For the second group of mathematical structures mentioned in literature, it is much easier to obtain value information.

B. Structure recognisable from the mathematical expressions in the problem formulation

For the structures in group A discussed above, analysis of the mathematical expressions is required for the recognition. The mathematical structures in group B, given the optimization problem formulation, are not difficult to recognise so that the possible occurrence of multiple optima is easily explained. By reformulating the model one structure often can be transformed to another, indicating the interdependence of the structures. The structures of group A, as discussed above, can be recognised in specific structures of group B; group A is more general. This shows that the structures are not mutually exclusive. In literature, during the discussion of a class of functions (structure), one often tries to stress the relevance of studying the class by demonstrating how other structures fall into the same class of functions.

Quadratic functions

Much attention in literature on mathematical optimization is devoted to quadratic functions. The objective function f is quadratic, if it can be written as

$$f(x) = x'Qx + d'x + c.$$

It is not difficult to recognise quadratic functions in a given model structure, as only linear terms and products of two decision variables occur in he model description.
The matrix Q is a symmetric matrix which defines the convexity of the function is in each direction. Eigenvectors corresponding to positive eigenvalues define the directions in which f is convex, negative eigenvalues give the concavity (negative second derivatives) of the function in the direction of the corresponding eigenvectors. Depending on the occurrence of positive and negative eigenvalues of Q, the function can be either concave (all eigenvalues negative), convex (all positive) or indefinite (positive as well as negative eigenvalues).

- A global property is that an indefinite function can be regarded as a differentiable convex function, by splitting the function in a convex and concave part.

In contrast to the general concave problem much more value information is available from the matrix Q, vector b and scalar c.

- The derivatives of the quadratic function are linear.

This property gives the possibility to derive a Lipschitz constant on a bounded set relatively easily. This property is used in Chapter 4. A bound on the second derivative can be directly extracted from the eigenvalues of Q.
Quadratic programming problems, f is minimized on a polyhedral set X, have the following property, due to the linearity of the derivatives.

- The Karush-Kuhn-Tucker conditions for the local optima of quadratic programming are a special case of the so called *Linear Complementarity Problem*, which is often discussed in optimization literature.

This property is used further in Section 4.5 for the derivation of a specific algorithm. A relation of concave quadratic programming with integer programming is discussed in Chapter 3. For an overview of quadratic functions in global optimization we refer to Pardalos and Rosen (1987).

Bilinear functions

For bilinear functions the vector of decision variables can be partitioned into two groups (x,y) and the function can be written in the form

$$f(x,y) = c'x + x'Qy + d'y\ ,$$

in which Q is not necessarily a square matrix. The function *is linear whenever either the decision variables x or the decision variables y are fixed.* Actually the function becomes affine in one group of variables when the other group is fixed; bi-affine would be a better name, nevertheless the general term bilinear is used in literature. An example is the function

$$f(x,y) = 2 - 2x - y + xy$$

On the interval $0 \le x \le 4$, $0 \le y \le 3$ this function has a minimum of -1 for $(x,y) = (0,3)$ and a minimum of -6 for $(x,y) = (4,0)$.

Several properties of bilinear functions are:
- Bilinearity can be regarded as a special case of quadratic functions.
- The optimum is attained at the boundary of the feasible set.

A specific way of constructing a **lower bound** is based on the observation in Al-Khayyal (1990, 1992) that xy for $l^x \le x \le u^x$, $l^y \le y \le u^y$ can be underestimated by

$$xy \ge l^x y + l^y x - l^x l^y$$
$$xy \ge u^x y + u^y x - u^x u^y.$$

For our example with $l^x=0$, $u^x=4$, $l^y=0$ and $u^y=3$ this means that

$$xy \ge 3x$$
$$xy \ge 4y - 12.$$

So the function $\varphi(x,y) = \max\{ 3x , 4y - 12\}$ is a minorant of xy on $0 \le x \le 4$, $0 \le y \le 3$.

The appearance of a bilinear structure will be discussed in Section 2.4. In Chapter 4 a practical case containing a bilinear structure is discussed and a specific algorithm is derived after analyzing the properties of the structure.
An extension of the bilinear structure is the idea of *biconvex* functions; i.e. the function is convex whenever one of the sets of variables is fixed. An extensive discussion of this topic can be found in Al-Khayyal (1992).

Multiplicative functions
A function is called a multiplicative function when it consists of a multiplication of convex functions. Besides the multiplication of two variables, as in bilinear programming, higher order terms may occur. A multiplicative function consists of a product of several affine or convex functions. It may not be hard to recognise this structure in a practical model formulation. As it is not used further in the examples of this work we do not go into detail on the mathematical properties, but refer to an overview due to Konno and Kuno (1995).

Fractional or rational functions
A function f is called fractional or rational when it can be written as one ratio or the sum of several ratios of two functions, $f(x) = g(x)/h(x)$. The ratio of two affine functions got most attention in literature. Depending on the structure of the functions g and h, the terminology of linear fractional programming, quadratic fractional programming and concave fractional programming is applied. A basic property which we will use in Chapter 4 due to Dinkelbach (1967) is the following.
Let the function $\varphi(x)$ be defined as $\varphi(x)= \{g(x) - \lambda h(x)\}$.

- If λ^*, the (global) minimum of $f(x)$, is used as the parameter in the function $\varphi(x)$, then the minimumpoint of φ (with objective value zero) corresponds to a global minimumpoint of f.

For an overview on the topic of fractional programming, we refer to Schaible (1995) and the extensive bibliography therein.

General functions with a given explicit mathematical expression
We have assumed that the explicit expression of an objective function f is available of the given derived optimization problem, the problem is of type I. If not any of the structures mentioned above is used then there is a possibility to apply the so called technique of *interval arithmetic*. The function f is considered on a closed hyperrectangle, an interval. In the theory of interval arithmetic, the function is regarded as built up by several mathematical operations, which each have their own so called inclusion function to determine the range of the function values on the interval. By combining those inclusion functions, finally a range is calculated in where the function values on the interval are situated. Better, sharper inclusion functions give tighter upper and lower bounds. For an overview we refer to Hansen (1992), the survey of Ratscheck and Rokne (1995) and to Kearfott (1997).

In the literature on deterministic global optimization these mathematical structures have been analyzed and used for algorithmic development. When a potential user would like to make a choice for the algorithm to be applied, it is necessary first try to **recognise** several of the structures mentioned here, in the mathematical formulation of the model. Some examples are given in the following sections of problems containing a certain structure. The mathematical structure **explains** the possible multi-extremal character of the problem to be solved. The construction of a specific algorithm can be started after analyzing the model. This may require some effort, as will be illustrated in Chapter 4.

We have seen in this section how properties of structures can be **used to derive bounds** on the function value. The sketch of the Danilin-Piyavskii-Shubert algorithm shows how this may lead to a **guarantee** that the global optimum is approximated in the end. The essential property of all the structures is that the objective function value (and that of constraint functions) can be bounded on a set.

2.3. *An example of concave programming*

A simple instance is discussed of a global optimization problem which by Tuy et al. (1996) has been called a *concave production-transportation* problem. It illustrates two aspects. First it shows how a user may discover the existence of multiple optima applying standard nonlinear optimization software. Second, the existence of multiple optima is explained by one of the mathematical structures, concavity, which has been described in Section 2.2.

We use a simple example whose solutions can be calculated by hand. A plan has to be developed for the transportation of biomass from 6 supply units $i=1,..,6$ to two processing centres $j=A,B$ and for the throughput of those centres. The throughput costs are assumed to be concave due to economies of scale and are modelled here by a square root function. The transportation costs are assumed to be linear.

$$\min \{10(\sqrt{TP_A}+\sqrt{TP_B}) + \Sigma\Sigma c_{ij}x_{ij}\}$$

$$\begin{array}{ll} x_{iA} + x_{iB} = S_i & i=1,..,6 \\ \Sigma_i x_{ij} = TP_j & j=A,B \\ TP_j \geq 0,\ x_{ij} \geq 0 & j=A,B \quad i=1,..,6 \end{array}$$

Variable x_{ij} describes the total transportation from supply unit i to centre j and variable TP_j describes the total throughput through centre j. The data on the supply S_i by the units and on the transportation costs c_{ij} from unit i to centre j are given in Table 2.1.

Table 2.1: Transportation costs and biomass supply by the units

units →	1	2	3	4	5	6
A	1	1	1	2	2	1
B	1	2	2	1	1	1
S_i	10	30	10	20	10	20

When throughput costs are neglected, it is cheapest to transport from units 2 and 3 to centre A and from 4 and 5 to centre B. For units 1 and 6 there is no difference in transportation costs to A or B. When throughput costs are included the extreme (concave optimization) global optimum corresponds to using only centre A, resulting in a total costs of 230 (=130 + 10$\sqrt{100}$).

Standard nonlinear programming codes will not directly find the global optimum. A survey on mathematical programming software, 'Optimization Software Guide' (Moré and Wright, 1994) gives several routines for nonlinear optimization. We use the GINO software and GAMS/MINOS here. The GINO software (see Liebman et al., 1985) uses a so called reduced gradient procedure (see Lasdon and

Waren, 1978). The starting values of the variables are in general set to zero. After running the solver, optimization procedure, the software returns the following plan with total costs of 238.4 (=100 + 10√70 +10√30).

Table 2.2: Plan generated by GINO. Starting values all zero.

units→	1	2	3	4	5	6	*TP*
A	10	30	10			20	70
B				20	10		30

The GAMS program (see Brooke et al. 1988) makes it possible to enter the problem in the MINOS solver (routine) which also applies among others a reduced gradient approach (Murtagh and Saunders, 1978). The MINOS solver gives more error messages than GINO on the fact that the objective function is not differentiable at the starting value of zero. After giving starting values to the throughput variables, $TP_A = TP_B = 1$, MINOS returns the following solution with an objective value of 241.4. (=100 + 10√50 +10√50)

Table 2.3: Plan generated by MINOS. Starting values $TP_A = TP_B = 1$, $x_{ij} = 0$.

units→	1	2	3	4	5	6	*TP*
A	10	30	10				50
B				20	10	20	50

This corresponds to another local optimum. A starting value of $TP_A = 100$, returns the global optimum and some warnings on the nondifferentiability of the objective function in the optimum.

For this simple example it is not extremely difficult to identify the global optimum. However, for a larger instance of the same problem with external data or for the same problem as a part of a larger problem, it will not be directly clear where the optimum can be found. Information on the structure of the problem tells us that it is a concave optimization problem and nondifferentiable for a part of the feasible region. Special global optimization algorithms may now be developed for the problem. The development of a specific algorithm for the problem described here can be found in Tuy et al. (1996). The algorithm described in that paper is similar to the Branch-and-Bound approaches used for Mixed Integer Linear Programming (MILP). In fact, it is common in Operational Research to approximate the problem with a fixed charge variant (location-allocation problem) or approximate the costs with piecewise linear programming, which leads to an MILP formulation. This will be worked out in Section 3.2.

2.4. *The pooling problem, a bilinear model*

Another example showing a specific mathematical structure, namely bilinearity, is due to the pooling problem, which got much attention in literature due to its relevance in petroleum industry. The first significant applications of mathematical programming (especially Linear Programming) were to oil refinery, food processing and steel manufacturing planning, see Dantzig (1963) for a discussion of these early applications and Bodington and Baker (1990) for a history of applications in the petroleum industry. Nonlinearities arise in refining, petrochemical and other process manufacturing models due to a variety of factors. An important factor is the need to model properties or qualities of product flows as well as the flows themselves.
We start with a simple blending problem where the relations can be described linearly. By adding the restriction on combining several products in one tank or pool, nonlinear relationships are introduced to capture pool qualities.

In our oil refinery example, crudes A, B and C are bought and blended to products R and S which are sold. We consider one quality item, such as the sulphur content, to describe the quality of raw materials, intermediate products and endproducts. The parameters q_a, q_b, q_c describe the quality of the crudes *A*, *B* and *C*, and the parameters q_r and q_s give maxima (requirements) on the quality of the products *R* and *S*. The prices of crudes and products p_i i = a, b, c, r and s, are given. The decision variables z_a, z_b and z_c give the amount of crudes bought, z_r and z_s the amount of product sold and the variable x_{ij} gives the amount of crude i blended in product j. Now the corresponding blending problem is given as follows.

$$\max \{p_r z_r + p_s z_r - p_a z_a - p_b z_b - p_c z_c\}$$

Component balance:

$$x_{ar} + x_{as} = z_a$$
$$x_{br} + x_{bs} = z_b$$
$$x_{cr} + x_{cs} = z_c$$
$$x_{ar} + x_{br} + x_{cr} = z_r$$
$$x_{as} + x_{bs} + x_{cs} = z_s$$

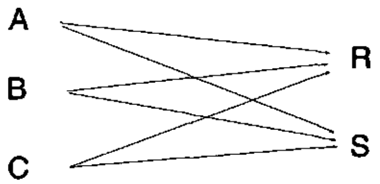

Figure 2.5: The classic blending problem

Product quality constraints:

$$q_a x_{ar} + q_b x_{br} + q_c x_{cr} \le q_r z_r$$
$$q_a x_{as} + q_b x_{bs} + q_c x_{cs} \le q_s z_s$$

Upper and lower bounds on crudes and products:

$$L_r \le z_r \le U_r \text{ etc.}$$
$$x_{ij} \ge 0 \qquad i = \text{a, b, c} \qquad j = \text{r,s}$$

The blending problem described thus far is linear. The pooling problem (Haverly, 1978) now appears, when in the classical blending problem only one tank or pool is used to store streams *A* and *B*. This combined storage (the pool) may be due to storage capacity restrictions. The quality q_0 of the stream out of the pool is now a variable, not predetermined by the data. Let y_r and y_s be the streams from the pool to the products *R* and *S*. The model is changed in the following way.

Component balance:

$$x_{cr} + x_{cs} = z_c$$
$$z_a + z_b = y_r + y_s$$
$$y_r + x_{cr} = z_r$$
$$y_s + x_{cs} = z_s$$

Product quality constraints:

$$q_0 y_r + q_c x_{cr} \leq q_r z_r$$
$$q_0 y_s + q_c x_{cs} \leq q_s z_s$$

The pool quality definition which is added describes that the sulphur stream which goes into the tank, the pool, equals the outgoing stream:

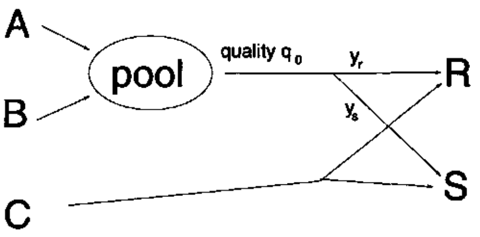

Figure 2.6: Blending with combined storage, the Haverly pooling problem

$$q_a z_a + q_b z_b = q_0(y_r + y_c).$$

This pooling equation and the quality constraints cause an inherent nonlinearity in the otherwise linear model. The problem is possibly multi-extremal; nonlinear programming codes, as outlined in Section 2.3., may fail to find the global optimum. Solving the problem got much attention in algorithmic development, e.g. Haverly (1978), Lasdon et al. (1979), Foulds et al. (1990) and Floudas and Aggarwal (1990). The structure of the problem is known and has been used to develop specific algorithms. The equation describing the pool quality can be called bilinear; a product of two variables occurs next to linear terms. The equation can also be called a quadratic restriction. Moreover, when the equation is divided by $y_r + y_c$, the problem becomes a special case of fractional programming.

The attention in literature is due to the relevance of the pooling problem in petroleum industry. The (multi)blending problem is also well known in the agricultural context of fodder production. Raw materials are bought and mixed to a final product with pre-specified quality requirements. The same type of nonlinearity also occurs in environmental planning models, when a model describes flows of products and waste. The combination of describing measures, storage and changes in concentrations of nutrients (polluting materials) may lead to the pooling property. This will be discussed in Chapter 4.

2.5. *Factorial and quadratic regression models*

Regression analysis is a technique which is very popular in scientific research and in design. Very often it is a starting point for the identification of relations between inputs and outputs of a system. In a first attempt one tries to verify a linear relation between output y, called regressand or dependent variable, and the input vector x, called regressor, factor or independent variable. A so called linear regression function is used:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

For the estimation of the coefficients β_j and the check how good the function 'fits reality', either data from the past can be used or experiments can be designed to create new data for the output and input variables. The data for the regression can be based on a design of a computer experiment which uses a simulation model to generate the data on input and output. The generation of regression relations out of experiments of a relative large simulation model is called *metamodelling* and is discussed in Kleijnen and van Groenendaal (1988). The regression model is called a metamodel, because it models the input-output behaviour of the underlying simulation model. In theory about design, the word *response surface methodology* is more popular and promoted among others by Taguchi (see e.g. Taguchi et al., 1989).

The regression functions based on either historic data, special field experiments or computer experiments can be used in an optimization context. As long as the regression function is linear in the parameters β, and in the input variables x_j, linear programming can be applied. The optimization becomes more complicated when interaction between the input variables is introduced in the regression function. Interaction means that the effect of an input variable depends on the values of another input variable. This is usually introduced by allowing so called two-factor interaction, i.e. multiplications of two input variables in the regression function. An example of such a *factorial regression model* is

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2 .$$

The introduction of multiplications also implies possible multi-extremality, when the functions are used in an optimization context. Consider for example the minimization of $y = 2 - 2x_1 - x_2 + x_1 x_2$ with $0 \leq x_1 \leq 4$ and $0 \leq x_2 \leq 3$. This problem has two minima: $y = -1$ for $x = (0,3)$ and $y = -6$ for $x = (4,0)$.

A further extension in regression analysis is the inclusion of quadratic terms which results in a complete second order Taylor series approximation of the relation one intends to find. In two dimensions the quadratic regression function is:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2 + \beta_{11} x_1^2 + \beta_{22} x_2^2 .$$

Notice that in regression terms this is called linear regression, as the function is linear in the parameters β. When these functions are used in an optimization context, it depends on the second order derivatives β_{ij} whether the function is convex and consequently whether it may have only one or multiple optima.
We were involved in several studies which included the approach of fitting linear and quadratic models and using the models for optimization.

An example from our experience in using optimization combined with regression in optimal control can be found in de Blank et al. (1997). A method is described for the optimization of the pelleting process of animal feed which is based on data of batch runs in the past. For the generation of an advice on the control variables for a new batch, first batches are selected from a database, which have been produced under the same circumstances such as outside temperature, fibre content of the production material, etc. Those data are used to fit linear regression relations between output, i.e. quality indicators, energy consumption and productivity, and the control variables as regressors. Those relations are used in a linear programming problem to create an advice on the values of the control variables. By preselecting the data, linear models can be used and for the optimization the problem of having multiple optima is avoided.

Another example is due to a study on the feasibility of an introduction of a new crop in the Netherlands for the production of fibres. A research group had the task to find optimal production circumstances. A set of experiments was planned to measure the production and the quality under different circumstances given by settings of temperature, chemical control etc. The resulting quadratic regression functions were not all concave or convex; nevertheless the resulting optimization problem had one, relatively 'wide', local optimum. Like in the response surface method it would be wise to perform new experiments in the neighbourhood of the optimum found.

Another application, which has been described in Hendrix and Pintér (1991) and in Hendrix et al. (1993), originates from a design department of a lubricant factory. Experiments are done to find the relation between quality y and input variables x describing the production circumstances and the composition of the mixture. Given the derived regression models, the optimization question was to find a product which meets predefined requirements formulated as $y_i(x) \leq b_i$. Moreover, technical restrictions on the design variables x are given. Possible optimization formulations involve the minimization of

$$f(x) = \max_i \ \{y_i(x) - b_i\}$$

or the minimization of

$$f(x) = \sum_i \max \{y_i(x) - b_i, 0\}.$$

When all quadratic relations $y_i(x)$ are convex, the objective functions imply the minimization of a convex nondifferentiable function; there is one local optimum. In general the regression functions $y_i(\mathrm{x})$ will not be convex, so that we need global optimization techniques to solve the problem. This case is used in Chapter 4 to discuss global optimization approaches to solve the resulting optimization problem. The same methodology of fitting quadratic models appeared to be common at a design department of a large firm in electronic industry. In a special project tolerances of the designs were studied such that finally robust products are produced. Some results are shown in Section 4.5.

The use and construction of factorial and quadratic regression models occurs frequently in research and development. When the models are applied in an optimization context, an optimization problem appears which is possibly multi-extremal containing a specific mathematical structure given the underlying quadratic relations. In Chapter 4 it is illustrated how this structure can be applied for the construction of specific algorithms.

2.6. *Alternative solutions due to symmetry in the model formulation*

2.6.1. Introduction

It has been shown so far how by analyzing the model, particular mathematical structures may be detected. This work focuses on how to exploit the structure and information of the model. In this section it will be shown how the existence of multiple optima may not be caused by the mathematical structure of the model, but by the translation from object system to model. The translation is represented by arrow a in Figure 1.1. The user who after finding multiple optima starts analyzing the problem formulation should keep in mind the following pitfall. Symmetry in the model formulation may cause several solutions of the model to represent the same object in the 'real world', object system. The solutions can be exchanged and have equal objective function values. Simple examples can be found in integer programming, when ordering of points has no special meaning for the solution.

Consider a discrete optimal design problem in which four experimental points should be selected from a predefined set S of candidate points, $S = \{1, 4, 7, 10, 11, 17\}$ (e.g. Rasch et al., 1997). This problem has $\binom{6}{4} = 15$ feasible solutions. When binary decision variables in an optimization problem are defined as

x_k: Candidate point selected as design point number k,

there are several solutions of the optimization problem with the same practical meaning. Because the ordering of the points is not important, solution (4,7,11,17) is the same as solution (11,7,17,4), both representing the set of points {4,7,11,17}.

Another simple example is due to the travelling salesman problem: Given n cities, find that permutation of the numbers 1, 2, ..., n which represents a tour with minimum costs. When the begin and end point of the tour is not fixed, solution 1, 3, 2, 5, 4 represents the same tour as solution 5, 4, 1, 3, 2.

This property of the possibility to exchange points, leading to solutions which have the same practical meaning, does not refer directly to the mathematical structures which are summarized in Section 2.2. However, in the construction of an algorithm to find a solution of the problem a modeller should make use of this property. In the following sections some practical nonlinear programming problems will be discussed which have the symmetry property.

2.6.2. Learning of neural nets as parameter estimation

The use of a neural net as a model to translate input into output is quite in fashion. It is doubtless that neural nets have been successfully applied in pattern recognition tasks. In literature on Artificial Intelligence a massive terminology has been introduced around this subject. Here, the learning of neural nets is discussed from the viewpoint of parameter estimation in which the terminology of AI is avoided. An introduction in the OR literature can be found for example in Masson and Wang (1990). For the interested reader we refer to other introductory texts from the same period, such as Beale and Jackson (1990) and Eberhart and Dobbins (1990).

It is the task of a neural net to translate input x into output y. Therefore a neural net can be considered a model in the sense of Figure 1.2. Parameters, formed by so called weights and biasses, are used to tune the net, which can be seen as a large regression function. We quote a colleague, Dr. Zwietering: "A neural net is often an euphemism for fitting with too many parameters". The tuning of the parameters, called learning in the appropriate terminology, can be considered a parameter estimation problem. In Chapter 6 some other topics on this subject will be discussed. In this section, the appearance of multiple optima of a goodness of fit criterion in the parameter space of the neural net is discussed.

To start with the analysis, the net is considered as a directed graph with arcs and nodes. Every node represents a function which is a part of the total model. The output y of a node is a function of the weighted input $z = \Sigma w_i x_i$ and the so called bias w_0. So a node in the network has weights (on the input arcs) and a bias as corresponding parameters. The input z is transformed to output y by a so called transformation function. Usually the sigmoid or logistic transformation function is used:

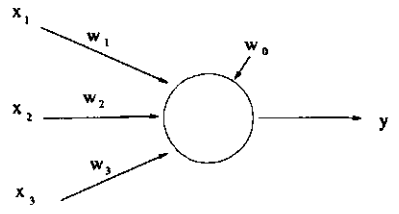

Figure 2.7: One node of a neural net

$$y = 1/(1 + \exp(w_0 - z)).$$

For the analysis, only this function will be used. This means that every individual node corresponds to a logistic regression function. The difference with the application of such functions in growth models and in logit models is that the nodes are connected by arcs in a network. Therefore, the total net represents a large regression function. The parameters w_i can be estimated to describe relations as revealed by data as good as possible. For the illustration of the analysis a very small net is used as given in Figure 2.8. It consists of two so called hidden nodes H_1 and H_2 and one output node y. Each node represents a logistic

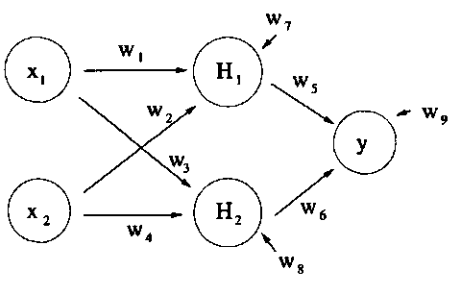

Figure 2.8: Small neural net with two inputs, one output and two hidden nodes

transformation function with three parameters, two on the incoming arcs and one bias. Parameters w_1, w_2, ..., w_6 correspond to weights on arcs and w_7, w_8 and w_9 are biasses of the hidden and output node. The corresponding **regression function** is given by:

$$y = 1/[1 + \exp(w_9 - \frac{w_5}{1 + \exp(w_7 - w_1x_1 - w_2x_2)} - \frac{w_6}{1 + \exp(w_8 - w_3x_1 - w_4x_2)})].$$

The net is confronted with data. Usually the index p of "pattern" is used in the appropriate terminology. Input data x_p and output t_p (target) are said to be "fed to the network to train it". From a regression point of view, one wants the parameters w_i to take values such that the predicted $y_p(x_p,w)$ fits well the output observations t_p according to a goodness of fit criterion such as

$$f(w) = \sum_p (y_p - t_p)^2 ,$$

in which y_p is the regression function calculated for x_p and the weights w. A usual way to train the net is by using a so called back propagation method. In order not to go too deep into the subject, only the assessment of the values for the weights on the arcs between hidden and output node is considered here. In Figure 2.4 those are the weights w_5 and w_6. Input x_p leads to input z_p into node y which finally leads to an output y_p which is compared with the target t_p. For the sigmoid transformation function the back propagation method will lead to the following updating rule for the weights between hidden and output node:

$$\Delta w_i = \eta(t_p - y_p)z_{ip}y_p(1 - y_p).$$

In this rule η is a parameter and z_{ip} represents the input in node y corresponding to weight i. It has been shown in e.g. Hung and Denton (1990), that feeding a complete batch of data to this rule corresponds to updating the weights according to the **descent gradient** of $f(w)$. This is outlined here.
Classic nonlinear programming learns that in the corresponding terminology:

$$\frac{\partial f}{\partial w_i} = 2\sum_p (y_p - t_p)\frac{\partial y_p}{\partial w_i}.$$

For the logistic transformation:

$$\frac{\partial f}{\partial w_i} = z_{ip}y_p(1 - y_p).$$

Since the weights in the back propagation method are changed proportionally to location space. Moreover, the symmetry property holds; i.e. exchanging two vectors p_j and p_k (in fact the indices of the vectors) represents the same solution of the real system and thus has the same objective value. Given a solution of the vectors $p_1,..,p_n$, the indices of the locations can be presented in any order, representing the same solution. So, when n facilities are to be located, then there are $n!$ equivalent variants of the same solution.

The same property propagates to extensions of this base variant of the problem. One variant of the facility location problem can be formulated by adding capacities (capacitated plant location) on the throughput of the facilities:

$$\sum_i x_{ij} \leq \text{cap}_j \qquad j = 1,...,n.$$

The calculation of the objective $f(p)$ for a set of location vectors p now requires solving a transportation problem. When the given capacities cap_j differ (and are restrictive), the possibility to exchange the (indices of the) locations becomes less. When the capacities are all the same, the indices can be ordered again in any order, representing the same solution.

A second variant is to introduce concave throughput costs as shown in Section 2.3. The variable TP_j and the definition

$$\sum_i x_{ij} = TP_j \qquad j = 1,...,n$$

are added to the original problem together with a (throughput or production) cost function on variable TP_j, e.g. a square root function. Now the problem can be considered from the joint decision space of location vectors p_j and throughput variables TP_j with a corresponding objective function $f(p,TP)$, which, when evaluated, requires the solution of a transportation problem. It is useful to introduce a penalty function for nonfeasibility of the transportation problem i.e. when the total supply ΣS_i exceeds the throughput ΣTP_j. The concavity which is in general used to express economies of scale in the throughput costs, adds the effect of multiple optima due to concavity, as illustrated in Section 2.3. When the cost function is equal for every facility, the locations (in fact the indices) of a solution can be exchanged, without changing the practical meaning of it.

The general problem was studied in a special project at Wageningen Agricultural University. Global optimization algorithms were used in the location (and throughput) part of the decision space of all three variants of the problem. Considering the problem from the joint space means that at every function evaluation, calculation of $f(p,TP)$, either an allocation or a transportation problem was solved. It is not difficult to construct worst case instances of the problems, which are extremely difficult to solve, even for a small number of facilities.

For testing the approaches on a realistic problem, data were taken from a larger study on environmental problems caused by manure in The Netherlands, see de Mol and van Beek (1991), which focused on opening facilities (from a candidate set of locations), transportation from surplus areas to shortage areas and on export to diminish the surplus of manure. For this case we took the data on locations of groups of farmers from the area with surplus. The locations for the facilities could be chosen freely in contrast with the fixed candidate locations. A group i of farmers wants to transport a surplus S_i to one of the facilities j which has to be set up. Figure 2.10 illustrates the situation with one depot. In Figure 2.10 the size of the spheres represents the size of the supply at location i. The optimal situation with one facility corresponds to calculating a weighted barycentre of the locations d_i. In Figure 2.11 a solution is shown of a problem instance with n = 4 facilities with given capacities. The solution was generated by applying among others the controlled random search method of Price (1979), which is discussed further in Section 6.3. Many methods could be used to try to solve the problem. The figure illustrates the **symmetry**: indices of the locations can be exchanged. Moreover it shows that graphical information can be fed back to the user of an algorithm. As in vehicle routing problems, the interaction between algorithms which can evaluate many solutions very quickly, and the user who has graphical insight may be very useful to speed up the solution process. In this particular example, one could think of selecting starting points for a local search in an attempt to avoid local optima. We have seen here that the exchangeability of the indices due to symmetry in a problem leads to multiple optimal points which graphically have the same interpretation.

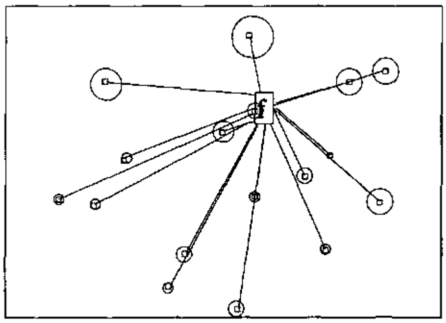

Figure 2.10: Location and sizes of the 'customers' and corresponding weighted barycentre

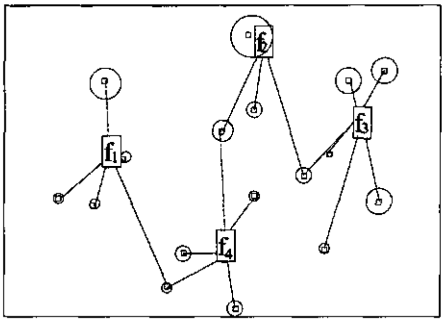

Figure 2.11: Solution of a problem with given capacities

2.7. *Concluding remarks*

In Section 2.2. a summary has been given of the mathematical structures which are distinguished in literature as being useful for the choice and development of special (deterministic) algorithms. Some examples have been discussed of models from practical problems and where mathematical structures can be recognised.

- The facility location model with concave throughput costs illustrated how widely used local search nonlinear programming software may fail to find the global optimum starting from default starting values.
- The pooling problem showed how a description of quality within a flow planning model may lead to bilinear model constructions. This kind of descriptions are not only useful in the classic oil refinery models, but also relevant for environmental planning problems, as will be discussed in Chapter 4.
- The discussion on the use of metamodels and response surface modelling shows how multiple optima in a very simple way of modelling may appear when factorial and quadratic regression is used. This also leads to one of the mathematical structures, viz. quadratic programming, which got much attention in the global optimization literature.

Section 2.6. shows that it is useful, first to have a look at the practical meaning of various optima of an optimization problem. Two examples are given where symmetry in the model formulation plays a role; the estimation of parameters in a neural network and the continuous facility location problem. They show that several optima may have the same meaning in the object system which has been modelled and thus have the same objective value. After the discovery of the multi-extremal character of the optimization problem one can first have a look at the practical interpretation of the solutions before starting looking for mathematical structures to explain the existence of multiple optima and which can be used further in deterministic global optimization algorithms.

In Chapter 4 the exploitation of the mathematical structure for the development of algorithms is worked out for cases containing elements of the examples shown in this chapter. First, in Chapter 3 a link will be made between global optimization and integer programming. Algorithms in integer programming are in general better known to our target groups and help to understand the algorithms in Chapter 4.

Chapter 3. *Relation between multi-extremal and integer programming*

3.1. *Introduction*

Focus in this chapter is on the relation between integer programming, which is generally known by our target groups, and global optimization. There are several reasons to have a better look at similarities between the two fields, combinatorial and global optimization.

1. From a theoretical point of view a relation between the two fields is emphasized in literature on global optimization, to discuss the complexity of global optimization; global optimization is "at least as difficult as integer programming". If there would exist an algorithm which can solve global optimization problems in polynomial time, we would be able to solve combinatorial problems in polynomial time. This relation which is proven in literature is illustrated in this section by some notes and an illustrative example. In Section 3.5. an example is given, derived from a practical problem, which illustrates that no matter how we consider the problem, from a continuous viewpoint or from a combinatorial viewpoint, it remains very hard to solve it.

2. On the other hand, there are many implemented and elaborate algorithms in integer programming, which can be applied to solve variants of global optimization problems; there exist multi-extremal models which are suited to be solved by using integer programming techniques. One of the early methods to solve nonlinear programming is to approximate a problem by piecewise linear programming. For nonconvex optimization this requires the use of binary variables and integer programming algorithms. This link between nonlinear and linear programming will be discussed in Section 3.2. In Section 3.4. a practical example is discussed of a global optimization problem which was approximated by an MILP formulation and solved via integer programming techniques.

3. There is a strong similarity in algorithms of combinatorial optimization and algorithms of global optimization. Knowledge of algorithms in one group can help to understand algorithms in the other group. Moreover, it is our opinion that the algorithms can be globally divided in a similar way. This will be discussed in Section 3.3.

4. Considering an integer programming problem from a continuous viewpoint may help to analyze the existence of many optima. Ideas from stochastic global optimization can be applied to find good solutions of integer problems. An example of a practical decision problem which can be viewed in this way, is discussed in Section 3.6.

We start here by discussing the complexity considerations which can be found in any introductory text on global optimization. This is followed by a discussion of a well known problem, the so called Quadratic Assignment Problem, to illustrate the complexity relation. Consider the following binary programming formulation.

$$\min f(x) \quad , \quad x_i \in \{0,1\} \quad i = 1,\ldots,n \ . \tag{3.1}$$

The real valued function f has a minimum on the compact set defined by the integrality constraints. Without loss of generality we left out other constraints in (1). Integer problem (3.1) is equivalent with the following problem (i.e. the set of global optimal solutions is the same):

$$\min \{f(x) + Mx'(e-x)\}, \ 0 \leq x_i \leq 1 \quad i = 1,\ldots,n. \tag{3.2}$$

An early publication is Raghavachari (1969), but we were assured that the relation is one century older. In (3.2), vector e is the vector with all elements 1. The penalty M forces the fractional variables x_i to take a binary value; M should be sufficiently big. Another formulation analogous to (3.2) (continuous and equivalent) is the following:

$$\min \{f(x) - M \ \Sigma|x_i - 0.5|\}, \ 0 \leq x_i \leq 1 \quad i = 1,\ldots,n. \tag{3.3}$$

Without going too much into detail on the topic of complexity we can say that hard combinatorial optimization problems, so called NP hard problems, are believed to have no polynomial time algorithms (see e.g. Aarts and Lenstra, 1997). This means that the calculation time to solve those problems (worst case variants) grows more than polynomial with the dimension n, which is also the case for generic problem (3.1). If we could be able to develop algorithms which solve general multi-extremal problems within a calculation time which grows polynomially in the dimension n, then we would be able to solve (3.2) and (3.3) in polynomial time and thus also (3.1). In our experience we encountered modellers who did reformulate a problem of the form (3.1) towards (3.2) with the aim to obtain an easier to solve problem and thus constructed an unsolvable global optimization problem with, in general, many optima.

These complexity issues and the transformation are illustrated by discussing the so called Quadratic Assignment Problem (QAP). This illustration is based on the discussion of this subject in Horst et al. (1995). We will use an example here. Let us assume that a department of 20 persons gets a new building with 20 rooms. To prevent a fight over the new rooms the manager decides to formulate a quantitative model to support the decision on which person gets which room. The binary decision variable is formulated as:

x_{ij}: person i gets yes/no room j.

One of the objectives of the manager is to get a clustering of persons which cooperate a lot, have much interaction. For all 190 pairs of rooms he measures the distance d_{jl} and for all 190 pairs of persons he measures the amount of interaction or information between the persons a_{ik}. Indices i and k stand for persons and j and l for rooms. With this information he creates weighted interaction coefficients $q_{ijkl} = a_{ik} \times d_{jl}$. Let us assume that also preference or suitability indicators c_{ij} exist.

The result is a variant of the so-called Quadratic Assignment Problem:

$$\min f(x) = \sum_{i=1}^{20} \sum_{j=1}^{20} c_{ij}\, x_{ij} + \sum_{i=1}^{20} \sum_{j=1}^{20} \sum_{k=1}^{20} \sum_{l=1}^{20} q_{ijkl}\, x_{ij}\, x_{kl}$$

$$\sum_i x_{ij} = 1 \quad j = 1,\ldots,20 \qquad \text{(QAP)}$$

$$\sum_j x_{ij} = 1 \quad i = 1,\ldots,20$$

$$x_{ij} \in \{0,1\} \quad i,j = 1,\ldots,20.$$

This problem is known in literature to be hard to solve; it is a so called NP-complete problem. The feasible set in the QAP problem mentioned here defines 20! possible assignments. No algorithm exists which solves the problem in a calculation time which grows polynomially with the number of persons and rooms.

We consider the problem from a mathematical viewpoint. When the elements x_{ij} are stored in a vector x with $20^2 = 400$ elements, the objective function can be written as:

$$f(x) = c^T x + x^T Q x\ .$$

The matrix Q contains 400×400 = 160,000 elements q_{ijkl}. In a realistic situation the distance between rooms is symmetric $d_{jl} = d_{lj}$ and $d_{jj} = 0$. A similar reasoning applies for the stream of information a_{ik} so that there are in fact at most 190×190 = 36,100 unique nonzero elements in Q.

We have seen in Chapter 2 that the convexity of $f(x)$, determined by matrix Q, tells us something about multi-extremality of the objective function in global optimization. Let us therefore consider the continuous (nonlinear programming) variant called CQAP of QAP. Variable x_{ij} is the fraction of the working time that person i is in room j. The integrality constraint is replaced by $0 \le x_{ij} \le 1$. Some extreme, nonrealistic instances show the complexity of the two associated problem.

<u>Extreme case 1</u>: Let f be convex. A simple analysis can be done when we consider the nonrealistic situation where all values c_{ij} and q_{ijkl} are zero apart from all 400

diagonal values q_{ijij} which are taken positive and equal. In this case the unique solution of CQAP is given by $x_{ij} = 1/20$ for all i and j. The solution gives a value of 1/400 for all products $x_{ij}x_{ij}$ and results in an optimal objective value of q_{ijij}. General nonlinear programming routines will find this unique solution. Every feasible (integer) solution of QAP contains 20 assignments and results totally in 20 values of 1 for the products $x_{ij}x_{ij}$ giving an objective value of $20 \times q_{ijij}$. This illustrates that relaxing the integrality constraints to generate an approximation of the solution of QAP by solving CQAP, may result in a bad approximation.

<u>Extreme case 2</u>: The opposite occurs when f is concave and all values c_{ij} and q_{ijkl} are zero apart from all 400 diagonal values q_{ijij} which are taken negative and equal. In this case all 20! feasible assignments of QAP are global optima of CQAP; we have created a washing board with 20! holes. As we are dealing with concave minimization, the optimum can be found at the extremes of the feasible set. The optimal solution will not take fractional values, as the vertices of the feasible area are integral. A general nonlinear programming algorithm will find one of the global optimal points. By adding a small slope to this washing board, having different values for the suitability indicators c_{ij}, the result is a CQAP problem with numerous local optima. Local nonlinear optimization routines will only identify one of the optima given a starting point.

For extreme case 2, the combinatorial QAP and global optimization CQAP are equivalent, have the same solution set and both are very hard to solve. For extreme case 1, the CQAP problem is easy to solve, but its optimum is far from the optimum of the QAP problem. Approximating one problem with the other has no use.

Pardalos (see Horst et al., 1995) shows that any QAP problem has an equivalent CQAP problem, which appears by a transformation which is similar to the relation between (1) to (2). According to this idea the objective function of the CQAP variant is made concave (concave quadratic programming) by adding to $f(x)$ the term $-\frac{1}{2}\alpha\Sigma\Sigma x_{ij}^2$. When α is chosen greater than the largest eigenvalue of the Hessean $2Q$ of $f(x)$, then $f(x)$ becomes concave. Minimizing a concave function over a polytope implies that the optima can be found at the vertices of the feasible set. This means that the optima take a binary value and thus are solutions of QAP. The modification of the objective function by adding $-\frac{1}{2}\alpha\Sigma\Sigma x_{ij}^2$, gives that for all integer solutions the objective value decreases by $\frac{1}{2}\alpha(20)^2$. In this way the modified CQAP problem is equivalent, has the same solution set, to the QAP problem. If we would be able to develop algorithms which solve concave quadratic programming problems in polynomial time, we could solve the general quadratic assignment problem and with that all problems which are called NP-complete in polynomial time.

To complete the discussion on the QAP problem, we will finish with the Mixed Integer Linear Programming (MILP) variant which we frequently observed to be used by OR practitioners to treat the problem. The QAP problem can be transformed to an MILP equivalent by introducing variables z_{ijkl} to describe and replace the interaction $x_{ij}x_{kl}$. Notice that for the 20 person problem this are at most 160,000 variables z_{ijkl}.

$$\min f(x) = \sum_{i=1}^{20}\sum_{j=1}^{20} c_{ij}\,x_{ij} + \sum_{i=1}^{20}\sum_{j=1}^{20}\sum_{k=1}^{20}\sum_{l=1}^{20} q_{ijkl}\,z_{ijkl}$$

$$\sum_i x_{ij} = 1 \quad j = 1,\dots,20 \qquad \text{(MILP-QAP)}$$

$$\sum_j x_{ij} = 1 \quad i = 1,\dots,20$$

$$x_{ij} \in \{0,1\} \quad i,j = 1,\dots,20.$$

In addition, the link between the assignment variables x_{ij} and the interaction variables z_{ijkl} has to be established by adding constraints:

$$z_{ijkl} \le x_{ij} \ \text{ and } \ z_{ijkl} \le x_{kl} \quad \text{if} \quad q_{ijkl} < 0 \tag{3.4}$$

and

$$z_{ijkl} \ge x_{ij} + x_{kl} - 1 \quad \text{if} \quad q_{ijkl} > 0. \tag{3.5}$$

Of course the variable z_{ijkl} can be left out when $q_{ijkl} = 0$. In this formulation the variable z_{ijkl} can be defined fractional between 0 and 1. Another often used variant in which the two linking constraints (3.4) are replaced by $z_{ijkl} \le (x_{ij} + x_{kl})/2$, requires z_{ijkl} to be integral too.

The solution methods for MILP problems are often based on a Branch-and-Bound approach, which is discussed further in Section 3.3. The bounds are based on a so-called LP relaxation, leaving out the integrality constraints. This is sometimes called by researchers the "cross your fingers approach" as one hopes te generate an integer solution automatically. Unfortunately this happens very seldom for the MILP-QAP formulation making the QAP problem hard to solve.

Let us consider some extreme cases again. For an instance with all $q_{ijkl} = 0$, the problem becomes the general assignment problem, for which the LP solution is integral automatically. Apparently the hard part of the problem is the quadratic, interaction part. For extreme case 1, all values of q_{ijij} equal and positive, the relations (3.5) apply. All nonintegral solutions with $x_{ij} \le 1/2$, give optimal LP solutions with an objective value of zero. This is even more far from the optimal objective $20{\times}q_{ijij}$ of every feasible (integer) assignement, than the optimal corresponding CQAP solution. This is not a very hopeful observation for the "cross your fingers" approach. For extreme case 2, where (3.4) applies and one tries to maximize the quadratic variables z_{ijij}, the optimal LP bound coincides with the optimal QAP objective.

The **QAP** problem has been used here to do some experimental thinking on complexity issues. The QAP problem is known in literature to be **hard to solve**, it is so called NP-complete. It has been shown that there is an equivalent concave quadratic programming variant to emphasize the message:

If we could solve global optimization problems in polynomial time, we could solve difficult combinatorial problems.

Relaxing the integrality of the QAP problem results in the CQAP problem, which has in general multiple optima. We have shown a worst case with $n!$ optima and a "best" case with one optimum. The analysis of the "best" case teaches us that there may be a large gap between the optimal CQAP problem and the corresponding (same parameters) QAP problems. The LP relaxation of the MILP variant of the QAP problem is not equivalent to the CQAP problem. It also showed that the bound generated by an LP relaxation of the MILP problem may deviate largely from the optimal value. As will be discussed in Section 3.3, this makes a problem hard to solve by a Branch-and-Bound approach.

3.2. *Problems handled by piecewise linear programming*

3.2.1. Introduction

Not every practical nonconvex optimization problem is doomed to be unsolvable. In this section it will be discussed how ideas from Linear Programming and Integer Programming may be applied in solving problems with some additional nonlinearity in a further linear model. From the early start of linear programming, mathematicians have been interested in including nonlinearities in linear models. We quote the well known remark "*But we all know the world is nonlinear*", after the first major lecture in 1948 of Dantzig on the subject of linear programming (Dantzig, 1991). The models are linear but one wants to include a nonlinear function $f(x)$ which is a function of one variable.

In textbooks on linear programming besides the terminology 'piecewise linear programming' (resulting, more correctly, in a piecewise affine function) the expression *separable programming* is used. In this way one expresses that the function $f(x)$ consists of a sum of nonlinear functions in one variable. We will show that the same concept can also be applied for functions of several variables. The advantage of approximating nonlinear programming problems by linear or, as will be discussed, mixed integer programming problems is that there exist many standard solution routines for these problems.

It will be discussed how nonlinearities can be included in a Linear Programming formulation by Piecewise Linear Programming (PWLP). First, a convex programming formulation with nonlinear functions of one and several variables is considered in 3.2.2. Consecutively the idea of performing PWLP iteratively (Sequential Piecewise Linear Programming, SPLP) to save calculation time and computer memory, is shown in 3.2.3. Then the treatment of non-convex programming by including binary variables in the Linear Programming formulation is discussed in 3.2.4. In Section 3.3 we proceed by discussing the corresponding solution approaches.

3.2.2. Piecewise linear approximations in convex programming

In textbooks on mathematical programming several formulations can be found under the name Piecewise Linear Programming (PWLP) or separable programming. We will discuss one way of including a function of one variable into an LP problem.
Let the one-dimensional function $f(y)$: $\mathbb{R} \rightarrow \mathbb{R}$ be defined on an interval $[l,u]$. In the interval the interpolation points $Y_1 = l, Y_2, ..., Y_N = u$ and the corresponding function values $f(Y_1), ..., f(Y_N)$ are defined. Now interpolation variables λ_i, sometimes called non-negative weightings, are added to the LP problem with the restrictions

$$\begin{aligned}&\Sigma\, \lambda_i\, Y_i = y\\ &\Sigma\, \lambda_i f(Y_i) = \varphi\\ &\Sigma\, \lambda_j = 1,\ \lambda_i \geq 0\ .\end{aligned}$$

The last constraint is called the convexity row as a convex combination is taken of the interpolation points.

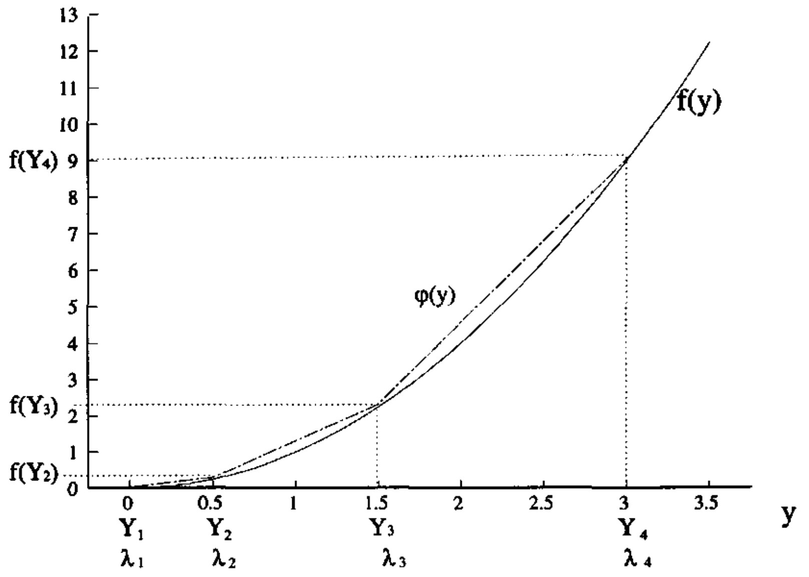

Figure 3.1: Piecewise approximation of $f(y)=y^2$

For example the approximation of $f(y) = y^2$ on [0,3] can look as follows. Take as interpolation points $Y_1 = 0$, $Y_2 = 0.5$, $Y_3 = 1.5$ and $Y_4 = 3$:

$$\begin{aligned}
&0.5\lambda_2 + 1.5\lambda_3 + 3\lambda_4 = y \\
&0.25\lambda_2 + 2.25\lambda_3 + 9\lambda_4 = \varphi \\
&\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 \quad = 1,\ \lambda_1,\lambda_2,\lambda_3,\lambda_4 \geq 0.
\end{aligned}$$

The linear programming variable φ is now implicitly a piecewise linear approximation $\varphi(y)$ of $f(y)$ on the interval $[l,u]$ when at most two adjacent λ_i take a positive value. This happens automatically when we deal with a convex programming problem i.e. $f(y)$ is convex and minimized or occurs in an appropriate way in a constraint. The choice of the interpolation points Y_i is also a topic mentioned in handbooks. If one wants a good approximation the distance between the points can be chosen bigger on parts where $f(y)$ is nearly linear than on parts where $f(y)$ is very curved.

Approximating nonlinear convex functions of several variables

As pointed out in Williams (1990) and Hendrix (1990), the approximation idea by interpolation is not limited to functions of one variable. Therefore the expression *separable programming* for the idea of PWLP is not appropriate. Consider the function of two variables $f(y)$: $\mathbf{R}^2 \to \mathbf{R}$ on the box constraints $l_k \leq y_k \leq u_k$, $k = 1, 2$, defining a two dimensional interval. A not necessarily equidistant grid with interpolation points Y_{ij}, $i = 1, ..., N_1, j = 1, ..., N_2$ is defined.

The corresponding PWLP formulation is as follows:

$$\Sigma\,\Sigma\,\lambda_{ij}\,Y_{ij} = y$$
$$\Sigma\,\Sigma\,\lambda_{ij}\,f(Y_{ij}) = \varphi$$
$$\Sigma\,\Sigma\,\lambda_{ij} = 1,\ \lambda_{ij} \geq 0\ .$$

Theoretically this approach can be extended to include functions of more than two variables f: $\mathbf{R}^K \to \mathbf{R}$ in the LP formulation. The number of indices K grows correspondingly. An important observation is that the number of interpolation points, when N_k is taken fixed, grows exponentially with the dimension K of the function which is approximated. A practical relevance of this remark is due to the implicit appearance of approximations in farm management (see Hendrix, 1990) and land use models. This statement is elaborated here. In farm models usually the main variable, called activity x, denotes the part of an area which is used or cultivated with a certain crop which is treated in a certain way. One index is used to describe the crop. Other indices may be used to describe the level of a certain input such as fertilizer, use of pesticide and cattle density per hectare. For a number of values of an input (index), coefficients are generated. If for example

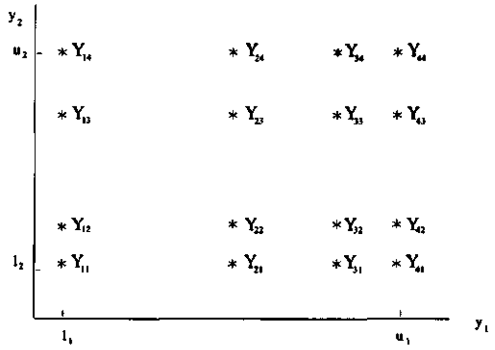

Figure 3.2: Interpolation points in approximating functions of several variables

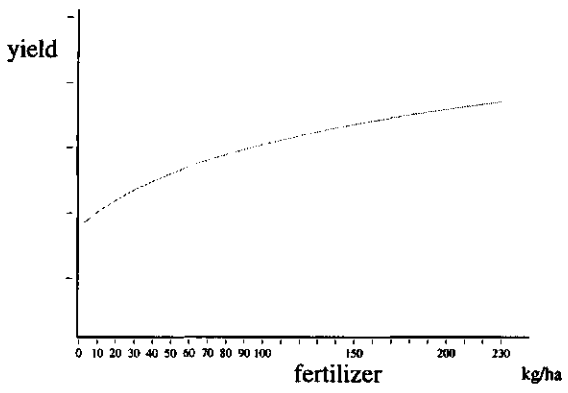

Figure 3.3: Nonlinear relation in farm management

the optimal outcome represents 2 hectare maize with a fertilizer level of 100 and 3 hectares of maize with fertilizer level of 150, the outcome can be interpreted as an approximation of an average fertilizer use of 130 per hectare. In this way the variables x can be interpreted as the weightings λ_{ij}. It is not uncommon in large studies (see e.g. Rabbinge and van Latesteijn, 1992), to use large crop growth simulation models to generate LP coefficients for several (N_k in fact) input levels for several (K) inputs. The number of activities and required coefficients grows exponentially with the number of inputs K which is included in the model.

3.2.3. Sequentially applying Piecewise LP for convex functions

The inclusion of a nonlinear function $f(y)$ into an LP problem by adding its piecewise linear programming approximation $\varphi(y)$ to the LP problem, may lead to an enormous increase in the size of the LP problem due to the number of interpolation points. In Hendrix (1990) it has been suggested to reduce the initial number of interpolation points for the construction of $\varphi(y)$, by iteratively calculating the LP problem with an updated set of interpolation points. This is called Sequential Piecewise Linear Programming. This idea is elaborated here.

In nonlinear programming the so called linesearch, looking for the optimum over a line, is often done by so called **interval reduction techniques**. The purpose is to bracket a minimum point over a line by an interval that shrinks in size iteratively. Two points are taken in the interior of the interval and their function values are used to cut away a part of the interval in which the minimum cannot be situated. A similar interval reduction approach can be used in PWLP. Let us consider a univariate function $f(y)$ on the interval $[Y_1,Y_4]$. Instead of using a large set of interpolation points, the approximation $\varphi(y)$ is constructed by adding two points in the interior so that there are four interpolation points, $Y_1 < Y_2 < Y_3 < Y_4$ like in figure 3.4. The optimal value $\bar{y}$ which has been found by the LP problem can now be used to **shrink** the interval and to bracket a minimum point y^* of the original problem. The shaded area between Y_3 and Y_4 does not contain the optimum and can be discarded. Analogously to the usual rules in interval reduction one can apply the rule:

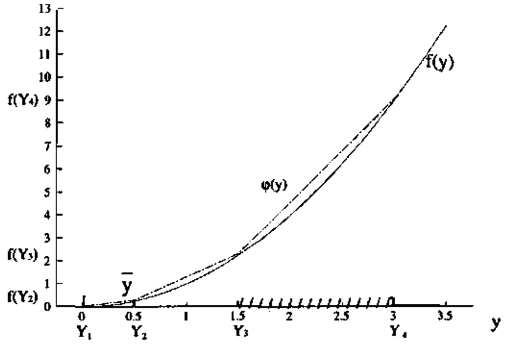

Figure 3.4: Interval reduction in iterative PWLP

If $\bar{y} > Y_2$ then replace $[Y_1,Y_4]$ by $[Y_2,Y_4]$ else
if $\bar{y} < Y_3$ then replace $[Y_1,Y_4]$ by $[Y_1,Y_3]$.

In the next iteration a new point can be added or two new interpolation points can be generated in the interior of the new interval and the procedure can be repeated until y^* has been approximated with a predefined tolerance. In this way, at every iteration the same LP problem is run; only the data differ. The same approach can be taken when the nonlinear function concerns $y \in \mathbb{R}^K$. For every index k, one can take 4 points and follow the procedure such that this requires in total 4^K interpolation points at every iteration. In this way it is not necessary to run a hugh LP problem with many interpolation points, but one can iteratively solve a smaller problem of the same size to approximate the solution of the problem.

Of course one should be certain that this procedure is valid, converges to the optimum. Can the optimum y^* be located in the part of the interval which is thrown away? We will show that the **procedure is correct** when $f(y)$ is a **convex** function on $[Y_1,Y_4]$. Unfortunately, this proof is not very straightforward and requires some additional formalisation and introduction of symbols and definitions. We try to sketch the idea **for the interested reader**.

The formalisation also provides some space to generalise the discussion. The idea of including the nonlinear function $f(y)$ in an LP problem is generalised to including $f(y)$ in a general convex programming problem.
Consider problem P:

$$P: \qquad \min \{ g(x) + f(y) \mid (x,y) \in X \}$$

in which $g(x)$: $\mathbb{R}^n \rightarrow \mathbb{R}$ convex, $f(y)$: $\mathbb{R} \rightarrow \mathbb{R}$ convex and $X \subset \mathbb{R}^{n+1}$ convex. The optimal solution of P is denoted by x^*, y^*. One can associate the minimization of $g(x)$ with a relatively easy to solve problem, such as the LP problem and the function $f(y)$ with a function for which the evaluation is difficult, for instance it requires experiments. One wants to evaluate f as few as possible or, in the PWLP context, we want to generate as few interpolation points as possible.
We assume that P has feasible solutions for all $y \in [Y_1,Y_4]$. The question raised is what will happen when $f(y)$ is approximated by a piecewise linear function $\varphi(y)$. Given are the interpolation points $Y_1 < Y_2 < Y_3 < Y_4$ and the piecewise linear approximation $\varphi(y)$ of $f(y)$ on the interval $[Y_1,Y_4]$; so in general $\varphi(y) = \Sigma\, \lambda_i f(Y_i)$ and $\varphi(y)$ coincides at the interpolation points, $\varphi(Y_i) = f(Y_i)$, $i = 1,2,3,4$.

Problem P is approximated by problem Q:

$$Q: \qquad \min \{ g(x) + \varphi(y) \mid (x,y) \in X \}$$

The optimal solution of Q is denoted by $\bar{x}$, $\bar{y}$.
The idea to be analyzed is to approximate P iteratively by Q and to shrink the initial interval according to the interval reduction methods for functions of one variable.

Interval reduction scheme for iterative piecewise linear programming

```
0.   Given an initial interval $[Y_1,Y_4]$ and a stopping tolerance for the final
     interval.

1.   Generate the interpolation points $Y_2$ and $Y_3$, for instance with the golden
     section rule and construct $\varphi$. Solve $Q$ giving an optimal point $\bar{y}$.

2.   a.   If $\bar{y}$ is an interior point such that $Y_i < \bar{y} < Y_{i+1}$ for a value
          of $i = 1,2,3$ then replace $[Y_1, Y_4]$ by $[Y_i, Y_{i+1}]$.

     b.   If $\bar{y} = Y_1$  replace $[Y_1, Y_4]$ by $[Y_1, Y_2]$.
          If $\bar{y} = Y_2$  replace $[Y_1, Y_4]$ by $[Y_1, Y_3]$.
          If $\bar{y} = Y_3$  replace $[Y_1, Y_4]$ by $[Y_2, Y_4]$.
          If $\bar{y} = Y_4$  replace $[Y_1, Y_4]$ by $[Y_3, Y_4]$.

3.   If the interval $[Y_1, Y_4]$ fulfils the stopping criteria, STOP, else go to 1.
```

An analogous scheme can be formulated for $y \in \mathbb{R}^K$, which implies the use of 4^K interpolation points at every iteration.

The question is whether the scheme is valid; **will it converge to the optimum y^*?** Interval reduction techniques bracket the minimum correctly when there is one minimum on the interval. This is the case when the function to be minimized is convex. Now there are two problems when we relate this to our interval reduction scheme:

i. One has to show that we are minimizing a convex function on the interval $[Y_1, Y_4]$.
ii. One has to show that the minimumpoint $\bar{y}$ found by the approximation is in the same subinterval as minimumpoint y^*. This is not hard for step 2.b, where the two points coincide, but unfortunately not straightforward for step 2.a. The fact that y^* is in the same interval as $\bar{y}$ has to be proven.

i. We start with the convexity question i. The function we are actually minimizing appears when we fix variable y and minimize over variable x. Let us call this function

$$\Psi(y) = \min \{ g(x) + f(y) \mid x \in X(y) \}$$

in which $X(y)$ is set X with a fixed value of y (so it is convex). One assumption told us that for every y in the interval a feasible point x exists. So we have to show that $\Psi(y)$ is a convex function. We first split $\Psi(y)$ in $f(y)$, which was assumed to be convex and the function

$$\pi(y) = \min\{\ g(x) \mid x \in X(y)\ \},$$

so that $\Psi(y) = \pi(y)+f(y)$. Function $\pi(y)$ is a generalisation of the perturbation function which is used in convex analysis, see Bazaraa et al. (1993). For this function the convexity can be proven. The proof and a more precise formulation of the theorem are provided in Appendix 3.A.

Theorem 3.1. (Appendix 3.A) Given all assumptions on problem P, function $\pi(y)$ is convex on $[Y_1, Y_4]$.

As $\pi(y)$ is convex also $\Psi(y) = \pi(y)+f(y)$ is convex for any convex function $f(y)$. This shows that the interval reduction procedure reduces the interval correctly; y^* cannot be situated in the subinterval which is thrown away. This is only valid if in 2.a. y^* can be found in the same interval as $\bar{y}$. This is investigated now.

ii. This needs some elaboration of the properties of the approximation of $f(y)$ by $\varphi(y)$ and consequently of approximating P by Q.
The functions $\Psi(y) = \pi(y)+f(y)$ and $\Xi(y) = \pi(y)+\varphi(y)$ are both convex and coincide at the interpolation points Y_i, $i = 1,2,3,4$. $\Xi(y)$ is a convex majorant of $\Psi(y)$, $\Xi(y)\geq\Psi(y)$ for $y\in[Y_1, Y_4]$. Note that $\Xi(y)$ in contrast to $\varphi(y)$ is not necessarily piecewise linear (with the same interpolation points); this causes $\bar{y}$ to be an interior point in case 2.a. We have to show now that if $\Xi(y)$ has a minimum point $\bar{y}$ in the interior of an interval, then $\Psi(y)$ also has a minimum point y^* in the same interval. This is done by theorem 3.2.

Theorem 3.2.
Given two convex functions $\psi(y)$, $\xi(y) : [a,b]\subset\mathbb{R} \rightarrow \mathbb{R}$, $\xi(y)\geq\psi(y)$ and two points $y_1, y_2 \in [a,b]$ such that $y_1<y_2$, $\psi(y_1) = \xi(y_1)$ and $\psi(y_2) = \xi(y_2)$.
If $\xi(y)$ has a minimumpoint in the interior of $[y_1, y_2]$ then $\psi(y)$ also has a minimum-point $y^* \in [y_1,y_2]$.

The proof can be found in Appendix 3.A.

The theorems 3.1 and 3.2 have answered the questions i. and ii. If $f(x)$ is convex, the procedure as outlined in the interval reduction scheme for iterative piecewise linear programming, is valid in the sense that the minimum of P is not thrown away when iteratively approximated by Q. Convergence can be forced by choosing the interpolation points Y_2 and Y_3 correctly. In this way

> *nonlinear programming problems can sometimes be solved by iteratively running LP models, for which there exist well known routines.*

3.2.4. Piecewise linear programming and nonconvex functions

After the discussion of the relation between linear programming and the minimization of convex functions let us return to the central theme of this work on nonconvex optimization. A variant of Hotellings remark could be: "*We all know the world is not always convex*". The question is how to include nonconvex nonlinear functions in an *LP* problem. The problem is that the approximation of $f(y)$ by $\varphi(y)$ is now frustrated, as not two adjacent interpolation variables are selected. This can be derived from the simple example in figure 1. Consider again the approximation of $f(y) = y^2$ on [0,3] by φ:

$$0.5\,\lambda_2 + 1.5\lambda_3 + 3\lambda_4 = y \qquad (3.6)$$
$$0.25\,\lambda_2 + 2.25\lambda_3 + 9\lambda_4 = \varphi \qquad (3.7)$$
$$\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1,\ \lambda_1,\lambda_2,\lambda_3,\lambda_4 \geq 0. \qquad (3.8)$$

Given a fixed value of $y = 1$, the minimization of φ given equations (6) and (8) (convex optimization) results in the optimal values of $\lambda_2=1/2$ and $\lambda_3=1/2$ and an approximation of $f(1)$ with $\varphi=0.25*0.5+2.25*0.5=1.25$. This approximation is as good as possible given the interpolation points. Nonconvex optimization includes the maximization of a convex function (this is called concave programming) and leads to extreme choices. The corresponding maximization of φ given a fixed value of $y=1$ gives an optimal solution of $\lambda_1=2/3$ and $\lambda_4=1/3$ and an approximation of $f(1)$ with $\varphi=9*1/3=3$. This bad approximation is caused by the choice of extreme and **not adjacent** interpolation points by the *LP* formulation.

As nonconvex optimization leads to multiple optima, one would expect (given the message of this chapter), that there is a relation with integer programming. This was indeed found by Dantzig (1960). In order to force the *LP* formulation to select at most two positive adjacent interpolation points he introduced the well known δ-formulation (see Dantzig, 1960). Binary variables δ_i, $i= 1,\ldots,N-1$ are used to describe whether the solution $\bar{y}$ can be found (yes/no) in the interval $[Y_i,Y_{i+1}]$. By linking the δ_i variables to the λ_i variables only two neighbouring variables λ_i take a positive value when an integer solution of the MILP problem has been found. For the example of figure 1 this requires the addition of the following restrictions to equations (6), (7) and (8):

$$\lambda_1 \leq \delta_1$$
$$\lambda_2 \leq \delta_1 + \delta_2$$
$$\lambda_3 \leq \delta_2 + \delta_3$$
$$\lambda_4 \leq \delta_3$$
$$\delta_1 + \delta_2 + \delta_3 = 1$$

and δ_i binary i=1,..,3.

As only one binary variable δ gets a value of 1, at most two adjacent interpolation variables λ will get a positive value. The equivalence between integer programming and global optimization as introduced is that solving the MILP formulation means looking globally over the feasible set to find the global solution of $\varphi(y)$. The solution method requires a global search over the domain. In Section 3.3, a specific algorithm is outlined and an example of the δ-formulation is given and solved in different ways.

Approximating nonconvex functions of several variables

The extension of the δ-formulation to the approximation of a multidimensional function $f(y)$: $\mathbb{R}^2 \rightarrow \mathbb{R}$ on the box-constrained region $l_k \leq y_k \leq u_k$, $k = 1,2$ is not straightforward. The point is that we first need to define the concept of adjacent points in a multidimensional space.

Let us observe that the optimization of φ under the constraints (3.6) and (3.8) leads to at most two interpolation variables to be positive. This is no coincidence. In *LP* the number of basic variables (potentially nonzero) in the optimum solution equals the number of constraints. If we extend this observation to a two-dimensional space,

$$\Sigma\, \Sigma\, \lambda_{ij}\, Y_{ij} = y \tag{3.9}$$

$$\Sigma\, \Sigma\, \lambda_{ij}\, f(Y_{ij}) = \varphi \tag{3.10}$$

$$\Sigma\, \Sigma\, \lambda_{ij} = 1,\ \lambda_{ij} \geq 0\ , \tag{3.11}$$

the optimization of φ in (3.10), given values for the vector y, leads to an LP problem with three equalities, namely two equalities (3.9) and one in (3.11). The LP solution will select at most three positive interpolation variables. In general, for a K-dimensional space the LP approximation will interpolate the most profitable K+1 points; the best simplex for the optimization.

For the δ-formulation, one should define the concept of adjacent K+1 points. Let us elaborate this in two dimensions (K=2). One possibility, discussed by Williams (1990), adds interpolation variables μ_i for the y_1-axis and variables ν_j for the y_2-axis to (3.9), (3.10) and (3.11) in the following way:

$$\mu_i = \sum_j \lambda_{ij} \qquad i = 1,\dots,N_1$$

$$\nu_j = \sum_i \lambda_{ij} \qquad j = 1,\dots,N_2\ .$$

Those variables can be used to describe the interpolation between the values on the y_1-axis and y_2-axis which correspond with the interpolation grid (figure 3.2). Now the δ-formulation can be applied for the μ_i and ν_j variables separately to force interpolation of adjacent values on both axes.

From a simplicial point of view, one can define directly what are considered adjacent points and **subdivide the interpolation space in simplices**. A binary variable δ can now be connected with each simplex. We use an example to

illustrate the simplicial subdivision and corresponding δ-formulation. Moreover, it shows that it is neither necessary to use a grid nor a rectangular domain of y. A simplicial partition of the domain with a corresponding list of interpolation points is sufficient, though the coding and **bookkeeping** may be cumbersome. We partition the rectangular domain $l_k \le y_k \le u_k$, $k = 1,2$, in four equal simplices using the vertices and barycentre as five interpolation points (two-dimensional vectors).
The vectors

$$Y_1 = \begin{pmatrix} l_1 \\ u_2 \end{pmatrix},\ Y_2 = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix},\ Y_3 = \begin{pmatrix} l_1 \\ l_2 \end{pmatrix},\ Y_4 = \begin{pmatrix} u_1 \\ l_2 \end{pmatrix} \text{ and } Y_5 = \begin{pmatrix} (l_1 + u_1)/2 \\ (l_2 + u_2)/2 \end{pmatrix}$$

define the interpolation vectors and the variables δ_i correspond to the four simplices which only contain three (adjacent) interpolation points. Now the corresponding δ-formulation to approximate a nonlinear nonconvex function on this box is given by:

$$\begin{aligned}
&\Sigma\, \lambda_i\, Y_i = y \\
&\Sigma\, \lambda_i\, f(Y_i) = \varphi \\
&\Sigma\, \lambda_i = 1 \\
&\lambda_1 \le \delta_1 + \delta_2 \\
&\lambda_2 \le \delta_2 + \delta_3 \\
&\lambda_3 \le \delta_1 + \delta_4 \\
&\lambda_4 \le \delta_3 + \delta_4 \\
&\lambda_5 \le \delta_1 + \delta_2 + \delta_3 + \delta_4 \\
&\delta_1 + \delta_2 + \delta_3 + \delta_4 = 1 \\
\\
&\delta_i \text{ binary } i = 1,2,3,4 \\
&\lambda_i \ge 0 \quad i = 1,2,3,4,5.
\end{aligned}$$

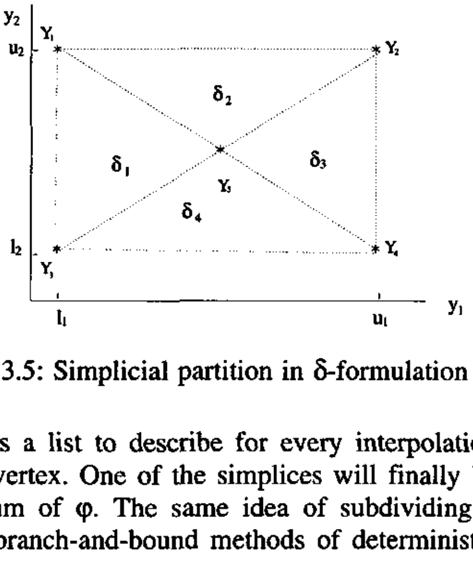

Figure 3.5: Simplicial partition in δ-formulation

The use of a simplicial division requires a list to describe for every interpolation point in which simplices it occurs as a vertex. One of the simplices will finally be selected and contains the global optimum of φ. The same idea of subdividing a domain in subsets can be found in the branch-and-bound methods of deterministic global optimization.

Linear programming is the most applied technique of convex optimization. From the early days, it has been tried to include nonlinear optimization functions in the model formulation. For some types of problems such an approach might be succesful. We have seen that extension to the inclusion of nonconvex funcions is not straightforward. One cannot solve (complex) multi-extremal optimization by simple convex programming. The δ-formulation shows how integer programming can be added in an attempt to solve nonconvex (global) optimization problems. If one could solve integer programming problems in reasonable time, also many types of global optimization problems could be solved easily. The solution approaches of both classes are discussed in Section 3.3.

3.3. *Similarity in algorithms*

The similarity in algorithms between the two fields, combinatorial optimization techniques at one side and global optimization techniques at the other side is interesting. We more or less assume the reader to be familiar with ideas from combinatorial optimization and do not intend to give a summary on either groups but just refer to literature on the subject; the Handbook on Global Optimization (Horst and Pardalos, 1995) and Combinatorial Optimization, Algorithms and Complexity (Papadimitriou and Steiglitz, 1982).

The GLOP methods can be globally divided into two categories (Chapter 1): One group based on deterministic methods (see Horst and Tuy, 1990), *GD*, using as much information on structure as possible and the other group based on stochastic methods and (nonlinear optimization) local searches (see Törn and Žilinskas, 1989), *GL*. The same categories can globally also be found in integer programming.

In combinatorial optimization one group of algorithms is based on enumeration, the name Polyhedral Techniques is sometimes used. For convenience we will call those methods **Integer-Deterministic**, *ID*. Other methods are based on the idea to obtain a good approximation of the optimum in reasonable calculation time using what is called heuristics, random search and local search. Let us name those methods Integer-Local, *IL*.

Table 3.1: Categories of algorithms.

fields	Determ.	Loc. Stoch.
Global Opt.	*GD*	*GL*
Integer Pr.	*ID*	*IL*

IL versus *GL*

Heuristics to construct good but not necessarily optimal solutions have been used for a long time in integer programming. Random search methods have become more popular in combinatorial optimization with the appearance of fast computing power in the last decades. Methods as simulated annealing and genetic algorithms are based on a random generation of many candidate solutions. The idea of local searches to find improvements of feasible solutions (improvement heuristics) also has become popular in integer programming, see Aarts and Lenstra (1997). However, unlike the nonlinear programming problems where there exists a natural way to define a neighbourhood of a solution, for every integer programming problem this concept has to be worked out and is often not unique. Nonlinear programming local searches are based on identifying, by simple function (or gradient) evaluations in the neighbourhood of the current iterate, a promising (descent) direction. For integer programming problems one has to look at the structure of the problem and define neighbouring solutions. For many combinatorial problems, local searches have been defined, see Lawler et al. (1985). The most famous example is the function $f(TP_j) = 10\sqrt{TP_j}$ can for instance be approximated by the line $0.6TP_j + 40$. The fixed charge model includes a binary variable FI_j to model the decision whether (yes/no) a location will be used. The throughput costs are approximated by $\varphi(TP_j) = 0.6TP_j + 40FI_j$ and a constraint $TP_j \le 100\ FI_j$ is added. Calculating the optimal transportation plan with this approximation results in the global optimum of the original problem with costs 230 and $TP_A = 100$ and $TP_B = 0$. the results are so good, because the approximation around the global optimum is correct. In general this optimum is not priori known and one looks for an approximation which fits the original problem as good as possible over the total feasible domain.

This can be done using the δ-formulation over the range [0,100] in which the values of TP_j are situated. Using as interpolation points:

Y_k	0	4	16	36	64	100
$f(Y_k)$	0	20	40	60	80	100

gives the following approximation with the δ-formulation:

$$\min\{\ \Sigma\ \varphi_j + \Sigma\ \Sigma\ c_{ij}x_{ij}\}$$

$$\begin{aligned}
&\Sigma\ x_{ij} = S_i && i = 1,\dots,6 \\
&\Sigma\ x_{ij} = TP_j && j = A,B
\end{aligned}$$

$$\begin{aligned}
&4\lambda_{2j} + 16\lambda_{3j} + 36\lambda_{4j} + 64\lambda_{5j} + 100\lambda_{6j} = TP_j && j = A,B \\
&20\lambda_{2j} + 40\lambda_{3j} + 60\lambda_{4j} + 80\lambda_{5j} + 100\lambda_{6j} = \varphi_j && j = A,B \\
&\lambda_{1j} + \lambda_{2j} + \lambda_{3j} + \lambda_{4j} + \lambda_{5j} + \lambda_{6j} = 1 && j = A,B
\end{aligned}$$

$$\begin{aligned}
&\lambda_{1j} \le \delta_{1j}\ ,\ \lambda_{6j} \le \delta_{5j} && && j = A,B \\
&\lambda_{kj} \le \delta_{k-1\,j} + \delta_{kj} && k = 2,3,4,5 && j = A,B \\
&\delta_{1j} + \delta_{2j} + \delta_{3j} + \delta_{4j} + \delta_{5j} = 1 && && j = A,B \\
&x_{ij}\ ,\ TP_j,\ \lambda_{kj} \ge 0 \quad \delta_{kj}\ \text{binary}
\end{aligned}$$

The LP solution without integrality constraints on the δ variables, which is the LP relaxation in the first node of the branch-and-bound tree, corresponds with the plan of Table 2.2, i.e. $TP_A = 70$ and $TP_B = 30$. The selected positive interpolation variables are typically not adjacent, but taken as the extremes (concave programming), $\lambda_{1A} = 0.3$, $\lambda_{6A} = 0.7$ and $\lambda_{1B} = 0.7$, $\lambda_{6B} = 0.3$ which results in an objective value of 200. The real costs of the plan of Table 2.2 amount to 238.4. After calculating 15 nodes the branch-and-bound procedure reaches the optimum in which $TP_A = 100$ and $TP_B = 0$, so that $\lambda_{6A} = \lambda_{1B} = 1$.

Special ordered sets
We discuss another type of branching here, which was specifically developed for these kind of MILP approximations by Beale (see Beale and Tomlin, 1969). The set of binary variables $\{\delta_1,...,\delta_N\}$ with the constraint $\Sigma\delta_i = 1$ is called a special ordered set of type 1, SOS_1, when there is a certain ordering like in our case by the corresponding interpolation points. Beale suggested to perform the branching not on posing integrality constraints on one branching variable, but to generate subproblems in the following way. One variable (index) δ_r is selected. Two new subproblems, nodes, appear by adding for one node the restriction $\delta_1 +...+ \delta_r = 0$ and for the other one $\delta_{r+1} +...+ \delta_N = 0$. This branching rule defines a branching tree with a depth of approximately ${}^2\log(N)$ and can be applied successfully with various rules for choosing the splitting point (index) in the set.

The ordered set ideas have been implemented in the Sciconic software, which has been used intensively in agricultural research in the Netherlands in the eighties. It also contains the so called SOS_2 concept (Beale and Tomlin, 1969) for piecewise linear programming. This concept does not use the binary variables δ_i of the δ-formulation, but branches directly on the interpolation variables λ_i. As only (at most) two adjacent variables are allowed to be positive, the two new subproblems, nodes, are defined by using a branching variable λ_r and imposing for one of the problems the additional constraint $\lambda_0 +...+ \lambda_{r-1} = 0$ and for the other $\lambda_{r+1} +...+ \lambda_N = 0$. The concept of SOS_2 sets has also been developed for nonconvex functions of several variables (see Beale, 1980) for the case of Figure 3.2. In contrast to the algorithm for functions of one variable, it never has been implemented in Sciconic for several variables.

With this automatic branching procedure and an automatic routine for selecting interpolation points the user only had to provide the one dimensional function and a range for the argument. In this sense the Sciconic software was one of the first packages on nonlinear optimization and on global optimization. Notice that this works for one particular class of functions, i.e. one dimensional functions embedded in an LP environment. Note further that the global optimum of the approximation φ is found and not necessarily that of the original problem. As choosing interpolation points implies a kind of grid search, it is still possible to miss the global optimum.

Algorithms in both fields can be divided in stochastic (local search) and deterministic methods. Stochastic methods using (integer) local searches need specific problem dependent formulations for the neighbourhood concept. In deterministic methods the concepts of cuts and Branch-and-Bound appear in both fields. In fact one of the early implementations (Sciconic) to solve nonconvex problems via piecewise linear programming and Branch-and-Bound, can be called one of the **early global optimization software packages**.

3.4. *A maximum distance problem*

As in Section 3.2, it is shown here how integer programming can sometimes be used to approach global optimization problems with a particular structure. The equivalence with integer programming implies that also the complexity of this class of problems is related, as will be shown. A specific property which is used, is the convexity of a distance function. This causes the maximization of a distance to be a case of concave programming, which (see Section 2.2.) in general leads to multiple optima which are attained at the boundary of the feasible region.

The particular problem which is discussed here, originates from the application of multi-objective programming in land use planning. An example which describes land use options for the European Community can be found in Rabbinge and Van Latesteijn (1992). In this context, variables x_j describe the use of an area in a certain region for a particular crop or land use option. Criteria variables z_k describe the various objectives and are used in a multi-objective framework. It is not necessary to go into detail on the numerous methods for multi-objective programming here, of which the so called Interactive Multiple Goal method is popular in land use modelling. What is important is that finally an optimal (for the decision maker) plan x^* with corresponding objective vector z^* is selected.

The question under discussion originates from robustness and reliability studies on the plans that are generated. One idea is to have a close look on the size or extremes of the set of optimal or 'near optimal' solutions. This is similar to the robustness and level set discussions in Sections 4.5. and 6.3. respectively. An approach could be to have a careful look at the reduced costs of the variables x_j^* which are not taken into the plan. Due to the existence of several objectives this may be rather complicated. Therefore the specific question was derived to look for a plan which is **as different as possible** in the space of the land use plans x, but **similar** in the **objective space** of z. The given plan x^*, z^* is used as a reference plan. The conceptual mathematical formulation is the following.

$$\begin{array}{ll} \max & D_1\,(x,x^*) \\ & D_2\,(z,z^*) \le \varepsilon \\ & Cx = z \\ & Ax \le b \\ & l \le x \le u \end{array}$$

in which matrix C is K to n, A is m to n, b is an m-vector, l and u are given vectors of dimension n and ε is a given scalar. We are dealing with **two distance** functions.

Distance in the restrictions.
The feasible set is convex, as also the distance function D_2 is convex. When the Euclidean distance function is used for D_2, the first inequality is quadratic. In our applications we used the maximum relative difference

$$D_2(z,z^*) = \max_k \frac{|z_k - z_k^*|}{z_k^*},$$

which came closest to the perception of the modeller. Moreover, this can easily be implemented in an *LP* environment.

Distance in the objective.
When the distance function in the objective is taken to be Euclidean, the problem becomes a concave quadratic programming problem which has been extensively discussed in literature, e.g. Horst et al. (1995). Methods to solve such problems are given in Horst and Tuy (1990) and are outlined in Chapter 4. A variant for the distance function which gave the desirable result for our application, was the use of the one-norm, the sum of the absolute deviations from x^*. No weights were used, as all activities in the model have the same dimension (area):

$$D_1\ (x,x^*) = \Sigma\,|\,x_j - x_j^*\,|\,.$$

In general LP, variables d_j^+ and d_j^- can be introduced to describe a positive or negative deviation $x_j - x_j^*$. For the implementation in an MILP environment, binary variables are necessary now. Variable δ_j describes whether a deviation $x_j - x_j^*$ is positive, d_j^+ or negative, d_j^-. In fact the binary variables describe the choice between the local optima of the maximum distance problem. A general MILP formulation is given as follows:

$$\begin{array}{lll}
\max\ D = & \Sigma\, d_j^+ + d_j^- & \\
& x_j^* - x_j + d_j^+ - d_j^- = 0 & j = 1,\dots,n \\
& d_j^- \le (x_j^* - l_j)\,\delta_j & j = 1,\dots,n \\
& d_j^+ \le (u_j - x_j^*)(1-\delta_j) & j = 1,\dots,n \\
& -\varepsilon \le (z_k - z_k^*)/z_k^* \le \varepsilon & k = 1,\dots,K \\
& & \\
& Cx = z & \\
& Ax \le b & \\
& l \le x \le u & \\
& d_j^+,\ d_j^- \ge 0,\quad \delta_j \text{ binary} & j = 1,\dots,n
\end{array}$$

At first sight, the approach requires an high amount, *n*, of binary variables. However, δ_j is not needed when x_j^* is either on its lower bound l_j, or on its upper bound u_j. This applies for most of the variables in an optimal plan.

To illustrate the approach, a small example is presented. Let a multi-objective problem be given as follows.

Multi-objective formulation:

$$\max z_1 = 25 + 5x_1$$
$$\max z_2 = 20 + x_1 + 5x_2$$

$$\begin{aligned} x_1 + x_2 &\le 8 \\ -x_1 + x_2 &\le 2 \\ 0 \le x_1 &\le 5 \\ 0 \le x_2 &\le 4 \end{aligned}$$

In the applications the problem stems from, the two objectives would have an economic and environmental interpretation. Those objectives are in general conflicting. In the example some constants are present in the objective functions, which is in general also the case in the applications. Let us assume that the decision maker selected as an optimal plan $x^* = (5,3)$ with corresponding objective values $z^* = (50,40)$.

Now the corresponding maximum distance problem can be formulated. Notice first that x_j^* attains a value at its upper bound. Only for the second index a binary variable will be introduced.

Corresponding maximum distance formulation:

$$\begin{aligned} \max\quad & D = d_1^- + d_2^+ + d_2^- \\ & 5 - x_1 - d_1^- = 0 \\ & 3 - x_2 + d_2^+ - d_2^- = 0 \\ & d_2^- \le 3\,\delta_2 \\ & d_2^+ \le (1-\delta_2) \\ & z_1 - 50 \le 50\,\varepsilon \\ & 50 - z_1 \le 50\,\varepsilon \\ & 40 - z_2 \le 40\,\varepsilon \\ & z_2 - 40 \le 40\,\varepsilon \\ & z_1 - 25 + 5x_1 \\ & z_2 = 20 + x_1 + 5x_2 \\ & x_1 + x_2 \le 8 \\ & -x_1 + x_2 \le 2 \\ & 0 \le x_1 \le 5 \\ & 0 \le x_2 \le 4 \\ & d_1^-, d_2^-, d_2^+ \ge 0, \quad \delta_2 \text{ binary} \end{aligned}$$

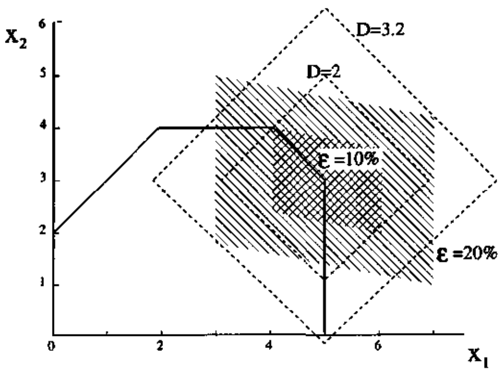

Figure 3.6: Maximizing the distance from x=(5,3)

The formulation given here contains quite some redundant constraints, which can be left out to improve the efficiency when solving a realistic case. For illustrative purposes the 'most distant' plan $x(\varepsilon)$ is calculated for several values of ε. In Figure

3.6 the so called ε-level set of the second distance function is depicted. The results can be found in Table 3.1.

Table 3.1: Solutions of the maximum distance problem for varying values of ε.

ε	z_1	z_2	x_1	x_2	D	δ_2
0%	50.0	40.0	5.0	3.0	0.0	0/1
5%	47.5	42.0	4.5	3.5	1.0	0
10%	45.0	44.0	4.0	4.0	2.0	0
15%	42.5	43.5	3.5	4.0	2.5	0
20%	40.0	32.0	3.0	1.8	3.2	1

The binary variable δ_2 shows here the choice between two local optima of the maximum distance problem. When the number of binary variables increases, there may be very many corresponding optima and thus the solution process by branch-and-bound may require a long time. In this problem the modeller cannot oversee all local optima and in general there is no guess where the global optimum is located or a guess of a bound on the objective function. For the illustration of the complexity a worst case example of the maximum distance problem can be found in literature.
Consider the problem

$$\max \quad f(x) = \Sigma\,(x_i - \varepsilon)^2$$

$$-1 \le x_i \le 1 \qquad i = 1,\ldots,n.$$

For a given small value of ε, this problem not only has 2^n local optima but also a multiple of Kuhn-Tucker points. However, as already outlined in the introduction of Section 3.1, the complexity result not necessarily tells us something about the solvability of practical cases. For real instances of the land use model with thousands of variables, and consequently implying hundreds of binary variables for the corresponding maximum distance problem, it was possible to generate 'near optimal' solutions very distant from the reference plan by the suggested approach.

3.5. *A minimum volume hyperrectangle problem*

In this section a particular problem is discussed which illustrates again the complexity link between global optimization and combinatorial optimization as discussed in Section 3.1. Unlike the former sections it does not provide a suggestion to use techniques from integer programming in global optimization.

The particular problem is a mathematical puzzle, with practical relevance originating among others from parameter bounding problems as outlined by Walter and Piet-Lahanier (1990) and with particular relevance to the algorithm suggested by Keesman (1990). The complexity of finding an optimal solution contrasts with the simplicity of its formulation. Given is a set of points $X = \{x_1,...,x_K\} \subset \mathbb{R}^n$. The question is to find an enclosing hyperrectangle with minimum volume around the points of which the axes are free to be chosen (Keesman, 1992).

Mathematically this can be translated into finding an orthonormal matrix $(a_1,...,a_n)$ such that the objective

$$\prod_i \{v_i = (\max_j a_i'x_j - \min_j a_i'x_j)\}$$

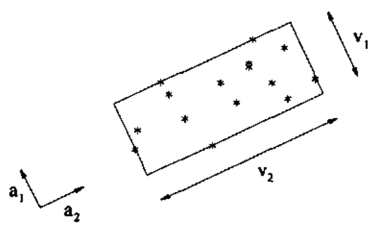

Figure 3.7: The minimum volume hyperrectangle problem

is minimized. Notice first that only the points in the convex hull of X are of interest. The others can be left out. However, this observation does not reduce the complexity of the problem. Notice further that if the corresponding polytope defined by the convex hull of X is not full dimensional (the points are situated in a lower dimensional plane), the optimal objective value is zero as one of the v_i takes a value of zero.

A problem similar to the minimum volume hyperrectangle problem, is to find an enclosing or an inscribed ellipsoid, as discussed for example by Khachiyan and Todd (1993). The enclosing minimum volume ellipsoid problem can be formulated as finding a positive definite matrix and a centre of the ellipsoid, such that it contains a given set of points or a polytope. A problem which is easier from a complexity point of view is to calculate a minimal sphere around a polytope or given set of points, e.g. Konno et al. (1994) or Botkin and Turova-Botkina (1992). The topic on finding an inscribed sphere in a polytope can be found in Chapter 4.

From a nonlinear (non-differentiable) point of view, the minimal enclosing sphere problem can be formulated as finding the central vector c, such that the sphere enclosing a set of points X is minimal:

$$\min_c \max_j \| c - x_j \|.$$

The corresponding centre c is called the Chebychev centre of the corresponding polyhedron. It can be shown that the optimum c must be a convex combination of a subset of affine independent points of X (see Botkin and Botkina-Turova, 1992) which are so called active with respect to c. The corresponding points are called active, because they are situated at the largest distance from c.

With a combinatorial view, one can construct an algorithm which generates all affine independent subsets of X, takes c as the average of those points and checks the points in the subset to be active in comparing the distance of the other points from c. This combinatorial view on the "min–max" problem does not imply a large complexity of the minimal sphere problem. Analogously, the solution of LP problems does not require the generation of all basic solutions. Botkin formulated an algorithm analogous to the simplex method, which pushes the centre c in the right (descent) direction of the current active points until it is in the convex hull of the active points and thus is optimal.

A similar approach will be shown here for a part of the minimum volume hyperrectangle problem. However, in contrast to the minimal sphere problem, the approach ends up in local optima. Let us first illustrate the multi-extremal character of the objective function Πv_i. The requirement of the orthonormality of the matrix of axes of the hyperrectangle, implies the degree of freedom in choosing the matrix to be $n(n-1)/2$. In two dimensions this can be illustrated by using as one parameter the angle of the first vector. The multi-extremal character of the objective for an instance with only four points is shown in Figures 3.8 and 3.9. In the corresponding simple instance the set $X = \{(2,3), (4,4), (4,2), (6,2)\}$ is enclosed by rectangles defined bij the angle α of the first axis a_1. This means that vector $a_1 = (\cos\alpha, \sin\alpha)$. Notice that case $\alpha=0$ represents the same situation as $\alpha=90$, because the position of

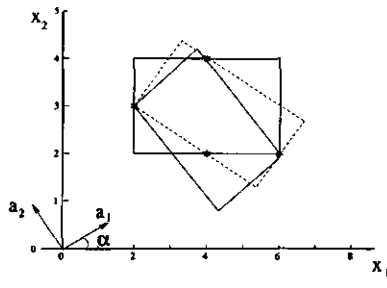

Figure 3.8: Rectangles enclosing a set of points

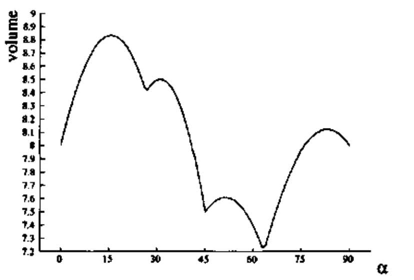

Figure 3.9: Volume of the rectangle as a function of angle α of axis a_1

the two axes switches. This phenomenon has been discussed in Section 2.6.

In a bad case instance for example when an increasing number of points is scattered over the boundary of an ellipse, the number of local optima in the parameter space increases correspondingly. For a further analysis, the problem is considered now from the viewpoint of an iterative procedure.

n-steps procedure

```
0.  $i = 1$

1.  $v_i = \min_{\|a_i\| = 1} (\max_j a_i' x_j - \min_j a_i' x_j)$

    in which $a_i$ is bound to be on the orthoplement of the formerly
    generated vectors $a_1,\ldots,a_{i-1}$.

2.  if $i < n$ then $i := i + 1$ and go to 1

3.  calculate the objective $\Pi v_i$.
```

The subproblem in step 1 is already multi-extremal as can also be derived from the example in Figures 3.8 and 3.9. In Figure 3.10. the objective function of the first subproblem has been depicted for the small example. It can be shown that the optimal values of v_i are nondecreasing in the iterations.

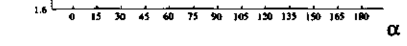

Figure 3.10: Objective v_1 of the first subproblem as function of angle α

To simplify the analysis we will focus on the subproblem when $i = 1$. So the vector a with $\|a\| = 1$ should be found which defines the planes $a'x = b^+$ and $a'x = b^-$ with $b^+ = \max_j a'x_j$ and $b^- = \min_j a'x_j$ such that $v = b^+ - b^-$ is minimized i.e. the planes are as close as possible. The subproblem can be seen as an orthogonal regression in which the infinite norm is used as a criterion.

Analogous to the minimal sphere problem a set of active points is defined, which are now divided over the "upper plane" and "lower plane". Let set I^+ contain the (indices of) the subset of X such that $a'x_j = b^+$ for $j \in I^+$ and let I^- be defined correspondingly. Notice that by considering the vector $-a$, the position of upper and lower plane switches, but the same situation is represented.

It will be shown that for a local optimal vector a at least $n + 1$ affine independent vectors x_j are active. From a combinatorial point of view one can select $n + 1$ (affine independent) points out of the (convex hull of the) K points in set X and divide the points in 2^n-1 ways into two groups I^+ and I^-. Now vector a and the upper and lower plane can be constructed. If the two planes sandwich all other points we have a local optimum with an objective value of b^+-b^-. By enumerating all selections and divisions, requiring an enormous amount of calculation time, one can select the best of the local optima to be the global optimum. In contrast to the minimal sphere problem, there are several selections of $n + 1$ points (and divisions in two subsets) which correspond to an enclosure of the points and thus to a local optimum in the parameter space of a.

In order to discuss the construction of the planes and the active status of $n + 1$ points, the sets B^+ and B^- are introduced. Given a vector a and the corresponding sets I^+ and I^-, set B^+ is defined as the linear space spanned by the vectors $x_j - x_1^+$ in which $j \in I^+$ and x_1^+ the first vector (corresponding to the first index) in I^+. Let B^- analogously be defined. So vector a is orthogonal to B^+ and B^-. When $|I^+| + |I^-| = n + 1$, so that $\dim(B^+) + \dim(B^-) = n - 1$, vector a is defined uniquely and, as will be discussed, local optimal with respect to the subproblem. When $|I^+| + |I^-| \leq n + 1$, the objective can be improved by changing vector a. In this way a local optimization procedure is defined. For an arbitrary starting vector a, it may be valid that $|I^+| + |I^-| = 1$ and the corresponding sets B^+ and B^- are empty. The procedure stops when $\dim(B^+) + \dim(B^-) = n - 1$ and there are $n + 1$ affine independent active points. For the objective function

$$v(a) = \max_j a'x_j - \min_j a'x_j$$

a descent direction d is constructed such that vector $a + \lambda d$ remains orthogonal to B^+ and B^-.

Let vector q be defines as $x_k - x_l$, $k \in I^+$ and $l \in I^-$, so that $v(a) = a'q/\sqrt{a'a}$, which is also valid when a does not have a unitary length. By taking the steepest descent direction

$$d = \frac{1}{\sqrt{a'a}}\left(\frac{a'q}{a'a}a - q\right)$$

we have a direction in which the objective decreases. Now d should be projected on the orthoplements of B^+ and B^- so that the active set stays active. If there are already $n + 1$ active points, the projection reduces d to the zero vector, because d is already orthogonal to vector a. Going in the direction of d reduces the objective function in the following way:

$$v(a + \lambda d) = \frac{a'q + \lambda q'd}{\sqrt{1 + \lambda^2}},$$

which is monotonously decreasing in λ as $a'q$ is positive and $q'd$ is negative. Now λ can be increased until one of the inactive points x_j ($a'x_l < a'x_j < a'x_k$) becomes active. When $d'(x_k - x_j) < 0$ then x_j comes into the "upper plane", this means j comes into I^+, for that values of λ_j of λ such that

$$a'(x_k - x_j) + \lambda_j d'(x_k - x_j) = 0$$

so

$$\lambda_j = - \frac{a'(x_k - x_j)}{d'(x_k - x_j)}.$$

Analogously x_j hits the lower plane when $d'(x_k - x_j) > 0$ for

$$\lambda_j = - \frac{a'(x_l - x_j)}{d'(x_l - x_j)}.$$

By determining all λ_j and taking the smallest value the next point to become active can be identified and the procedure can be repeated until $n + 1$ affine independent active points have been determined.

This procedure defines a local search which arrives at a local optimum of the subproblem given a starting vector. As starting vectors for instance eigenvectors of the dispersion matrix QQ', in which $Q = [x_1,...,x_K]$, can be used (Keesman, 1992). A laborious algorithm has been outlined here to create a local optimum for the minimum volume hyper-rectangle problem. The problem illustrates the relation between difficult "unbeaten" global optimization problems and the combinatorial difficulty of choosing the right active points.

3.6. *An investment problem in nature conservation*

In this section a management problem is discussed, which illustrates how a combinatorial problem when it is analysed by a continuous global optimization view, reveals a high multi-extremal structure. Moreover, as pointed out in Section 3.3., it shows how ideas from global optimization algorithms, based on random and local search, can be used in an integer programming environment.

The background of the decision problem originates from government managers which take decisions on investing funds for the support of rare species in nature conservation. Species such as the badger in the Netherlands are considered that live on certain spots, so called patches. Because the animal travels, migrates from one patch to another there are dangers when crossing a road and encountering a barrier such as creaks, which it cannot cross. To improve the living condition of the total population which lives in a patchy habitat, investments can be done in "infrastructure" such as tunnels and bridges which decreases the death rate of the animals.

The question to environmental scientists is to support these decisions with the aid of models. One type of population dynamics model is considered here, which is based on Adler and Neurnberger (1994). First the model is presented followed by an analysis of various decision problems which can be derived. In the population model K is the number of patches. The reproduction and migration (number of animals leaving the patch in one time period) is assumed to be proportional to the number of animals living in a patch with a reproduction rate r and a migration rate d respectively.

The model can be written as:

$$N_{i,t+1} = rN_{i,t} + (1 - d)N_{i,t} + d\sum_{j\neq i}^{K}\lambda_{ij}N_{j,t}$$

in which

$N_{i,t}$: number of animals living in patch i at period t,
λ_{ij}: so called connectivity between i and j.

The connectivity is defined as the fraction of the animals migrating from patch i, which survive and arrive at patch j, so that $\Sigma\lambda_{ij}\leq 1$. We will assume that $\lambda_{ij} = \lambda_{ji}$. The population growth is the largest eigenvalue of a matrix with $r-d+1$ on the diagonal and the connectivity λ_{ij} as other elements. So the eigenvalue is that of the connectivity matrix Λ (zeros on the diagonal) plus $r-d+1$. The investments have the purpose to improve the connectivity. As $r-d+1$ is not influenced by the investment, the largest eigenvalue of Λ can be taken as a criterion to judge a particular investment plan. It has been shown by Adler and Neurnberger (1994) by numerical experiments with a large simulation model that the maximum eigenvalue is a good criterion to describe the growth of the population. Moreover they show that the maximum eigenvalue can be approximated by a parameter μ. To facilitate the definition of this parameter, the so called immigration potential S_i is introduced:

$$S_i = \sum_j \lambda_{ij}.$$

Now μ is defined as

$$\mu = \frac{\Sigma S_i^2}{\Sigma S_i} = \frac{\Sigma(\Sigma\lambda_{ij})^2}{\Sigma\Sigma\lambda_{ij}}.$$

The investment problem with criterion μ is analyzed here from a global optimization viewpoint and a combinatorial viewpoint. First the relation between investment and connectivity is taken from a **continuous (GLOP) point of view.** Let x_{ij} be the amount of the money invested in connectivity λ_{ij}. The function f_{ij}, describing the relation between connectivity λ_{ij} and investment x_{ij} is an increasing function and in a practical environment may be hard to estimate. The total amount of money which can be spend for an investment plan is assumed to be constrained by a budget. The continuous problem (*P*) is now given by:

$$\max \mu\ (\Lambda)$$

$$\begin{array}{lll} \lambda_{ij} = f_{ij}(x_{ij}) & i = 1, ..., j-1 \;\; j = 2, ..., K & \\ \lambda_{ji} = \lambda_{ij} & i,j = 1, ..., K & \text{(P)} \\ \sum_{j=2}^{K}\sum_{i=1}^{j-1} x_{ij} \le \text{Budget}. & & \end{array}$$

First, according to the ideas of Section 2.2, the problem is analyzed to reveal an underlying structure. As can be observed, the objectfunction is fractional in λ_{ij}. The denominator of the ratio is linear, whereas the numerator is convex in λ_{ij}. This gives the tendency of the objective function to be **multi-extremal** (see Schaible, 1995). When the functions f_{ij} are taken affine with equal derivative:

$$f_{ij}(x_{ij}) = \lambda_{ij}^0 + ax_{ij},$$

this can easily be seen. The increase of the denominator of μ by an investment plan is now fixed (full use of the budget), leaving a convex quadratic objective function.

As a consequence (section 2.2) the optimal plan will be an extreme point of the feasible set. This means that it is not a good idea to base algorithms for finding the global optimum on derivative information, as is usually done in nonlinear programming. The partial derivatives of the objective are given by:

$$\frac{\partial\mu}{\partial x_{kl}} = \frac{\partial\mu}{\partial\lambda_{kl}}\frac{\partial\lambda_{kl}}{\partial x_{kl}}$$

where

$$\frac{\partial\mu}{\partial\lambda_{kl}} = 2\frac{S_l + S_k}{\Sigma S_i} - 2\frac{\Sigma S_i^2}{(\Sigma S_i)^2}$$

in which $\lambda_{kl} = \lambda_{lk}$ is accounted for.

Marginally seen, additional investments best take place in a connection between patches which already have a large immigration potential S_i. However, the pattern of "the best connected" patches changes completely when first a large investment is done around a poorly connected patch. In this case a derivative driven algorithm will run to another local optimum. This means that nonlinear programming local searches will return **different local optima** depending on the starting point as illustrated in the simple example in Section 2.3. In a more practical context, this effect is mitigated slightly, because the functions f_{ij} will be concave.

Deterministic approaches in global optimization (Horst and Tuy, 1990), as outlined for some practical problems in Chapter 4, can be developed for the investment problem. In fact the investment problem is very similar to the quadratic assignment problem (section 3.1) in complexity and from the combinatorial optimization point of view.

Integer point of view

The investments usually take place in an integer mode; a number of tunnels and bridges is constructed which are selected from a group of candidate projects. In general one project influences various connectivities. It is assumed here that x_{ij} represents the number of projects to improve λ_{ij} only, to simplify the analysis. So the decision variables in problem (P) are restricted to take integer values. The budget restriction is translated such that a predefined number n of projects will be executed. When investments can take place in all $N = K(K-1)/2$ trajects simply all $\binom{N+n-1}{n}$ feasible investment plans can be enumerated and evaluated to select the best one. Also in this context it might be possible to construct Branch-and-Bound algorithms to speed up the enumeration.

For a similar problem originating from design of experiments Rasch et al. (1997) discuss various algorithms. In the design problem n measurement points have to be selected out of N candidate points. The bounding in a Branch-and-Bound context can simply be based on a monotonicity observation: more measurements, experiments improves the statistical criterion. Notice that for the investment problem this may not be true (see derivative). There may be investments which, when they are added to the plan, lead to a worse objective function value.

As pointed out in Section 3.3., approaches of combinatorial optimization techniques can be applied now which are similar to ideas from global optimization methods which are based on local searches and random search (Chapter 5). For many combinatorial problems, local searches have been defined, see Lawler et al. (1985) and Aarts and Lenstra (1997). The most famous example is the travelling salesman problem. A given tour may be changed by exchanging two (or more) cities in the tour. The exchanging operation defines an environment in mathematical sense of a feasible solution and therefore also gives an interpretation of the terminology of local search and local optimum; a local optimum is a feasible solution which cannot be improved by performing the corresponding exchange operation. The same ideas can be applied for the investment problem. Some algorithms were implemented and tested on a small case study in the south of The Netherlands for a limited area where the badger lives.

Possible structure of a random search algorithm

```
Given $n$: number of investments in the plan
        $N$: $K(K-1)/2$: number of connections which can be improved
        (in a practical case less connections will be distinguished)
        $L$: number of plans to be generated

For $q$:= 1 to $L$ do
        set all $x_{ij} = 0$
        For $p$:= 1 to $n$ do
                generate one of the connections $i,j$
                $x_{ij}: = x_{ij} + 1$
        endo
        evaluate the generated plan
endo
```

Notice that not all feasible investment plans have the same probability to be generated in this way. Plans with high investments in one connection have a higher probability of occurring than plans with a high diversification. A simple experiment with test instances comparing the frequency distribution of the objective value of the random generated set of plans and the distribution of the plans generated by enumeration (every plan occurs once) gives a large similarity. Apparently the bias does not affect the objective space.

In practical decision support a good heuristic is to present the best plans which are generated, let say the top ten, to the decision maker. The best plans can also be used as starting points for a local search. There are various ways of constructing a local search. One possibility is outlined here.

Local search outline

```
Give a starting plan $x$ (in which some $x_{ij} > 0$ and some $x_{ij} = 0$).
Repeat
  -   perform all possible exchanges by lowering one $x_{ij}$ with $x_{ij} > 0$ by
      one unit and increasing another. Evaluate the resulting plan.
  -   If the criterion value of the best exchange plan is better than the
      current value, perform the exchange.
Until no improvement can be found.
```

By performing a simple random search and local searches as outlined here, the optimal solution of the test cases could be found in much less calculation time than by plain enumeration of all feasible plans. This illustrates how ideas of random search, local search and multistart (Chapter 5), which have been successfully applied in global optimization can be applied for complex combinatorial decision problems. The drawback of the random search based methods is of course that there is no guarantee that the global optimum is reached. On the other hand, a group of different local optima can be generated which provide the decision maker with several alternative "good" plans. Moreover, the explicit or implicit (Branch-and-Bound) enumeration may for practical instances require an unrealistic amount of calculation time and computer memory as will be discussed in Chapter 4.

3.7. *Concluding remarks*

We have seen four aspects on the relation between combinatorial optimization and global optimization. First of all, the exercises with the Quadratic Assignment Problem have shown how an integer problem can be translated to a global optimization problem. This means that the theoretical **complexity** is related between the two fields. If general purpose algorithms in global optimization would appear which can solve problems in polynomial time (in the dimension of the problem), one would be able to solve problems from integer programming which are known to be hard to solve. This indicates it will be impossible to construct such algorithms.

Secondly, the application of piecewise linear programming for nonconvex programming and the maximum distance problem have shown, that despite the theoretical complexity relation, **practical** problems may be **solved** by an integer programming technique.

A third aspect is the similarity in **classification** of the algorithms. Algorithms in both fields can be divided in stochastic and deterministic methods. Stochastic methods using (integer) local searches need specific problem dependent formulations for the neighbourhood concept. It has been shown by the investment problem for infrastructure in nature conservation, how such a neighbourhood can be defined and how corresponding algorithms can be used based on the idea of local searches. In deterministic methods the concepts of cuts and Branch-and-Bound appear in both fields. In fact one of the early implementations (Sciconic) to solve nonconvex problems via piecewise linear programming and Branch-and-Bound, can be called one of the early global optimization software packages.

The fourth aspect is that we can **analyze** an integer problem with the eyes of deterministic global optimization. The investment problem in Section 3.6. showed how a gradient way of thinking, add the investment which is most profitable (add-heuristic), will lead to local optima. The underlying maximization of a convex objective was the apparent reason. The same appeared in the analysis of the Quadratic Assignment Problem. Given negative interaction coefficients, the problem tends to concave maximization, so that nor the continuous variant, not the LP relaxation of the MILP formulation will result in integer solutions; the 'cross your fingers' strategy will not work. In this way analysis of the problem (using the structure of the problem) can help the same way in integer as in global optimization for the selection of algorithms.

Appendix 3.A.

Theorem 3.1. Given problem P: min $\{ g(x) + f(y) \mid (x,y) \in X \}$ with $g(x)$: $\mathbb{R}^n \to \mathbb{R}$ convex, $f(y)$: $[a,b] \subset \mathbb{R} \to \mathbb{R}$ convex, $X \subset \mathbb{R}^{n+1}$ convex and let P have feasible solutions for all values of $y \in [a,b]$.
Function $\pi(y) = \min\{ g(x) \mid x \in X(y) \}$, with $X(y)$ set X when the value of y is fixed, is convex on $[a,b]$.

Proof
Given any two points $y_1, y_2 \in [a,b]$ and corresponding vectors $x_1 \in X(y_1)$ and $x_2 \in X(y_2)$ such that $\pi(y_1) = g(x_1)$ and $\pi(y_2) = g(x_2)$.
For any value of $\lambda \in (0,1)$ applies: $\lambda(x_1, y_1) + (1-\lambda)(x_2, y_2) \in X$ so that $\lambda x_1 + (1-\lambda)x_2$ is a feasible point of $X(\lambda y_1 + (1-\lambda)y_2)$ with a corresponding function value of $g(\lambda x_1 + (1-\lambda)x_2)$.
In this way $\lambda\pi(y_1) + (1-\lambda)\pi(y_2) = \lambda g(x_1) + (1-\lambda)g(x_2) \geq g(\lambda x_1+(1-\lambda)x_2) \geq$ $\min\{g(x) \mid x \in X(\lambda y_1+(1-\lambda)y_2)\} = \pi(\lambda y_1 + (1-\lambda)y_2)$.
So π is convex on $[a,b]$. □

Theorem 3.2.
Given two convex functions $\psi(y)$, $\xi(y)$: $[a,b] \subset \mathbb{R} \to \mathbb{R}$, $\xi(y) \geq \psi(y)$ and two points $y_1, y_2 \in [a,b]$ such that $y_1 < y_2$, $\psi(y_1) = \xi(y_1)$ and $\psi(y_2) = \xi(y_2)$.
If $\xi(y)$ has a minimumpoint in the interior of $[y_1, y_2]$ then $\psi(y)$ also has a minimum-point $y^* \in [y_1, y_2]$.

Proof
Point $\bar{y}$ is a minimumpoint, so that $\xi(\bar{y}) \leq \xi(y_1) = \psi(y_1)$.
Suppose ψ has no minimumpoint in the interval $[y_1, y_2]$ so that without loss of generality it can be assumed that ψ has a minimumpoint $y^* < y_1$ and $\psi(y^*) < \psi(y_1)$.
Then there exists a $\lambda \in (0,1)$ so that $y_1 = \lambda y^* + (1-\lambda)\bar{y}$.
Given that $\psi(\bar{y}) \leq \xi(\bar{y}) \leq \xi(y_1) = \psi(y_1)$ and by convexity of ψ this implies that $\psi(y_1) = \psi(\lambda y^* + (1-\lambda)\bar{y}) \leq \lambda\psi(y^*) + (1-\lambda)\psi(\bar{y}) < \lambda\psi(y_1) + (1-\lambda)\psi(y_1) = \psi(y_1)$.
This shows that $y^* < y_1$ cannot be valid. The same reasoning applies for $y^* > y_2$, so that at least one minimumpoint y^* of ψ should be located in the interval $[y_1, y_2]$. □

Chapter 4. *Analyzing models and developing specific deterministic solution procedures: some cases*

4.1. *Introduction*

In Chapter 2 it has been discussed which mathematical structures are useful for the development and choice of specific deterministic global optimization methods. Furthermore, it has been shown with practical examples how mathematical structures can be recognised and how they explain the occurrence of multiple optima. In Chapter 3, one step further has been worked out, namely actually solving problems exploiting their structure. This was done by applying integer programming techniques which are similar to deterministic global optimization methods.
In this chapter some larger cases are worked out. We follow the complete route from formulation of models (arrow a), analysis of their mathematical structure (arrows c and e) and the construction of specific deterministic global optimization methods (arrow f).
We start with the discussion of the so called nutrient problem which is similar to the pooling problem discussed in Section 2.4. This is a high dimensional problem close to linear programming, but hard to solve. In this chapter we further focus on the (quadratic) design problem which is in general not larger than ten dimensional. Although multi-extremal, deterministic methods may reach the level of being implemented in a Decision Support System for this problem, as will be shown.

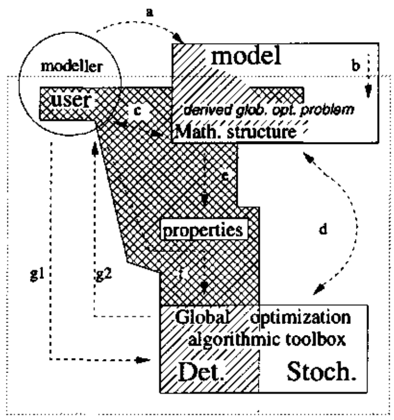

Figure 4.1: Route for applying deterministic methods

For all cases discussed in this chapter, traditional optimization tools, may not give the answer, as will be illustrated. Deterministic global optimization methods guarantee the solution, but require quite some effort in analyzing the problems and constructing specific algorithms. A first step is the recognition (arrow c) of the special structure of the problem, which explains its multi-extremal behaviour. We will start with this topic in Sections 4.2. and 4.3. In Section 4.4. specific deterministic global optimization algorithms are developed. As the reader may be less familiar with the concept of branch-and-bound procedures for global optimization, illustrations are elaborated to give a flavour of the method. In Section 4.5. finally, a more detailed discussion takes place on the topic of finding robust solutions.

4.2. *The nutrient problem*

4.2.1. Introduction

In Section 2.4 the pooling problem has been discussed. It has been shown how including joint storage in the classical linear blending problem leads to bilinearity in the model formulation. The general similarity in bilinear models, is the appearance of flows which are combined and the description of concentrations which is included in the model. When there is a chain of activities on flows (quantity) of products, such as happens in environmental planning, the modelling of polluting material (quality) may lead to balance equations which are bilinear.

One of the simplest examples is to consider a dynamic sequence of decisions such as in reservoir management. We consider a simple reservoir system with given inflow f_t in period t and decision variables x_t and I_t (state variable) describing the the release and storage of water respectively in period t. A simple mass balance equation describes that the amount of water stored at the end of a period equals the amount of water at the beginning plus the inflow minus the release in that period:

$$I_t = I_{t-1} + f_t - x_t, \tag{4.1}$$

which is linear and can be applied in a Dynamic Programming context. When the model is extended to describe the quality of the water with a concentration (of salt for example), bilinear balances are included in the mathematical structure. Let φ_t be the given concentration of inflow f_t and let variable q_t describe the concentration in the reservoir. Again a balance equation describes how the amount of salt at the end of the period equals the amount at the beginning minus the outflow of salt plus the inflow. The new concentration q_t is determined by:

$$q_t I_t = q_{t-1}(I_{t-1} - x_t) + \varphi_t f_t\,. \tag{4.2}$$

The resulting concentration q_t depends directly on the release decisions x_t. The classical linear blending problem uses concentrations and is not bilinear. So it is not the modelling of concentrations alone, but also the additional description of consecutive decisions on flows of materials which are combined, causing bilinearity in environmental models.

The remainder of this paragraph will be devoted to the nutrient problem which is an extension of another classical model, namely the farm management problem. The discussion is based on Bloemhof-Ruwaard and Hendrix (1993, 1996). First the formulation of the model is discussed. Then the mathematical structure is thoroughly analyzed by comparing the model with the pooling problem, considering the problem as a bilinear problem, examining its multimodal structure and analyzing the boundary characteristic of the solution. Furthermore, it is shown that classical nonlinear programming heuristics may not lead to the global optimum. In Section 4.4. a specific branch-and-bound method will be developed, based on the mathematical structure revealed here.

4.2.2. Model formulation

A popular method in farm management and land use planning is linear programming (LP). It has shown to be very useful in finding a profit maximizing combination of farm activities that is feasible with respect to a set of fixed farm constraints. Inputs for a dairy farm are fertilizer, labour, fodder etc., and saleable outputs are milk, maize or meat. Dairy cows do not only deliver milk, but also produce a large amount of manure that has to be disposed of at some cost. However, this manure (containing useful nutrients) can also be applied for fertilizing the meadow, replacing expensive inorganic fertilizers. Nowadays, the treatment of manure for fertilizing land is an important political issue and a subject of many scientific studies in the Netherlands. The government regulates the use of manure for fertilizing by setting standards per hectare for each type of land use. The amount of manure to be used for this purpose depends on the land use, which is a decision variable in the model.

Linear farm management models can be used to determine a profit maximizing combination of farm activities that is feasible with respect to a set of fixed farm constraints (Hazell and Norton 1986, Williams 1990). The variable x_i represents the level of activity i, denoted by the number of hectares. The set of activities $i = 1, ..., n$ (also the index j will be used) defines the possibilities in the farm organisation, such as the amount of cattle treated and fed in a particular way. The vector x gives the combination of activities chosen to be the farm plan. Each activity has a profit γ_i (sales minus input costs) and some constraint parameters such as cattle density, amount of labour needed, milk production per hectare, and amount of fodder needed. The constraint parameters build the matrix A, also known as the technology set of the farm model. The set of feasible farm plans is determined by the vector of bounds b, representing e.g. total land, available labour and allowable milk production (European milk quota).
The traditional farm management model formulation is:

$$\begin{aligned} \max \quad & \gamma' x \\ & Ax \leq b \\ & x \geq 0 \,. \end{aligned}$$

The extended model covers on the one hand decisions on the farm plan with corresponding amounts of manure produced and demands for fertilizing. On the other hand, it covers decisions on the use of manure for fertilizing purposes. Choices in the farm plan determine the possibilities for the application of manure. Figure 4.2 outlines the situation.

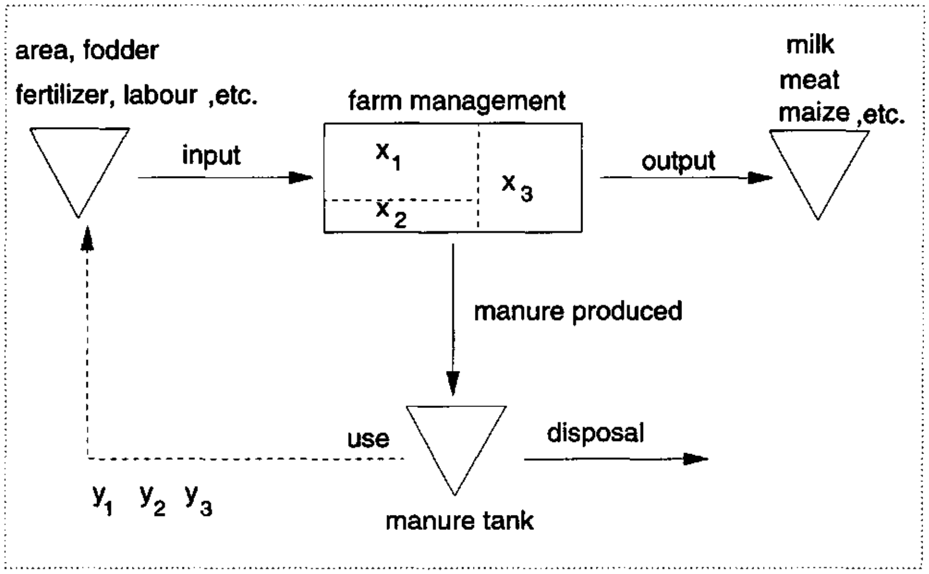

Figure 4.2: Schematic view on the nutrient problem

We assume that all manure of the cattle arrives in one manure tank. The manure in the tank can be used to fertilize the land thus reducing the amount of inorganic fertilizer that has to be bought and applied on the meadow. Furthermore, the re-use of manure saves disposal costs. Disposal costs arise due to manure excess rules in the intensive livestock sector. The variable y_i is introduced to model the decision on the use of the manure for fertilizing the meadow going with farm activity i:

y_i: fraction of produced manure used on the land with activity i

No more manure can be applied on the land than has been produced, and thus y_i is restricted by the following constraint:

$$\Sigma y_i \leq 1$$

The cost saving of using the manure produced by activity j, instead of disposing it, is given by the parameter δ_j. This cost saving is higher when the manure contains more of the useful nutrients that the land requires, and also when the prices of the replaced inorganic fertilizer are high. The parameter q_{jk} represents the amount of nutrient k produced in the manure of activity j. The index k represents all relevant nutrients, viz. nitrate, phosphate and potassium. In agreement with the approach of the farm management model, q_{jk} is given in kilogram per hectare. The total amount of nutrient k in the manure is given by $\Sigma\, q_{jk}x_j$. The government will set standards (parameter d_{ik}) for the utilization of nutrient k per hectare land use of activity i, due to large environmental problems connected with the increased use of nutrients in the intensive livestock sector. The amount of nutrient k applied for activity i should not exceed the environmental standards:

$$y_i \times \Sigma_j q_{jk} x_j \leq d_{ik} x_i \qquad i = 1, ..., n \quad k = 1, ..., K$$

The left-hand side of these nutrient constraints represents the amount of nutrient k (in kilograms) which is applied on land of activity i by using manure as fertilizer. Notice that by modelling the treatment of manure in this way, the model looses its linearity. The nutrient problem formulation is:

$$\begin{array}{llll}
\max & \Sigma \gamma_i x_i + \Sigma y_i \Sigma \delta_j x_j & & (4.3) \\
& Ax \leq b & & (4.4) \\
& \Sigma y_i \leq 1 & & (4.5) \\
& d_{ik} x_i - y_i \Sigma_j q_{jk} x_j \geq 0 & i = 1, ..., n \quad k = 1, ..., K & (4.6) \\
& x_i, y_i \geq 0 & i = 1, ..., n & (4.7)
\end{array}$$

Let us summarize the notation.

Indices

i,j	: indices for the farm activities
k	: index for the nutrients

Parameters

γ_i	: profit per hectare of farm activity i minus the costs of disposing all produced manure and buying inorganic fertilizer
δ_j	: costs of disposing the produced manure and buying inorganic fertilizer for activity j (is saved when Σy_i is positive)
A	: technology set of the farm model
b	: bounds on available scarce resources
q_{jk}	: content of nutrient k in manure from farm activity j (kg/ha)
d_{ik}	: environmental standards for the utilization of nutrient k for farm activity i (kg/ha)

Decision variables

x_i	: number of hectares used for farm activity i
y_i	: fraction of total produced manure used on land with activity i

To illustrate the formulations, ideas and solution methods in this paragraph, we use a simple example with two possible farm activities (1 and 2) and environmental rules for three nutrients. The numbers in this example are arbitrary and have no resemblance with real data.

The model formulation for this example is

$$\max \quad \{20x_1 + 70x_2 + (y_1 + y_2)(70x_1 + 95x_2)\} \tag{4.8}$$
$$x_1 + x_2 \le 10 \tag{4.9}$$
$$x_1 + 1.5x_2 \le 12 \tag{4.10}$$
$$y_1 + y_2 \le 1 \tag{4.11}$$
$$y_1(x_1 + 5x_2) \le x_1 \ ; \ y_2(x_1 + 5x_2) \le x_2 \tag{4.12}$$
$$y_1(4x_1 + x_2) \le 5x_1 \ ; \ y_2(4x_1 + x_2) \le 4x_2 \tag{4.13}$$
$$y_1(x_1 + x_2) \le x_1 \ ; \ y_2(x_1 + x_2) \le x_2 \tag{4.14}$$
$$x_1, x_2, y_1, y_2 \ge 0 \tag{4.15}$$

The constraints (4.9) and (4.10) define the feasible set of the classical farm plan (e.g. upper bounds for the acreage and the production of milk). Constraint (4.11) defines that no more manure can be applied than is available. The nutrient constraints (4.12), (4.13) and (4.14) represent the environmental regulations for fertilizing. In this example, constraint (4.12) dominates constraints (4.13) and (4.14), i.e. for non-negative variables (4.13) and (4.14) are redundant due to (4.12). Specific characteristics can be derived for data instances with this property (the one-nutrient problems).

4.2.3. Analysis of the model

Pooling property

The nonlinearity in (4.3) and (4.6) is caused by the use of one manure tank to store the manure of various activities. The bilinearity in the constraints is called the *pooling property* as introduced in Section 2.4. The nutrient problem has the same characteristics as the pooling problem because it is linear whenever either the decision vector x or y is fixed. However, the pooling problem has fixed bounds on the concentrations of the final product, whereas in the nutrient problem the concentrations may vary and the use of the end product is bounded by the input, i.e. the farm plan. One could say that the output (concentrations) is cyclically connected with the input (the farm plan), see Figure 4.2. The pooling problem as described in Foulds *et al.* (1990) has a linear objective function with bilinear balance restrictions whereas the nutrient problem has a bilinear objective function and bilinear constraints.

The nutrient problem as a bilinear problem

The roots of bilinear programming can be found in Nash (1951), who introduced game problems involving two players. Each player must select a mixed strategy from fixed sets of strategies open to each, given knowledge of the payoff based on selected strategies. These problems can be treated by solving a bilinear program.

A bilinear program is

$$\min_{(x,y)\in\Omega} f(x,y) = c'x + x'Ay + d'y$$

The objective is a function of two groups of variables. The problem is linear (actually affine) in one group of variables if the other group is fixed, and vice versa, over the feasible region Ω. Bilinear problems are interesting from a research point of view, because of the numerous applied problems that can be formulated as bilinear programs (dynamic Markovian assignment problems, multi-commodity network flow problems, quadratic concave minimization problems).

In the first model formulation, the feasible set Ω for x and y is the cartesian product $X \times Y$, where

$$X := \{x\colon B_1x \le b_1,\ x \ge 0\}$$
$$Y := \{y\colon B_2y \le b_2,\ y \ge 0\}$$

Global optimization algorithms to solve these *traditional bilinear programs* (TBP) have been developed e.g. by Falk (1973) and Sherali and Shetty (1980). These algorithms guarantee finite convergence for all instances. The key property of the traditional model, used in almost all methods, is that the feasible region is expressed as the cartesian product of two polyhedra. This structure ensures the existence of an extreme-point global solution (Falk, 1973).

Al-Khayyal (1990) relaxed this assumption by considering the feasible set

$$\Omega := \{(x,y)\colon Cx + Dy \le b,\ x \ge 0,\ y \ge 0\}$$

in the *jointly constrained program* (JCBP). In this case, interaction in the constraints between the x and y decision variables is allowed.

A further extension of the model is to include bilinear constraints, which results in the *generalized bilinear program* (GBP):

$$\min_{(x,y)\in\Omega} c_0'x + x'A_0y + d_0'y$$

with

$$\Omega := \{(x,y)\colon c_p'x + x'A_py + d_p'y \le b_p\ \forall p,\ \ Cx + Dy \le b,\ \ x \ge 0,\ y \ge 0\}.$$

The nutrient problem now becomes a special case of (GBP) where constraints (4.4) and (4.5) represent the linear parts and constraints (4.6) the bilinear part. Note that for this special case the jointly constrained part of the (GBP) reduces to the cartesian product $X \times Y$, where

$X := \{x: Ax \leq b, x \geq 0\}$ and $Y := \{y: \Sigma y_i \leq 1,\ y \geq 0\}$.

The objective function of a generalized bilinear problem is a *bi-affine* function, that is affine for x and y for fixed y and x respectively. The feasible set Ω is a *bipolyhedron*, i.e. a polyhedron in x and y for fixed y and x respectively. A general way to analyze such a bilinear problem is to fix x and y successively, iteratively leading to an LP problem in either y or x. We will elaborate further on this successive LP approach. First the problem is considered in another decomposed way.

The nutrient problem as a two step problem, a specific property

The mathematical structure of the nutrient problem makes it possible to analyze the problem from the following point of view. For every x, the optimal y can be calculated. Given this optimal value for y (in terms of x variables), we only have to solve a maximization problem in the x variables. The branch-and-bound algorithm in Section 4.4. uses this feature.

For convenience, we introduce S as the sum of the y variables, and $S(x)$ as the optimal fraction of the produced manure, applied on the land given farm plan x. Now, the nutrient problem in x variables is:

$$\max_x\, (\gamma + S(x) \times \delta)'x$$
$$Ax \leq b$$
$$x \geq 0$$

The objective function is monotonously increasing in S since δ is strictly positive. The objective value (4.3) corresponding to the maximization problem is denoted as $z(x, S(x))$; it depends on x directly as well as on the optimal fraction $S(x)$ which is applied. In general, the expression $S(x)$ is equivalent to $\Sigma_i \min\limits_k \dfrac{d_{ik}x_i}{\Sigma_j q_{jk}x_j}$.
If only one nutrient constraint is active, the expression for $S(x)$ is much easier. For the example, the sum $S(x)$ can be expressed as $\dfrac{x_1 + x_2}{x_1 + 5x_2}$, since (4.12) is the only active constraint. This expression can be substituted in the objective function. For the resulting nonlinear problem, the first-order Karush-Kuhn-Tucker conditions can be analyzed. The global maximum is $x^* = (10,0)$ with corresponding optimal value (1,0) for the y variables and an objective value of 900. Figure 4.3 gives the graphical representation of the model for this example. The example shows that it is interesting to examine the boundary of the feasible region of the variables $x_1, ..., x_n$ in particular, since all local optima can be found there. Moreover, Al-Khayyal and Falk (1983) show that jointly constrained bilinear problems always have boundary solutions.

Boundary solutions

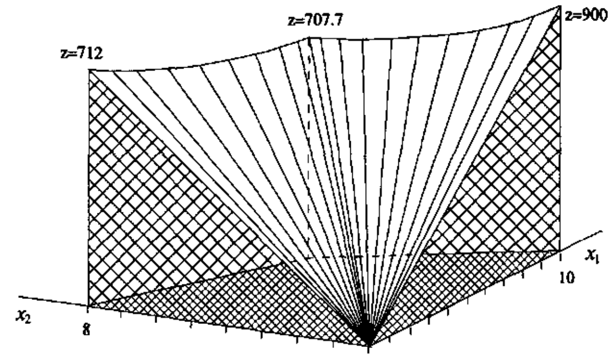

Figure 4.3: Graphical representation of the nutrient problem in x

In this paragraph, we analyze the existence of boundary solutions for the nutrient problem. The property of boundary solutions can be used explicitly in branch-and-bound methods as will be outlined in Section 4.4, since this characteristic can reduce the computation time of these algorithms.

We first consider the one-nutrient problem, where one nutrient constraint (say k^*) makes the other nutrient constraint redundant. For this simplification, the optimal value for $y_i(x)$ is $\frac{d_{ik}.x_i}{\sum_j q_{jk}.x_j}$.

Therefore, $S(x) := \sum y_i(x)$ can be written as $\frac{d'x}{q'x}$, and the objective function (1) can be replaced by $\min -(\gamma' x + \frac{x'\delta d'x}{q'x}) = \min \frac{x'Cx}{q'x}$ with $C = -\gamma q' - \delta d'$.
The feasible region $X = \{x: Ax \leq b \ ; x \geq 0\}$ is compact.

Theorem 4.1 *If the matrix C has at least one negative eigenvalue, the optimal solution x^* can be found on the boundary of the feasible region X.*

The proof of this theorem is by indirect demonstration and is based on the existence of a direction in which the objective is concave. If x^* is an interior point then it is always possible to obtain a better function value in the neighbourhood of x^*. So, x^* cannot be a minimum point. More details on this proof can be found in Bloemhof-Ruwaard and Hendrix (1993) or alternatively in Pardalos (1986).

In general, matrix C in the one-nutrient problem has such a negative eigenvalue. As the vectors d, q and δ are positive, it is sufficient to have at least one positive element of the vector γ if the corresponding q-element has a strictly positive value. So, at least one farm plan should have a positive contribution, which is not a severe assumption.

We can generalize this result to the general nutrient problem. First, Al-Khayyal and Falk (1983) show that the solution must be on the boundary of the $X \times Y$ space.

Theorem 4.2 (Al-Khayyal and Falk, 1983) *If an optimal solution exists to the (GBP) problem, at least one optimal solution will be on the boundary of the bipolyhedral feasible set.*

This theorem states that some optimal solution (x^*, y^*) of the nutrient problem is found on the boundary of Ω. This solution is not necessarily an extreme point (Al-Khayyal, 1990). For the nutrient problem, theorem 4.3 shows that (x^*, y^*) is on the boundary of X.

Theorem 4.3 *Given the nutrient problem defined by (4.3) - (4.7), and* $X := \{x \in \mathbb{R}^n : Ax \leq b;\ x \geq 0\}$ *is a compact set. Then the solution* (x^*, y^*) *of the nutrient problem is found on the boundary of X.*

Proof:
Given the optimal y^* for a (GBP) problem, (x^*, y^*) is a solution of a linear programming problem in x only. Let $(NUP(x \mid y^*))$ denote the nutrient problem given a fixed value y^* for y. Constraints (4.6) define a cone
$\kappa := \{x \in \mathbb{R}^n\text{: } (d_{ik}e_i - y_i^* q_k)^T x \geq 0,\ i = 1, \ldots, n,\ k = 1, \ldots, K\}$, with e_i the i^{th} unit vector. The solution (x^*, y^*) can be found at a vertex of the feasible set of $(NUP(x \mid y^*))$ consisting of $X \cap \kappa$. Apart from the origin, κ does not contain a vertex. If x^* is not the origin, then it is not a vertex of κ, so the solution is found on the boundary of X.

4.2.4. Classical solution approaches

Successive linear programming approach

One promising heuristic to solve a bilinear programming problem is *successive linear programming*. Starting with any $x^0 \in X$, find y^0 that optimizes the objective function over Y, then find x^1 that optimizes the objective function over X, and so on until the objective does not improve between two successive optimizations.

If the feasible set Ω is a cartesian product $X \times Y$, with X and Y convex sets and the objective function is differentiable and convex over $X \times Y$, then any limit point to this alternating procedure globally solves the problem (Wendell and Hurter, 1976). If the objective function is allowed to be biconvex, every limit point is a Karush-Kuhn-Tucker point (Konno, 1976), but necessary conditions are needed for a limit point to be a local optimal point of the biconvex problem. However, the (GBP) problem does not have these characteristics. Therefore, the approach by successive linear programming can be very disappointing, as is illustrated by the nutrient problem.

First note the following observations for the nutrient problem:

- For any fixed $y^t \in Y$ in step t, the LP-problem in the variables x is equivalent to

$$
\begin{aligned}
& z(NUP(x \mid y^t)) = \max\ (\gamma + S^t\delta)'x && (4.16)\\
& d_{ik}x_i \le (q_k'x)y_i^t \qquad i = 1,\dots,n && (4.17)\\
& Ax \le b && (4.18)\\
& x \ge 0 && (4.19)
\end{aligned}
$$

- For any fixed $x^t \in X$ in step t, the LP-problem in the variables y is equivalent to

$$
\begin{aligned}
& z(NUP(y \mid x^t)) = \max\ (S = \Sigma y_i\ ) && (4.20)\\
& y_i(q_k'x^t) \le d_{ik}x_i^t \qquad i = 1,\dots,n \quad k = 1,\dots,K && (4.21)\\
& S \le 1 && (4.22)\\
& y \ge 0 && (4.23)
\end{aligned}
$$

- A feasible solution for the y-variables can be found for any given $x \in X$. The reverse statement does not necessarily hold.

The successive linear programming procedure for the nutrient problem is described as follows:

Successive Linear Programming Algorithm

```
Step 1    Start with any feasible $y^0$, $t = 1$.
Step 2    Given $y^{t-1}$ (with corresponding value for $S^{t-1}$), find $x^t$ that minimizes
          $z(NUP(x \mid y^{t-1}))$ over the feasible region for $x$.
Step 3    Given $x^t$ find $y^t$ that minimizes $z(NUP(y \mid x^t))$ over the feasible
          region for $y$.
Step 4    If $|y^t - y^{t-1}| \le \varepsilon$ then STOP else $t := t + 1$ and go to Step 2.
```

For the example, the successive linear programming algorithm stops after at most two iterations. The LP solution $x^* = x^1$ is a point on the boundary of X. Only for the starting vector $y^0 = (1,0)$, the successive linear programming method finds the global maximum $x^* = (10,0)$. This example illustrates that the successive linear programming approach may be very disappointing for the nutrient problem.

Nonlinear programming: local search

Perhaps the most common method to solve problems like the nutrient problem is to apply standard nonlinear programming software. As illustrated by Figure 4.3, the nutrient problem may contain many local optima. One could try to discover the global optimum by starting a nonlinear programming local search from various starting points. An overview of global optimization methods, based on local searches, can be found in Törn and Žilinskas (1989). One of the known packages for solving constrained nonlinear programming is GAMS/MINOS which was also used in Section 2.3 to experiment with a concave programming model. Implementation of the example in this package (MINOS 5.3) gives the different local optima for various starting points (Table 4.1):

Table 4.1: Solutions local search

starting point				local optimum				objective
x_1	x_2	y_1	y_2	x_1	x_2	y_1	y_2	
0.0	0.0	0.0	0.0	10	0	1.00	0.00	900.0
3.0	3.0	0.2	0.2	6	4	0.23	0.15	707.7
3.0	0.0	0.0	0.0	0	8	0.00	0.20	712.0

Starting point (0,0,0.2,0.2) did not converge to a local optimum, but stayed at the starting point. In the approach of using various starting points, at least one of the points should be situated in the region of attraction of the global optimum, which is hard to verify.

The traditional optimization methods, which in general are applied first, deliver local optima, or worse, no optima. They cannot avoid the pitfall created by the problem. Here is where global optimization methods can help. Following the route towards deterministic methods, requires use of the special properties of the mathematical structure of the problem. In Section 4.4. this is elaborated in the construction of a specific heuristic for the nutrient problem.

4.3. *Quadratic design problems*

4.3.1. Problem formulation

In many situations with respect to the design of products, mathematical models are used to describe the properties of a product to be designed. Our observations are based on experience in design departments of several companies. Abstracting from all kinds of practical technical details, one can say that a model (in the sense of Figure 1.2) calculates (or predicts) the consequences in terms of a vector of properties y of the values from a vector x of design parameters. Often a graphical presentation is used to describe the resulting design.

In this context the n-vector x represents the design parameters or factor variables of the product and the model, or function $y(x)$, describes the properties represented by an m-vector y. Because this study concerns nonlinear optimization, it is first of all assumed that $y(x)$ is a continuous function. Another assumption is the ability of x to vary in a continuous way within a so called experimental region X, whereas in practical design problems the values for the parameters are sometimes selected from a given finite set. For the algorithms we developed, the region X was taken to be a polytope, but in most cases it consists of lower and upper bounds on the design parameters. In a technological context it is common to use an expression about a relation $y(x)$ like "the relation is valid on this range".

The words "experimental region" originates from cases in which the function $y(x)$ is derived from (computer) experiments. As indicated in Section 2.5, it is common to use linear, factorial and quadratic regression functions as a first description $y(x)$ of measured properties of a set of experiments, which are taken from an experimental region described by a polytope or simple box constraints. Many of the observations in this section do not only apply for quadratic relations, but are general for the design problem. Now the designer aims to find the designs fulfilling target values for properties within the experimental region.

In the definition of the design problem, we restrict ourselves to the inequality form. The designer formulates target values on the quality of the product by setting lower or upper bounds bl_i and bu_i on the properties $y_i(x)$. Now the design problem is to find a product x in experimental region X which fulfils the requirements on the properties. To facilitate the mathematical analysis the slack function $g_i(x)$ is introduced as either

$g_i(x) = bu_i - y_i(x)$ when there is an upper bound requirement
or $g_i(x) = y_i(x) - bl_i$ when the quality is described by a lower bound.

Two sided requirements imply the introduction of more slack functions. To facilitate the discussion (without loss of generality), index i is used both for requirements as for properties.

Numerous ways in formulating design problems exist, see Taguchi et al. (1989). In order to focus on inequalities we do not define an objective function on the properties which is usually included in these problems. This results in a product design problem mathematically formulated as "find an element of $F \cap X$", with $F := \{x \in \mathbb{R}^n \mid g_i(x) \geq 0,\ i = 1, ..., m\}$. As formulated in Section 2.5, an objective (penalty) function can be constructed to define an equivalent optimization problem. By maximizing over X for instance

$$f(x) = \min_i g_i(x) \tag{4.24}$$

or

$$f(x) = \sum_i \min\{g_i(x), 0\} \tag{4.25}$$

a solution is sought of the inequality problem. Some remarks on the mathematical characteristics of these functions will be made when we discuss the analysis of the problem.

For the design problem, an additional complicating factor is when the value of parameter x represents the component as a fraction of a mixture product, as in the classical blending problem, in mixing and processing industries. This will be called the *mixture design* problem, which can be found in Hendrix and Pintér (1991) and Hendrix et al. (1996). Besides the restrictions of X and F, the possible mixtures are bound to be on the unit simplex $S = \{x \in \mathbb{R}^n \mid \Sigma_j x_j = 1,\ x_j \geq 0\}$. In 4.4 a specific algorithm is presented for the quadratic mixture design problem.

We were involved in setting up a Decision Support System for the design department of a chemical company. To give a flavour of the information the user can give and obtains, the input screen and output screen are sketched in Tables 4.2 and 4.3 respectively. As in the discussion of the nutrient problem a simple illustrative example, the rum-coke example, is used. The design parameters x_j represent the rate of coke, rum and ice in a glass of drink and the properties y_1 and y_2 describe the taste and colour of the product.

Table 4.2: Sketch of the input screen of the product design problem, rum-coke example

Parameter	*L*	*U*	property	*bl*	*bu*
coke	0.0	0.75	taste	−2.0	−1.0
rum	0.0	1.00	colour	0.1	0.4
ice	0.0	0.90			

In an experimental setting, where the relations $y_i(x)$ are found by regression on experiments, default values exist for lower and upper bounds on the design parameters $L_j \leq x_j \leq U_j$. The quadratic functions describing the properties in our example are:

$$y_1(x) = -2 + 8x_1 + 8x_2 - 32x_1x_2$$
$$y_2(x) = 4 - 12x_1 - 4x_3 + 4x_1x_3 + 10x_1^2 + 2x_3^2.$$

The initial values for the bounds on the properties can be taken as the lowest and highest values found in the experiments. Those do not necessarily equal the minima and maxima of $y_i(x)$ on the experimental region X. Now the user can iteratively set bounds on the design parameters and formulate requirements by setting target bounds on the properties. For the example it will be assumed that upper bounds $bu_1 = -1$ and $bu_2 = 0.4$ have been formulated as requirements on the properties $y_i(x)$. The user can feed the input to a solver which tries to find solutions of the inequality problem.

Experience learns that it is a good heuristic to present the best, let say ten solutions which have been found. The ranking of the "best" can be done with respect to a score function defined by (4.24), (4.25) or any other multi-criteria function. In Table 4.3 an example is reported of an infeasible solution with a score based on (4.25).

Table 4.3: Sketch of the report screen of the product design problem, rum-coke example.

solution 9	parameter	value	property	value	slack
score −1.125	coke	0.25	taste	0.000	−1.000
	rum	0.25	colour	0.625	−0.125
	ice	0.50			

For the simple example the solution space can be represented in a figure.
The slack functions are:

$$g_1(x) = 1.0 - 8x_1 - 8x_2 + 32x_1x_2$$
$$g_2(x) = -3.6 + 12x_1 + 4x_3 - 4x_1x_3 - 10x_1^2 - 2x_2^2.$$

Because the example is a mixture design problem (x_1+x_2+x_3=1), a projection of the simplex S on the x_1, x_2 plane is generated. In Figure 4.4 vertex x_p represents a product consisting for 100% of component p, p = 1,2,3. The area in which the feasible products are situated is given by F. The problem of maximizing (4.25) over S has a local optimum in x_{loc} = (0.125, 0, 0.875), $f(x_{\text{loc}})$ = −0.725, and of course a global optimum (= 0) for all elements of $F \cap S$.

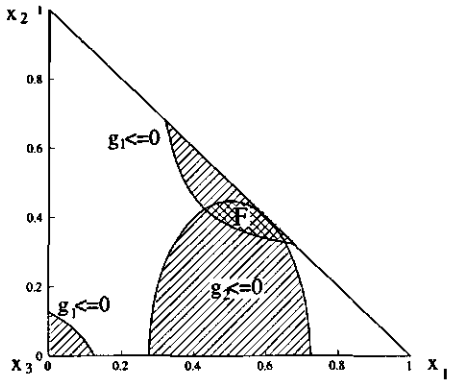

Figure 4.4: The feasible set F of the rum-coke example

In practical mixture design, an additional complication is the appearance of variables x_j which do not represent components of the mixture but additional process variables, such as temperature and mixing speed. Another important aspect is that the mixing industry is typically interested in stable solutions, as irregularities, fluctuations may appear during the production process. Mathematically, this means that a subset of $F \cap D(\cap S)$ with a given volume ε should be looked for or alternatively an interior point located at a given distance from the boundary of F is sought. This is discussed in Section 4.5.

4.3.2 Analysis and solution approaches

Having outlined the design problem, we will now have a look at possible solution approaches and an analysis of the mathematical problem.

Grid search

One of the simplest approaches to solve the problem is doing a grid search over the experimental region X. This engineering approach requires no assumptions on the function $y_i(x)$ and is often applied in a first attempt to find a solution. We also implemented a version of this method to have a reference which easily can be explained to users. Mixture design problems need a special treatment; an equidistant grid on x_1, x_2 in Figure 4.4 with different step sizes for x_1 and x_2 does not necessarily result in an equidistant grid for $x_3 = 1-x_1-x_2$. For the general problem, a do-loop testing $g_i(x)$ on being positive or negative can be cut off if for one of the properties (indices) a negative slack has appeared. This means that ordering the indices towards their expected difficulty may speed up the search, as it needs less function evaluations.

An obvious drawback of the method is the possibility to miss, step over, a feasible solution of $F \cap X$. Another drawback is the relatively high number of evaluations, although not expensive when quadratic functions are used. Given K steps in every dimension, the method requires K^n points to be tested on their feasibility (worst case). For practical problems the dimension is in general low, $n \leq 10$, but the computational effort increases quickly when the stepsizes are reduced (K becomes larger). One can have the user following the search process by counting down the number of points which (maximum) still have to be generated and by depicting the best score values found, such that the user can interact in the search process. Because of the appearance of fast computer power the simple grid search approach is possibly effective; it worked for all test cases we had. However, many evaluations are done in areas which are not interesting and the user should exclude by changing bounds.

In the theoretical field, Manas (1968) proposed a grid search approach for solving indefinite quadratic programming problems. Making use of the property that the functions $g_i(x)$ are quadratic, a minimum size for the grid can be identified on the observation that a Lipschitz constant is available. We will not go into detail on this topic.

Local search

The easiest use of the mathematical structure of the problem under consideration is the observation that (4.24) and (4.25) are (nonsmooth) continuous functions. By applying local searches from possibly random starting points, the iterates walk uphill to look for positive slacks and thus feasible products. We implemented a local search method based on Powell's method (see Powell, 1964) to find the maximum of (4.25). For the mixture design problem a penalty function was composed. A method of generating random starting points and doing local searches (this is called *multistart*), required much less function evaluations than plain grid search and appeared more effective in looking for interior solutions of F when (4.24) was used.

A drawback of the multistart approach is that there is no guarantee to find the global optimum. If after some calculation time no solution of the inequality problem has been found, it is not certain whether there exists one. The approach of combining random search and local searches will be discussed further in Chapter 5. It is useful however in the context of design problems to elaborate on the ideas of random search methods.

Methods based on the generation of random numbers such as Pure Random Search, Genetic Algorithms and Simulated Annealing, which start to be popular in the world of modelling based design, are successful when the number of optima is really big, as will be discussed in Chapter 5. Doing many local searches on a washing board is not very effective. A statement in Chapter 5 is that one can better perform a global search (random search), rough exploitation of the feasible set, before starting a local search. When the number of optima is limited on the contrary, one can better put energy in tracking the local optima by performing local searches. The practical design problem under consideration here has **a limited number of optima** in terms of (4.24) and (4.25). The number of optima, and components of $F \cap X$, was limited (≤ 3) for all practical cases we had (around 10). By analyzing the model, bounds can be found on the number of possible optima.

When all functions $g_i(x)$ are concave, there is only one local and global optimum of (4.24) and (4.25) and the area $F \cap X$ consists of one compartment, because the problem is equivalent to convex minimization. For the worst case, $g_i(x)$

is convex on a polytope X; every vertex may be a local optimum. In the common case where X is defined by box constraints and $g_i(x)$ are quadratic, the number of optima of $g_i(x)$ is bounded to be 2^p, where p is the number of positive eigenvalues of the Hessean of $g_i(x)$ (see Pardalos and Rosen, 1987). Notice that the design problem contains m slack functions $g_i(x)$.

Quadratic functions
A further analysis of quadratic functions $g(x) = x'Qx + d'x + c$, shows one may consider an indefinite objective $x'Qx$ as a separable function by decomposing Q in $Q = UDU'$, in which D is the diagonal matrix with eigenvalues and U consist of n orthonormal eigenvectors. On a polytope the convex part of the quadratic function may be overestimated, creating the possibility of constructing bounds. Another overestimation can be based on the Lipschitz continuity of the function. On a bounded set the "maximum derivative" $\| \nabla g_i(x) \|$ is bounded. For the algorithm in Section 4.4.3, we will make use of this property. Another mathematical structure which is related to quadratic functions is bilinearity. The quadratic term $x'Qx$ may be replaced by $x'Qy$ with the linear restriction $x = y$. The dimension is multiplied in this way and the mathematical consequences as discussed in Section 2.2. and with respect to the nutrient problem are valid.

This analysis learns how one can look with various pairs of glasses or tools (the so-called hammer-nail story in Chapter 1) to the same mathematical problem. We have seen how traditional heuristics such as grid search, random search and multistart may be effective, but give no guarantee on the (lack of) existence of a feasible solution for the problem. The deterministic methods based on branch-and-bound provide this guarantee with a corresponding price on computational effort and computer memory, as will be discussed in 4.4.

Moreover, the application of standard nonlinear programming is not straightforward, as nondifferentiable functions are optimized. At least the analysis has shown that:
- the number of possible optima is limited compared to the nutrient problem,
- value information due to the quadratic structure is available for the derivation of bounds.

4.4. *Application of branch-and-bound*

4.4.1. Introduction

The analysis of the mathematical structure of models and optimization problems as in Sections 4.2. and 4.3. can be used to develop and select specific deterministic optimization algorithms. An overview on the toolbox of possible approaches, such as outer approximation, cutting planes, inner approximation and branch-and-bound can be found in literature on the subject. For instance Horst and Tuy (1990) give a large overview and in "Introduction to Global Optimization" (Horst et al., 1995) a didactic introduction is given. In this Section a sketch will be given on the structure of the branch-and-bound approach, the most applied technique for deterministic global optimization.

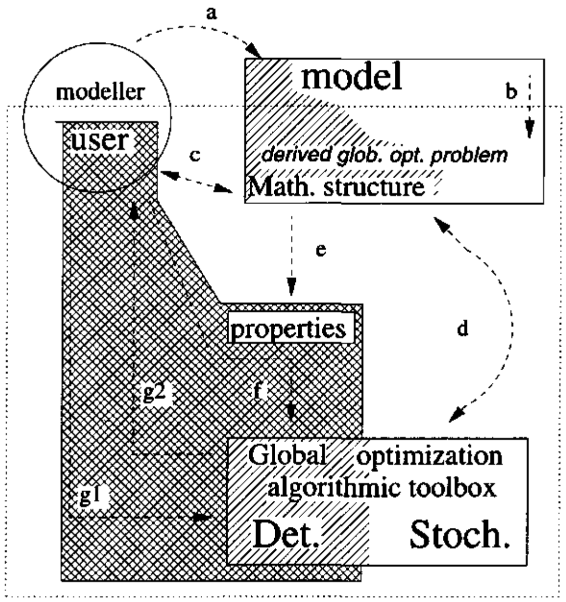

Figure 4.5: Using the properties to construct specific algorithms

As discussed in Chapter 3, the users in the target groups may be more or less familiar with the branch-and-bound approach for Mixed Integer Linear Programming (MILP). Therefore, first some similarities are indicated, before an example algorithm is presented. This is followed by some simple numerical examples in which the reader may follow the steps of the algorithm. After this sketch of the branch-and-bound approach, in 4.4.3 and 4.4.4 specific algorithms for the Nutrient problem and Quadratic Mixture Design problem are discussed.

4.4.2. The branch-and-bound procedure

In the branch-and-bound method the feasible set is relaxed and subsequently partitioned in more and more refined parts (branching) over which lower and upper bounds of the minimum objective function value can be determined (bounding). Parts of the feasible set with lower bounds exceeding the best upper bound found at a certain stage of the algorithm are deleted from further consideration (pruning), since clearly these parts of the domain do not contain the optimum.

We discuss some similar elements in the branch-and-bound approach for global optimization (GLOP) and MILP. In both methods we start with a set C_1 enclosing the feasible set X of the optimization problem (minimization is assumed). For MILP, set C_1 arises from relaxing the integrality constraints. At every iteration the branch-and-bound method has a **list Λ of subsets** (partition sets) C_k of C_1. The method stops when the list is empty and starts with C_1 as the first element. For every set C_k in Λ, a **lower bound** z_k^l is determined of the minimum objective function value on C_k. For MILP this bound is usually based on the continuous LP solution over C_k. As will be shown, for global optimization exist numerous ways of calculating lower bounds.

At every stage, there also exists a **global upper bound** z^U of the minimum objective function value over the total feasible set defined by the objective value of the best feasible solution found thus far. The **bounding** (pruning) operation concerns the deletion of all sets C_k in the list with $z_k^l > z^U$. Besides this rule for deleting subsets from list Λ, a subset can be removed when it does not contain a feasible solution. For MILP this is verified when running a continuous LP problem for the determination of the lower bound. However, as will be shown in 4.4.4, for the GLOP-variant it is not always easy to perform this so called 'deletion by infeasibility' step.

The **branching** concerns the further refinement of the partition. Although in the GLOP-variant every subset may be divided in several new subsets for any group of subsets on the list simultaneously, we restrict ourselves to a binary search tree variant as in MILP. This means that one of the subsets is selected to be split in two new subsets. In the referred literature, other strategies can be found. For the MILP-variant, in general the two new subsets are constructed by adding two bounds on a variable which got a nonintegral value in the continuous solution. In global optimization there exist several ways for splitting a subset in two or more subsets.

The **selection rule** which determines the subset to be split next, influences the performance of the algorithm. One can select the subset with the lowest value for its lower bound (most promising) or for instance the subset with the largest size (relatively unexploited). Considering the binary way of searching, with the corresponding data structures, one can look for a depth first search or a breadth first search. The target is to obtain sharp bounds z^U soon, such that large parts of the search tree (of domain C_1) can be pruned. The selection rule is one of the elements of the algorithm where a user can influence the performance. Software for MILP problems in general contain features to monitor the search and parameters can be set to influence the search. There are several reasons to remove subsets C_k from the list, or alternatively, not to put them on the list in the first place.

- C_k cannot contain any feasible solution.
- C_k cannot contain the optimal solution as $z_k^l > z^U$.
- C_k has been selected to be split.
- It has no use to split C_k any more. This happens for example for MILP problems when the continuous LP solution over C_k has integral values and in global optimization this may happen when the size of the partition set has become smaller than a predefined accuracy ε.

Outline of a branch-and-bound algorithm

```
0.  Given accuracy $\varepsilon$, $\Lambda = \emptyset$.
    Determine a set $C_1$ enclosing the feasible set $X$, $C_1 \supset X$ .
1.  Determine a lower bound $z_1^l$ on $C_1$.
    Determine a feasible point $x^1 \in C_1 \cap X$.
    If there exists no feasible point STOP
        else $z^U := f(x^1)$, put $C_1$ on list $\Lambda$, $r := 1$.
    endif.
2.  If list $\Lambda$ is empty STOP.
3.  Take (selection rule) a subset $C$ from list $\Lambda$ and split it into two new
    subsets $C_{r+1}$ and $C_{r+2}$.
4.  Determine lower bounds $z_{r+1}^l$ and $z_{r+2}^l$.
5.  For $p := r+1$ to $r+2$ do
        If $C_p \cap X$ contains no feasible point $z_p^l := \infty$ endif.
        If $z_p^l < z^U$
            then calculate a feasible point $x^p$ and $f_p = f(x^p)$.
                if $f_p < z^U$
                    then $z^U := f_p$ and remove all $C_k$ from
                        $L$ with $z_k^l > z^U$.
                endif
                if $|z_p^l - z^U| < \varepsilon$
                then save $x^p$ as an approximation of the optimum
                else add $C_p$ to list $\Lambda$
                endif
        endif
    endo
6.  $r := r + 2$, go to step 2.
```

In step 3 the subset with the most promising (lowest) lower bound z_k^l is chosen, defining the selection rule. In this scheme, the selected subset is split into two new subsets, thus defining a binary tree. This specific way of splitting is not necessary in global optimization. The selected set C_k can be partitioned in any number of subsets. Index r represents the number of subsets which have been generated. Note that r does not give the number of subsets on the list. The global bound z^U is updated, every time a better feasible solution is found.

After a successful search, list Λ will be empty and a guarantee is given that either the global optimum has been found or that there exists no feasible solution. When the search is not successful the size of list Λ keeps increasing and fills up the available computer memory, despite possible use of efficient data structures. Therefore it is important to obtain good bounds. Various ways of deriving bounds will be illustrated in this section. Furthermore, it is important to inform the user on the course of the algorithm by showing e.g. z^U and the number of subsets in list Λ.

Examples
A somewhat more specified algorithm is outlined at this point for the determination of a global optimum for the general (nonconvex) quadratic programming problem.

$$\min\{f(x) = x'Qx + d'x + c\}$$
$$x \in X$$
with X given by $\{x \in \mathbf{R}^n \mid Ax \le b, x \ge 0\}$ a polytope (bounded).

As partition sets (hyper)rectangles C_k are used, defined by the two extreme corners l^x_k and u^x_k i.e. $l^x_{jk} \le x_j \le u^x_{jk}$, $j = 1, ..., n$. Initially the global upper bound z^U can be put on infinity or given the objective function value of a feasible solution X which can be found by LP. For the illustration, two ways of calculating a lower bound z^l_k are given and elaborated for small numerical examples.

Lower bound 1
The first lower bound is based on the core of concave minimization and can be applied when $f(x)$ is concave (see Chapter 2). It uses directly the definition of concave functions:

$$f(x) = f(\lambda v_1 + (1-\lambda)v_2) \ge \lambda f(v_1) + (1-\lambda)f(v_2).$$

Many observations in Horst and Tuy (1990) are based on this so called affine underestimating function. Calculate the function values $f_i = f(v_i)$ for all vertices v_i, $i = 1, ..., 2^n$, of C_k. Solve the following LP problem bij minimizing over x and λ_i

$$\begin{aligned} z^l_k := \min\ & \Sigma F_i\lambda_i \\ & \Sigma\lambda_i v_i = x \\ & \Sigma\lambda_i = 1 \\ & \lambda_i \ge 0, \quad i = 1, ..., 2^n \\ & x \in X \end{aligned}$$

Lower bound 2
The second lower bound is based on the equivalence of general indefinite quadratic programming with bilinear programming as discussed in Al-Khayyal (1990, 1992). The quadratic term $x'Qx$ is equivalent with

$$x'y, \quad y = Qx.$$

The used observation is that xy for $l^x \le x \le u^x$,$l^y \le y \le u^y$ can be underestimated by (Al-Khayyal, 1990):

$$xy \ge l^x y + l^y x - l^x l^y$$
$$xy \ge u^x y + u^y x - u^x u^y.$$

To use this underestimating technique, the bounds l^y and u^y of $y = Qx$ have to be determined given $l^x \le x \le u^x$. This is not very hard. For every component $y_i = \Sigma q_{ij} x_j$ the lower bound l^y_i can be determined by

$$l^y_i = \sum_j q_{ij}\, I(q_{ij})$$

$$\text{with} \quad I(q_{ij}) = \begin{cases} = l_j^x & \text{if} \quad q_{ij} < 0 \\ = u_j^x & \text{if} \quad q_{ij} \ge 0 \end{cases}$$

Analogously

$$u^y_i = \sum_j q_{ij}\, I\,(-q_{ij})$$

can be determined. Now the second way of calculating a lower bound can be derived. Given rectangle C_k with corresponding bounds l^x, u^x. Determine the bounds l^y and u^y of $y = Qx$ for $x \in C_k$.
Solve the LP problem

$$\begin{array}{ll} z_k^l := & \min\{\Sigma w_j + d'x + c\} \\ & w_j \ge l^x_j y_j + l^y_j x_j - l^x l^y \qquad j = 1, \ldots, n \\ & w_j \ge u^x_j y_j + u^y_j x_j - u^x_j u^y_j \qquad j = 1, \ldots, n \\ & y = Qx \\ & x \in X \cap C_k. \end{array}$$

This lower bound calculation exploits the mathematical structure of bilinearity whereas lower bound 1 exploits the concavity. A more sophisticated use of the quadratic structure is given in Pardalos et al. (1987) by the separation of convex and concave parts of the objective via an eigenvalue decomposition of Q. The concave part is underestimated by an affine minorant as in lower bound 1 and the convex part is left unchanged. This results in a convex quadratic programming problem describing a convex envelope of $f(x)$ on a subset which can be solved by nonlinear programming solvers like MINOS.

Illustrative problem (*CQP*)

First a concave quadratic programming problem (*CQP*) is given.

$$\min\{f(x) = -(x_1-1)^2 - x_2^2\}$$
$$x \in X$$

X is given by

$$\begin{aligned} -x_1 + 8x_2 &\le 11 \\ x_1 + 4x_2 &\le 7 \\ 6x_1 + 4x_2 &\le 17 \\ 0 \le x_1 &\le 25 \\ 0 \le x_2 &\le 2 \end{aligned}$$

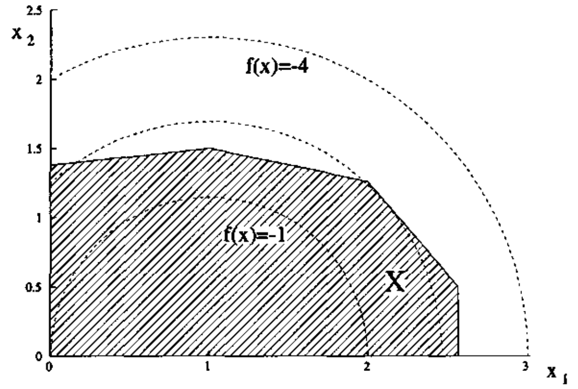

Figure 4.6: Example (*CQP*) of a concave quadratic program

Some contour lines and the feasible area are depicted in Figure 4.6. Problem (*CQP*) has 4 local optima in the points (0, 1.375), (1, 1.5), (2, 1.25) and (2.5, 0.5). Moreover, points (0, 0), (1, 0) and (2.5, 0) are Kuhn-Tucker points which are no local optima.

Now the outlined branch-and-bound method can be used with the lower bound 1. The branching is performed by bisecting the rectangles over their longest edge. The starting rectangle C_1 is defined by $l^x = (0, 0)$, $u^x = (2.5, 2)$. For the accuracy $\varepsilon = 0.05$ is taken. The resulting course of the algorithm is given in Table 4.4. The first column indicates the generated subset. The second column gives the parent, the subset which has been split. Point x^k is a feasible solution (if it exists) resulting from the lower bound calculation. Its function value can be used to improve the global upper bound z^U.

Table 4.4: Course of the branch-and-bound algorithm for *CQP*.

k	Par	l_k^x	u_k^x	x^k	z_k^l	f_k	z^U	L
1		(0, 0)	(2.5, 2)	(1, 1.5)	−4.5	−2.25	−2.25	{1}
2	1	(0, 0)	(1.25, 2)	(0, 1.375)	−3.75	−2.89	−2.89	-
3	1	(1.25, 0)	(2.5, 2)	(2, 1.25)	−3.875	−2.56	−2.89	{2,3}
4	3	(1.25, 0)	(2.5, 1)	(2.5, 0.5)	−2.75	-	−2.89	_
5	3	(1.25, 1)	(2.5, 2)	(2, 1.25)	−3.125	−2.56	−2.89	{2,5}
6	2	(0, 0)	(1.25, 1)	(0, 1)	−2	-	−2.89	-
7	2	(0, 1)	(1.25, 2)	(0, 1.375)	−3.125	−2.89	−2.89	{5,7}
8	7	(0, 1)	(0.625, 2)	(0, 1.375)	−3.125	−2.89	−2.89	-
9	7	(0.625, 1)	(1.125, 2)	(1, 1.5)	−2.59	-	−2.89	{5,8}
10	8	(0, 1)	(0.625,1.5)	(0, 1.375)	−2.9375	−2.89	−2.89	-
11	8	(0, 1.5)	(0.625, 2)	Inf	-	-	−2.89	{5}
12	5	(1.25, 1)	(1.875, 2)	(1.875, 1.28)	−2.62	-	−2.89	-
13	5	(1.875, 1)	(2.5, 2)	(2, 1.25)	−2.81	-	−2.89	∅

It can be observed from Table 4.4 that for the second subset the global optimum $x^* = (0, 1.375)$ has already been found, leading to the sharpest upper bound. The further iterations only serve as a verification of the optimality of x^*. This feature can often also be found when running a branch-and-bound algorithm for MILP. The bounding is successful for subsets 4, 6, 9, 12 and 13 where $z_k^l > z^U$. Subset 11 appeared to be infeasible. In subset 10 finally x^* was recognised as being a global minimum point as $|z_{10}^l - z^U| < \varepsilon$. In Figure 4.7, the final partition and location of generated points x^k are depicted.

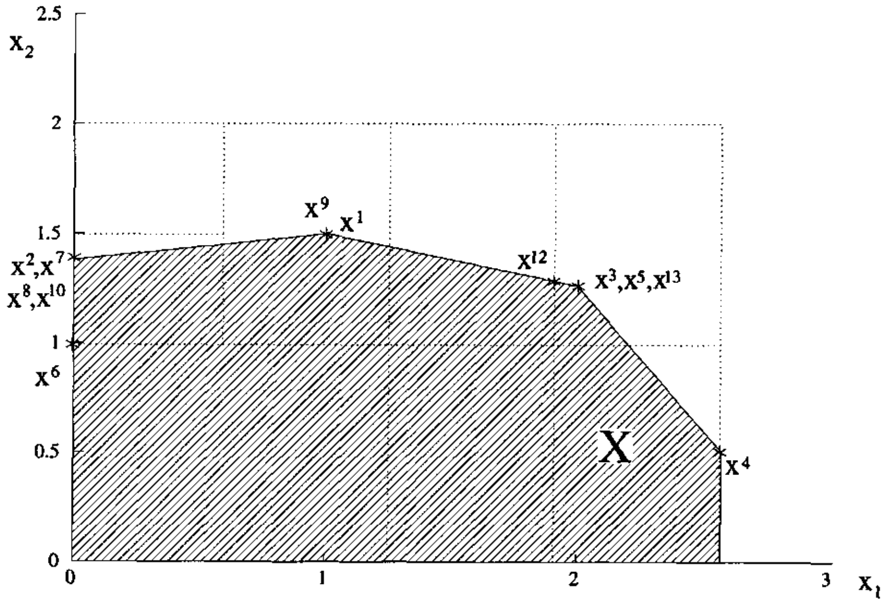

Figure 4.7: Final partition of the branch-and-bound method for the (CQP) problem

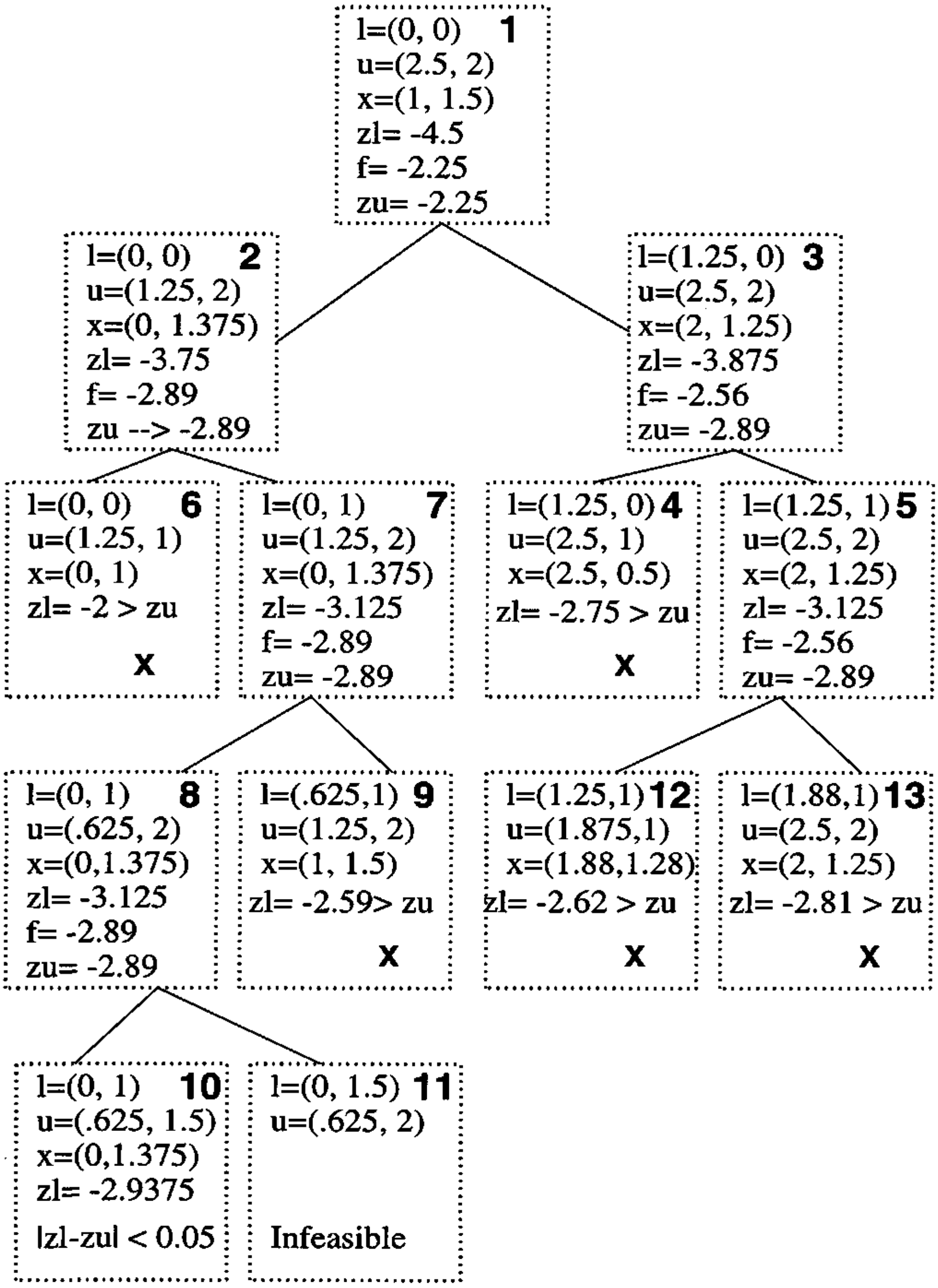

Figure 4.8: Branch-and-bound tree for the (*CQP*) problem

Illustrative problem (*IQP*)

The second numerical example is an indefinite quadratic programming (*IQP*) problem for which lower bound 1 cannot be applied.

$$\min\{f(x) = (x_1-1)^2-(x_2-1)^2\}$$
$$x \in X$$

X is given by

$$x_1 - x_2 \le 1$$
$$4x_1 - x_2 \ge -2$$
$$0 \le x_1 \le 3$$
$$0 \le x_2 \le 4$$

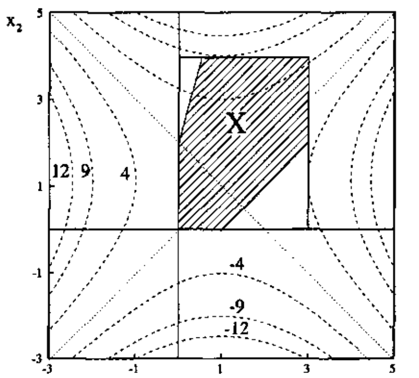

Figure 4.9: Contour lines and feasible set of (*IQP*)

Some contour lines and the feasible set of (*IQP*) are given in Figure 4.9. The problem has two local minimum points, namely (1, 0) and (1, 4) (the global one). The outlined branch-and-bound method is used again with the lower bound 2. The calculation of the bounds for the y variable is extremely simple for (*IQP*) as $y_1 = x_1$ and $y_2 = -x_2$. For the accuracy again $\varepsilon = 0.05$ is used. The starting subset C_1 is defined by $l_1^x = (0, 0)$, $u_1^x = (3, 4)$. The course of the algorithm is given in Table 4.5. The final partition is depicted in Figure 4.10.

Table 4.5: Course of the branch-and-bound algorithm for *IQP*.

k	Par	l_k^x	u_k^x	x^k	z_k^l	f_k	z^U	L
1		(0, 0)	(3.4)	(1.5, 4)	−11	−8.75	−8.75	{1}
2	1	(0, 0)	(3.2)	(1.5, 0.5)	−3	–	−8.75	-
3	1	(0, 2)	(3.4)	(1.5, 4)	−11	−8.75	−8.75	{3}
4	3	(0, 2)	(1.5, 4)	(0.75, 4)	−9.5	−8.94	−8.94	_
5	3	(1.5, 2)	(3.4)	(2,4)	−5	–	−8.94	{4}
6	4	(0, 2)	(1.5, 3)	(0.75, 3)	−4.5	-	−8.94	-
7	4	(0, 3)	(1.5, 4)	(0.75, 4)	−9.5	−8.94	−8.94	{7}
8	7	(0, 3)	(0.75, 4)	(0.75, 4)	−8.94	−8.94	−8.94	-
9	7	(0.75, 3)	(1.5, 4)	(1.125, 4)	−9.125	−8.98	−8.98	{9}
10	9	(0.75, 3)	(1.5, 3.5)	(1.125, 3)	−6.375	–	−8.98	-
11	9	(0.75, 3.5)	(1.5, 4)	(1.125, 4)	−9.125	−8.98	−8.98	{11}
12	11	(0.75, 3.5)	(1.125, 4)	(0.94, 4)	−9.03	−8.98	−8.99	-
13	11	(1.125,3.5)	(1.5, 4)	(1.125, 4)	−8.98	-	−8.99	∅

Finally $x_{12} = (0.94, 4)$ is taken as an approximation of the global optimum, as $|z_{12}^l - z^U| < \varepsilon$ at this stage. This happens once more during the iterations, because $|z_8^l - z^U| < \varepsilon$, so that x^8 is temporarily saved as an estimation of the optimum. For this numerical example the selection rule appears to be not important, as list Λ only contains one subset during the iterations.

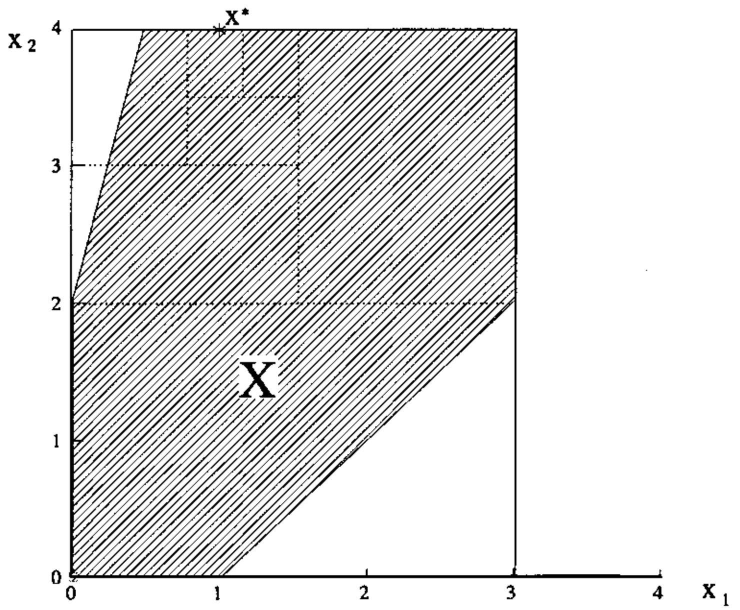

Figure 4.10: Partition for the (*IQP*) problem after 6 iterations

Implementation aspects
One of the problems, which does not follow from the simple examples, of implementing a branch-and-bound algorithm in a computer program is that information concerning the partition sets has to be kept in the computer memory. This can be done by maintaining a list of subsets and a list of points generated, but special data structures based e.g. on vertices or edges are also possible. The problem is that the program should not run out of memory, before a solution has been found. In branch-and-bound methods efficient use can be made of the fact that memory can be "recycled" if partition sets are deleted from further consideration. This can be done e.g. by applying linked list structures. Numerical aspects which are for instance important when dealing with differential equations, play a minor role in branch-and-bound. Good handling of required memory by storing the appropriate part of the tree on hard disk can speed up the search, but does not solve the overall capacity problem.

In this section the working of the branch-and-bound approach has been sketched and illustrated. In the following sections specific algorithms are derived for the Nutrient problem and the Quadratic Mixture Design problem.

4.4.3. A branch-and-bound method for the nutrient problem

Section 4.2.4. showed that traditional optimization methods may not lead to the solution of the nutrient problem. We now follow arrow f of Figure 4.1. and use the properties derived by the analysis of the mathematical structures in 4.2.3. to construct a specific branch-and-bound scheme for the problem, similar to the illustrations in 4.4.2. In the analysis the following appeared:

- The nutrient problem is a special case of generalized bilinear programming.
- The nutrient problem has solutions on the boundary of the feasible x-space X.
- The nutrient problem can be considered in a decomposed (two step) way.

A first consideration for a modeller who wants to solve the problem is to have a look at **literature** on specific methods for bilinear programming. Falk (1973) developed the first branch-and-bound procedure for the traditional bilinear program. It guarantees convergence to an exact solution in a finite number of steps. Bounds are obtained by solving linear programming relaxations and branching is accomplished by holding individual variables out of the basis for each branch when using the simplex method to solve the subproblems. Al-Khayyal and Falk (1983) develop a branch-and-bound algorithm for the (JCBP) problem. The algorithm branches into four nodes based on a partition of the parent node's rectangular set $C = \{(x,y)\colon l^x \leq x \leq u^x,\ l^y \leq y \leq u^y\}$ and bounding is achieved by minimizing the convex envelope of the objective function over the subset of the feasible region, intersected with the p^{th} partition set. The same can be done for the (GBP) problem (Al-Khayyal, 1992).

For the nutrient problem it is not necessary to make use of the general bilinear approach. Analogous to the *(IQP)* example of Section 4.4.2, partition sets can be defined on the X-space, not in the $X \times Y$-space as in Al-Khayyal (1992). The first ingredient of the algorithm is the definition of the partition sets.

Partition sets

Subset C_p: $[l_p, u_p]$ is defined as the *p-th* block, hyperrectangle, created in the branching tree (the vectors l and u have the same dimension as x). Because the optima can be found on the boundary, each partition set that does not intersect with the boundary of X can be deleted. To check this, it is not necessary to run a linear program. An algorithm to check whether box C:$[l,\ u]$ is interior with respect to polyhedron $X=\{x \in \mathbb{R}^n \mid Ax \leq b\}$ (A includes the nonnegativity constraints) is given in Bloemhof-Ruwaard and Hendrix (1996). Let $I(a_{ij})$ analogously to the lower bound calculation in 4.4.2 be defined as

$$I(a_{ij}) = \begin{cases} u_j & \text{if } a_{ij} \geq 0 \\ l_j & \text{if } a_{ij} \leq 0 \end{cases}$$

Now the algorithm works as follows.

Interior check algorithm

```
0.  $i := 1$

1.  If $\Sigma_j a_{ij} I(a_{ij}) \geq b_i$    STOP; box is not interior
    else $i := i+1$
    endif.

2.  If $i := n+1$  STOP; box is interior
    else go to step 1
    endif.
```

Calculation of the bound

A following ingredient of the algorithm is the construction of the bound. As the nutrient problem is a maximization problem, we are looking for an upper bound of the objective z on $C_p \cap X$. In the decomposition view of the analysis it has been shown that given a feasible solution $x^p \in X$, $S^p = S(x^p)$ can be determined by solving an LP problem, $S(x^p) = z(NUP(y \mid x^p)) = \min\limits_k \dfrac{d_{ik} x_i^p}{\Sigma q_{jk} x_j^p}$.

The relevant observation in the analysis is that the objective $z(x, S)$ is monotonic in S. Therefore using an overestimate σ^p of the maximum of $S(x)$ over $C_p \cap X$ results in an overestimating LP problem with objective $z(x, \sigma^p) \geq z(x, S(x))$.

The maximum of $S(x)$ over $C_p \cap X$ is not easy to derive, so the task is to find a good upper bound σ^p for $S(x)$ that is easy to obtain.

The upper bound derivation of the maximum value for S in (the feasible part of) each box C^p is as follows:

$$\max_{x \in C_p \cap X} S(x) = \max_{x \in C_p \cap X} \left\{ \Sigma_i \min_k \frac{d_{ik} x_i}{q_k' x} \right\} \leq \max_{x \in C_p} \left\{ \Sigma_i \min_k \frac{d_{ik} x_i}{q_k' x} \right\}$$

$$\leq \max_{x \in C_p} \min_k \left\{ \frac{d_k' x}{q_k' x} \right\} \leq \min_k \max_{x \in C_p} \left\{ \frac{d_k' x}{q_k' x} \right\}$$

So we choose $\sigma^p := \min\{1, \min\limits_k \max\limits_{x \in C_p} \{\dfrac{d_k' x}{q_k' x}\}\}$.

First, a fractional programming problem has to be solved for each k. Appendix 4.A describes a solution method for such a problem, the threshold algorithm. It is easy to determine σ^p once the optimal solution for each fractional programming problem is found.

Having formulated ingredients as partition sets, interior check and upper bound calculation, now a branch-and-bound algorithm is constructed analogous to the outline in 4.4.2. Notice that we are dealing with a maximisation problem, so that there is a global lower bound z^L and there are local upper bounds z_p^u.

Branch-and-bound algorithm for the nutrient problem

```
0.  Given tolerance $\varepsilon$, $r := 1$, $\Lambda := \emptyset$
    Determine a box $C_1:[l_1, u_1] \supset X$

1.  Calculate $\sigma^1$.
    $z_1^u = \max_{x \in X} z(x,\sigma^1)$.
    Denote $x^1$ as the corresponding maximum point.
    Initiate $z^L$ as $z^L := z(x^1, S^1)$.

2.  Bisect the box under consideration over the longest edge into $C_{r+1}$, $C_{r+2}$,
    $r := r+2$. For both boxes perform step 3.

3.  Perform the interior check.
    If the box is interior, stop analyzing the box, endif.
    Determine $\sigma^p$.
    Calculate $z_p^u := \max_{x \in X \cap C_p} z(x,\sigma^p)$; $x^p$ is the corresponding maximum point.
    If the box has no feasible solution, stop analyzing it, endif.
    If $z_p^u > z^L$ then
        determine $z_p^l := z(x^p, S^p)$, (function value of a feasible point)
        if $z_p^l > z^L$ then
        $z^L := z_p^l$ and save $x^p$ as approximation of the optimum.
        endif.
        Add $C_p$ to the list of boxes.
    endif.

4.  Delete all boxes from the list with $z_p^u < z^L$.
    If $\Lambda = \emptyset$ STOP.

5.  Take box $C_p$ from the list with the highest upper bound $z_p^u$.
    If $|z_p^u - z^L| < \varepsilon$ STOP.
        else go to step 2.
    endif.
```

The algorithm starts with the smallest rectangular box C_1 that includes
$X := \{x\colon Ax \leq b;\ x \geq 0\}$ as a whole. In step 1, the value of σ^1 is obtained using the threshold algorithm of Appendix 4.A. A global upper bound to $z(x, S(x))$, z_1^u, is

found by maximizing $z(x, \sigma^1)$ over X. For this solution x^1, we calculate the optimal y and the corresponding $S(x^1)$ solving the linear programming problem $NUP(y \mid x^1)$ and the first lower bound $z^L = z(x^1, S(x^1))$ is found. At each iteration, there is a list Λ of boxes C_p. One of them is split by dividing the box into two parts of equal volume over the longest edge (bisection). This creates two new boxes (step 2). For each of these boxes C_p it is checked whether the box is totally interior with respect to X. If it is, it is discarded and we proceed with another box. If

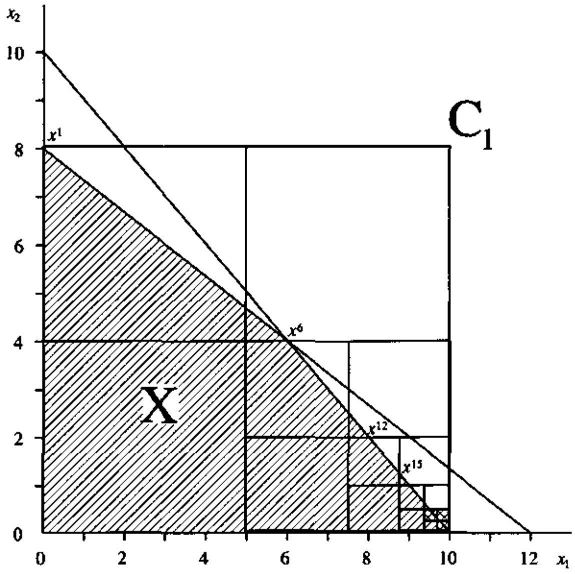

Figure 4.11: Partition after 15 iterations

the box is not interior, we continue with step 3. An upper bound $z_p^u := z(x^p, \sigma^p)$ is calculated, based on the upper bound σ^p obtained from the threshold algorithm described in Appendix 4.A. A feasible solution (x^p, S^p) is obtained by solving the simple LP problem $NUP(y \mid x^p)$, which results in a the objective value of a feasible solution, a local lower bound $z_p^l := z(x^p, S^p)$. If this lower bound is better than the global lower bound z^L, then $z^L := z_p^l$ and the solution x^p is saved. In step 4 the list of boxes is checked: If the upper bound z_p^u of some C_p is below the new lower bound z_L, there is no reason to analyze this box C_p further, so this box can be neglected.

We select a new box for further division by means of the highest upper bound z_p^u (step 5). During the branch-and-bound process, the gap between upper and lower bound, $\mid z_p^u - z^L \mid$, is decreasing. The algorithm stops as soon as this gap is less than a preset tolerance ε, or if the list of boxes is empty. In order to guarantee convergence of the overestimate σ^p to the maximum of $S(x)$ and to speed up the algorithm, boxes that are too small may also be discarded. Application of the branch-and-bound procedure to the example in 4.2.2 leads to the following result. After 15 iterations, the point $x_{\text{best}} = (9.75, 0.25)$ gives the best lower bound $z^L = 855$. The best upper bound is $z^u = 919$, so these bounds enclose the optimum value $z^* = 900$ at (10,0). Only three small partition sets in the neighbourhood of $x^* = (10,0)$ remain on the list (see Figure 4.11).

The branch-and-bound approach gives a guarantee that the optimum is approximated. However, just as the pooling problem the nutrient problem is high dimensional, so that in a worst case situation it may be practically impossible to reach this guarantee, as the computer memory fills up with boxes. Here is where the user can interact (arrow g1). It is not hard to generate a "good feasible solution" of the nutrient problem. The objective value can be used as a lower bound z^L on the optimum. The branch-and bound algorithm is continuing to verify whether the good feasible solution is indeed optimal. The user can interact by deleting boxes which, he thinks, will not contain the global optimum.

4.4.4. Branch-and-bound in a DSS for mixture product design

In 4.3 the design problem has been introduced as finding a feasible point of a set of inequalities $g_i(x) \geq 0$, $i = 1,...,m$. Some mathematical properties were given of quadratic functions $g_i(x)$, which can be used to develop a specific branch-and-bound method. In this section simplicial subsets are used in a branch-and-bound frame, which is a natural choice when considering mixture design problems. The use of simplicial sets is a variation on the application of rectangular partition sets used in the earlier sections. First the partition sets and outline of the algorithm are discussed. This is followed by a discussion on the implemented upper bound and alternative ways of generating upper bounds. Finally some remarks on implementation aspects and performance are given.

The problem to be solved consists of identifying mixture products, each represented by a vector $x \in \mathbb{R}^n$, which meet certain requirements. The set of possible mixtures is mathematically defined by the unit simplex $S = \{x \in \mathbb{R}^n \mid \Sigma x_j = 1,\ x_j \geq 0\}$, in which the variable x_j expresses the fraction of components j in product x. Note that set S lies in the $n-1$ dimensional hyperplane given by $\Sigma x_j = 1$.

Partition sets

As the first ingredient for the branch-and-bound algorithm, subsets C_k, being simplices (having n vertices) are distinguished. The points which are generated by the algorithm are denoted by x_{pk}, $p = 1,...,n$ and represent the vertices of a simplex C_k. In general one point appears as a vertex of several partition sets. The implementation requires special data structures to store and link the list of points and subsets. The information obtained in the n vertices of C_k is used to calculate upper bounds z^u_{ik} for the function values of g_i on C_k. In contrast to the treatment of the nutrient problem, the information is not used to derive a global bound z^L_i (compare the algorithm for the nutrient problem), as the global bound is defined by zero. Namely, when an upper bound $z^u_{ik} < 0$, simplex C_k cannot contain any feasible product and therefore can be discarded. The upper bounds can also be used in the selection rule to decide on which subset is to be split further.

The algorithm is summarized in the scheme. Let us devote some words to various steps. A new point is generated when one of the subsets is split. Because there is no update of the global $z^L_i = 0$, as in the earlier presented branch-and-bound schemes, the partition sets remain on the list Λ until they are selected for further refinement in step 3. The practical interest in robust solutions can be implemented by not putting a newly generated subset on the list when its size is too small, the diameter $\delta(C_k)$ is smaller than a preset tolerance; ($\delta(C_k) < \varepsilon$). In this case the algorithm either finds a feasible solution or gives the guarantee that no robust feasible composition exists. In a worst case solution the algorithm may require an exponential number of iterations (in n) to verify that all subsets are smaller than ε.

Branch-and-bound algorithm for the mixture design problem

```
0.  $C_1 := S$ , $r := 1$ , $\Lambda := \varnothing$

1.  If one of the vertices $x_{p1}$ $p=1,..,n$ is feasible ($g_i(x_{p1}) \geq 0$, $i=1,...,m$)
            STOP; solution found endif.
    Determine upper bounds $z^u_{i1}$ ($z^u_{i1} \geq g_i(x)$, $x \in C_1$).
    If $z^u_{i1} \geq 0$, $i = 1,...,m$, then put $C_1$ on list $\Lambda$.

2.  If $\Lambda = \varnothing$ STOP; no feasible composition exists.

3.  Take that $C_k$ from list $\Lambda$ with the highest value for $\sum_i z^u_{ik}$ and split it into
    two parts $C_{r+1}$ and $C_{r+2}$ of equal volume over the longest edge. Evaluate
    $x_{\text{new}}$, the midpoint of the longest edge of $C_k$.
    If $g_i(x_{\text{new}}) \geq 0$, $i = 1,...,m$,
            STOP; a solution has been found
    endif.

4.  For both new subsets $k := r+1$ to $r+2$ do
    Determine upper bounds $z^u_{ik}$.
    If $z^u_{ik} \geq 0$, $i = 1,...,m$, put $C_k$ on list $\Lambda$
    (The corresponding new product $x_{\text{new}}$ is saved as one of the vertices $x_{pk}$)
    endif
    endo.

5.  $r := r+2$, go to step 2.
```

Let us remark that the algorithm can be modified in such a way that it is not necessary to calculate all upper bounds at every iteration. The check in step 4 for every property i, can be interpreted as the question: is it possible that C_k contains a vector x for which $g_i(x) \geq 0$? To answer this question, it is not necessary to determine the upper bound z^u_{ik}, if for one of the vertices x_{pk} of C_k holds $g_i(x_{pk}) \geq 0$. The algorithm easily can be modified so that in such situations the upper bound is not calculated. Note that - as a consequence - the selection criterion in step 3 should be changed accordingly. In the implemented version it was also checked in step 4, whether a partition set is completely out of the bounds ($C_k \not\subset X$). This can be done by checking whether all vertices of C_k are at one side of the bounds (or linear restriction).

Before illustrating the way the algorithm proceeds, the various ways of calculating an upper bound for quadratic properties $g_i(x)$ are discussed.

Upper bound calculation
The mathematical structure as shown in earlier sections gives that bounds can be generated from a quadratic, bilinear and Lipschitzian point of view. Let us first consider the Lipschitzian point of view as described in Hendrix and Pintér (1991).
Let again x_{pk} denote the vertices of C_k, $p = 1,...,n$. The value of z^u_{ik} can be based on the following relations:

$$g_i(x) \le g_i(x_{pk}) + L_{ik} \, \|x - x_{pk}\|, \; x \in C_k, \; p = 1,...,n. \tag{4.26}$$

In (4.26) L_{ik} is the Lipschitz-constant of g_i on C_k. The Lipschitz-constant L_{ik} can be (over)estimated by solving

$$L_{ik} = \max_{x \in C_k} \|\nabla g_i(x)\|. \tag{4.27}$$

As the function g_i are quadratic, problem (4.27) means the maximization of a convex function over a polyhedron; hence it can be solved by simply evaluating $\|\nabla \, g_i(x)\|$ at every vertex x_{pk}. Note that the estimate of L can be made sharper, by projecting first the gradient $\nabla g_i(x)$ on the hyperplane of S. By convexity of $\|x - x_{pk}\|$, in the implemented algorithm the upper bound z^u_{ik}, based on (4.26) and (4.27) is:

$$z^u_{ik} = \min_p\{g_i(x_{pk}) + L_{ik} \max_{v \ne p} \|x_{vk} - x_{pk}\|\} \tag{4.28}$$

(x_{vk} being the vertices of C_k, different from x_{pk}).
We shall discuss some alternative ways to determine an upper bound.

In Lipschitzian global optimization on interval (box) regions, Pintér (1986, 1988) uses rectangular subsets $[l_k,u_k]$, for which the information available is based on the "lower-left" vertex l_k and the "upper-right" vertex u_k. His approach can be termed a diagonal extension of known univariate methods e.g. of the Danilin-Piyavskii-Shubert method (cf. the references). The selection of the subset to be refined (cf. step 3) is based on the (rectangular subset) selector function:

$$(g_i(l_k) + g_i(u_k))/2 + L_i\|l_k - u_k\|.$$

For an upper bound used in step 4 of the algorithm (elimination), Pintér uses the expression

$$\min\{g_i(l_k), \, g_i(u_k)\} + L_i\|l_k - u_k\|.$$

Applying directly this idea to the simplicial algorithm given above would lead to the upper bound:

$$z_{ik}^{u} = \min_{p} \{g_i(x_{pk})\} + L_{ik}\delta(C_k), \tag{4.29}$$

where $\delta(C_k) = \max \{ \|x_{vk} - x_{pk}\| : v \neq p\}$ is the diameter of C_k.

Observe that this would yield a more crude estimate than the upper bound given by (4.28); on the other hand, (4.29) requires somewhat less calculation per iteration than (4.28). For practical implementations several interior points of C_k can be additionally evaluated and included into the estimations (4.28) or (4.29), usually improving the bounds.

The sharpest upper bound given all information of (4.26) can be found by solving explicitly the following problem (for convenience leaving out index k):

$$\begin{aligned} &\text{maximize } \{z\} \\ &\text{subject to} \\ &x \in C \\ &z \leq g_i(x_p) + L_i\|x - x_p\| \quad p = 1,\dots,n. \end{aligned} \tag{4.30}$$

Meewella and Mayne (1988) approximate (4.30) by a piecewise linear problem replacing $\| \cdot \|$ by the infinite norm. They apply further rectangular subsets C_k storing all 2^n-vertices. In every iteration they solve 2^n-problems.

The n-dimensional variant of (4.30), in which C_k is defined by n+1 vertices, defines the problem of finding the highest point of a "turned around pommes-frites bag" and was studied a.o. by Mladineo (1986). The maximum point x_i of (4.30) in most cases can be found by solving n linear equations given by (leaving out k again):

$$(x_p - x_t)'x = (x_p - x_t)'[\frac{x_p + x_t}{2} + \frac{g_i(x_p) - g_i(x_t)}{2L_{ik}\|x_p - x_t\|}(x_p - x_t)], \quad p \neq t,$$

in which $x_t = \text{argmin}_{x_p}\{g_i(x_p)\}$, x_p the vertices of simplex C_k, $p = 1, \dots, n+1$. Solving (4.30) is less laborious, if regular simplices are used. Relations for this can be found in the bracketing procedures and geometrical observations of Wood (1992) and Baritompa (1993).

Besides the Lipschitzian property which has been elaborated thus far, one can make use directly of the quadratic structure of $g_i(x)$. If the Hessean Q_i is negative semi-definite, the maximum of $g_i(x)$ over C_k can be found by a local search algorithm; there is one global optimum. In the situation where Q_i is positive semi-definite, an affine over estimation based on Horst (1986) can be used, as has been illustrated in 4.4.2. The possibility of Q_i to be indefinite makes the problem of finding a valid upper bound more complex. A possible approach is to "make" $g_i(x)$ concave by replacing $\frac{1}{2}x'Q_ix$ by $\frac{1}{2}\mu_i\|x\|^2$, in which μ_i is the most negative eigenvalue of Q_i. An overestimating concave function Θ_{ik} can be found in the following way.

Let X_k be the n by n matrix with the vertices of C_k as columns. The majorant:

$$\Theta_{ik} = \beta'_{ik}x + \tfrac{1}{2}\mu_i\|x\|^2$$

can be found by solving $\beta_{ik} = (X_k^{-1})h_k$, in which the n-vector h_k gives the difference between $g_i(x)$ and the concave function $\mu_i\|x\|^2$ in the vertices of C_k:

$$h_{pk} = g_i(x_{pk}) - \mu_i\|x_{pk}\|^2 .$$

The majorant $\Theta_{ik}(x)$ equals $g_i(x)$ in the vertices of C_k, further on, it is concave. The upper bound can then be calculated by determining the maximum of Θ_{ik} over C_k. This way of determining the upper bound requires the solution of an unimodal optimization problem at every iteration.

A more sophisticated elaboration of the convex-concave splitting of a quadratic function based on an eigenvalue decomposition has been mentioned several times in this work and is due to Pardalos et al. (1985). Another idea which has been elaborated and illustrated in earlier sections is due to the bilinear view (see Al-Khayyal, 1990).

Illustration

We will illustrate the performance of the algorithm based on upper bound calculation (4.28) now and discuss some implementation aspects. Application of the branch-and-bound algorithm to the rum-coke example (4.1.3) results in a partition of S as is indicated in Figure 4.12. After 29 iterations, 26 points have been evaluated, a feasible point in F is found and two subsets have been deleted. The feasible point found is $x^* = (0.5, 0.375, 0.125)$ with "acceptable" properties expressed by $y_1(x^*) = -1$ and $y_2(x^*) = 0.281$.

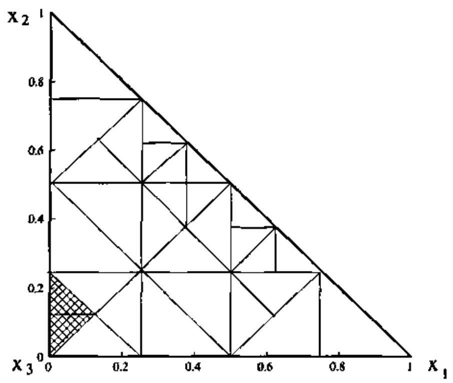

Figure 4.12: Partition after 29 iterations for the rum-coke example

An **implementation aspect** for the mixture design routine is that by the symmetry of the partition, a generated point may be used as a splitting point several times, as can be seen in Figure 4.12. It is important that such a point is evaluated only once and will not be added again to the list of points, which occupies most of the memory. Applying this simple idea, only a single new point which is not already part of the search information, is to be evaluated at every iteration cycle.

According to our numerical experience, the algorithm suggested can solve problems with a few variables (say, up to $n = 5$) in several hundred iteration steps. This will be illustrated by the following example.

Example
We consider a test problem originating from a practical application with $n = 3$ components and $m = 5$ properties. The coefficients of $y_i(x)$ can be found in Appendix 4.B. Assume that the following requirements are given for the properties:
$y_1 \leq 1.496$, $y_2 \geq .92$, $y_3 \leq 10$, $y_4 \geq 179$, $y_5 \leq 85$.
Applying the branch-and-bound algorithm, a feasible solution is found after 80 new points have been generated. The solution found is given by (the components sum to 100%):

$$x = (37.5, 43.75, 18.75) \text{ and } y(x) = (1.48, .921, 9.114, 180.77, -42.81)$$

Naturally, it is possible to generate more feasible points by not terminating the algorithm. For the problem given, this action required 100 points to be evaluated additionally, in order to find 10 more feasible points. Note that it can be more easy to carry out an "exhaustive" search say, in a ball around the first solution found to estimate how large one of the connected components of the set $S \cap F$ is (is discussed in 6.3). The fact that the number of iterations becomes larger, if the maximum of (4.25) is a small negative number, can be illustrated by modifying the above example as follows:

Let $0 \leq y_1 \leq 85$, $y_2 \geq .963$; the other requirements are left unchanged. The algorithm needs 56 iterations to conclude that the problem does not have a solution. If the problem is "near to feasibility" e.g. $y_2 \geq .94$, then concluding that there is no solution needs 157 iterations; if $y_2 \geq .93$, the "no solution" conclusion requires 239 iterations. Having these simple examples in mind, it may be intuitively clear that such "bad" cases can be very hard to solve by branch-and-bound techniques in higher dimensions.

In a decision support environment it is useful to provide the user with information on the performance of the algorithm. Compared to the solution approaches as discussed in 4.3.2, the elegance of the branch-and-bound approach is the guarantee character. After a successful finish, the existence of a feasible solution for the design problem is identified. The drawback of the illustrated approach is its need for computer memory. Therefore it is useful to inform the user on the number of subsets and points kept in memory. Other indicators are the objective in terms of (4.24) or (4.25) of the least-worst product which has been found thus far and the largest sum of the upper bounds (step 3) which is diminishing during the iterations.

4.5. *Finding robust solutions for product design problems*

4.5.1. Introduction

The product design problem, formulated as finding a point in a feasible area defined by inequalities, was introduced in earlier paragraphs. We now focus further on the ideas introduced at the practice of generating a DSS for finding robust solutions of the design problems. First the notion of robust solutions of design problems is discussed. We have to be precise with this definition as it concerns the variation in the decision variables and not the variation in the external data, which is common in literature on optimal control. Therefore, a mathematical description of the problem is provided subsequently. Now again the mathematical structures can be used on linear and quadratic functions to derive specific algorithms. Topics on linear and quadratic properties are discussed separately as well as the topic of finding robust mixture products (see 4.3.2).

Let us first reintroduce the concepts of the product design problem in mathematical terms. The vector $x \in \mathbb{R}^n$ is called a product with factor variables or components x_j. This product should be situated in the so called experimental area given by the polytope X, which represents the technical possibilities for the values of the variables x_j. A convenient concept is the range R_j of the variable x_j, defined as the difference between the upper and lower bound of x_j. The properties of a product x are formalised by the functions $y_i(x)$, $i=1,..,m$. In this chapter the product design problem is assumed to be formulated in the inequality form: find a product x in the experimental area X fulfilling m requirements on the properties of the form $y_i(x) \leq b_i$, $i=1,..,m$. Let the slack functions $g_i(x)$ be defined as $g_i(x) = b_i - y_i(x)$, $i=1,..,m$, then $F := \{x \in \mathbb{R}^n \mid g_i(x) \geq 0, i=1,..,m\}$ represents the set of products meeting all requirements on the properties.

The terminology of design problems and robust solutions can be found in Parkinson et al. (1990). They distinguish in their paper strictly between controllable (design variables, factor variables) and uncontrollable (external) parameters. When there is variation in the uncontrollable parameters, there exists a probability that the design may appear to be infeasible. In this section we concentrate on variation in the controllable variables. After the feasible design x has been found, it appears that due to fluctuations in the production, the realised product might not alwys be feasible, may not fulfil the requirements.

The research question on this topic originates from a practical problem in chemical industry, in which it appeared that the tolerance of equipment used for the production made it hard to produce a recipe (design) exactly. An analogous problem may be found in Kristindottir et al. (1993) who describe the design of laminates in aircraft industry, which also can only be produced with given tolerances. A further investigation was stimulated by our contribution to a project on the design of electronic circuits for an electronics company. The calculated values for parameters such as the capacity of a capacitor and value for the resistance of an included resistor, are in practice varying between ranges when the circuit is assembled. This means that a part of the production might not fulfil the requirements on the performance exactly. By taking into account the variations one can obtain a kind of 'confidence optimization'. Other key-words which are used in this context are: reliability, risk analysis and tolerance optimization.

In this section methods are developed to find designs which still lead to products meeting all requirements although during production small deviations are made from the recipe. The words 'deviation' and 'small' are formalized mathematically. Robustness of a design is defined as the (maximum) size of the deviation from this design that can be made in such a way that the product still meets all requirements on the properties. If the products under consideration are so called mixture designs (the sum of the components equals 1), the concept of robustness needs a special mathematical treatment.

This concept of robustness is related to (but differs from) the probabilistic concept which one can find in risk analysis such as in environmental engineering (another target user group). When designing environmental systems, e.g. a waste water treatment plant, one is interested in the probability that it fails to perform correctly. The design parameter is seen as an average of a random variable, see Bjerager (1988) and Liu and Der Kiureghian (1991). One is interested in the probability that a model output exceeds a critical value. The analogy of our term infeasible area is called the failure zone, or failure area. Computation of this probability is identical to integration of the joint density function of all stochastic parameters over that region of the stochastic parameter space where a criterion is exceeded. The analogy of this concept in product design is that one tries to minimize the number of products which do not fulfil the requirements when one design is produced frequently, such as in electronics industry. The robustness concept we formulated, differs slightly from this probabilistic idea. The similarity in the two concepts is worked out when the mathematical translation is discussed.

In the following section the idea of robustness and the problem of finding the most robust solution is introduced and formalized in a mathematical way. To clarify the ideas, examples in $\mathbb{R}^2$ are given. Further topics are the consequences of the robustness concept for the mixture design problem, the possibility to solve large problems when the properties are linear (affine) functions and the difficulty of the idea when having quadratic properties.

4.5.2. Mathematical formulation of robustness

The vague term 'small deviation from the design' means that not design x is produced, but a product with a certain deviation in one or more factor variables x_j. A design x has a certain robustness, if this deviation does not lead to an infeasible product; a vector outside set F.

Let $d \in \mathbb{R}^n$ be a vector with length 1, $\|d\|=1$, representing a possible direction of the deviation and T represent the size of the deviation, then $x+Td$ should be situated in set F. The robustness $T(x)$ of a feasible design $x \in X \cap F$ is formulated as:

$$\begin{aligned} &\max \{T\} \\ &g_i(x+Td) \geq 0 \qquad i=1,..,m \\ &\text{for all } \|d\| \leq 1 \end{aligned} \tag{4.31}$$

The problem of finding the feasible design with the largest robustness can be formulated as:

$$\max T(x) \quad x \in X \cap F \tag{4.32}$$

The choice of the norm of vector d determines the form of the area in which the deviations are allowed to take place. Three cases will be considered. The robustness corresponding to:

a) the 1-norm, $\|d\|_1 = \Sigma_j \, |d_j|$

b) the infinite norm, $\|d\|_\infty = \max_j \, |d_j|$

c) the 2-norm, $\|d\|_2 = \sqrt{\Sigma_j \, d_j^2}$

a) Consider the 1-norm, $\|d\|_1 = \Sigma_j \, |d_j|$. The value of the maximum deviation T_j of a factor variable x_j can be determined, such that $x \pm T_j e_j \in F$. The vector e_j is the j-th unit vector. For the linear (or in general convex) case, the value $T = \min_j T_j$ will be called the one-component robustness or the 1-norm robustness, because it focuses on deviations in one design variable x_j at a time. The value T is the maximum deviation possible in all directions d with $\|d\|_1 \leq 1$. The example illustrates the robustness of a feasible point. The vertices $x \pm T_j e_j$ should be in F.

Example problem P
Given the linear design problem P:

Let $X = \mathbb{R}^n$

$$\begin{aligned} y_1 &= -2x_1 - 3x_2 && \leq -0.6 \\ y_2 &= 3x_2 && \leq 0.45 \\ y_3 &= 5x_1 + 5x_2 && \leq 1.5 \end{aligned}$$

Let set F be the feasible region of problem P. A feasible point of this problem is x=(0.14, 0.13). The deviations which are allowed to occur in the factor variables, x_1 respectively x_2, while the product remains an element of F, are given in Figure 4.13. The 1-norm robustness T(x)=0.020 of x is equal to the minimum of

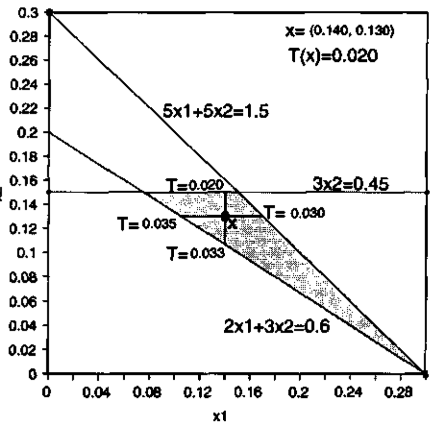

Figure 4.13: The 1-norm robustness T(x)=0.020 of the feasible point x=(0.14, 0.13)

these deviations. The feasible point with the best 1-norm robustness is the solution of problem (4.33):

$$\begin{aligned} &\max \{T\} \\ &x \in X \cap F \qquad (4.33) \\ &g_i(x+Td) \geq 0 \;\; i=1,..,m \\ &\text{for all } \|d\|_1 \leq 1 \end{aligned}$$

The solution of problem (4.33) is the feasible product with the largest 45° rotated square (in $\mathbb{R}^2$) around x which is subset of F. For problem P this is product x=(0.150, 0.125). The 1-norm robustness for this feasible point is 0.025 (see Figure 4.14).

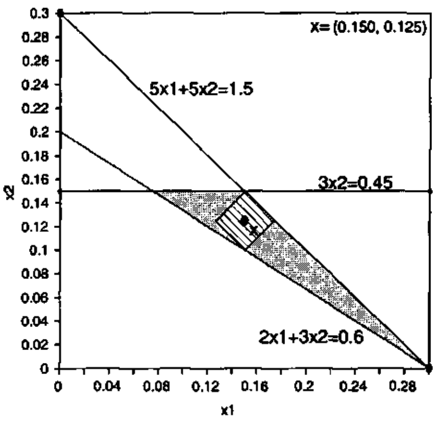

Figure 4.14: The feasible point x=(0.150, 0.125) with the best 1-norm robustness T(x)=0.025

A variant of the 1-norm robustness is the scaled one norm robustness in which the ranges R_j are used as weights: $\|d\|=\Sigma|d_j|/R_j$. The feasible point with the best scaled robustness is the solution of problem (4.34):

$$\begin{aligned} &\max \{T\} \\ &x \in X \cap F \qquad\qquad (4.34) \\ &g_i(x+Td) \geq 0 \;\; i=1,..,m \\ &\text{for all } d \text{ with } \Sigma|d_j|/R_j \leq 1 \end{aligned}$$

The optimal solution of (4.34) for problem P is the feasible product with the largest area around it, which is completely part of the feasible area. The shape of this area is a rhomb. Suppose the range R_1 of x_1 is twice as big as the range R_2 of x_2,then the allowed deviation in the direction of x_1 is twice the allowed deviation in the direction of x_2. The feasible point x=(0.127, 0.138) has the largest robustness in set F (see Figure 4.15).

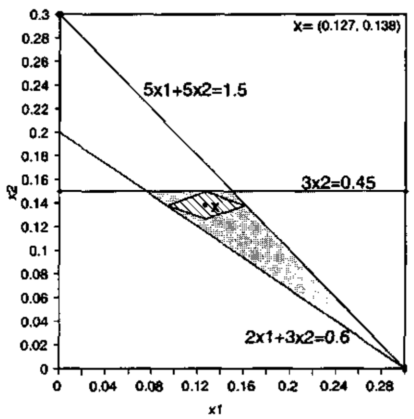

Figure 4.15: The feasible point x=(0.127, 0.138) with the best scaled 1-norm robustness

b) If the infinite norm: $\|d\|_\infty = \max_j |d_j|$ is considered, the robustness T(x) corresponds to the largest square (hypercube) around x. Consequence of this choice is that deviations may take place in all variables simultaneously and the components d_j of the vector of change d can vary between -1 and 1. The corresponding maximum value of T will be called the simultaneous or ∞-norm robustness of x. For the linear case the whole area defined by $x+Td$, $\|d\|_\infty \leq 1$ is subset of F if its 2^n vertices are element of F.

The feasible point with the best ∞-norm robustness is the solution of problem (4.35):

$$\begin{aligned}&\max\ \{T\}\\ &x \in X \cap F \qquad (4.35)\\ &g_i(x+Td) \geq 0 \quad i=1,..,m\\ &\text{for all } \|d\|_\infty \leq 1\end{aligned}$$

The optimal solution of (4.35) for problem P is x=(0.135, 0.135); the feasible point with the largest square around it, which is subset of F (see Figure 4.16).

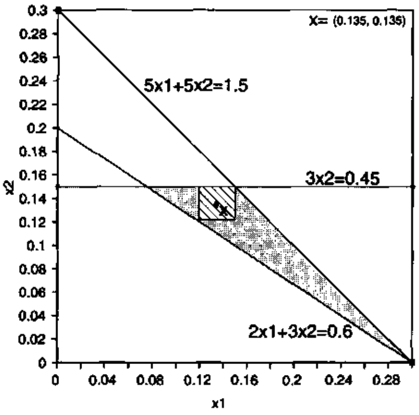

Figure 4.16: The feasible point x=(0.135, 0.135) with the best infinite norm robustness T(x)=0.010

If not the ∞-norm is used, but a scaled variant $\|d\| = \max_j |d_j|/R_j$, the robustness corresponds to the largest (hyper)rectangle around x. The ratio of the lengths of the sides of the rectangle depends on the ratio of the ranges R_j of the factor variables. Kristindottir et al. (1993) use the same concept of tolerance around a given design. By interval arithmetic they construct a hyperrectangle around a given design in which all points are feasible. They allow different deviations T_j for every factor variable x_j, whereas we focus on one (global) maximum deviation T which applies in every direction.

c) If the 2-norm: $\|d\|_2 = \sqrt{\Sigma_j d_j^2}$ is considered, the robustness T(x) corresponds to the largest circle around x. The feasible point with the best Euclidean or 2-norm robustness is the solution of problem (4.36):

$$\begin{aligned}&\max\ \{T\}\\ &x \in X \cap F \qquad\qquad (4.36)\\ &g_i(x+Td) \geq 0 \quad i=1,..,m\\ &\text{for all } \|d\|_2 \leq 1\end{aligned}$$

The optimal solution of (4.36) for problem P is x=(0.142, 0.130); the feasible product with the largest circle around it which is subset of area F (see Figure 4.17). Problem (4.36) is known in literature as the inscribed sphere problem and is related to the so called weighted distance problem in flexible programming (de Vet, 1980). The general problem of finding the maximum volume of a body with a given shape within a given set, is called the design centring problem (Horst and Tuy, 1990).

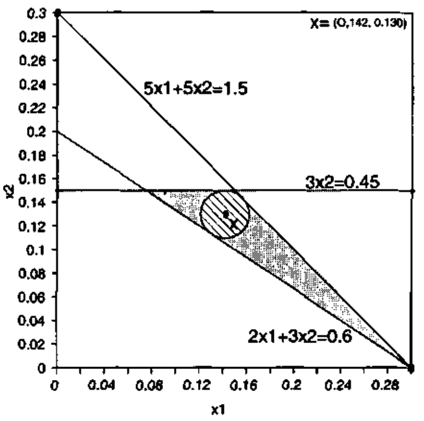

Figure 4.17: The feasible point x=(0.142, 0.130) with the best Euclidean robustness T(x)=0.020

If not the 2-norm is used but the scaled variant $\|d\|=\sqrt{\Sigma_j(d_j/R_j)^2}$ the robustness corresponds to the largest ellipse around x. The idea of finding the largest ellipse relates to the probabilistic view on robustness and reliability. Each parameter x_j, can be seen as the average of a stochastic variate with variance σ_j^2. Often the inaccuracies, deviations in the individual design variables can been seen as independent. In the design of electronic circuits, the deviation of one resistor from the average has nothing to do with another resistor, which can be from a completely different source. When the distribution of the mistakes, errors, is assumed to be normally distributed, the confidence region is an ellipsoid. When the deviates are independent the axes of the ellipsoid coincide with the Euclidean axes and the size is proportional to $1/\sigma_j$. Using $1/\sigma_j$ as weights in (4.36) leads to a reliability optimization in the sense that we are looking for the maximum volume confidence region enclosed by set F:

$$\|d\|=\sqrt{\Sigma_j(d_j/\sigma_j)^2} \tag{4.37}$$

Note that the corresponding probability mass is smaller than the integration of the joint density function over the total feasible area, as in general, there will be some volume left which is in F and which is not a part of the ellipsoid. So by using (4.37) in (4.36) a lower bound is optimized of the probability that good (feasible) products are produced.

The first implementations of optimizing the robustness with (4.31) and (4.32) in chemical industry were successful. Practical solutions were obtained with a safeguard against inaccuracies in the production process. The practical errors have a kind of uniform behaviour; application of the infinite norm and Euclidean norm were appropriate. In electronics industry however, the inaccuracies are due to tolerances in the assembled components and tend to be proportional to the size of the design value (e.g. resistance and capacity):

$$\sigma_j = r_j x_j \,. \tag{4.38}$$

This so called *proportional tolerance* property, complicates the search for robust products with (4.37) even more. We will discuss some mathematical properties in the following sections.

Another topic is the mathematical treatment of the concept of robust products for mixture designs. For a mixture design the condition $\Sigma x_j = 1$ holds. In the practical chemical problem, which stimulated this research, it appeared that tolerances in the production process made it hardly possible to produce the designed mixture exactly. If deviations in the production process occur in the non-mixture case, it has to be checked if the changed product $x+Td$ still fulfils the requirements on the properties. However in the mixture design case $x+Td$ probably does not satisfy $\Sigma(x_j+Td_j)=1$. For the mixture design problem the properties y_i are in fact defined on the ratio's $x_j/\Sigma x_j$. So for the linear case, a deviation Δx_j in one of the components x_j does not only have a direct effect $a_{ij}\Delta x_j$ on property i, but also an indirect effect on all properties (including i), since the value of Σx_j will change. This will be illustrated with an adaptation of example problem P. The problem is adapted such that one extra factor variable x_3 and an extra restriction $\Sigma x_j = 1$ is added. This results in mixture design problem P':

Example problem P'

$$\begin{array}{ll} y_1 = -2x_1 - 3x_2 & \leq -0.6 \\ y_2 = \;\; 3x_2 & \leq 0.45 \\ y_3 = \;\; 5x_1 + 5x_2 & \leq 1.5 \\ x_1 + x_2 + x_3 = 1 & \end{array} \tag{P'}$$

The point x=(0.14, 0.13, 0.73) is a feasible design of problem P'. If a deviation in the dosage of x_2 in positive direction (an overdose) of 0.03 occurs, the changed product will be: $(x+0.03e_2)$ and the properties are defined on $(x+0.03e_2)/(1.03)$ = (0.136, 0.155, 0.708). The robustness T(x) of design x is defined as the (maximum) size of the deviations that can occur in such a way that the product still meets all requirements on the properties. The deviations in the factor variables which are allowed to occur, considering the design x of problem P' are given in Table 4.6.
The 1-norm robustness of a solution is equal to the maximum deviation which is allowed to occur in every direction x_j. For x of problem P' the robustness T(x) is equal to 0.024 (see Table 4.6).

Table 4.6: Allowed deviations in the factor variables for x=(0.14, 0.13, 0.73)

component	x_1	x_2	x_3
max. dev. in pos. direction	0.024	0.043	0.117
max. dev. in neg. direction	0.029	0.050	0.100

If in the case of a mixture design problem deviations from the design during production occur, the changed product still fulfils the requirements if:

$$g_i((x+Td)/(1+T\Sigma d_j)) \geq 0,\ i=1,..,m.$$

Let set G be defined as:

$$G:=\{x\in \mathbb{R}^n \mid g_i(x/\Sigma x_j)\geq 0,\ i=1,..,m\} = \{x\in \mathbb{R}^n \mid g_i(x/e'x)\geq 0,\ i=1,..,m\}$$

with e'=(1,1,..,1).
The changed product $x+Td$ has to be an element of set G. Robustness T(x) of a mixture design can analogous to definition (1) be defined. Let $x\in X\cap G$, $\Sigma x_j=1$ be a feasible mixture design. T(x) is defined as:

$$\begin{array}{l}\max\{T\}\\ g_i((x+Td)/(1+T\Sigma d_j)) \geq 0,\ i=1,..,m\\ \|d\|\leq 1.\end{array} \tag{4.39}$$

The robustness T(x) is the size of the largest area (corresponding to the chosen norm) around x which still fits in G. The solution with the best robustness can analogously to definition (2) be found by solving:

$$\begin{array}{l}\text{Max T}(x)\\ x\in X\cap G\\ \Sigma x_j=1\end{array} \tag{4.40}$$

Analogously to the nonmixture case, for the 1-norm, ∞-norm and Euclidean norm robustness the corresponding norms should be used in (4.39) and (4.40).

First the mathematical structure of the derived problems is analyzed for the cases where the properties y_i are linear and quadratic functions. Those functions are interesting when the design problem has been derived by quadratic regression models as discussed in 2.5 and 4.3.2. This is followed by a discussion of possible solution approaches.

Summarizing theorem

Let $\|d\|_p = (\Sigma_j d_j^p)^{1/p}$ in definition (4.31). The solution with the best p-norm robustness can be found by solving LP problem:

$$\begin{array}{ll} \max \{T\} & \\ x \in X & \\ Ax + g = b & \\ g_i / (\Sigma_j d_j^q)^{1/q} \geq T & i=1,..,m \\ \text{with } q = p/(p-1) & \end{array}$$

Theorems 4.4 and 4.5 are a limit situation for the limit $p\downarrow 1$ and $p\rightarrow\infty$ and are proven on the bases of the feasibility of the vertices of the area defined by $\|d\|\leq 1$. The proof of the summarizing theorem is based on the explicit expression which is available for the point of contact z_i. This is shown in Appendix 4.C.

The theorems show that it is possible to formulate and solve even large linear design problems including the robustness concept. Moreover the theorems demonstrate that robustness seen as tolerance in the controllable parameters leads to including weighted slacks in the model formulation. When there is variation, uncertainty in an uncontrollable parameter such as b_i, the slack variables in general also play a central role, which can be observed in literature on stochastic programming (See Kall and Wallace 1994).

It has also be shown in Hendrix et al. (1993, 1996) how the linear problems do not become more complicated when considering the mixture design case. Analogous theorems can be derived for the linear mixture design problem. For the linear design problem the requirements on the (linear) properties imply that G defines a cone:

$$G := \{x \in \mathbb{R}^n \mid A(x/e'x) \leq b\} = \{x \in \mathbb{R}^n \mid (b_i e - a_i)'x \geq 0,\ i=1,..,m\}.$$

So finding the solution with the maximum robustness can be seen as: finding the largest area (depending on the chosen norm), which still fits in cone G, with its centre satisfying $\Sigma x_j = 1$. Theorems 4.4, 4.5 and 4.6 can simply be extended for the mixture design problem resulting in the following theorems (see Hendrix et al.,1996), which are proven in Appendix 4.C.

Theorem 4.7

Let $\|d\|_1 = \Sigma_j |d_j|$ in (4.39). The solution with the best 1-norm robustness can be found by solving LP problem P4:

$$\begin{array}{ll} \max \{T\} & \\ x \in X & \\ \Sigma x_j = 1 & \\ Ax + g = b & \\ g_i / \max_j |a_{ij} - b_i| \geq T & i=1,..,m \end{array} \qquad \text{(P4)}$$

Theorem 4.8

Let $\|d\|_\infty = \max_j |d_j|$ in definition (4.39). The solution with the best ∞-norm robustness can be found by solving LP problem P5:

$$\begin{aligned} &\max \{T\} \\ &x \in X \\ &\Sigma x_j = 1 \\ &Ax + g = b \\ &g_i / \Sigma_j |a_{ij} - b_i| \geq T \quad i=1,..,m \end{aligned} \tag{P5}$$

Theorem 4.9

Let $\|d\|_2 = \sqrt{\Sigma_j d_j^2}$ in definition (4.39). The solution with the best 2-norm robustness can be found by solving LP problem P6:

$$\begin{aligned} &\max \{T\} \\ &x \in X \\ &\Sigma x_j = 1 \\ &Ax + g = b \\ &g_i / \sqrt{\Sigma_j (a_{ij} - b_i)^2} \geq T \quad i=1,..,m \end{aligned} \tag{P6}$$

The results show that including robustness in a linear mixture design, can simply be solved by weighting the slacks in a correct way. It is possible to solve large mixture design problems including the robustness criterion. The same does not apply for the quadratic design problem.

Robust products and nonconvex properties

In Sections 2.5 and 4.3 it has been shown that in practical applications often quadratic functions for the properties are used, because it is common to use quadratic regression models to establish the relations between design and properties. The robustness problem becomes harder to solve when the properties $y_i(x)$ and the corresponding slack functions $g_i(x)$ are nonconvex functions.

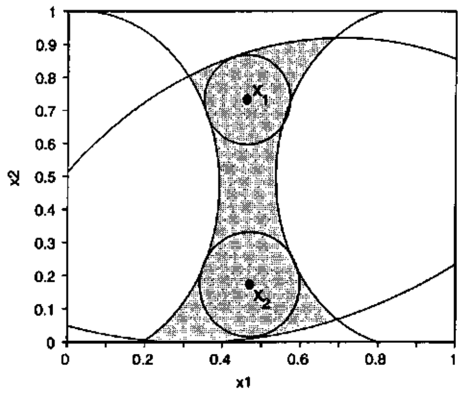

Figure 4.18: Multiple local optima of the maximum robustness problem

In Section 4.3.2, it has been discussed and illustrated that the design problem with quadratic properties is is not more complicated. Considering a transformed space with variables $\hat{x}_j = x_j/\sigma_j$, the inequality $\Sigma a_{ij}x_j \le b_i$ becomes $\Sigma\sigma_j a_{ij}\hat{x}_j \le b_i$. To formalise further, the matrix $D(x) = \mathrm{diag}(\sigma)$, the diagonal matrix with the tolerance vector on the diagonal, is introduced. The robustness problem with the weighted norm (4.44) becomes a problem with the standard Euclidean norm in the transformed space with

$$\hat{A}\hat{x} \le b,\ \hat{A} = D(x)A,\ \hat{x} = D^{-1}(x)x.$$

This means that theorem 4.6 applies in the transformed space. So the robustness of a point is given by

$$T(x) = \min_i \frac{b_i - \hat{a}_i\hat{x}}{\|\hat{a}_i\|} = \min_i \frac{b_i - a_i'x}{\sqrt{\Sigma_j(\sigma_j a_{ij})^2}} \tag{4.45}$$

The idea of transforming the space is used further in the derivation of algorithms.

For fixed tolerances the most robust product (linear properties) can be found by linear programming. A base algorithm is derived now for this relatively simple case, which will be extended for more complex cases. First, it is useful to introduce the concept of *active constraints*. In the discussion of the minimum volume hyperrectangle problem (section 3.5), the idea was used to call those points active which are responsible for building the volume and which are touched by the hyperrectangle. The same thing can be done for the maximum sphere problem given by (4.45). The (index) set of active constraints $I(x)$ is given by those constraints which are touched by the sphere around x:

$$T(x) = \|x - z_i\|, \qquad i \in I(x)$$

where z_i are the points where the sphere or ellipse touches requirement i.
More robust solutions can be found by "walking away" from the active constraints. In this way an improving direction d_k can be constructed and by linesearch a better product can be found. There are several points to take into account when constructing such an algorithm.

- The iterate x_k can reach the boundary of the experimental region X. The search direction should be adapted.
- At a certain moment the gradients $\nabla g_i(x_k)$ of the active constraints $I(x_k)$ can form n+1 affine independent vectors. This means that x_k is "pushed in all directions simultaneously" and has reached an equilibrium.

For a practical algorithm both points should be considered for the construction of stopping criteria. We will leave out those details in the outline of a base algorithm. A base algorithm is presented intended for linear properties and fixed tolerances.

Base algorithm linear properties, fixed tolerances

```
0.  Define stopping criteria, $k := 1$
1.  Find a feasible $x_1$ (LP)
2.  Determine $T(x_k)$ and $I_k = I(x_k)$
    $d_k := \sum_{i \in I_k} - a_i / \|a_i\|$
3.  $\lambda_k :=$ argmax$_\lambda$ $T(x_k + \lambda d_k)$
    $x_{k+1} := x_k + \lambda_k d_k$
4.  Check stopping criteria
    Either STOP or $k := k+1$ and go
    to step 2.
```

The structure of the algorithm is a general way of constructing nonlinear programming codes and will converge to the global optimum for this simple case, when implemented correctly. An improving direction d_k can be constructed by taking any convex combination of the gradients $-a_i$ which push the iterate away from the active constraints (border and affine independence considerations left out). In the base algorithm they are simply added. The algorithm is not useful for the base case itself (linear properties, fixed tolerances) as such, because that case can also be solved by LP.

Linear properties, proportional tolerances

Looking for the most robust point becomes more complicated when the tolerances are proportional, $\sigma_j = r_j x_j$, as occurs in electronics industry. The determination of the robustness of a product itself is still relatively easy when the properties are linear; (4.45) becomes (4.46).

$$T(x) = \min_i \frac{b_i - a_i' x}{\sqrt{\sum_j (r_j x_j a_{ij})^2}} \tag{4.46}$$

In the base algorithm the directions away from the active constraints should be adapted to account for the search for smaller parameter values ($|x_j|$ is smaller), because their tolerance is also smaller. Mathematically this can be derived from the tolerance $T_i(x)$ with respect to an active constraint i:

$$T_i(x) = \frac{b_i - a_i' x}{\sqrt{\sum_j (r_j x_j a_{ij})^2}}$$

Before finding an improving direction, we return to a question which has not been analyzed: Does $T_i(x)$ or $T(x)$ possibly contain multiple optima? At first sight, one would think so as there are implicitly two objectives when maximizing $T_i(x)$:

- One is looking for products far away from the boundary $g_i(x)=0$ (numerator),
- One is looking for small products, which have low variation (denominator).

However, it is impossible to construct an instance of the robustness problem with linear properties, proportional tolerance and several optima.

To explain this one has to go into the theory of fractional programming (Chapter 2 and Schaible, 1995). The fractional function $T_i(x)$ has a linear numerator and convex denominator. This leads to a so called quasiconcave function which contains one global maximum. The minimum of a set of quasiconcave functions (4.46) is again quasiconcave (see e.g. Bazaraa et al., 1993). This implies that when we follow the gradient in the base algorithm we will end up in the global maximum.

An improving direction with respect to i can be found by following the gradients $\nabla T_i(x_k)$ of the active constraints $i \in I_k$. The gradient $\nabla T_i(x_k)$ consists of the partial derivatives $\partial T_i(x)/\partial x_j$ which are proportional to

$$-a_{ij}(\sqrt{\Sigma(r_jx_ja_{ij})^2} + a_{ij}r_j^2x_jT(x)) \quad . \tag{4.47}$$

Compared to the derivative $-a_{ij}$ of the base case (fixed tolerances), following the derivative tends to a reduction of the values x_j for that components j with a larger tolerance factor r_j. To find the most robust solution, the search direction d_k in the base algorithm is adapted.

Algorithm linear properties, proportional tolerances

```
0.  Define stopping criteria, $k := 1$
1.  Find a feasible $x_1$ (LP)
2.  Determine $T(x_k)$ and $I_k = I(x_k)$
    $d_k := \sum_{i \in I_k} \nabla T_i(x_k)$
3.  $\lambda_k := \text{argmax}_\lambda\ T(x_k+\lambda d_k)$
    $x_{k+1} := x_k + \lambda_k d_k$
4.  Check stopping criteria
    Either STOP or $k := k+1$ and go
    to step 2.
```

Quadratic properties

The search for robust solutions when considering quadratic properties concerns several difficulties.

- Finding an initial solution in step 1 of the algorithm. As discussed in 4.3.2, it may be hard to find a feasible solution of the design problem.
- The determination of $T(x_k)$ is not straightforward.
- Specific improving directions for step 2 have to be derived.

Let us first consider the determination of the robustness for a given product x. As given by (4.41) we are looking for the point z_i, where the sphere or ellipsoid with centre x "touches" the surface $g_i(z) = z'Q_iz + d_i'z + c = 0$.

Again a transformation can be used to show that the problem with nonuniform tolerances is equivalent to a problem with uniform tolerances, the maximum sphere problem. Let again $D(x) = \text{diag}(\sigma)$. Now the ellipsoid problem

$$\begin{aligned}&\min\|x-z\|\\&g(z) = z'Qz + d'z + c = 0 \qquad (4.48)\\&\text{with}\quad \|x\| = \sqrt{\Sigma(x_j/\sigma_j)^2}\end{aligned}$$

can be transformed to a problem which searches the maximum sphere around the origin which touches a quadratic surface.

Let $\hat{z} = D^{-1}z - D^{-1}x$, then

$$g(z) = \hat{z}'\hat{Q}\hat{z} + \hat{d}'\hat{z} + \hat{c}$$

with

$$\begin{aligned}\hat{Q} &= DQD\\ \hat{d} &= 2DQx + Dd\\ \hat{c} &= x'Qx + d'x + c.\end{aligned}$$

Problem (4.48) is now equivalent to the maximum sphere problem (4.49):

$$\begin{aligned}&\min\|\hat{z}\|\\&\hat{z}'\hat{Q}\hat{z} + \hat{d}'z + \hat{c} = 0 \qquad (4.49)\\&\text{with}\quad \|x\| = \sqrt{\Sigma x_j^2}\text{ , the Euclidean distance.}\end{aligned}$$

In this way, it is sufficient to focus on finding a solution of (4.49).

The Karush–Kuhn–Tucker conditions (4.43) are for the specific case (4.49) equivalent to the set (4.50):

$$\begin{aligned}&z = -\mu\nabla g(z) \qquad (4.50)\\&g(z) = 0\end{aligned}$$

in which μ is a Lagrange multiplier.

An algorithm to obtain a solution of (4.50), closest to the origin, is given as follows.

Core of the algorithm to determine $T(x)$

```
0.  Give a stepsize $\gamma$ and a tolerance $\varepsilon$, $k := 1$
1.  Find an initial solution $z_k$ such that $g(z_k) = 0$
2.  Go into the direction $-\nabla g(z_k)$:
        $p_k := z_k - \gamma \nabla g(z_k)$
3.  Find the zero point $\lambda_k$ of $g(\lambda p_k) = 0$,
        $z_{k+1} := \lambda_k p_k$
4.  If $\|z_{k+1} - z_k\| < \varepsilon$          STOP
    else $k := k+1$ and go to step 2
    endif
```

The procedure is depicted in Figure 4.21. Only the main steps are given; some safeguards are needed to treat worst cases. For instance the determination of λ_k in step 3 is in general relatively easy as $g(z)$ is quadratic, $g(0) > 0$ (x is a feasible product) and $g(p_k) < 0$ in general. The smallest of the two solutions for λ_k can be chosen. A safeguard is needed when there is no solution (then $g(p_k) > 0$).

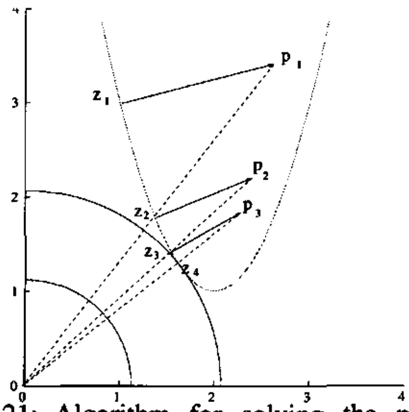

Figure 4.21: Algorithm for solving the maximum sphere problem, quadratic properties

There are several alternatives possible for an algorithm to find a solution of (4.50). In the literature on "structural reliability" we found the following approach developed by Hasofer and Lind (1974) and Rackwitz and Fiessler (1978), the HLRF-approach, see also for an overview Liu and Der Kiureghian (1991). In this approach, not a fixed stepsize γ is chosen, but a linesearch is performed in the direction

$$d_k = \frac{\nabla g(z_k)' z_k}{\|z_k\|^2} z_k - \nabla g(z_k)$$

which is the projection of the steepest descent on the orthoplement of z_k. Now p_k in step 1 of the algorithm corresponds to the line minimum $\min_\alpha g(z_k+\alpha d_k)$. The HRLF approach aims at finding the point closest to the origin of a general nonlinear surface. Every function evaluation may require a considerable amount of calculation time (Chapter 5). It is therefore surprising that a crude grid search is suggested in their approach to find an optimal value for α.

The suggested transformation and algorithm (or any alternative) gives the possibility to calculate the robustness $T(x)$ for quadratic properties for the weighted Euclidean distance case. Let us now return to the outer loop of the base algorithm to find the most robust design. Figure 4.18 illustrated that one can only aim at finding a local solution of (4.32). In the framework of the base algorithm, improving directions for step 2 have to be constructed to push the iterate away from the active constraints. Following the line from linear to quadratic design, the properties when having requirements $a_i'x \le b$ can be approximated by using the tangent hyperplane $\nabla g(z_i)'(z_i-x) \ge 0$.

For **fixed tolerances** simply a_i in step 2 of the base algorithm can be replaced by $\nabla g(z_{ik})$.

For **proportional tolerances** the derivatives (4.47) in the gradient $\nabla T_i(x_k)$ in the algorithm for linear properties and proportional tolerances are replaced by $\partial T_i(x)/\partial x_j$ which is proportional to:

$$-\frac{\partial g_i}{\partial x_j}(z_i)\left[\Sigma(r_j x_j a_{ij})^2 + \nabla g(z_{ik})'(z_i-x_k)\,r_j^2 x_{kj}\frac{\partial g_i}{\partial x_j}(z_i)\right] . \tag{4.51}$$

Although (4.51) seems very complicated it is relatively easy to calculate, as we are dealing with quadratic functions. In a computer implementation the points of contact z_i are available from the algorithm which solves (4.50) and the approximation of the hyperplane can be substituted in the subroutine which evaluates (4.47) to derive a search direction.

The algorithm was implemented and used for several design problems in a electronics industry leading to local optima for the robust design problem. Further analysis learns that worst cases can be constructed which frustrate the performance of the algorithm. The approach outlined here represents a specific local search procedure which can be used given various starting points. The structures of quadratic and linear properties and of the proportional tolerance model are fully exploited.

4.6. *Concluding remarks*

In this chapter several cases have been discussed to show the complete route of model formulation, analysis and construction of algorithms. Focus was on applying the Branch-and-Bound method. Several points can be concluded from the viewpoint of the potential user.

- Analysis of the expressions is required for the discovery of useful mathematical structures (Chapter 2).
- The elegance of the techniques is the guarantee that we are certain about the global optimality of the optimum when it has been discovered and verified.
- The methods are hard to implement. Thorough use should be made of special data structures to store the necessary information in memory.

The last point refers to the difficulty that in Branch-and-Bound methods (including integer programming) the computer memory may fill up, before the guarantee has been reached. In a practical computer implementation there are several instruments to reduce this problem.

- Classical instruments in Branch-and-Bound (arrows g1 and g2) are that we should keep track of the memory occupation by the number of subsets and that the selection rule can be influenced and alternative bounds can be generated.
- Another decision in the implementation is the trade-off which information to keep in memory and which to recalculate several times.
- A technical solution is to make clever decisions on which part of the Branch-and-Bound tree to store temporarily on hard disk and which part to keep directly available. This does not solve the overall capacity problem, but may speed up the search.
- A specific Global optimization feature is to make good use of special data structures linking subset information to information on evaluated points.

Cases

The *nutrient problem* is an example of a high dimensional problem for which a branch-and-bound approach with rectangular subsets has been outlined and illustrated. The problem is hard to solve in practice as one may not have sufficient storage capacity and/or time to get to the global optimum and to verify it. The similar so-called pooling problem has been a challenge for many researchers and despite the relevance for petrochemic industry, remains hard to solve. The nutrient problem also shows how analysis of a given problem can lead to many useful properties:

- boundary solutions of the problem
- successive LP is very unsuccessful
- standard NLP leads to many local not global optima.

The *(quadratic) design problem* can also be handled by a branch-and-bound procedure to guarantee the (non) existence of feasible designs. The quadratic and Lipschitzian structure contain value information for the derivation of bounds. The way the problem has been formulated requires no update from a global lower bound; when all slacks are positive a feasible solution has been found. The performance of the procedure can only be influenced by changing the selection rule, which subset is to be split next. This is similar to branch-and-bound procedures for integer programming. The information during the execution can be the lowest bound, and the number of subsets and points to be held in memory, to follow the storage in computer memory. For the final information we advise to show the best let say ten products which have been found. The algorithm has been built into a Decision Support System in combination with local search (traditional) algorithms.

A solution is often understood to be **robust**, when it remains "good" in uncertain situations due to uncertainty in data or changing circumstances. We discussed it in the context of product design, where robustness is defined as a measure of the error one can make from the solution such that the solution (product) is still acceptable. Looking for the most robust product is looking for that point which is as far away as possible from the boundaries of the feasible (acceptable) area. For the solution procedures, we had a look at the appearance of the problem in practice, where boundaries are given by linear and quadratic surfaces, properties of the product.

- For linear boundaries, finding the most robust solution is a Linear Programming problem and thus rather easy.
- For quadratic properties the development of specific algorithms is required. The user still should interact in delivering good starting designs, probably generated by the DSS of the design problem.

Appendices

Appendix 4.A

For each nutrient k the fractional programming problem $\max\limits_{x \in C_p} \varphi_k(x) := \dfrac{d_k' x}{q_k' x}$ has to be solved, where the demand and supply vectors (d and q) are strictly positive.

A well-known approach to solve fractional programming problems (FP): $\max_{x \in X} \dfrac{f(x)}{g(x)}$ is to consider the global optimization problem (GP): $\max_{x \in X}\{f(x) - \lambda g(x)\}$ where $\lambda \in \mathbb{R}$ (Pardalos and Phillips, 1991). The fundamental result which relates the (GP) problem to the (FP) problem is

Theorem (Dinkelbach, 1967) x^* solves the fractional programming problem (FP) if and only if x^* solves the global optimization problem (GP) with constant $\lambda^* = \dfrac{f(x^*)}{g(x^*)}$.

For our fractional programming problems, according to this theorem, $\lambda_k^* = \dfrac{d_k' x^*}{q_k' x^*}$.

For this constant λ_k^* the global optimization problem is the linear programming problem

$$\max_{x \in C_p} \Sigma(d_{ik} - \lambda_k^* q_{ik}) x_i \quad .$$

If $(d_{ik} - \lambda_k^* q_{ik}) < 0$ then it is optimal to decrease x_i as far as possible, ($x_i^* = l_{ip}$) and if $(d_{ik} - \lambda_k^* q_{ik}) > 0$ then it is optimal to increase x_i as far as possible ($x_i^* = u_{ip}$).

$$\text{So} \quad x_i^* = \begin{cases} l_{ip} & \text{if } \dfrac{d_{ik}}{q_{ik}} < \lambda_k^* \\[2ex] u_{ip} & \text{if } \dfrac{d_{ik}}{q_{ik}} \geq \lambda_k^* \end{cases} \quad .$$

For each k, λ_k^* can be seen as the threshold value, which determines in which corner of the box C_p the value x^* is obtained. An algorithm to find this x^* is described below:

Threshold Algorithm

Step 1	Fix all x_i at l_{ip} and calculate the objective value φ_k. Initiate φ_k^* as φ_k.
Step 2	Renumber all x_i, such that $\frac{d_{1k}}{q_{1k}} \le \frac{d_{2k}}{q_{2k}} \le \ldots \le \frac{d_{mk}}{q_{mk}}$. Pick up the first element of the sorted vector x.
Step 3	Put the chosen element x_i on its upper bound u_{ip} and calculate the new φ_k. If φ_k exceeds φ_k^* then go to Step 4 else put the chosen element back to its lower bound l_{ip} and STOP: φ_k^* is the optimum.
Step 4	Replace φ_k^* by the new φ_k and take the next element of the x vector, go to Step 3.

Appendix 4.B

The test example for the mixture design problem is based on the following data. The property y_i is calculated as:

$$
\begin{aligned}
y_i(x) = {} & c_i + d_{i1}\hat{x}_1 + d_{i2}\hat{x}_2 + d_{i3}\hat{x}_3 \\
& + q_{i12}\hat{x}_1\hat{x}_2 + q_{i13}\hat{x}_1\hat{x}_3 + q_{i23}\hat{x}_2\hat{x}_3 \\
& + q_{i11}\hat{x}_1^2 + q_{i22}\hat{x}_2^2 + q_{i33}\hat{x}_3^2
\end{aligned}
$$

in which $\hat{x}_j = x_j - 0.5$.

The data that are used are given by the following Table:

property i	c_i	d_{i1}	d_{i2}	d_{i3}	q_{i12}	q_{i13}	q_{i23}
1	1.495	−0.006	−0.024	0.050	−0.002	0.017	−0.021
2	0.951	−0.014	−0.048	0.108	−0.001	−0.004	0.006
3	15.986	4.729	−16.657	−8.974	−10.174	−21.977	−86.952
4	178.708	−0.687	12.800	−7.347	0.241	−4.947	−3.967
5	52.002	−5.217	−201.326	192.989	−6.180	337.106	1030.228

i	q_{i11}	q_{i22}	q_{i33}
1	0.001	0.008	−0.021
2	0.004	0.001	−0.014
3	20.605	32.003	−81.278
4	−0.766	−0.528	7.822
5	116.750	−67.424	−845.215

5.2. *The design of a continuous sugar centrifugal screen*

The first practical example is taken from a project in cooperation with a metallurgic firm which among others produces screens for sugar refiners. In the sense of Section 1.3 (Figure 1.2) a mathematical model has been constructed to describe the behaviour and performance of the process of separating sugar from molasses in a continuous sugar centrifugal. An optimization model has been derived (arrow b) from the model by distinguishing the parameters which can be influenced and the criteria to be optimized.

We first give a flavour of the mathematical model. The continuous centrifugal works as follows (Figure 5.2). The fluid (molasses) including the sugar crystals streams into the middle of the rotating basket. By the centrifugal force and the angle of the basket, the fluid streams uphill. The fluid goes through the slots in the screen whereas the crystals continue their way uphill loosing all fluid which is still sticking on the material. Finally the crystals are catched at the top of the basket. The constructed model describes the stream of the fluid from the start, down in the basket, until the end, top of the screen. The flux of the fluid through the screen does not only depend on the geometry of the slots, but also on the centrifugal force and height of the fluid film on a certain position. Reversely, the height depends on how quick the fluid goes through the screen, so how fast the height profile decreases. Without going into detail, this interrelation can be described by a set of differential equations which can be solved numerically. Other, simpler, relations were found to describe the strength of the screen, as wear is a big problem.

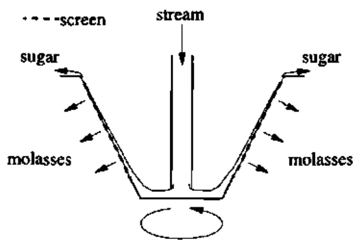

Figure 5.2: Continuous sugar centrifugal

In this way a model exists in the sense of Figure 1.2, which given technical data such as the size and angle of the basket, revolutions per second, the stream into the refiner, the viscosity of the material, the shape of the slots and the slot grid pattern, calculates the behaviour described by the fluid profile and the strength of the screen. Now an optimization problem is derived, by selecting at one side that parameters which can be influenced by the firm producing the screens, and at the other side the criteria that one intends to optimize. A part of the parametrization of the pattern is given in Figure 5.3. Another parameter was among others the thickness of the screen. Two criteria were formulated; one to describe the strength of the screen and one to measure the dryness of the resulting sugar crystals. There are several ways to combine the two criteria in a multicriteria approach as described in 4.3. Actually we are looking for several designs on the Pareto set describing screens which are strong and deliver dry sugar crystals when used in the refiner.

The prototype DSS to be described in Section 5.5. was used to perform local searches, to generate random points to be used in a global search, monitor the location of the local optima etc. Notice that every function evaluation requires invoking the subroutine which performs the numerical calculation of the differential equations describing the performance of the separating process. Closed form expressions are not available.

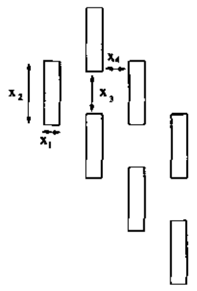

Figure 5.3: Parametrization of the slot grid pattern

Several designs were generated that were predicted to perform better than existing screens. The use of a mathematical model in this design context is very useful, because it is extremely difficult to do real life experiments. The approach followed here, lead to an advisory system to make statements on what screens to use in which situation. Furthermore it lead to insights for the design department which generated and tested a few new designs for the screens.

The optimization is a typical case where the user can apply his knowledge about the domain, the process of filtering sugar, during the execution of the algorithms. Parameters can be fixed, bounds can be changed, the global search can be influenced, local optimization can be stopped etc. In our experience, graphical information is also very useful. At one window, one can view the performance in the parameter space and at other windows one can monitor the predicted performance of a design which is evaluated.

5.3. *Dynamic decision making in water management*

The problems in this section involve sequential decision making. The performance, objective function, not only depends on the sequence of decisions, but also on fluctuating data over a given time period, often considered as a stochastic variable. The calculation of the objective typically requires the simulation of the behaviour of a system over a long period.

In the first example operating rules have to be derived for pumping water in a higher situated lake in the Netherlands. In general the rainfall exceeds the evaporation and the seepage. In summer however, water has to be pumped from lower areas and is treated to maintain a water level above the minimum with a good water quality. Not only the pumping, but certainly also the treatment to get out phosphate costs money. The treatment installation performs better when the stream is constant, so that the pumps should not be switched off and on too frequently. The behaviour of the system is given by equation

$$I_t = \min\{I_{t-1} + \xi_t + x_t, \text{Max}\} \tag{5.3}$$

with I_t: water level of the lake
ξ_t: natural inflow i.e. rainfall - seepage - evaporation
x_t: amount of water pumped into the lake.

When the water level reaches its maximum (Max), the superfluous water streams downwards through a canal system towards sea. For the case we studied, two pumps were planned to be installed, so that x_t only takes values in $\{0, B, 2B\}$, where B is the capacity of one pump. Decisions are taken on a daily basis. In water management, it is common practice to derive so-called operating rules, decision strategies including parameters. A decision rule instructs on what decision to make in which situation. An example is rule (5.4) with parameters β_1 and β_2.
For

$$\begin{array}{rl} I_t < \beta_1 & x_t = 2B \\ \beta_1 \le I_t \le \beta_2 & x_t = B \\ I_t \ge \beta_2 & x_t = 0. \end{array} \tag{5.4}$$

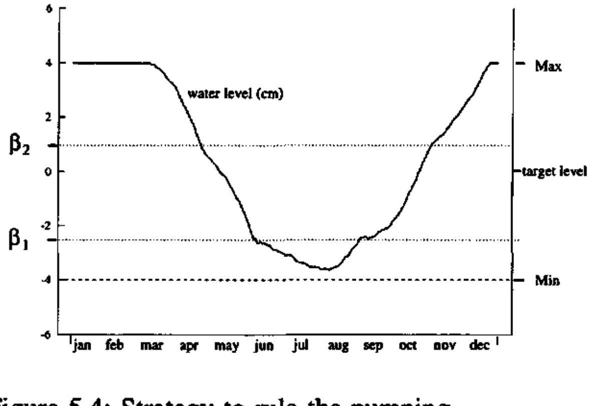

Figure 5.4: Strategy to rule the pumping

Given weather data of a certain period now the resulting behaviour of a sequence of decisions x_t can be evaluated by measuring performance indicators such as the amount of water pumped Σx_t and the number of switches of the pumps $\Sigma | x_t - x_{t-1} | /B$. Assessment of appropriate values for the parameters β_1 and β_2 is a black-box optimization problem in the sense of Figure 5.1. For every parameter set, the model (5.3) with strategy (5.4) can be simulated with weather data (rainfall and evaporation) of a certain time period. Some 20 years of data on open water evaporation and rainfall were available. The performance can be measured leading to one (multi-) objective function value. At every iteration the global optimization algorithm delivers a proposal for the parameter vector β, the model simulates the performance and after some time returns an objective function value $f(\beta)$.

One possibility is to create a stochastic model of the weather data and resulting ξ_t and to use the model to "generate" more years by *Monte Carlo simulation*, i.e. simulation using (pseudo) random numbers. In this way, it is possible to extend the simulation run over many years. The model run can be made arbitrary long. Notice that in our context it is useful for every parameter proposal to use the same set of random numbers (seed), otherwise the objective function $f(\beta)$ becomes a random variate.

Various strategies as variants of (5.4) can be formulated taking into account for instance the level of ξ_t, the increase in water level etc. with several parameters. The approach of optimizing the parameters by iteratively invoking Monte Carlo simulation remains the same. As can be derived from Figure 5.4, due to seasonal influences the underlying stochastic process is not stationary, so that decision rules such as (5.4) are too rigid. Actually the parameters should depend on time. In this specific case *Stochastic Dynamic Programming* can be used to optimize expected values of the criteria. The result is a so called operating Table. It tells exactly what actions should be undertaken in which situation in what period. Stochastic programming using hydraulic data is very popular in water management, reservoir planning (see e.g. Butcher, 1971). In multireservoir systems however, the state variables I_t in (5.3) are linked, which makes the application of Dynamic Programming very complicated. See for instance Kularathna (1992), who described the application for a large system in Sri Lanka.
Before elaborating further on the topic of multiple reservoir systems, first two other areas in which **parametrized decision strategies** play a role are mentioned.

In Management Science and Logistics a well known application is that of *Stochastic Inventory Control*. Equation (5.3) reads as follows (see Hax and Candea, 1984).

I_t: level of inventory
x_t: amount produced or ordered
ξ_t: (negative) demand, considered stochastic.

Criteria that play a role are inventory costs, production costs and backordering or out of stock costs. Depending on the construction of the parametrized decision rule similar to (5.4) names are used such as (B,Q)-policy or (s,S)-policy. In the context of optimizing oracle functions the optimization of the parameters can be carried out in the same way as in the water management problem. Given the data over the past, it can be evaluated how the system would have performed if the parameters would have had a certain value. This way of approaching the problem is not very common in inventory control. Usually assumptions are made about the (stationary) probability distribution of the demand and criteria are matched in such a way that by integrating over the distribution function explicit (closed form) expressions can be derived of how the objective function (expected criterion) depends on the parameters, thus leading to a simpler to solve optimization problem.

A totally other field where we came across the idea of parametrized strategies, is the *financial world* of the stock exchange. By making assumptions on the underlying stochastic processes of the stock price and an analysis of the price of options, researcher have been able to derive a so called optimal hedging strategy (see Black and Scholes, 1973). This strategy defines how much money an investor should put into various funds, stocks and options, in order to have a risk free portfolio. To keep the portfolio risk free, an investor should adapt the ratio continuously. In practice however, transaction costs are involved in selling and buying stocks and options, so that changing the portfolio continuously is infinitely expensive. We carried out a simulation study to evaluate various parametrized strategies on when to change the portfolio depending on state variables such as the deviation from the ideal portfolio and the expiration time of the options. The criteria are the risk involved, the expected final value of the portfolio and the expected transaction costs. To make accurate estimates, long simulation runs are necessary to evaluate only one set of parameters.

In all examples, there is a **dynamic decision situation** with a state variable, water level, inventory level and stockprice, and a **decision strategy with parameters**.

We return now to **multiple reservoir** systems which implies state variables I_{it} for every reservoir i. We discuss a study on a seven reservoir system in Northern Tunesia due to Milutin and Bogardi (1996) (see Figure 5.5). For every reservoir i, equation (5.3) defines the water volume I_{it} given stochastic river inflow ξ_t and release (decision) x_{it}. Operating strategies can be derived on a monthly basis taking into account the stochasticity of the river inflows for which 44 year data are available. As can be observed in Figure 5.5, there are six demand centres supplied by more than one reservoir. This means that the demand d_{jt} (demand centre j, month t) partly has to be fulfilled by a fraction of the release x_{it}, when reservoir i delivers to centre j. The decomposition approach chosen by Milutin and Bogardi is to use a fixed release distribution γ_{ij} among groups of reservoirs towards their common demand. The demand of water D_{ijt} of centre j from reservoir i is taken as

$$D_{ijt} = \gamma_{ij} \; d_{jt}$$

where γ_{ij} is thought to be fixed over the year, meaning that $\Sigma_i D_{ijt} = d_{jt}$ and $\Sigma_i \gamma_{ij} = 1$, the summation with respect to that reservoirs which deliver water to centre *j*. Given a distribution γ_{ij}, an operating rule can be derived for every reservoir to fulfil total water demand $\Sigma_j D_{ijt}$. The approach is to apply simple operating rules and to simulate their performance over a 44 year period of monthly inflow data. The objective is to minimize the expected value of the annual sum of squared deviations between releases and corresponding demand for water. The restriction that the distributions sum to unity, such as in the mixture design problem (4.3), can be dealt with in a penalty form. Optimization of the relative contributions γ_{ij}, involves iteratively calling a larger program (subroutine), which simulates the performance of operating strategies.

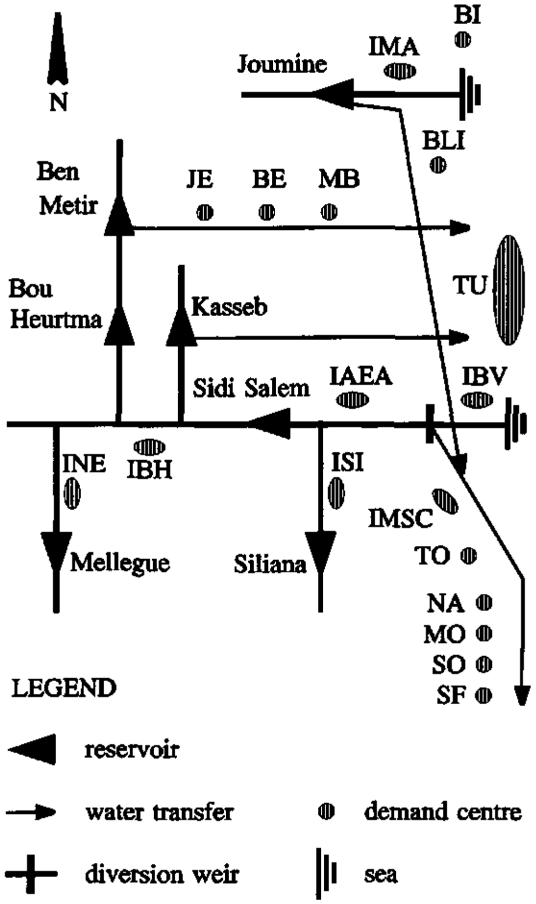

Figure 5.5: Seven reservoir system in Tunesia. Source: Milutin and Bogardi, 1996

Milutin and Bogardi applied genetic algorithms which can be seen as a random search to generate good distributions. Later on we used local searches in a multistart way, leading to several local optimal distributions. The user typically has a direct interpretation of the decision variables and can influence the search by fixing parameters and changing bounds. He can also increase the number of random points to scan the feasible area before starting a local search.

The cases in this section have shown optimization problems where evaluation of the objective function involves running a dynamic model with data either from a given time period in the past or from (extended) Monte Carlo simulation. The explicit dependence of objective function on parameters to be optimized is therefore hidden. A user, due to his knowledge of the modelled domain, may have a direct interpretation of the parameter values and can interact during the search for the global optimum.

5.4. *Multiple source river pollution management*

A combination of a continuous model description as in the sugar centrifugal case and a Monte Carlo simulation as in the water management problem can be found in Boon et al. (1989). They describe the so called waste load allocation problem, i.e. design of a system for wastewater treatment in a multiple source pollution of a river. The situation is sketched in Figure 5.6 (Pintér, 1996a). There are several pollution sources along the river, locations where discharges of untreated (domestic) wastewater take place. The pollution is expressed in levels of BOD (Biochemical Oxygen Demand), which can be used directly as a measure for the oxygen demand in the receiving stream. The management question is to design a cost effective allocation of treatment capacities such that water quality standards on oxygen are met. For the details we refer to Boon et al. (1989) and Pintér (1996a) and the numerous references therein.

For our purpose, it is sufficient to know that there are decision variables x_i describing the site specific treatment efficiency, BOD-load removal capacity, at site i, which can vary between prespecified technological bounds. There are convex cost functions $C_i(x_i)$ describing the costs of constructing and operating a treatment installation with removal capacity x_i at site i.

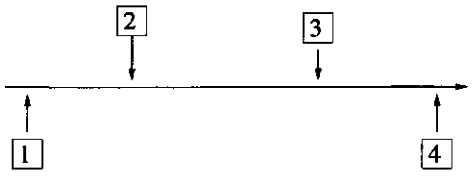

Figure 5.6: Multiple point-source pollution sites along a river. Source: Pintér, 1996a

So far the problem is simple and straightforward and has been solved in water quality engineering for many years, see Thomann and Mueller (1987). The quality determination requires following the quality development of Dissolved Oxygen (DO) interacting with the development of the Biochemical Oxygen Demand (BOD) along the river. This development can be described by classical water quality models consisting of a set of differential equations derived from mass balances. For the equations concerning one river section, analytical expressions can be derived to calculate the lower peak, minimum, of the oxygen (DO) level. However, the purpose is to stay above the DO water quality standards in all river sections of interest i (Pintér, 1996a, uses the same index). The interrelation between the sections complicates the calculation of all lower levels, i.e. the minimum DO level may not be reached in one section, before the water flows into another. Moreover, the development of the DO level also depends on all kinds of factors which vary in practice such as water temperature, streamflow etc.

Now Boon et al. (1989) want to design a system which performs well in 'nearly all situations' (robust design) and therefore include a stochastic criterion:
The joint probability of satisfying a minimum DO target level in all river sections.
This water quality criterion in a probabilistic sense, can be estimated by using Monte Carlo simulation. Probability distributions for the fluctuating (input) parameters are derived and Monte Carlo simulation is used to measure the probability of exceedence of the minimum oxygen standards in the various river sections. This defines the reliability criterion next to the costs of a design.

Pintér (1996a) uses global optimization algorithms to generate Pareto optimal points (minimum costs, maximum reliability) of the design. Every function evaluation requires running the continuous model for many Monte Carlo simulations, which implies quite some calculation time. In a joint project of Wageningen university and the water institute VITUKI in Hungary the methodology was applied successfully to a case of the Zala river in Hungary, see Boon et al. (1989).

Again the cases illustrate the oracle character of an objective function. Calculation of differential equations and Monte Carlo simulation are used for every function evaluation. Also here the user has more knowledge on the domain which can be used for deriving bounds on the criterion functions and indicating promising areas. In this example there is a monotonicity consideration; a higher removal rate leads to a higher reliability and to higher costs. The maximum reliability is reached using the most expensive design. The "weak" river sections where the standards are exceeded very often can be spotted to derive which efficiencies can be expected to be high in the optimum. In this way the domain knowledge can be used to interact during execution of the algorithms.

5.5. A DSS for minimizing a continuous function under box constraints

In the former sections, we have seen how optimization problems can be derived from descriptive models such that the corresponding function evaluations are relatively expensive and the underlying expression is implicit. In this section we are discussing some experience with a prototype DSS, called BOP (Bounded OPtimization), which was used to examine the search process for the optimum of some practical optimization problems where a function *f*(*x*) is minimized under box constraints, i.e. lower and upper bounds on the parameter vector *x*. Some notes are given on the implementation aspects and, what mainly interests us, some experience is given on what might be useful information for the user (arrow g2) and what might be useful instruments (arrow g1) with which the user can influence the search process. The indicators and instruments are summarized at the end of this section.

The idea of the implementation is outlined in Figure 5.7. The derived objective function *f*(*x*) is represented by a separate program (func.bat in Figutre 5.7). In some applications only the executable of the model used by the optimization problem is available due to confidentiality. Therefore we choose to use files as an interface between search program and objective function (arrow d in Figure 5.1). The in.dat file contains suggested values of the decision vector *x* and after the evaluation program func.bat has been run, the resulting objective value is written in the file out.dat. In this way there are no further software requirements on the problem to be optimized. Furthermore, the construction puts emphasis on the black-box character of the objective; the optimization routine has no structural information on the problem to be solved.

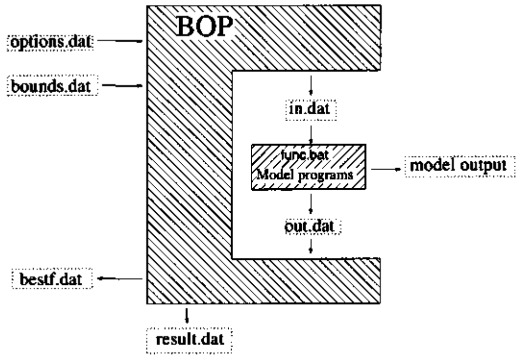

Figure 5.7: Outline of file interactions around BOP

Some of the information the user directs towards the search process is more of a general nonlinear optimization nature (local search) whereas other information has a more global character. The options.dat file may contain the (initial) *tolerances* for the local search routine. In the implementation we used a variant of Powells' method (Powell, 1964) adapted for the box constraints. Search directions are projected on the boundary when this is approached too close. In the linesearch routines, initially small steps are used to find an interval, bracketing a line minimum. This has a global optimization purpose; we try not to miss, rush over, a small but deep minimum. Another global optimization adaptation is that it is checked during the local search if the iterate is close to an optimum already found. This requires some additional checking time, but saves function evaluations which we assumed to be the expensive part of the calculation. A typical other global optimization information topic is to put a random seed number in the options file to start the random generator. Possibly the options file may contain directions for the final report. The information in this options file is considered fixed during the search process. The tolerances are in general used in criteria which are relative e.g. to the ranges in which the variables vary and therefore need no updating when implemented properly.

An important feature is the ability of the user to STOP and RESUME the search process. In BOP this was done by hitting any key. Note that it may still take some time for the program to stop, as it may be in the middle of the process of evaluating the function. Not only may the user want to change some parameters during the search process, but he may also want to examine the results graphically. Often a user may want to keep track of the progress by examining the results of the suggested parameter values for his model, translate the values back to the world the model represents, e.g. the fluid profile in the sugar screen design problem. This can be done either in a separate window continuously, but also by stopping the search and running some graphical programs. Notice that the optimization may lead to remarkable results for the user. Often the user has already carried out many runs with the underlying simulation model using realistic values for the input parameters. The feasible set defined by the bounds given by the user may not only contain realistic combinations, points. The optimization looks into all corners of the feasible space in order to detect points which lead to good objective function values.

Therefore the other important feature is the ability to set and change *bounds* during the search or to fix some parameters. For global optimization the user during the search may adapt his intuition about where to find the best local optima, the most promising areas, and may also change the bounds then. This may require some bookkeeping, as one may not want to throw away all points which have been evaluated and optima which have been found in areas that are temporarily not considered any further. The bounds.dat file can be used to give initial values for the bounds, give the names of the parameters for the translation in the direction of the user and may possibly contain initial values of the parameters as a *starting point* for a local search.

For the output information it is first of all useful to keep track of the *best point* which has been found sofar and its function value during the search, the so called record value. By storing this solution in a separate file bestf.dat, this point does not get lost when the system breaks down due to a failure in the execution of the func.bat program which runs the model. The final report in the result.dat file can contain various topics. By many users it was considered a good idea to list the best let say 10 points which have been found. In global optimization all optima can be reported and the number of times that they are detected. This information should be available somewhere during the search for the user to adapt his search strategy. First the graphical interface is discussed now before going into detail on how the user may influence the search further. The graphical interface of the BOP prototype DSS is outlined in Figure 5.8. A three dimensional picture is drawn of the search space. Two axes are used to project the parameter (decision) space in two dimensions, putting two variables at the axes, in the figure referring to the design problem of Section 5.2. The third axis is used to record the function value corresponding to given parameter values. Points in this plot represent function evaluations. Local optima which have been detected are represented by a number. The scaling of the objective axis is not straightforward. One can use the minimum and maximum value which have been found. In general we used the minima and maxima of let say the 50 best points which have been evaluated. Of course the user can adjust this. Notice furthermore that due to graphical reasons the objective axis is turned around (maximization graphically).

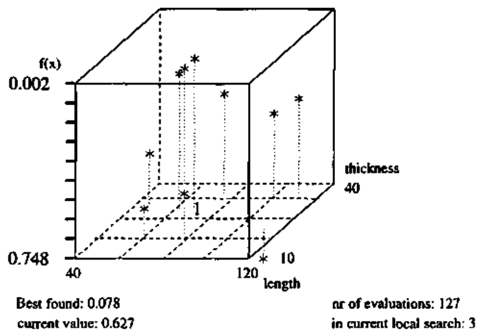

Figure 5.8: Sketch of the graphical information screen of BOP

Now the user can scan through the decision space by changing the parameters at the axes. For instance the user can suspect that the function is monotonic in one of the variables, so that he decides to fix its value. There is also a limited possibility to discover patterns in the location of the optima. If the optima are on a line in the plane of two variables, this can be discovered. However if the points are in a higher dimensional plane or line not orthogonal to the parameter axes, this might not be discovered. Another pattern is due to symmetry in the optima (Section 2.6), when the variables are interchangeable. This also may be discovered.

There are several **indicators** possible around the graphical interface with respect to the progress of the (global) search process and several **instruments** to influence the process. One piece of information is a tabular form of all optima which have been detected and the number of times they have been discovered. Another piece is the number of function evaluations which have been performed. This gives rise to two other indicators: an estimate of the amount of function evaluations per local search and an estimate of the calculation time per function evaluation. This last one might not be straightforward. We came across an application where the time of a function evaluation depended on the objective function value; for a bad function value the underlying computer program did not need all calculation steps.

This kind of indicators are useful in the context of a limited solution time; the user wants to start up the program and observe the results the next morning. One suggestion by a user was to derive from theory a confidence figure; what is the probability that the global optimum has been reached. In fact in the theory on stochastic global optimization this has been considered from the opposite point of view: Given a probability, when can we stop sampling further, the so called stopping rules.

To give an example of such **stopping rule** analysis, we mention an early result due to Karnopp (1963). If we consider a pure random search strategy, the probability after having generated N points that a better point will be found with sampling N_l points additionally, is $P = N_l/(N+N_l)$. At first sight this is a remarkable result, as it does not depend on the function which is minimized. The result can be derived by analyzing extreme value distributions as will be studied in Section 5.6. In a limited time situation one knows the remaining number of function evaluations N_l, and with that the probability to reach a better point. However, it does not give a probability that the global optimum actually has been reached. For a multistart strategy this kind of analysis has been done by Betro en Schoen (1987) and Boender and Rinnooy Kan (1987). In their approach a priori distribution of the number of optima and the size of the regions of attraction of the optima is given and adapted during performing the local searches. The adapted probability distributions result in estimates of the probability that a new local optimum is discovered or that a better point will be found. Schoen as well as Boender use the probability to derive stopping rules. Our user was interested in the idea to obtain a kind of certainty estimate, the probability itself, that the optimum is reached after a night of calculation, a probabilistic statement about the best point which has been found. Notice that when the number of evaluations or local searches is low (expensive function evaluations), the probability will depend much on the initially assumed prior distribution.

As one of the main instrument for influencing the local versus global search we used the number of generated random points before a local search is started from the best point of this sample. This basic approach is called **Multi-singlestart** and is analyzed further in 5.6. When there are many local (not global) optima, the user can put more effort in global search instead of spending all "ammunition" of function evaluations on detecting local optima and thus spend more energy on global scanning. Another possible approach is to switch to other methods. One could switch towards simulated annealing or other adaptive search methods which, as outlined in 5.1, are more or less combinations of global and local search.

In a multistart environment, *clustering* is a useful approach and an option for influencing the search, see e.g. Timmer (1984). This can be implemented such that when a random point is generated it only serves as a starting point for a local search, if it is not 'too close' to a cluster of points around a local optimum. This is called *Simple Linkage*. One parameter in this way of looking for the optimum is the critical distance which decides on what is 'close to'.

The last option mentioned here is the possibility to change the local search method. Direct search methods such as that of Nelder and Mead (1965) are known to be able to handle nonsmooth objective values. If the nonsmooth character of the function was not already clear from the way the objective has been constructed, it is only discovered during running the search process by a bad convergence performance of methods based on iterative linesearch.

Let us summarize the information between user and search process as mentioned in this section.

g1 information: from user towards search process

general for nonlinear optimization
- -stop and resume
- -tolerances for the local search
- -choice of local search method
- -setting bounds and fixing parameters
- -deliver a starting point
- -instructions for the report

global optimization
- -trade-off between local and global search
- -random seed
- -influence clustering

g2 information: from search process towards user
- -graphical information on the decision space
- -current function value
- -best function value found sofar
- -number of evaluations in the current local phase
- -graphical information from the user model
- -number of optima found
- -number of times an optimum has been detected
- -estimate of the time of one function evaluation
- -estimate number of function evaluations for one local search
- -"confidence figure": indicator on how certain the optimum has been reached

An interesting question is now how the user may adapt the main instrument in global optimization, the choice between local and global search, given the information which is generated by the search process. In a situation where the user orders his computer to search in the feasible area, walks out his office and when he returns in the morning wants to have obtained some good answers, good heuristics would be welcome. This question will be discussed in Section 5.6, where we further analyze the efficiency of possible decision rules for a situation where the calculation time is considered limited.

5.6. *A limited solution time*

5.6.1. Introduction

The box constrained global optimization problem, which is to find the global optimum of a real valued, in general multimodal objective function over a hyperrectangle $X \subset R^n$, has been studied by many researchers. An efficient and often applied approach is to perform a local search from points derived from a random sample from a probability distribution over X. Attention has been paid to the derivation of stopping rules to determine the sample size. In many practical situations the function evaluations necessary for the optimization can be rather time consuming e.g. it may need minutes, as a special subroutine or program has to be run to determine the function value. This has been illustrated by the problems in 5.2, 5.3 and 5.4. In such a case it is not uncommon that with a given amount of calculation time e.g. a night or a weekend, one wants to reach a point in X with a function value as low as possible. The limited time to find a good solution can also be found in cases such as power station decisions, where the decision time is restricted. In this section we introduce for this case the term Box Constrained global optimization problem with a given Budget of function evaluations, BCB problem for short. In 5.6.3 the Multi-singlestart method is presented and possible strategies within this method are discussed. In 5.6.4 criteria are introduced to measure the performance of solution methods for the problem class. The criteria are numerically illustrated for various methods and instances of the BCB problem. In 5.6.5 an analysis can be found on the performance of random search methods for the BCB problem. This is followed by a discussion of the results and conclusions in 5.6.6. The analysis of this section can be found in Hendrix and Roosma (1996).

5.6.2. Box constrained GOP with a limited solution time

In many applications in engineering one wants to find the minimum of a real valued function f over a region X defined by lower and upper bounds on the variables. (X can be called a closed hyperrectangle or a box.)

$$\min_{x \in X} f(x) \qquad \text{(BCP)}$$

It will be assumed that $f(x)$ is a real valued possibly multi-extremal continuous oracle function for which no derivatives or other global information such as a Lipschitz constant or concavity properties are available. Evaluating the function is similar to presenting parameter values to a black-box, possibly implemented in a subprogram, which calculates the function as a criterion on the parameters.
Zhigljavsky (1991) introduced a classification on global optimization problems based on the prior information on the problem. In this classification the BCP problem is classified as type a), it is only known that f is continuous.

In technical oriented literature, pragmatic approaches can be found to solve the problem. See e.g. Pronzato et al. (1984), Bohachevsky et al. (1986), Brazil and Krajevski (1987). In Mathematical Programming literature, analyses on various global optimization methods are presented. An overview on global optimization methods can be found in Törn and Žilinskas (1989). Most methods are based on the idea of globally exploring the feasible area (global search) e.g. by generating points in the feasible area, and performing local searches to arrive possibly (hopefully) at the global optimum. In Section 5.1 among others the following elements were distinguished:

Generating random points

If the target is to discover all optima, the purpose is to try to get starting points for the local search in every region of attraction. Following the idea that the region of attraction close to the global optimum contains the lowest function values, one can also only start a local search from the best point found during the global search. This idea will specifically be explored in this section.

Local search

The purpose is to get a local optimum from a "good" starting point.

Clustering

By clustering random generated points, a part of the local minimum structure can be discovered and one can save a number of function evaluations by only starting a local search once for each cluster.

Focus of this section is on the practical case, where there exists a budget for the computer time. Given this budget the search method should result in a function value (and corresponding point in X) as low as possible. It is assumed that the budget in computer time translates directly to a budget B on the number of function evaluations during the search process, i.e. every function evaluation requires the same calculation time. This problem will be called the box constrained global optimization problem with a given budget on function evaluations, the BCB problem. We concentrate on the combination of generating random points (global search) and local search, because these elements exist in the core of many methods. The solution method for this problem allocates budgets to local searches and to generating random points. A framework of this allocation is presented in the next section.

5.6.3. Multi-singlestart

In the literature on random search based methods the following approaches have been analyzed.

-Pure Random Search (**PRS**) (see e.g. Zabinsky and Smith, 1992): Generate a number N of random points from a uniform probability distribution over X and evaluate them. The lowest point is an approximation of the global optimum.

```
PRS
        1. Generate and evaluate random points in $X$
        2. Determine the best value $y_r$ and incumbent minimizer $x_r$
```

Random search methods have been studied, among others, by Zabinsky and Smith (1992), Zhigljavsky (1991), Romeijn (1992) and Klepper and Hendrix (1994), focusing mostly on adaptation of the distribution function over X.

-Singlestart (**SIS**): Generate and evaluate random points over X and start one local search from the lowest point found.

```
SIS
        1. Generate and evaluate random points in $X$
        2. Determine the best value $y_r$ and incumbent minimizer $x_r$
        3. Start a local search with starting point $x_r$
```

-Multistart (**MUS**): At every iteration, a random point is generated in X as a starting point for a local search.

```
MUS
        Do for $t$=1 to $N$
                1. Generate a random point $x_t$ in $X$
                2. Start a local search with starting point $x_t$
```

In Boender and Rinnooy Kan (1987) and Betro and Schoen (1987) studies can be found on when to stop the multistart process given some criteria on the trade-off between reliability and computational effort.

When we have a budget on the number of function evaluations, as in the BCB problem class, the application of the SIS method would consume as many function evaluations as possible for the global search, whereas the MUS method allocates the budget towards local searches. The success of both methods depends on the instance of the BCB problem, which a priori is unknown. For an instance of the BCB problem with a few local optima and sufficient budget to perform some local searches, it may happen that multistart proves to be the best strategy. For a problem with many local optima which are much different from the global optimum, it may be better to perform one local search from the best of a long list of random points, than to spend all "ammunition" on identifying local optima. The existence of many local optima may occur in practical cases due to numerical effects e.g. when an evaluation involves the numerical integration (fitting of continuous models) or inversion of a matrix (optimal design of experiments).

The message is that, if the surface of f is "rough", so that there are many local optima, then more effort should be put into global search, conversely if there are only a few optima, then more of the budget can be allocated to local searches. Due to the character of the BCB problem, this function structure is of course unknown when the search starts. During the search process more of the structure is revealed and the allocation of budget to local searches and global search can be adapted. Here we get to the idea of multi singlestart (MSIS), where the number of random points to be evaluated depends on the structure revealed during the search.

```
MSIS
    0. $t$=1
    1. Generate and evaluate $N_t$ random points on $X$.
    2. Identify the best point $x_t$ out of the $N_t$ points.
    3. Perform a local search with starting point $x_t$ .
    4. If budget is left, $t$=$t$+1 and go to 1.
```

At every stage t in step 1, the number of random points N_t is chosen before the next local search is performed from the best of those points (if enough budget B_t is left). So more or less effort can be put into the global search.

The number of function evaluations F_t necessary to perform one local search does not only depend on tolerances, but also on the starting point and the function under consideration. Thus, F_t can be considered as a random variable. During the search, estimates of (the expected value of) F_t become available. Note that F_t tends to decrease when N_t increases, due to the fact that part of the local search work is taken over by the random search.
As a variant of the decision parameter N_t we introduce the parameter K_t.

K_t: number of local searches (iterations) intended to be performed before the budget is exhausted.

If the part of the budget which is not used for local searches is equally divided over the intended iterations the number of random points N_t can be derived from :

$$N_t = (B_t - F_t - F_{t+1} -...-F_{t+K-1})/K_t \, .$$

A uniform estimate F for the expected number of function evaluations for a local search reduces the formula to:

$$N_t = B_t/K_t - F$$

The maximum number of local searches that can be performed is estimated by

$Kmax_t = [B_t/F]$. At every decision stage t, K_t is chosen between 1 and $Kmax_t$. The SIS method corresponds to K_t=1 and MUS can be approximated by choosing K_t=$Kmax_t$.

We now define a MSIS strategy as a choice rule to determine N_t (or K_t alternatively) at every iteration from the information generated by the previous iterations.
At step 1 of the algorithm among others the following information is available:

B_t:	budget left
$Nloc_t$:	number of different local optima found
t-1:	number of local searches performed sofar
F:	expected (estimate) number of function evaluations necessary for one local search
$Kmax_t$:	$[B_t/F]$

Now, various strategies can be constructed. If we follow the general idea described above, then if many optima are found, K_t should be tending to 1, which means the pure SIS strategy. If the prior expectation is the existence of many local optima, the corresponding strategy is to have K_1=1. One problem is that there is no good estimate for F_t when no local searches have been performed. Another approach is to start with the hypothesis that there exists only one local optimum. At the first and second iteration a local search with one random starting point (N_1=N_2=1) can be performed. If the local optima found are equal, the hypothesis still holds and one can proceed with multistart in an attempt to check the hypothesis by discovering other local optima. If they are not equal, there are apparently multiple optima and, in an attempt to detect the region of attraction of the global optimum, N_t can be increased. Another possibility is to base the choice of N_t on an estimate of the number of undetected local optima, see e.g. Boender and Rinnooy Kan (1987). To illustrate the idea of MSIS strategies we introduce the following rule (Hendrix and Roosma, 1996) which is called β heuristic (as depending on the parameter β≥0) :

$$K_t = Kmax_t - \frac{(Kmax_t - 1)(Nloc_t - 1)}{(Nloc_t - 1) + \beta\frac{t-1}{Nloc_t}} \tag{5.5}$$

The parameter β weights the relative number of different optima found compared to the number of local searches performed. When t-1/$Nloc_t$→∞, the rule approximates pure multistart (K_t→$Kmax_t$) in an attempt to discover new optima. If every local search results in a new local optimum, the rule tends to singlestart, K_t→1.

After the introduction of the BCB problem and possible solution methods there is a need to establish performance criteria.

5.6.4. Performance criteria for the BCB problem

In the OR literature it is common to use the number of function evaluations used and the indicator whether the global optimum has been found, as a performance criterion for global optimization methods. For the BCB problem the objective is to reach a point as good as possible given budget B, hopefully it is the global optimum. So the best function value found is the criterion. The score on these classical criteria does not only depend on the local search method, tolerances and stopping criteria, but in random search techniques also on the random series used. To filter out this random effect, the expected values for those classical criteria are suggested in this book to be applied as criteria for the BCB problem.

$PG(B)$:	the probability of a search method reaching the global optimum within budget B.
$ER(B)$:	expectation of the record value found with budget B.

The possibility to analyze the behaviour of algorithms with respect to the criteria is limited. In general, estimates for these criteria for various search strategies on test functions can be determined by Monte Carlo simulation. Only for multistart the probability of reaching the global optimum can also be approximated analytically by the following idea. Let D^* be the region of attraction of the global optimum and let $v=V(D^*)/V(X)$ be its relative size. Let F be the average number of function evaluations necessary to perform one local search (which is itself stochastic) and B the budget. The number of local searches that can be executed is $[B/F]$. This makes the probability of reaching the global minimum, as given in (5.2), at least

$$PG(B) = 1 - (1-v)^{[B/F]} \qquad (5.6).$$

The $PG(B)$ and $ER(B)$ criteria are illustrated here for various instances of the BCB problem. Monte Carlo simulations are done to estimate the score on the two criteria for the β-heuristic (5.5) for some values of β and for the MUS and SIS strategy. For the β-heuristic we choose $N_1=N_2=1$. This implies that for low budgets the heuristic performs exactly the same as MUS. For SIS an estimate should be available for the number of function evaluations necessary for one local search.

Test functions for which the number of local optima varies can be found e.g. in Törn and Žilinskas (1989). To see any difference, test functions with many optima are of interest. Therefore the Rastrigin function (50 optima), the Shekel functions (5,7 or 10 optima) and the Goldstein-Price function are taken from this reference.

For the local optimization a variant of Powell's method (Powell, 1964), adapted for the box constraints, is used. In the linesearch initially small steps are taken in an attempt not to miss the nearest optimum. The stopping criterion is defined on the progress in function value. Moreover, after every linesearch it is checked whether the iterate is close to an optimum already found. The tolerance of being close to an optimum is taken as 1 percent of the componentwise range of the variables. This check speeds up the search process, and causes F_t to decrease when the iterations proceed.

The estimation of the criteria *PG*(*B*) and *ER*(*B*) is done by running the random search many times with various random series for fixed values of the budget *B*. For lower values of *B*, the fluctuation of the criteria is larger, does more depend on the random series, than for higher values of the budget where the probability PG(B) approaches 1. Therefore more replications were done (10,000) for small values of *B* than for large values of *B* (200). As a numerical illustration we applied MUS to the Rastrigin test function (See Törn and Žilinskas 1989). The relative size of the region of attraction D^* for the local optimizer used was v=0.0346. In Figure 5.9 the theoretical smooth curve of (5.6) is confronted with two curves that were found by Monte Carlo simulation.

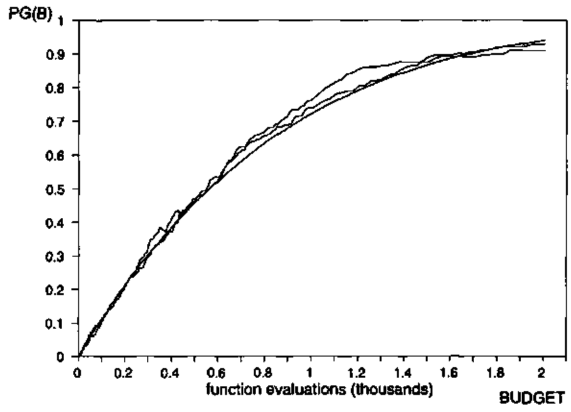

Figure 5.9: Probability to reach the optimum for multistart

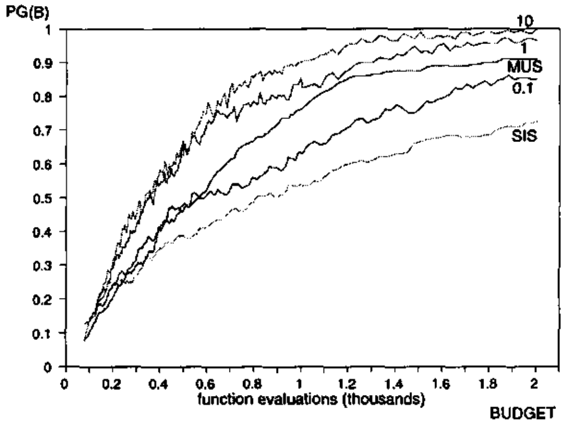

Figure 5.10: Results of criterion *PG*(*B*) for the Rastrigin function for various strategies

The performance of MUS, SIS and the β-heuristic have been estimated by Monte Carlo simulation for the Rastrigin test function. The results are given by Figure 5.10 and Figure 5.11.

The results illustrate the idea of the criteria. Given an instance of the BCB problem, every solution method has its *PG*(*B*) and *ER*(*B*) curve. For the five solution methods (MSIS strategies) for which the curve has been approximated, the β-heuristic with a value of β = 10 performs the best for this test function. This illustrates how the criteria introduced can be used to judge on search strategies for the BCB problem: A particular method is better than another for a certain instance of the BCB problem, if its *PG*(*B*) curve is higher or its *ER*(*B*) curve is lower. In Figure 5.11 we see that from the five strategies, the β=10 rule has the best expected value for the best function value found. The objective function value of the global minimum is -2.

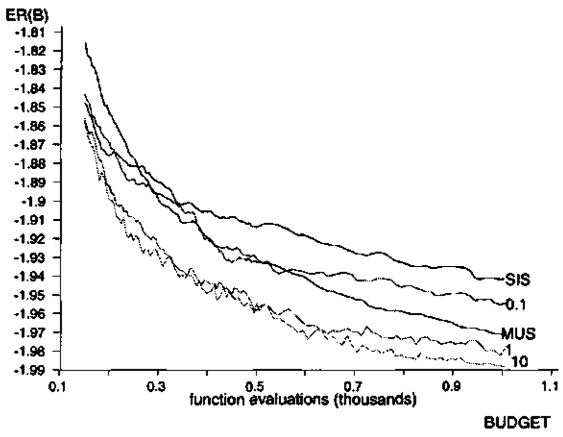

Figure 5.11: Results of criterion *ER*(*B*) for the Rastrigin function for various strategies

What determines the success of generating random points in the context of increasing the probability that the global optimum is detected? We first illustrate the difference in efficiency of generating points by two extreme numerical examples. In 5.6.5 we will try to analyze this extreme difference. Figures 5.12 and 5.13 give the performance, according to the *PG*(*B*) criterion, of the five strategies for the Shekel-5 function and the Goldstein-Price function respectively. The results show that generating many random points is efficient for the Goldstein-Price function, whereas it apparently is not efficient for the Shekel-5 test function.

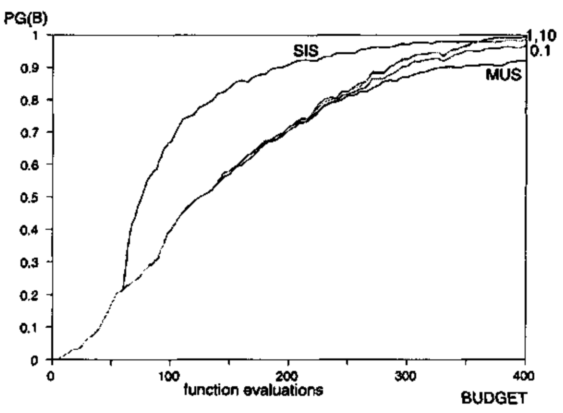

Figure 5.12: Results of criterion *PG*(*B*) for the Goldstein-Price function for various strategies

This illustration leads to the question whether there exists a MSIS strategy, or more generally a method, which performs better for all instances of the BCB problem for all values of the budget. To formalise this question we define the concept of dominating methods.

A method is called PG-Dominating if for all instances of the BCB problem, *PG*(*B*) is higher than (or equals) *PG*(*B*) of all other methods for all values of budget *B*.

5.6.5. Analysis of random search methods for the BCB problem

Can a dominating method exist for the BCB problem? To answer this question we first analyze for which cases it is profitable to put more effort in global search than is done by multistart. As mentioned in 5.6.2, the objective of generating random points (or increasing N) is to increase the probability that the starting point of the local search is situated in D^*. We use the following notation.

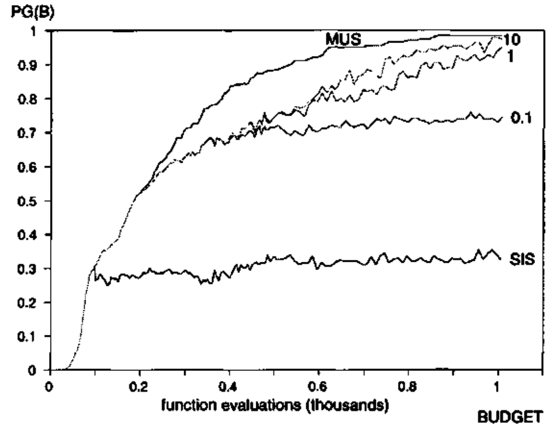

Figure 5.13: Results of criterion $PG(B)$ for the Shekel-5 function for various strategies

The relative size of a level set $S(y)$ is defined as

$$\mu(y) = V(S(y))/V(X).$$

When x is uniformly distributed over X, $y = f(x)$ is a random variable with cumulative distribution function $\mu(y) = P\{f(x) \le y\}$ and probability density function $\mu'(y)$. By performing a random search with N points the probability density function $M_N'(y_r)$ of the record value y_r (lowest function value found) is

$$M_N'(y_r) = N\mu'(y_r)(1 - \mu(y_r))^{N-1} .$$

The success of a global search depends on the probability that the point x_r corresponding to y_r is in the right region of attraction, D^*. We define

$$\varphi(y) = P\{ x \in D^* \mid f(x) = y\}$$

as the probability that a point x is in the right region of attraction given that x is situated at a contour with height y. The efficiency of going deeper into the level sets by generating random points depends on the shape of $\varphi(y)$. If one random starting point is used then the probability to reach the global optimum equals the relative size $\nu = V(D^*)/V(X)$ of D^*:

$$\nu = \int_{y_*}^{y^*} \varphi(y)\ d\mu(y) \tag{5.7}$$

in which $y_* = \min_{x \in X} f(x)$ and $y^* = \max_{x \in X} f(x)$.

By first generating N points and then starting a local search from the lowest of these points, the probability PS_N of reaching the global optimum is

$$PS_N = \int_{y_*}^{y^*} \varphi(y) M_N'(y) dy \quad . \tag{5.8}$$

Essential in the analysis is that PS_N may be worse (lower) than ν. This occurs when there is a wide relatively deep level set of which a large part does not belong to D^*. As an example from the standard test functions (see Törn and Žilinskas, 1989) the Shekel functions have this characteristic, as we have seen from Figure 5.13. This implies that SIS performs very bad versus MUS. The Goldstein-Price function gives the opposite result, see Figure 5.12.

The function $\varphi(y)$ is apparently very different for those two examples. However $\mu(y)$ also differs for every problem. To make $\varphi(y)$ more comparable, we introduce the following transformation. Let z be a uniformly distributed random variable defined as $z = \mu(y)$. In other words $\mu^{-1}(z)$ defines the quantiles of y. Notice that the function $\mu(y)$ is increasing and not necessarily continuous. The function $\mu^{-1}(z)$ exists. Now equation (5.7) can be written as

$$\nu = \int_{y_*}^{y^*} \varphi(y) d\mu(y) = \int_0^1 \varphi(\mu^{-1}(z)) dz \quad . \tag{5.9}$$

Every value of z is "equally probable". We will call the function $\psi(z) = \varphi(\mu^{-1}(z))$ a characteristic function (not to be confused with the probabilistic meaning), as it contains all information to calculate (5.8) and consequently gives the exact information on the efficiency of generating random points. Note that $\psi(z)$ approaches 1 when z goes to 0. Equation (5.8) can be replaced by

$$PS_N = \int_0^1 \psi(z) N(1 - z)^{(N-1)} dz \quad . \tag{5.10}$$

The characteristic function $\psi(z)$ determines the success of generating random points and contains much more information than e.g. the number of optima. It should be mentioned that the information of $\psi(z)$ is in general not available, so it cannot be applied in an algorithmic framework. It has been introduced for analytic reasons.
For illustrative purposes the function $\psi(z)$ is approximated numerically by Monte

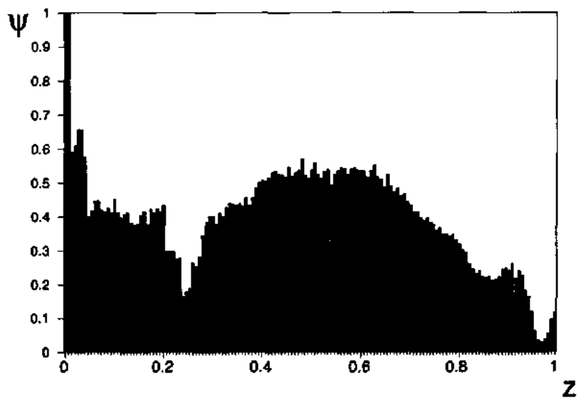

Figure 5.14: Characteristic function for the Goldstein-Price problem

Carlo simulation for the two extreme example functions. In Figure 5.14 and Figure 5.15 numerical approximations can be found.

In limit $\psi(z)$ approaches 1 when z goes to zero. However for the Shekel-5 function this limit is that distant that it cannot be observed in the numerical estimates. Increasing the number of random points leads the point x_t away from the right region of attraction D^*. For the Goldstein-Price function generating random points improves the probability that the global optimum is reached. This shows that given the information which becomes available during the search, it is impossible to determine a search strategy which performs better than all other strategies for all instances of the BCB problem; a PG-dominating method does not exist. We used the concept of the characteristic function to show that MUS has to be the optimal method over all possible MSIS strategies for the Shekel-5 problem. This strategy is not optimal for other instances of the BCB problem.

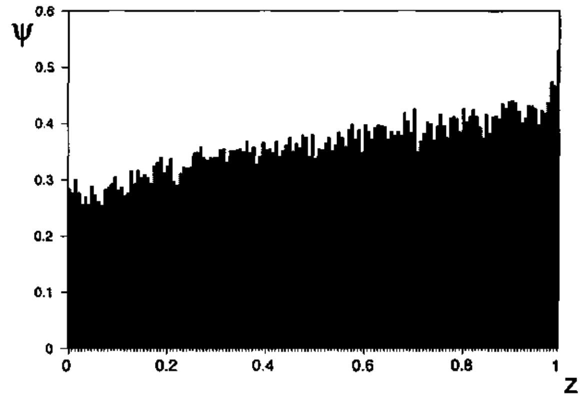

Figure 5.15: Characteristic function for Shekel-5

Furthermore we have seen that knowledge of the characteristic function, which for practical problem solving will be out of the question (given the budget on function evaluations), gives by calculating (5.10) how the probability PS_N of reaching the global optimum changes, when the lowest point x_t of a random search with N_t points is used as a starting point for a local search. This is illustrated by Figure 5.16 and 5.17. In those Figures, PS_1 equals v, the relative size of the region of attraction of the global optimum. The well known rule from stochastic methods that $PS \to 1$ when $N \to \infty$, applies here, but cannot be derived from Figure 5.17.

A less frequently used criterion is to look at the maximum absolute error $\max_i |e_i(\theta)|$ over the observations. The minimization of squared errors (6.3) has an important interpretation in statistics. When assumptions are made such as that the measurement errors are independent normally distributed random variables, the estimation of θ by minimizing $f(\theta)$ of (6.3) corresponds to a so called *maximum likelihood* estimate and probabilistic statements can be made, see e.g. Bates and Watts (1988). Parameter estimation by minimizing (6.3) given data on z_i and x_i is called an *ordinary least squares* approach.

The formalisation is sufficient to discuss the idea of identifiability. When a linear regression function

$$z(x, \theta) = \theta_1 + \theta_2 x$$

is fitted to the data of Figure 6.1, ordinary least squares (but also minimization of (6.2)) results in a unique solution, optimal parameter values (θ_1, θ_2). There is one best line through the points. Consider now the following model which is nonlinear in the parameters:

$$z(x, \theta) = \theta_1\theta_2 x \ .$$

The multiplication of parameters sometimes appears when two linear regression relations are combined. This model corresponds with a line through the origin. The best line
$z = \text{CONSTANT} \times x$ is uniquely defined; the parametrization however is not. All parameter values on the hyperbola $\theta_1\theta_2 = \text{CONSTANT}$ give the same goodness of fit. The set of solutions of the optimization problem is a hyperbola. The parameters are non-identifiable, i.e. cannot be determined individually. For the example this is relatively easy to see. For large models, analysis is necessary to determine the identifiability of the parameters, see Walter (1982) and Mous (1994).
For the optimization problem this phenomenon is important. Application of a multistart strategy will lead to a new (global) optimum at every local search. The number of optimal points is infinite. In 6.2 this problem is elaborated further.

In Section 2.6, the learning of a neural net was considered from a parameter estimation point of view. In this problem there is more than one parameter vector leading to the same model description. We have shown for this problem that the number of global optimal parametrizations is not necessarily infinite, but it grows more than exponential with the number of hidden nodes due to an inherent symmetry in the optimization problem.

We focus further on the least squares formulation (6.3). The quadratic character has been used for the derivation of specific algorithms. Moreover, there is a statistical link between the reliability question and (6.3). We will outline both aspects.
In both aspects the so called Jacobian plays a role. Consider the m measurements

(x_i, z_i), $i = 1, ..., m$ and a regression model with parameters θ_j, $j = 1, ..., n$. The residuals are $e_i = z(x_i, \theta) - z_i$, $i = 1, ..., m$. The Jacobian $J(\theta)$ of the m functions $e_i(\theta)$ is defined as the m by n matrix with partial derivatives as elements:

$$\partial e_i(\theta)/\partial\theta_j = \partial z(x_i,\theta)/\partial\theta_j.$$

For linear regression, the model $z(x_i, \theta)$ is linear in the parameters θ and the Jacobian is constant. In general, for standard nonlinear regression functions the derivatives can be derived analytically. For the logistic regression function (6.1) the row i of the Jacobian is given by

$$\left(\frac{1}{1 + \theta_2 e^{\theta_3 x_i}},\ \frac{\theta_1 e^{\theta_3 x_i}}{(1 + \theta_2 e^{\theta_3 x_i})^2},\ \frac{\theta_1 \theta_2 x_i e^{\theta_3 x_i}}{(1 + \theta_3 e^{\theta_3 x_i})^2}\right).$$

Notice that this matrix for some parameter vectors will appear singular, e.g. $\theta_1 = 0$. For complex models where the function $z(x, \theta)$ is in fact a routine it is not certain whether the derivatives exist at all. The analytical expressions are in general not available. Often the Jacobian is approximated numerically for these oracle models.

6.1.2. The Jacobian and finding the minimum

The Jacobian plays an important role in algorithms for finding the minimum of (6.3) iteratively. Given a trial parameter vector θ with corresponding residual vector $e(\theta)$ and objective $f(\theta) = e^T(\theta)e(\theta)$, one tries to find a direction d in which the function decreases. The function value in the direction d can be approximated by:

$$\begin{aligned} f(\theta + d) &= e^T(\theta + d)e(\theta + d) \\ &\approx (e(\theta) + J(\theta)d)^T(e(\theta) + J(\theta)d) \\ &= e^T(\theta)e(\theta) + 2d^TJ^T(\theta)e(\theta) + d^TJ^T(\theta)J(\theta)d. \end{aligned} \quad (6.4)$$

The approximation (6.4) is a convex quadratic function in d. The minimum is given by the so-called normal equations

$$J^T(\theta)J(\theta)d = -J^T(\theta)e(\theta). \quad (6.5)$$

Formula (6.5) results in the so-called Gauss-Newton direction d. A procedure based on (6.5) will fail if the matrix $J(\theta)$ is (almost) singular. A well known remedy is due to the Levenberg-Marquardt method (Marquardt, 1963). This changes (6.5) into

$$(J^T(\theta)J(\theta) + \lambda I)d = -J^T(\theta)e(\theta), \quad (6.6)$$

where parameter λ is chosen according to a certain strategy and I is the n×n unit matrix. The Gauss-Newton and Levenberg-Marquardt method are available in statistical packages which deal with nonlinear regression.

For complex models, no partial derivatives are available and numerical estimates may be expensive to evaluate. The model run may require considerable simulation time. For such large models we observed the use of direct search methods such as Nelder-Mead (1965) and the method of Powell (1964), to obtain optima of the goodness of fit function $f(\theta)$. In that case no use is made of the quadratic expressions in (6.3).

An intermediate method, exploiting the quadratic structure, but not requiring derivative information is the so-called DUD (Doesn't Use Derivatives) method (Ralston and Jennrich, 1978), which we will sketch here. An idea is taken from the method of Powell, which does not use derivatives to construct search directions, but derives new directions from former iterates. The DUD method follows the same line of reasoning for the Jacobian. The Jacobian is not approximated numerically at one point, but is replaced by an estimate F which is based on n+1 former iterates. The error function in (6.4) is approximated by

$$e(\theta + d) \approx e(\theta) + Fd \ . \tag{6.7}$$

Every error function $e_i(\theta + d)$ $i = 1,.., m$ is approximated by a plane $e_i(\theta) + F_i d$, in which F_i is row i of F. The matrix F is constructed such that (6.7) is exact for the former n iterates. This requires some formalisation. Given the iterates $\theta_1, ..., \theta_{n+1}$ and resulting directions $d_p = \theta_p - \theta_{n+1}$, F is constructed such that

$$e(\theta_p) = e(\theta_{n+1} + d_p) = e(\theta_{n+1}) + Fd_p \quad p = 1, ..., n \ . \tag{6.8}$$

Similar to (6.4) the DUD direction is a solution of

$$F^T F d = -F^T e(\theta).$$

Geometrically, because matrix F fulfils (6.8), the m planes described by (6.7) go exactly through the n + 1 previous values of $e(\theta)$. The base of the Gauss-Newton method, $e(\theta) + J^T(\theta)d$, describes a tangent hyperplane at θ of the error surface (Ralston and Jennrich, 1978). Due to algebraic operations, it is not necessary to calculate F explicitly. The plane (6.7) can be described similar to the way the affine minorants were generated in Chapters 2 and 4 and the piecewise approximations in Chapter 3. The DUD method can also be found in statistical packages (e.g. SAS).
At the end of the procedure an estimate of the Jacobian is available. This is important for the use of the Jacobian in the reliability question.

6.1.3. The Jacobian and reliability

Often the researcher is interested in the accuracy, precision of the estimated parameter values. An intuitive interpretation is that all parameter values with nearly the same goodness of fit are as likely or as probable as the optimal parameter vector. This intuition is indeed affirmed by statistical theory, see e.g. Bates and Watts (1988) and Ross (1990). Assumptions such as the existence of independent identically distributed measurement errors leads to the following general asymptotic result (number of observations grows to infinity). The Jacobian J at the real value of θ appears in the variance-covariance matrix of the least squares parameter estimator $\sigma^2(J^TJ)^{-1}$, with σ^2 the variance of the measurement errors. Notice that the expression does not exist, when the Jacobian J is singular. This appears when θ cannot be identified. The theoretic covariance matrix leads to a usual construction of an ellipsoidal confidence region around the estimated (optimal) parameter vector θ^*:

$$(\theta-\theta^*)^T J^T(\theta^*)\ J(\theta^*)(\theta-\theta^*) \le \frac{n}{m-n} f(\theta^*)\ F_\alpha\ (n,m-n)\ , \tag{6.9}$$

where $F_\alpha(n,m-n)$ is the upper α quantile for Fisher's F-distribution with n and $n-m$ degrees of freedom. One expects the true value of θ to be in this ellipsoidal region. Seen from an optimization point of view (6.9) defines a level set of a convex quadratic function. The quadratic function is an approximation of $f(\theta) - f(\theta^*)$ so that the ellipse approximates in fact the level set

$$f(\theta)\ - f^* \le \frac{n}{m-n} f^*\ F_\alpha(n,m-n). \tag{6.10}$$

According to the likelihood method (6.10) gives the region in which with a certain confidence the true value of θ can be found, see among others Donaldson and Schnabel (1987).

Klepper and Hendrix (1994) illustrate with a simple case of logistic regression that (6.9) may approximate (6.10) very poorly. The drawback of general formula (6.10) over the elliptic asymptotic approach (6.9) is that it is hard to represent the set with probable values of the parameters to a researcher. For the elliptic description only the elements of the Jacobian, f^* and n and m are required. How can a possibly banana shaped level set be described efficiently?

Several papers dealt with this question. A thorough study can be found in Donaldson and Schnabel (1987). A general measure to describe the curvature of the regression function is sought in their paper. Less curvature, a 'more linear' regression function, gives a better approximation of (6.10) by (6.9). Walter and Piet-Lahanier (1988) suggest to generate points on the boundary of level set (6.10) and to represent (6.10) by interpolations between the points. For nonconvex regions this is not a good idea. Klepper and Hendrix (1994) suggest to use a set of random generated points in (6.10) and their neighbourhood to represent the level set. An earlier study of Klepper and Rouse (1991) focused on the use of the resulting population, cloud of points of the method of Price (Chapter 5, Price, 1979) to represent the confidence level set (6.10). The final set of points of the method of Price appeared not to lead to a sample of a uniform distribution over the level set. This gives rise to a specific question in the field of random search methods: How to generate efficiently a set of points from a uniform distribution over a level set? This question is elaborated further in Section 6.3.

6.1.4. The Jacobian and optimal experimental design

The second optimization question in model validation, namely experimental design, also focuses on the asymptotic covariance matrix $V(\theta) = \sigma^2(J^T(\theta)J(\theta))^{-1}$ given a certain parameter vector θ. We outline the problem briefly here. The aim of obtaining reliable estimates for θ is translated in getting the variances determined by $V(\theta)$ as low as possible. A smaller variance of an estimator results in an estimate with a higher reliability.

The variance of the measurement error σ^2 cannot be influenced. The Jacobian $J(\theta)$ which consists of the partial derivatives $\partial z(x_i,\theta)/\partial\theta_j$ is not only influenced by the choice of the parameters, but also by the choice of the points of measurement x_i. The question of experimental design is how, given a model $z(x,\theta)$ and a parameter vector θ, the observations x_i should be chosen such that the reliability of the estimator of θ is as big as possible. To solve this question, first of all a criterion function on the matrix $V(\theta)$ is necessary, to decide on which matrix is better. Often the so called *D*-criterion, the determinant of the matrix is minimized, but other criteria, such as the trace, are also possible. Notice that for the experimental design problem, θ is given and the observations are the decision elements, opposite to the parameter estimation problem. A confusing terminology appears when in experimental design a *local optimal* design expresses the design to be optimal with respect to the parameter vector θ. This is a complete other notion of local optimality than in global optimization where it denotes optimality with respect to the environment of the solution.

The generation of realistic designs of experiments is a complex optimization problem. In a realistic situation there is a budget on a number of experiments which can be done. The question on where to sample in the experimental region is a continuous optimization problem. The question how many observations to take at the measurement points is an integer decision problem. In fact we are dealing with a mixed integer-continuous problem. We will give a flavour of the problem without going into detail.

Let us first consider linear regression. In linear regression the Jacobian does not depend on the parameter vector. Intuitively one should choose measurement points as distant as possible from each other, at the extremes of the experimental area, to fix a line or plane and consequently to fix the parameters as much as possible. This is indeed what comes out of an analysis of $V(\theta)$, see e.g. Rasch

(1995). For standard nonlinear regression functions, such as the logistic regression function (6.1), expressions of elements of the asymptotic covariance matrix are available. The determinant of the matrix for logistic regression can be expressed as a function on operations on the measurement points x_i.
For logistic regression the curve is not only determined by its endpoints, but also by the infliction point. The optimal design includes measurement points at the endpoints and also in the neighbourhood of the infliction point. In Rasch et al. (1997), we investigated the optimization problem of selecting measurement points which are bound to be selected from a predefined set of candidate points. At each measurement point only one observation is allowed. This turns the optimization problem into a typical combinatorial optimization problem. How to pick out m points out of a larger set such that some criterion on the covariance matrix is optimal.

6.1.5. Cases

The models of the parameter estimation problem range from linear regression, standard nonlinear regression functions to complex models existing of sets of differential equations. The application of linear and nonlinear regression functions is common in research and for the parameter estimation problem standard methods are available which in general lead to global optima. The use of one or a few differential equations occurs among others in chemistry, food science and physics. The parameters often have a physical meaning. Stortelder (1997) describes methods for parameter estimation in systems with partial differential equations. Moreover he describes a prototype DSS for the estimation problem similar to the BOP prototype in Chapter 5. At various research departments and institutes we observed that statistical consultants developed their own software to handle parameter estimation problems of researchers.

River flow models
A description of parameter estimation in large complex models seen from the global optimization point of view can be found in Pintér (1996a). He considers large river flow and water quality models of which the predictability is of great importance, certainly in The Netherlands. Sometimes parameters have no important interpretation, but are used to fit the data more accurately with the model. Every function evaluation, calculating the goodness of fit of a parameter vector, involves simulating a large model.

Runoff models in wastewater treatment
Similar experience at the Agricultural University can be found in Grum and Aalderink (1997). A large model was considered to describe the runoff flow rate and concentrations of suspended chemical oxygen demand at the overflow point in a sewer system. Six parameters were estimated by comparing the full model simula-

tion with observation series. The sum of squared errors was minimized using a direct search method. At the global optimal parameter vector the Jacobian was numerically estimated to create an estimate of the asymptotic covariance matrix.

A noticeable other achievement of the same group is the attempt to estimate parameters of a given black box model of which the simulation time is half an hour. The model consists of a program (executable only) provided by another institute, which was fit to local data. The estimation problem is an extreme case of the oracle type of optimization problems discussed in Chapter 5 and was handled with a variant of the BOP prototype. The function evaluations were recorded in a log-file so that after a long optimization period one could examine the progress. It may take days before some local optima are detected.

Estimating mass fractions in geology
Let us in contrast to the oracle example, describe a parameter estimation problem with many parameters and an explicit model formulation from geology, in which we were involved. Meijer and Buurman (1997) report on a method to explain data from soil samples with a case from the soil of a volcano in Costa Rica. For 20 samples $i = 1,...,20$ in total 14 properties $prop_{ik}$, $k = 1,...,14$ were measured. A common method is to use factor analysis to describe the observations. The researchers however, were convinced they would be able to determine the underlying mass fractions of the constituents of the samples by estimating those fractions $frac_{ij}$ and the parameters θ_{jk} which explain the properties from the mass fractions simultaneously:

$$prop_{ik} = \sum_j \theta_{jk} \times frac_{ij}.$$

Only measurements on the properties are available. In first instance the parameters of such a model cannot be identified; by increasing θ_{jk} with a factor and decreasing $frac_{ij}$ with the same factor one can end up with the same prediction of $prop_{ik}$. This effect disappears by the observing that the fractions sum to unity:

$$\sum_j \mathrm{frac}_{ij} = 1 \tag{6.11}$$

A second observation is that due to symmetry the indices j can be interchanged; after the estimation it is not clear which fractions represent which constituents (iron, aluminium, carbon, etc). This structure, mentioned in Section 2.6, causes the model to result in the same prediction for various parameter vectors. The researchers added so many relations on fixed proportions, maxima and minima that the solution tends to be unique and corresponds with their physical interpretation. As 7 constituents are distinguished, the resulting optimization problem calculates the goodness of fit of $20 \times 14 = 280$ observations with the aid of $7 \times 14 = 98$ parameters θ_{jk} and

$20 \times 7 = 140$ parameters frac$_{ij}$, which due to (6.11) can be reduced by 20. The estimation problem is a large scale nonlinear programming problem which can be handled with nonlinear programming software. Using various starting points finally an optimum was found which fits the data well. The result appeared very encouraging when finally expensive laboratory results of some of the samples appeared with the real mass fractions. The measured fractions of the constituents met the estimated fractions very well. If the model and parameters θ_{jk} are good, expensive laboratory experiments can be replaced by determining mass fractions with the model given data of observed properties. Regular factor analysis applies the same type of model. The hidden regressors $frac_{ij}$ are bound to be orthogonal vectors. The model used here delivers nonnegative fractions which apparently can be interpreted very well.

The geology problem gives the possibility to investigate the identifiability question, as the explicit expressions are available. The discussed water quantity models are examples where these expressions are possibly not available. It is more difficult to generate statements about reliability of the parameters and about identifiability. In Sections 6.2 and 6.3 some consequences for oracle problems are discussed.

6.2. *Finding infinitely many optima.*

6.2.1. Introduction into the problem

We have seen that in the problem of model calibration, several model outcomes are fitted to data. A criterion $f(\theta)$, not necessarily differentiable, determines the goodness of fit (GOF) for a given set of parameters. One of the phenomena which can occur when determining the solution, global minimum of $f(\theta)$, is that every local search (starting point) results in a new solution vector. It looks as if the parameter estimation problem has infinitely many optima.

It is known that if the minimum is obtained for all points at a lower dimensional subset in the parameter space, the values of the parameters cannot be identified uniquely. See for an extensive discussion on identifiability Mous (1994) and Walter (1982). In this section we are looking for an existing optimal parameter vector in cases in which some parameters are 'hard to identify'. An example will be given in 6.2.3. We are not trying to find a mathematical formalization of the concept 'hard to identify', but we are looking for practical tools to overcome the problems involved. The resulting optimization problem will be called ill-conditioned. The terminology has been introduced by Hendrix et al. 1994. As already noted by Loehle (1988) and Klepper and Slob (1994), the goodness of fit may be much more sensitive to one group of parameters than to another one. The so called level set of the response surface (graph of goodness of fit criterion) becomes very narrow in at least one direction.

It is the aim of this section to look for solution approaches when, due to this ill-conditioning, the local search procedure ends in nonoptimal points. Numerical considerations are left out of the analysis. We will switch back to the notation of Chapter 5, so that the symbol x will be used as an argument of the function $f(x)$ which is optimized. Now we are considering nonlinear programming problems for which no analytic expression of the gradient may be available. The feasible set X is either $\mathbb{R}^n$ (unconstrained) or contains bounds on the variables.

$$\min_{x \in X} f(x) \qquad (P)$$

The function f is a real valued continuous function.

Approaches to solve problem (P) can be so called direct search or pattern search methods such as grid search, the simplex method, see Nelder and Mead (1965), the method of Rosenbrock or the method of Hooke and Jeeves, see for an overview e.g. Bazaraa et al. (1993). The approaches we will consider are line search methods of which the search directions are e.g. the coordinates, based on Powells' method, see Powell (1964), or based on a numerical approximation of the gradient. There exist many practical problems among others in parameter estimation, for which such methods converge very slowly to the optimum or alternatively converge to a point which is not optimal. Adjusting the tolerances which determine the stopping criteria does not significantly improve the performance of the optimization procedures. The bad behaviour of the algorithms for this particular class of problems is caused by ill-conditioning.

In literature, see e.g. Bazaraa et al. (1993), the phenomenon of ill-conditioning has been discussed to describe that the graph of $f(x)$ is very flat in one direction and steep in another. This is formalized by the condition number of the second derivative matrix (Hessean) of f. In this section we are not looking for an extension of this formalization for non-differentiable objective functions, but focus on the consequences for the optimization methods. For ill-conditioned problems the contours near the optimum tend to lower dimensional surfaces, sets. Such a set can be a plane or a manifold. A well known example of this is the parabola which determines the banana shaped valley for the Rosenbrock function. A non-differentiable analogy of the Rosenbrock function is used as an illustration in Figure 6.2. The global unique minimum is attained at $x = (1,1)$, which is situated in a valley given by the Rosenbrock parabola $x_2 = x_1^2$.

A local optimization routine $L(x)$ aims at obtaining a local optimum of (P) given a starting point $x^s \in X$. The practical problem is that $L(x)$ due to its finite tolerance and stepsizes may terminate with a point which is not an optimal point of (P). An optimal point of (P) is by definition optimal in all feasible directions. However, the line search methods check a point on a finite number of directions. Therefore after a point on a very low contour has been reached (very low in the valley), the local search procedure may reach its stopping criterion at a point which is not optimal. It is quite possible that for every starting point the routine terminates at another point, which fulfils the stopping criteria. Therefore it looks as if (P) is a multi-extremal problem with an infinite number of optima.

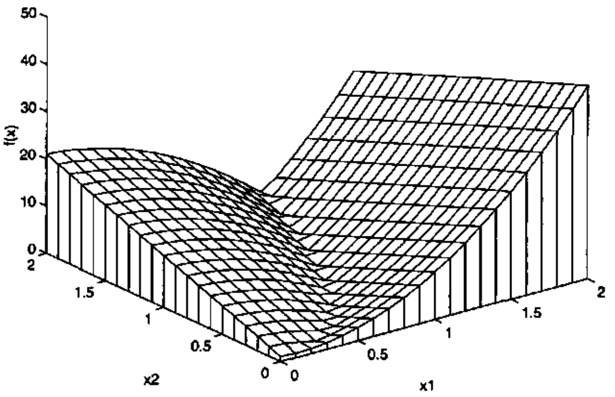

Figure 6.2: Surface of $f(x_1,x_2) = 10\,|\,x_2-x_1^2\,| + (x_1-1)^2$

Definition

Let L be a local search procedure with a given stopping tolerance, and let (P) be an NLP problem. A point p is called a numerical optimal point (NOP) of (P) if there exists a starting point $x^s \in X$ such that $L(x^s) = p$.

Note that the NOP character of a point depends on the local optimization routine L and the tolerances used during the search. The local optimal points coincide with a NOP point whereas this is not so the other way around. When there is one NOP point, it corresponds to the global optimum. The NOP concept is illustrated for the test problem in Figure 6.3. The coordinates of ten random points are given in Table 6.1. They are used as starting points for two local search procedures. In the first procedure the method of Powell is used for generating search directions; in the second a numerical approximation of the gradient is used.

Table 6.1 Application of two local search methods on 10 random starting points for $f(x_1,x_2) = 10\,|\,x_2-x_1^2\,| + (x_1-1)^2$, $0 \le x_1 \le 2$, $0 \le x_2 \le 2$

STARTING POINTS				NOP Powells method			NOP Num. Gradient method		
x_1	x_2	$f(x_1,x_2)$		x_1	x_2	$f(x_1,x_2)$	x_1	x_2	$f(x_1,x_2)$
0.92280	0.16448	6.87680	p_1	0.40411	0.16330	0.35511	0.58936	0.34735	0.16863
0.84802	1.18440	4.67571	p_2	1.08899	1.18589	0.00794	1.03694	1.07496	0.00420
1.95470	0.97075	29.4125	p_3	0.98873	0.97758	0.00025	1.08765	1.18299	0.00779
0.79882	0.29644	3.45720	p_4	0.54441	0.29644	0.20818	0.63466	0.40184	0.14303
1.66906	1.72666	11.0387	p_5	1.31159	1.71981	0.10156	1.34877	1.81913	0.12200
0.05822	0.21136	2.96664	p_6	0.46077	0.21136	0.30024	0.11838	0.01400	0.77740
0.56310	0.90307	6.05074	p_7	0.95031	0.90307	0.00260	0.82747	0.68524	0.03513
1.65135	0.40595	23.6343	p_8	0.63673	0.40595	0.13718	0.80767	0.65232	0.03700
0.73449	0.86155	3.29131	p_9	0.92797	0.86155	0.00935	0.87669	0.76815	0.01954
0.53845	0.73095	4.62324	p_{10}	0.85489	0.73095	0.02219	0.74477	0.55449	0.06710

In practical parameter estimation one may arrive at NOP points that do not correspond to local optima. One may even have the impression that the problem is a global optimization problem, whereas the fact that the local search procedure stops at different points for different starting points is due to ill-conditioning. For this class of problems we suggest some search strategies and a particular algorithm in 6.2.2. In

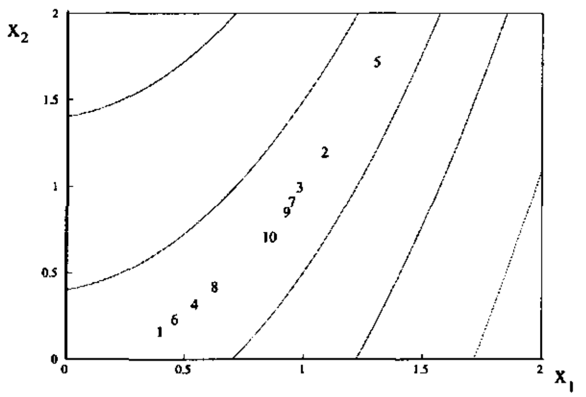

Figure 6.3: Numerical optimal points found by Powells method

6.2.3 these will be worked out for some cases. Numerical considerations are left out of the analysis. The ideas are based on Hendrix et al. (1994).

6.2.2. Search strategies

The search strategies suggested are based on a one dimensional minimization along a curve leading through a set of points found in earlier iterations. We start by generating a group of NOP points derived from performing a local search on a set of starting points. If all NOP points coincide, it is assumed that the minimum has been found.

If the points do not coincide (and there is one optimal point), they are situated in a lower dimensional set. Now we try to generate search directions along the surface by taking the difference between NOP points. If a better point is found along a direction, the local search procedure is started from this point until a new NOP point is found which replaces the worst point in the set. This total procedure is called the *line improvement* step.

If the lower dimensional surface is a hyperplane, this improvement forces the points in the direction of the optimum. It can be checked whether the points are on a hyperplane, before applying the improvement steps, in the following way. One can create a matrix B with columns b_k taken as the difference between the NOP points p_k and the first point p_1, $b_k = p_k - p_1$. When the columns b_k of matrix B are singular, the original points p_k are situated in a hyperplane. One can use standard algebraic software to check this, but a general spreadsheet also provides an answer on the question if B^TB is singular, its determinant is zero.

However, if this not the case, the improvement procedure will lead to a set of points which in addition to their NOP character have the property that they are optimal over all lines between the points. The set of points found in this way will be called a set of line optimal points. The next improvement step is based on searching along curves leading through three line optimal points in an attempt to search over the curved surface. A parabola can be constructed between every triple of points in the set. Again worst points may be discarded when better new points are found. Analogously we can introduce the term parabolic optimal points, when the line searches over parabolas through every triple of points gives no improvement and the points do not coincide. When the *parabolic improvement* is successful, two points are left over of which the lowest is an approximation of the minimum. One can proceed with cubic interpolation to generate cubic-optimal points etc. In general, one can use the term trajectory optimal points. An algorithm to generate a set of line optimal points is given by the following scheme.

Complexity
To reduce the advertisement character (this conflicts with Popperian science), which usually speaks from numerical results such as that of Table 6.7, let us focus on the analysis of failure rate and uniformity with the aid of a simple (extreme case) numerical experiment. The analysis of Section 6.3.3 is illustrated here by applying the UCPR algorithm for generating N=50 points in $S(1.05)$ for problem f7, a variant of $Q1$, with varying dimension and value for the parameter c. It has been argued that parameter c gives a kind of trade-off between effectiveness (uniformity of the final sample) and efficiency (number of function evaluations). The uniformity is measured by partitioning the parameter space in 4 parts of equal size determined by the sign of the two first parameters x_1 and x_2. By measuring and adding $(NR_{\text{sample}} - NR_{\text{expected}})^2/NR_{\text{expected}}$ for each partition set a statistic appears which approximately has a Chi-square distribution. As mentioned before, there are many other ways to measure the tendency of the sample to originate from a uniform distribution (see Ripley, 1981). An important factor in the analysis of Section 6.3.3 is the so called failure rate; the number of function evaluations that do not give an improvement of the level set divided by the total number of function evaluations. We measured those criteria for one run of the UCPR algorithm with values for the parameter c of 2, 1.5 and 1.2 for the problem f7 with dimension n of 2, 3 and 4. We just consider one run for each case, to be able to have a look at the final sample.

Table 6.8: Results of UCPR for problem f7 with varying dimension and values of c

	Number of function evaluations			Failure rate			Uniformity (Chi-square stat.)		
n c	2	3	4	2	3	4	2	3	4
2	437	1158	2809	18%	55%	77%	.88	110.96	3.28
1.5	344	709	1437	8%	28%	55%	10.48	16.24	.88
1.2	292	734	1155	4%	16%	25%	12.56	77.04	36.08

It is clear that for lower values of c the failure rate decreases, which causes the total number of evaluations that have to be done going down. The drawback is that the uniformity of the final set of points grows worse. An extreme example is the case with c=1.2 and n=3: In a certain iterations all points generated have negative values for x_2. The parameter c is too small to give that new points are generated at the other side of the axis.

6.3.5. Conclusions

The problem of generating a sample of N points over a level set $S(\alpha)$ was studied. The number of function evaluations necessary to generate N points from a uniform distribution over a level set $S(\alpha)$, is linear in the dimension and N for the theoretical ideal algorithm of Pure Adaptive Search (PAS). Like every good ideal, it looks impossible to be reached. This impossibility can be derived from considerations on complexity of global and integer programming. The same theoretical complexity may be reached by a modification of PAS which uses a sample of points called N-points PAS. The algorithm of Uniform Covering by Probabilistic Rejection (UCPR) is a heuristic practical approximation of N-points PAS. For higher dimensions the deviation from the ideal becomes bigger. The test results show that UCPR performs in general better than other practical alternatives such as Pure Random Search, Controlled Random Search and a variant of the Hit and Run algorithm. The development of better algorithms, i.e. closer to the ideal, is still a challenge to Popperian science.

Chapter 7. *Major Conclusions, GLOP at work*

7.1. *The problem*

If we want global optimization (GLOP) to get to work, roughly two sides can be distinguished. Most of the literature aims at one side, namely the mathematics of global optimization, i.e. the derivation of properties for special structured optimization problems and for specific methods and techniques. This leads to useful mathematical results and elegant exercises. To quote the mathematician P. Erdös,
"*A mathematician is a device for turning coffee into theorems*".
The other side is, what people like to call, the application side. This is not very well defined. In many research situations where mathematical models are used, researchers try to find parameter values such that a given performance criterion is at its best, optimal value. When the parameters can be varied in a continuous way, this defines a so-called Non-linear Programming Problem. Methods for Nonlinear Programming usually result in local optima, i.e. a solution, parameter values, which are the best with respect to values in the neighbourhood of that solution, not necessarily the best over the total admissible, feasible set of all possible parameter values, solutions. Cooperation between mathematicians and researchers which raised global optimization problems from practical problems, in this book called 'the modeller' or 'the potential user', has lead to application of GLOP algorithms to practical optimization problems. Some of those can be found in this book.

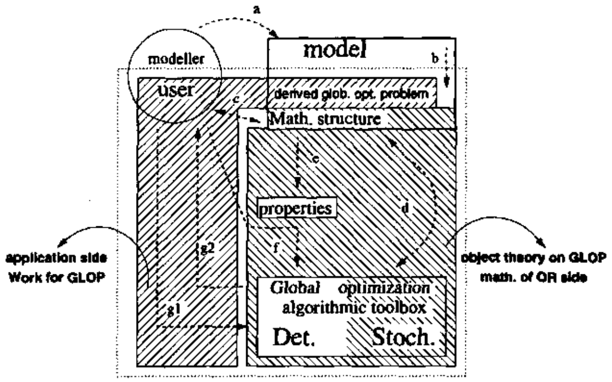

Figure 7.1: Two sides in GLOP at work

In this book we started with the question
Given a potential user with an arbitrary global optimization problem, which route can be taken in the GLOP forest to find solutions of the problem?

The target group of this study uses mathematical modelling for research, though it does not consist of experts in optimization. In this study, stimulated by experience at the Wageningen Agricultural University, cases were used of the following not mutually exclusive categories of modellers (and their typical models) and potential users of global optimization methods (Chapter 1): Engineers on agricultural and environmental sciences, designers and OR people in agricultural science. Those groups are not clearly defined, nor mutually exclusive, but have in common that mathematical modelling is used and there is knowledge of linear programming and possibly of combinatorial optimization. The further step of applying global optimization for this group requires bridging a large gap between this group and existing literature. The numerous examples and cases in this book are taken from experience of the target user groups with GLOP and may help to find similarities for potential users, but they also illustrate the way problems can be handled by GLOP techniques. The cases do not cover all possible applications. For instance successful use has been made of GLOP to find optimal crystal structures and molecule configurations (see e.g. Bollweg et al., 1997) and applications exist of concave programming in logistic planning. This book serves as a guide for the target user groups in the forest of global optimization and should help the potential user to get to the relevant literature.

Recognizing Nonlinear Programming

Our analysis in the dotted box of Figure 7.1 starts when the potential user has derived an optimization problem from a possibly descriptive model

$$\min f(x), \quad x \in X \subset \mathbb{R}^n . \tag{1.1}$$

Nonlinear programming (NLP) becomes interesting when $f(x)$ is a continuous function on a robust feasible set X. It is not difficult to recognize an NLP problem. Current practice is not that the user starts analyzing (1.1) to determine all useful mathematical structures. Either the modeller had already a solution method in mind when formulating the problem (hammer-nail story), or he starts by looking for methods to generate good solutions of (1.1) by applying a grid search or random search over X or proceeding in looking for nonlinear programming routines. Those routines generate a local optimum of (1.1) given a starting point.

Routines are mainly selected on their availability in the direct neighbourhood of the user.

We often observed the use of simple direct search methods in engineering applications and the use of standard nonlinear programming software such as GAMS/MINOS in economic research environments. Nowadays also standard solvers in spreadsheet programs are available.

The generation of local optima has a good and a bad side. When the local optima have about the same objective value, they have their interpretation on the problem which has been modelled and thus lead to information to the user about e.g. alternative good products, good economic policies etc. The bad part is, that if only one optimum has been discovered, a user may conclude wrongly that he has found the best solution of (1.1). Actually, we namely never know whether really the global optimum has been found, unless we have had a better look, analyze the problem. Global optimization starts to be interesting when we really want to find the best point and when (1.1) has several optima.

Major tracks
For the road to be taken we distinguish two major tracks, the potential user can go to solve the problem. One track we called the deterministic track and has been discussed in Chapters 2, 3 and 4. The other track we called the stochastic track and was discussed in Chapters 5 and 6. The two approaches intend to reach a different goal. The deterministic track aims at:

> *The global optimum is approximated (found) with certainty in a finite number of steps.*

Stochastic methods are understood to contain some stochastic elements and aim at

> *Approaching the optimum in a probabilistic sense as effort grows to infinity.*

Both tracks were investigated in this book from the viewpoint of a potential user. Which way to follow, first of all depends on the characteristics of the problem to be solved. In Chapter 1 we made the following rough distinction.

I. When the analytic closed form expressions are available of (1.1), they can be analyzed, i.e. mathematical structures can be extracted and properties can be derived (arrow c and e) to be used for specific methods. A recent statement of one of the leading researchers in the application of nonlinear programming and modelling languages is that "every practical optimization problem should be given in explicit formulae. Only in this way derivatives, bounds and other useful properties can be derived" (Drud, 1997).
For our target group this is simply not the case. Many practical problems not fulfilling this statement are given in II.

II. Oracle or black-box functions do not show mathematical expressions explicitly. The examples in Chapter 5 illustrated how calculating an objective function $f(x)$ may require the solution of a set of differential equations or running a Monte-Carlo simulation with a large model. For those functions in general it is not possible to calculate derivatives or to analyze the structure. Deterministic methods cannot be applied; the user is committed to nonlinear optimization routines combined with stochastic approaches.

A side question along the way was:
> *How can the user influence the search process of solution methods given the knowledge of the user of the underlying problem and which information becoming available during the search is useful for steering the search process?*

In Sections 7.2 and 7.3 remarks and sub-questions can be found for the potential user when he follows those tracks.

7.2. *The deterministic methods approach*

Let us first summarize some remarks on deterministic methods from the potential user point of view which came out of the study.

- Analysis of the expressions is required for the discovery of useful mathematical structures. Also interval arithmetic techniques can be applied on the mathematical expressions directly.
- The elegance of the techniques is due to the guarantee that when the global optimum has been discovered and verified, we are certain about its status.
- The methods are hard to implement. Thorough use should be made of special data structures to store the necessary information in memory.

In Chapter 2 the discovery of mathematical structures was discussed. In Chapters 3 and 4, the working of the methods was demonstrated.

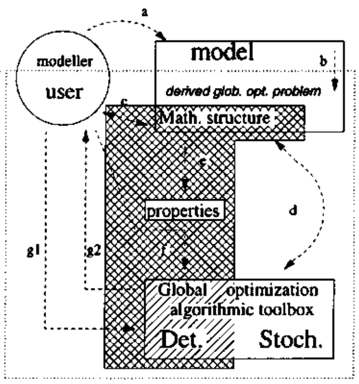
a
modeller
user
model
b
derived glob. opt. problem
c
Math. structure
e
properties
d
f
g1
g2
Global optimization algorithmic toolbox
Det.
Stoch.

Figure 7.2: Emphasis along the deterministic track

Mathematical structures
In Chapter 2, first for the problems in group I, the structures were enumerated which are useful for deterministic global optimization approaches (arrow c). At this point we used a partitioning from the point of view of the problem owner, which is not used in literature.

A. There are structures which only can be recognized after analyzing the mathematical expressions in (1.1).
B. Other structures can be discovered directly from the formulae in (1.1).

Of course first a structure has to be recognized, before it can be used in an algorithm. The use is based on the ability to create bounds on the function value. The most global structure is nonconvexity. If namely both $f(x)$ and X are convex, there do not exist local, non-global optima. Therefore the name nonconvex optimization is sometimes used for global optimization.

A. Structures found by analysis only.

-Concavity is not directly recognizable, but a very strong structure. The property is sufficient to generate lower bounds of a function on subsets of the domain X, without any further information on the function to be minimized.

-Differentiable[1] convex (*dc*) is a structure which is valid for nearly all practical objective functions. The drawback is that it can only be used when a user has found a so-called *dc*-decomposition, a partitioning of the function in a convex and concave part. The decomposition is not unique and a bad choice may lead to poor bounds on the functions.

-Lipschitz continuity is another structure which applies for nearly every practical problem. To make it working in an algorithmic context, however, the value of a so-called Lipschitz-constant is needed, which may be as hard to find as the optimum of the original problem. For some practical problems the derivation of a Lipschitz-constant may not be very difficult.

B. Structures directly recognizable from the expressions

Some structures are rather elaborated in literature, following the paradigm of making assumptions on the function to be minimized for the derivation of properties and algorithms as is usual in mathematics of OR. For the user those structures are not hard to recognise and literature is easily accessible. In Chapter 2 we mentioned

- quadratic functions
- bilinear functions
- fractional functions
- multiplicative functions.

As a last global structure we mentioned that the mathematical expressions can be used directly in a branch-and-bound context by so-called interval arithmetic techniques. As implementations become easily available nowadays, this approach may become more popular by users.

[1] The word differentiable concerns the difference of two functions and has nothing to do with the existence of derivatives.

Construction of Algorithms

To make a recognized structure useful, algorithms have to be selected, developed and implemented in the context of deterministic GLOP methods. In Chapters 3 and 4 we choose to illustrate the branch-and-bound approaches. The elegance of deterministic methods is due to the guarantee that the global optimum is approximated (found) with certainty in a finite number of steps. Whether we will find and verify the global optimum in practice depends on how the implementation of the algorithm proceeds. The problem simply can be too difficult to solve. In a branch-and-bound context this means that the available memory fills up with subsets, subproblems which have to be elaborated, refined further in the future. "Finite" may be beyond a humans lifetime, it may take too long to traverse the complete branch-and-bound tree. Global optimization only can do its best. Let us summarize with the following statement:

No global optimization method can guarantee to find and verify the global optimum for every practical situation, within a humans lifetime.

The user can select and implement an algorithm (arrow f) and use his information on the problem to influence the search process (arrow g1).

In Chapter 3 the relation between GLOP and Integer programming (IP) is highlighted for several reasons.

- Sometimes practical GLOP problems can be approximated by IP variants and solved by standard Mixed Integer Linear Programming techniques.
- The transformability of GLOP problems to IP problems vice versa shows that difficult problems in one class will not change to easy to solve problems in the other class.
- The algorithms of GLOP and IP can be classified similarly.
- Analysis of problems, which is common in Global Optimization, can be used to understand better the complexity of some IP problems.

We related the two fields, GLOP and IP, because we assumed that IP is more known to our target groups than GLOP. First it was shown how practical problems which consist of Linear Programming with one or some nonlinear functions on one or several variables can be approximated by the concept of Piecewise Linear Programming. Nonconvex functions require the use of Mixed Integer Linear Programming (MILP) and a corresponding branch-and-bound approach. The maximum distance problem illustrated that sometimes practical instances of problems which are known in literature to be 'unsolvable' can be solved by using an MILP formulation.

An advantage of using MILP is that standard routines are available so that it is relatively easy to implement and influence algorithms. A drawback of the use of Piecewise LP is that seen from a GLOP point of view, only a grid-search is performed on a grid in the space of the nonlinear variables, which is not refined

during the iterations.
The equivalence between the two fields not only gives a possibility to solve problems from one field in the other, but it also helps to see that a problem remains as difficult when it is transformed to the other field. Reformulation of difficult IP problems to difficult GLOP problems is not going to lead to easy solution procedures. No magic algorithm is going to solve difficult problems. The minimum volume hyperrectangle problem is a notorious problem, no matter from which side it is approached. The investment problem in Chapter 3 showed that analysis of the structure, as is usual in GLOP, helps to see why an IP problem is difficult.

In Chapter 4 we had the deterministic GLOP methods to get to work. Two bigger cases were worked out. First the branch-and-bound approach is explained with some simple cases.

The *nutrient problem* in Chapter 4 is an example of a high dimensional problem for which a branch-and-bound approach with rectangular subsets has been outlined and illustrated. The problem is hard to solve in practice as one may not have sufficient storage capacity and/or time to get to the global optimum and to verify it. The similar so-called pooling problem has been a challenge for many researchers and despite the relevance for petrochemic industry, remains hard to solve. The nutrient problem also shows how analysis of a given problem can lead to many useful properties:

- boundary solutions of the problem
- successive LP is very unsuccessful
- standard NLP leads to many local not global optima.

The second case, the *quadratic design problem* with a dimension lower than 10, leads to algorithms which can be implemented in a Decision Support System. The guarantee character of deterministic algorithms has been reached. For a potential user a relevant question is: "At what price". The necessary work contained analysis to find the best way to derive bounds and effort to implement a branch-and-bound algorithm efficiently. For the design problem these efforts were worthwhile, because the algorithm is run many times. For implementing deterministic algorithms one can invest in studying efficient use of memory or consult research groups which have experience on implementing branch-and-bound methods.

7.3. *The stochastic methods approach*

In general the implementation of stochastic approaches requires less effort than that of deterministic methods. Often the user has already selected nonlinear optimization routines which deliver a local optimum given a starting point. The implementation of a random generator or for instance the method of Price, which is popular among some groups of users, is not very hard.

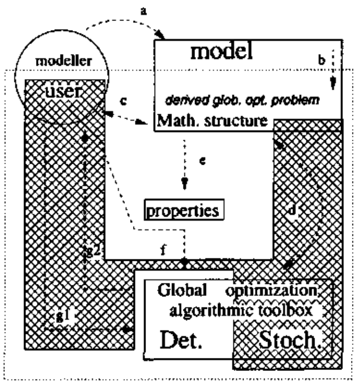
a
modeller
model
b
user
c
derived glob. opt. problem
Math. structure
e
properties
d
g2
f
Global optimization algorithmic toolbox
g1
Det.
Stoch.

Figure 7.3: Emphasis along the stochastic track

The main difference with deterministic methods is the final target. The aim is not to get a verified global optimum, but to approach the optimum in a stochastic sense as effort grows to infinity. Here we get to the practical point again that the user only has a lifetime or preferably less to solve the problem. In chapter 5 and 6 not all available ideas and methods were highlighted. We paid attention to the following.

The ideas of Adaptive Random Search which is to modify the distribution from which random points are generated such that the global optimum is reached easily, were studied in Chapter 6. Practise appears to be far away from ideals we would like to reach, namely solving problems in an expected calculation time which grows polynomially in the dimension of the problem:

It is unlikely that stochastic methods will appear solving problems in an expected calculation time which is polynomial in the number of variables of the problem.

The random function approach or sometimes called Bayesian Heuristic approach was not elaborated in this book. The idea elegantly tries to extract as much information as possible from the function evaluations which already have been performed. It does not use a random generator, but proceeds in a deterministic way, based on assumptions on a model of the stochastic behaviour of the objective function. The implementation on the algorithms is difficult and one needs faith in the underlying stochastic model.

Multistart types of approaches have always been very successful. It concerns the generation of random points as starting points of iterative local searches. Combined with clustering approaches which aim at not rediscovering the same optimum many times, the method has been successfully applied to find the global optimum in

situations where the function can be calculated, evaluated many times.
A question from practice which was discussed more thoroughly in Chapter 5 is:

What to do when the available solution time is finite, given the information we have on the problem?

Translated to the solution procedure this means that a limit exists on the number of function evaluations the procedure is allowed to do. To analyze this question, it was first enumerated which information becomes available during the search process (arrow g2) and how the user can influence the running algorithms (arrow g1). Besides possibilities which typically concern the local search, the user can influence the optimization problem directly and the main point for global optimization is that the user will influence the trade-off between global search and local search.

As every stochastic GLOP method consists of an heuristic which makes choices between global and local search, we arrived at the question:

Is there a best way to rule the choice between global and local search, given the information which becomes available?

To answer this question we experimented and analyzed with the two-phase approach, i.e. performing local searches and carrying out global search by generating random points. It appeared that **there is no best way** to choose between local and global search. When there is no overall best method we are only left with the task to filter out methods which are dominated, are inefficient in some sense. The conclusion shows again that mathematical analysis with extreme cases is a strong tool to demonstrate that magic algorithms, algorithms which are said in scientific journals to be very promising, because they perform well on some test cases, can be analyzed and 'falsified' in a Popperian style.

Let us summarize some of the remarks on stochastic methods from the potential user point of view.

- The methods require no mathematical structure on the problem and are therefore more generally applicable.
- The methods are relatively easy to implement.
- We are never completely certain that the global optimum has been reached.
- The optimum is approximated in a probabilistic sense when effort increases to infinity.

A specific phenomenon we paid attention to in Chapter 6 is:
"*Every local search leads to a new local optimum*".
We know from parameter estimation that this is a symptom in so called non-identifiable systems. The minimum is obtained at a lower dimensional surface or curve. In that case the points which are found have the same function value,

goodness of fit. From the optimization point of view, the symptom can be due to so-called ill-conditioning and is in fact an interaction between the local search methods and the problem. Some heuristics were given to overcome this problem. In Chapter 2 it was already noticed that there may be very many local optima which after translation to the meaning of the solution have the same interpretation. This is caused by symmetry in the model formulation and does not depend on the local search method.

Good solutions

There are two side questions of users derived from the general remark:
"*I am not interested in the best (global optimal) solution, but in good points*".
The first question is that of Robust Solutions, introduced in Chapter 4, and the other is called Uniform Covering, concerning the generation of points which are nearly as good as the optimum, discussed in Chapter 6.

A solution is often understood to be robust, when it remains "good" in uncertain situations due to uncertainty in data or changing circumstances. We discussed it in the context of product design, where robustness is defined as a measure of the error one can make from the solution such that the solution (product) is still acceptable. Looking for the most robust product is looking for that point which is as far away as possible from the boundaries of the feasible (acceptable) area. For the solution procedures, we had a look at the appearance of the problem in practice, where boundaries are given by linear and quadratic surfaces, properties of the product.

- For linear boundaries, finding the most robust solution is a Linear Programming problem and thus rather easy.
- For quadratic properties the development of specific algorithms is required.

The question of Uniform Covering concerns the desire to have a set of 'suboptimal' points, i.e. points with low function value given an upper level of the function value; the points are in a so-called level set. To generate low points, one could run a local search many times. However, we want the points not to be concentrated in one of the compartments or one sub-area of the level set, we want them to be equally, uniformly spread over the region. This is a very difficult problem for which we tested and analyzed several approaches in Chapter 6.

Final result

Whether an arbitrary problem of a user can be solved by GLOP requires analysis. There are many optimization problems which can be solved satisfactorily. Besides the selection of algorithms the user has various instruments to steer the process. For stochastic methods it mainly concerns the trade-off between local and global search. For deterministic methods it includes setting bounds and influencing the selection rule in Branch-and-Bound. We hope with this book to have given a tool, a guidance to solution procedures and to further literature on the subject.

References

Aardal, K. and van Hoesel, C.P.M. (1996), Polyhedral Techniques in Combinatorial Optimization, *Statistica Neerlandica*, 50, 3-36

Aarts, E.H.L. and Lenstra, J.K. (1997), *Local Search Algorithms*, Wiley, New York

Adler, F.R. and Nuernberger, B. (1994), Persistence in patchy irregular landscapes, *Theoretical Population Biology*, 45, 41-75

Al-Khayyal, F.A. and Falk, J.E. (1983), Jointly Constrained Biconvex Programming, *Mathematics of Operations Research*, 8, 273-286

Al-Khayyal, F.A. (1990), Jointly Constrained Bilinear Programs and Related Problems: An Overview, *Computers & Mathematics with Applications*, 11, 53-62

Al-Khayyal, F.A. (1992), Generalized Bilinear Programming, Part I, Models, Applications and Linear Programming Relaxation, *European Journal of Operational Research*, 60, 306-314

Baritompa, W. (1993), Customizing Methods for Global Optimization: A Bisection Viewpoint, *Journal of Global Optimization*, 3, 193-212

Baritompa, W.P. and Steel, M.A. (1993), Bounds on first hitting times of directionally-based random sequences, Research Report 94, department of mathematics and statistics University of Canterbury, Christchurch

Baritompa, W.P. (1994), Accelerations for a variety of global optimization methods, *Journal of Global Optimization*, 4, 37-45

Baritompa, W.P. and Cutler, A. (1994), Accelerations for global optimization covering methods using second derivatives, *Journal of Global Optimization*, 4, 329-341

Baritompa, W.P., Mladineo, R.H., Wood, G.R., Zabinsky, Z.B. and Zhang Baoping (1995), Towards Pure Adaptive Search, *Journal of Global Optimization*, 7, 73-110

Bates, D.M. and Watts, D.G. (1988), *Nonlinear regression analysis and its applications*, Wiley, New York

Bazaraa, M.S., Sherali H.D. and Shetty, C.M. (1993), *Nonlinear Programming*, Wiley, New York

Beale, E.M.L. and Tomlin, J.A. (1969), Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables, in J. Lawrence (Ed.), *Proceedings of the 5th International conference on Operations Research*, Tavistock, London

Beale, E.M.L. (1980), Branch and bound methods for numerical optimisation of non-convex functions, in Barritt, M.M. and Wishart, D. (Eds), *COMPSTAT 80: Proceedings in Computational Statistics*, 11-20, Physica Verlag, Wien

Beale, R. and Jackson, T. (1990), *Neural Computing: an introduction*, Adam Hilger, Bristol

Betro, B. and Schoen, F. (1987), Sequential Stopping Rules for the Multistart Algorithm in Global Optimization, *Mathematical Programming*, 38, 271-286

Bjerager, P. (1988), Probability integration by directional simulation, *Journal of Engeneering Mechanics*, 114, 1285-1302

Black, F. and Scholes, M. (1973), The pricing of options and corporate liabilities, *The Journal of Political Economy*, 82, 637-659

Blank de, H., Hendrix, E.M.T., Litjens, M.A. and van Maaren, J. (1997), On-line control and optimisation of the pelleting process of animal feed, *J. Sci. Food Agric.*, 74, 13-19

Bloemhof-Ruwaard, J.M. and Hendrix, E.M.T. (1993), Generalized Bilinear Programming: an application in farm management, Technical Note 9305, Centre for Environmental Studies Wageningen

Bloemhof-Ruwaard, J.M., van Beek, P., Hordijk, L. and Van Wassenhove, L.N. (1995), Interactions between operational research and environmental management, *European Journal of Operational Research*, 85, 229-243

Bloemhof-Ruwaard, J.M. and Hendrix, E.M.T. (1996), Generalized Bilinear Programming: an application in farm management, *European Journal of Operational Research*, 90, 102-114

Bodington, J.E. and Baker, T.E. (1990), A history of mathematical programming in the petroleum industry, *Interfaces*, 20, 117-132

Boender C.G.E., Rinnooy Kan A.H.G. (1987), Bayasian Stopping Rules for Multistart Global Optimization Methods, *Mathematical Programming*, 37, 59-80

Boender, C.G.E. and Romeijn, H.E. (1995), Stochastic methods, in *Handbook of Global Optimization*, (ed. R. Horst and P.M. Pardalos), 829-871, Kluwer, Dordrecht

Bohachevsky, I.O., Johnson, M.E., Stein, M.L. (1986), Generalized Simulating Annealing for Function Optimization, *Technometrics*, 28, 209-217

Bollweg, W., Kroll, H. and Maurer, H. (1997), Numerical prediction of crystal structures by simulated annealing, in: *Developments in Global Optimization*, I.M. Bomze, T. Csendes, R. Horst and P.M. Pardalos eds., Kluwer, Dordrecht

Boon, J.G., Pintér, J.D. and Solmyódy, L. (1989), A new stochastic approach for controlling point source river pollution, in: *Proceedings of the 3rd scientific assembly of the International Association for Hydrologic Sciences*, IAHS publication 180, 241-249, IAHS, London

Botkin, N.D. and Turova-Botkina V.L. (1992), An algorithm for finding the Chebyshev centre of a convex polyhedron, Report 395, Institut für Angewandte Mathematik und Statistik, Universität Würzburg.

Brazil, L.E. and Krajevski W.F. (1987), Optimization of Complex Hydrologic Models using Random Search Methods, in *Engineering Hydrology Proceedings*, Williamsburg ASCE, 726-731

Breiman, L. and Cutler, A. (1989), A deterministic algorithm for global optimization, *Mathematical Programming*, 58, 179-199

Brooke, A., Kendrick, D., and Meeraus, A. (1988), *GAMS, A User's Guide*, The Scientific Press, Redwood City

Butcher, W.S. (1971), Stochastic dynamic programming for optimum reservoir operation, *Water Resources Bulletin*, 1, 115-123

Cliff, A.D. and Ord, J.K. (1981), *Spatial processes, models and applications*, Pion Ltd, London

Csendes, T. (1988), Nonlinear parameter estimation by global optimization, efficiency and reliability, *Acta Cybernetica*, 8, 361-370

Danilin, Yu. and Piyavskii, S.A. (1967), An Algoritm for Finding the Absolute Minimum, *Theory of Optimal Decisions*, 2, 25-37, Institute of Cybernetics of the Ukrainian Academy of Sciences (In Russian)

Dantzig, G.B. (1960), On the significance of solving linear programming problems with some integer variables, *Econometrica*, 28

Dantzig, G.B. (1963), *Linear programming and extensions*, Princeton Univ. Press, Princeton

Dantzig, G.B. (1991), Linear programming, the story about how it began, in *History of Mathematical Programming*, eds. J.K. Lenstra et al., North Holland, Amsterdam

Davis, L. (1991), *Handbook of Genetic Algorithms*, Nostrand Reinhold, New York

Dennis, J.E. and Schnabel, R.B. (1996), *Numerical methods for unconstrained optimization and nonlinear equations*, SIAM, Philadelphia

Dinkelbach, W. (1967), On Nonlinear Fractional Programming, *Management Science*, 13, 492-498

Donaldson, J.R. and Schnabel, R.B. (1987), Computational experiencce with confidence regions and confidence intervals for nonlinear least squares, *Technometrics*, 29, 67-82

Drud, A.S. (1997), Interactions between Nonlinear Programming and Modelling Systems, Lecture at the ISMP, Lausanne, August 1997

Eberhart, R.C. and Dobbins, R.W. (1990), *Neural networks PC tools: a practical guide*, Academic Press, San Diego

Falk, J.E. (1973), A Linear Max-Min Problem, *Mathematical Programming*, 5, 169-188

Floudas, C.A. and Aggarwal, A. (1990), A decomposition strategy for global optimum search in the pooling problem, *ORSA Journal on Computing*, 2(3), 225-235

Foulds, L.R., Haugland D. and Jörnsten, K. (1990), A Bilinear Approach to the Pooling Problem, Working paper no 90/03 Chr. Michelsen Institute, Centre for Petroleumeconomics, Bergen, Norway

Grum, M. and Aalderink, H.R. (1997), A statistical approach to urban runoff pollution modelling, 4th International Conference on Systems Analysis and Computing in Water Quality Modelling, Quebec, June 1997

Hansen, E. (1992), *Global Optimization Using Interval Analysis*, Marcel Dekker, New York

Hansen, P. and Jaumard, B. (1995), Lipschitz optimization, in *Handbook of Global Optimization*, (ed. R. Horst and P.M. Pardalos), 407-493, Kluwer, Dordrecht

Hasofer, A.M. and Lind, N.C. (1974), An exact and invariant second-moment code format, *Journal of Engeneering Mechanics*, 100, 111-121

Haverly, C.A. (1978), Studies of the behaviour of recursion for the pooling problem, ACM *SIGMAP Bulletin*, 25, 19-28

Hax, A.C. and Candea, D. (1984), *Production and Inventory Mangement*, Prentice Hall, New Jersey

Hazell, P.B.R. and Norton, R.D. (1986), *Mathematical Programming for Economic Analysis in Agriculture*, Macmillan, New York

Hendrix, E.M.T. (1990), Linking Linear Programming to Simulation Models, some experiments for cattle husbandry, *Proceedings of the 32nd Georgikon scientific conference on OR and computer science in agriculture*, Kesthely August 1990, 291-295

Hendrix, E.M.T. and Pintér, J.D. (1991), An Application of Lipschitzian Global Optimization to Product Design, *Journal of Global Optimization*, 1, 389-401

Hendrix, E.M.T., Mecking, C.J. and Hendriks, Th.H.B. (1993), A Mathematical Formulation of Finding Robust Solutions for a Product Design Problem, Technical Note 9302, Department of Mathematics Wageningen

Hendrix, E.M.T., Mous, S.L.J., Roosma, J. and Scholten, H. (1994), Optimality and search strategies for ill conditioned parameter estimation, Technical Note 94-14, Department of Mathematics, Wageningen

Hendrix, E.M.T. and Roosma, J. (1996), Global Optimization with a Limited Solution Time, *Journal of Global Optimization*, 8, 413-427

Hendrix, E.M.T., Mecking, C.J. and Hendriks, Th.H.B. (1996), Finding robust solutions for product design problems, *European Journal of Operational Research*, 92, 28-36

Horst, R. (1986), A General Class of Branch-and-Bound Methods in Global Optimization with Some New Approaches for Concave Minimization, *Journal of Optimization Theory and Applications*, 51, 271-291

Horst, R. and Tuy, H. (1987), On the Convergence of Global Methods in Multiextremal Optimization, *Journal of Optimization Theory and Applications*, 54, 253-271

Horst, R. and Thoai, Ng.V. (1988), Branch-and-Bound Methods for Solving Systems of Lipschitzian Equations and Inequalities, *Journal of Optimization Theory and Applications*, 58, 139-145

Horst, R. (1988), Deterministic Global Optimization with Partition Sets whose Feasibility Is Not Known: Application to Concave Minimization, Reverse Convex Constraints, DC-Programming and Lipschitzian Optimization, *Journal of Optimization Theory and Applications*, 58, 11-37

Horst, R. and Tuy, H. (1990), *Global Optimization (Deterministic Approaches)*, Springer Verlag, Berlin

Horst, R. and Pardalos, P.M. editors (1995), *Handbook of Global Optimization*, Kluwer, Dordrecht

Horst, R., Pardalos, P.M. and Thoai, N.V. (1995), *Introduction to Global Optimization*, Kluwer, Dordrecht

Hung, M.S. and Denton, J.W. (1993), Training neural networks with the GRG2 nonlinear optimizer, *European Journal of Operational Research*, 69, 83-91

Jarvis, R.A. (1975), Adaptive global search by the process of competitive evolution, *IEEE Trans. on Syst., Man and Cybernetics*, 75, 297-311

Kall, P. and Wallace, S.W. (1994), *Stochastic Programming*, Wiley, New York

Karnopp, D.C. (1963), Random search techniques for optimization problems, *Automatica*, 1, 111-121

Kearfott, R.B. (1997), *Rigorous Global Search: Continuous Problems*, Kluwer, Dordrecht

Keesman, K.J. (1990), Membership-set estimation using random scanning and principal component analysis, *Mathematics and Computers in Simulation*, 32, 535-543

Keesman, K.J. (1992), Determination of a minimum-volume orthotopic enclosure of a finite vector set, MRS report 92-01, Wageningen Agricultural University

Khachiyan, L.G. and Todd, M.J. (1993), On the complexity of approximating the maximal inscribed ellipsoid for a polytope, *Mathematical Programming*, 61, 137-159

Kleijnen, J.P.C. and van Groenendaal, W. (1988), *Simulation, a statistical perspective*, Wiley, New York etc

Klepper, O. and Rouse, D.I. (1991), A procedure to reduce parameter uncertainty for complex models by comparison with real system output illustrated on a potato growth model, *Agricultural systems*, 36, 375-395

Klepper, O. and Hendrix, E.M.T. (1994), A Method for Robust Calibration of Ecological Models under Different Types of Uncertainty, *Ecological Modelling*, 74, 161-182

Klepper, O. and Hendrix, E.M.T. (1994), A comparison of algorithms for global characterization of confidence regions for nonlinear models, *Environmental Toxicology and Chemistry*, 13, 1887-1899

Klepper, O. and Slob, W. (1994), Diagnosis of model applicability by identification of incompatible data sets illustrated on a pharmacokinetic model for dioxins in mamals, in *Predictability and Nonlinear Modelling in Natural Sciences and Economics*, eds. J.Grasman and G. van Straten, Kluwer, 527-540

Konno, H. (1976), A Cutting Plane Algorithm for Solving Bilinear Programs, *Mathematical Programming*, 11, 14-27

Konno, H., Yajima, Y. and Ban, A. (1994), Calculating a minimal sphere containing a polytope defined by a system of linear inequalities, *Computational Optimization and Applications*, 3

Konno, H. and Kuno, T. (1995), Multiplicative programming problems, in *Handbook of Global Optimization*, (ed. R. Horst and P.M. Pardalos), 369-405, Kluwer, Dordrecht

Kristindottir, B.P, Zabinsky, Z.B., Csendes, T. and Tuttle M.E. (1993), Methodologies for tolerance intervals, *Interval Computations*, 3, 133-147.

Kularathna, M.D.U.P. (1992), *Application of dynamic programming for the analysis of complex water resources systems: a case study on the Mahaweli River Basin Development in Sri Lanka*, PhD dissertation, Wageningen Agricultural University

Kushner, H.J. (1962), A versatile stochastic model of a function of unknown and time varying form, *Journal of Mathematical Analysis and Applications*, 5, 150-167

Lasdon, L.S., and Waren, A.D. (1978), Generalized Reduced Gradient Software for Linearly and Nonlinearly Constrained Problems, in *Design and Implementation of Optimization Software*, H.J. Greenberg (ed), Sijthoff and Noordhoff, 363-397

Lasdon, L.S., Waren, A.D., Sarkar, S. and Palacios-Gomez, F. (1979), Solving the pooling problem using generalized reduced gradient and succesive linear programming methods, *SIGMAP Bulletin*, 22, 9-15

Lawler, E.L., Lenstra, J.K. and Rinnooy Kan, A.H.G. (1985), *The travelling salesman problem: a guided tour of combinatorial optimization*, Wiley, New York

Liebman, J.S., Schrage, L., Lasdon, L.S. and Waren, A.D. (1985), *GINO, General Interactive Optimizer*, Lindo Systems, Chicago

Liu, P-L. and Der Kiureghian, A. (1991), Optimization algorithms for structural reliability, *Structural Safety*, 9, 161-177

Loehle, C. (1988), Robust parameter estimation for nonlinear models, *Ecological Modelling*, 41, 41-54

Manas, M. (1968), An Algorithm for a Nonconvex Programming Problem, *Economic. Mathem. Obzor*, 4, 202-212

Marquardt, D.W. (1963), An algorithm for least squares estimation of nonlinear parameters, *SIAM Journal*, 11, 431-441

Masson, E. and Wang, Y.J. (1990), Introduction to computation and learning in artificial neural networks, *European Journal of Operational Research*, 47, 1-28

Meewella, C.C. and Mayne, D.Q. (1988), An Algorithm for Global Optimization of Lipschitz Continuous Functions, *Journal of Optimization Theory and Applications*, 57, 307-322

Meijer, E.L. and Buurman, P. (1997), Factor analysis and direct optimization of the amounts and properties of volcanic soil components, *Geoderma*, 80, 129-151

Milutin, D. and Bogardi, J.J. (1996), Application of genetic algorithms to derive the release distribution within a complex reservoir system, in *Hydroinformatics* (A. Müller ed.), A.A. Balkema, Rotterdam

Mladineo, R.H. (1986), An Algorithm for Finding the Global Maximum of a Multimodal, Multivariate Functions, *Mathematical Programming*, 34, 188-200

Mockus, J. (1989), *Bayasian approach to global optimization*, Kluwer, Dordrecht

Mockus, J., Eddy, W., Mockus, A., Mockus, L. and Reklaitis, G. (1997), *Bayasian Heuristic Approach to Discrete and Global Optimization*, Kluwer, Dordrecht

Mol de, R.M. and van Beek, P. (1991), An OR contribution to the solution of environmental problems in the Netherlands caused by manure, *European Journal of Operational Research*, 52, 16-27

Moré, J.J. and Wright, S.J. (1993), *Optimization Software Guide*, Siam, Philadelphia

Mous, S.L.J. (1994), *On identification of nonlinear systems*, PhD dissertation, Wageningen Agricultural University

Murtagh, B.A. and Saunders, M.A. (1978), Large Scale Linearly Constrained Optimization, *Mathematical Programming*, 14, 41-72

Nash, J.F. (1951), Noncooperative Games, *Annals of Mathematics*, 54, 286-295

Nash, S.G. (1995), Software Survey NLP, *OR/MS Today*, 22, 60-71

Nelder, J.A. and Mead, R (1965), A simplex method for function minimization, *The Computer Journal*, 8, 308-313

Papadimitriou, C.H. and Steiglitz, K. (1982), *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall Inc., New Jersey

Pardalos, P.M., Glick, J.H. and Rosen, J.B. (1987), Global Minimization of Indefinite Quadratic Problems, *Computing*, 39, 281-291

Pardalos, P.M. and Rosen J.B. (1987), *Constrained Global Optimization: Algorithms and Applications*, Springer, Berlin

Pardalos, P.M. and Phillips, A.T. (1991), Global Optimization of Fractional Programs, *Journal of Global Optimization*, 1, 173-182

Parkinson, A., Sorensen, C., Free, J. and Canfield B. (1990), Tolerances and robustness in engineering design optimization, in *Advances in Design Automation-1990*, Vol. 1, ASME Publication No. DE-Vol. 23-1, Proceedings of ASME Design Automation Conference, Chicago, IL Sept. 16-19, 1990

Patel, N.R., Smith, R. and Zabinsky, Z.B. (1988), Pure adaptive search in Monte Carlo optimization, *Mathematical programming*, 43, 317-328

Pintér, J.D. (1986), Extended Univariate Algorithms for n-Dimensional Global Optimization, *Computing*, 36, 91-103

Pintér, J.D. (1988), Branch-and-Bound Algorithms for Solving Global Optimization Problems with Lipschitzian Structure, *Optimization*, 19, 101-110

Pintér, J.D. and Pesti, G. (1991), Set partition by globally optimized cluster seed points, *European Journal of Operational Research*, 51, 127-135

Pintér, J.D. (1996a), *Global Optimization in Action; continuous and Lipschitz optimization: algorithms, implementations and application*, Kluwer, Dordrecht

Pintér, J.D. (1996b), Continuous global optimization software: a brief review, *Optima*, 52, 1-8

Polovinkin, A.J. (1970), Algorithms for the search of the global minimum in the design of engineering constructions, *Automation and Computers 1970*, 31-37 (in Russian)

Powell, M.J.D. (1964), An efficient method for finding the minimum of a function of several variables without calculating derivatives, *The Computer Journal*, 7, 155-162

Price, W.L. (1979), A controlled random search procedure for global optimization, *The Computer Journal*, 20, 367-370

Pronzato, L., Walter, E., Venot, A., Lebruchec, J.F. (1984), A General Purpose Global Optimizer: implementation and applications, *Mathematics and Computers in Simulation*, 24, 412-422

Rabbinge, R. and van Latesteijn, H.C. (1992), Long term options for land use in the European Community, *Agricultural Systems*, 40, 195-210

Rackwitz, R. and Fiessler, B. (1978), Structural reliability under combined load sequences, *Computers and Structures*, 9, 489-494

Raghavachari, M. (1969), On connections between zero-one integer programming and concave programming under linear constraints, *Operations Research*, 17, 680-684

Ralston, M.L. and Jennrich, I.J. (1978), Dud, a derivative-free algorithm for nonlinear least squares, *Technometrics*, 20, 7-14

Rasch, D.A.M.K. (1995), *Mathematische Statistik*, Joh. Ambrosius Barth, Leipzig

Rasch, D.A.M.K., Hendrix, E.M.T. and Boer, P.J. (1997), Replication-free optimal designs in regression analysis, *Computational Statistics*, 12, 19-52

Ratscheck, H. and Rokne, J. (1995), Interval methods, in *Handbook of Global Optimization*, (ed. R. Horst and P.M. Pardalos), 751-828, Kluwer, Dordrecht

Rinnooy Kan, A.H.G. and Timmer, G.T. (1987a), Stochastic Global Optimization Methods: I Clustering Methods, *Mathematical Programming*, 39, 27-56

Rinnooy Kan, A.H.G. and Timmer, G.T. (1987b), Stochastic Global Optimization Methods: II Multilevel Methods, *Mathematical Programming*, 39, 57-58

Ripley, B.D. (1981), *Spatial Statistics*, Wiley, New York

Romeijn, H.E. (1992), Global Optimization by Random Walk Sampling Methods, PhD dissertation Erasmus University Rotterdam

Romeijn, H.E., Zabinsky, Z.B., Graesser, D.L. and Neogi, S. (1997), A new reflection generator for simulated annealing in mixed integer/continuous global optimization, technical report, Rotterdam School of Management, Erasmus University Rotterdam

Romero-Morales, D., Carrizosa, E. and Conde, E. (1997), Semi-obnoxious location models: A global optimization approach, *European Journal of Operational Research*, 102, 295-301

Ross, G.J.S. (1990), *Nonlinear Estimation*, Springer, New York

Schaible, S. (1995), Fractional programming, in *Handbook of Global Optimization*, (ed. R. Horst and P.M. Pardalos), 495-608, Kluwer, Dordrecht

Scholten, H., De Hoop, B.J. and Herman, P.M.J. (1990), SENECA 1.2: A simulation Environment for ECological Application (Manual), DIHO, Yerseke, Ecolmod report EM-4, ISBN 90-9003978-3

Sherali, H.D. and Shetty, C.M. (1980), A Finite Convergent Algorithm for Bilinear Programming Problems using Polar Cuts and Disjunctive Face Cuts, *Mathematical Programming*, 19, 14-31

Shubert, B.O. (1972), A Sequential Method Seeking the Global Maximum of a Function, *SIAM Journal of Numerical Analysis*, 9, 379-388

Smith R.L. (1984), Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions, *Operations Research*, 32, 1296-1308

Stortelder, W. (1998), Parameter Estimation in Nonlinear Dynamical Systems PhD dissertation University of Amsterdam

Taguchi, G., Elsayed, E. and Hsiang, T. (1989), *Quality Engineering in Production Systems*, McGraw-Hill.

Tarantola, A. (1987), *Inverse Poblem Theory*, Elsevier, Amsterdam

Thomann, R.V. and Mueller, I. (1987), *Principles of surface water quality modelling and control*, Harper & Row, New York

Timmer, G.T. (1984), Global optimization: a stochastic approach, PhD dissertation Erasmus University Rotterdam

Törn, A. and Žilinskas, A. (1989), *Global Optimization*, Springer, Berlin

Tuy, H. (1995), D.C. Optimization: theory, methods and algorithms, in *Handbook of Global Optimization*, (ed. R. Horst and P.M. Pardalos), 149-216, Kluwer, Dordrecht

Tuy, H., Ghannadan, S., Migdalas, A. and Värband, P. (1996), A strongly polynomial algorithm for a concave production-transportation problem with a fixed number of nonlinear variables, *Mathematical Programming*, 72, 229-258

Vet, R.P. van der (1980), Flexible solutions to systems of linear equalities en inequalities, Dissertation, Eindhoven Technical University, Eindhoven.

Walter, E. (1982), *Identifiability of state space models*, Springer, New York

Walter, E. and Piet-Lahanier, H. (1988), Estimation of the parameter uncertainty resulting from bounded-error data, *Mathematical Bioscience*, 92, 55-74

Walter, E. and Piet-Lahanier, H. (1990), Estimation of parameter bounds from bounded-error data: a survey, *Mathematics and Computers in Simulation*, 32, 449-468

Wendell, R.E. and Hurter, A.P. (1976), Minimization of a Non-Separable Objective Function Subject to Disjoint Constraints, *Operations Research*, 24, 643-657

Williams, H.P. (1990), *Model Building in Mathematical Programming*, Wiley, New York

Winston, W.L. (1994), *Operations Research, applications and algorithms*, Duxbury, Belmont

Wood, G.R. (1992), The Bisection Method in Higher Dimensions, *Mathematical Programming*, 55, 319-337

Zabinsky, Z.B. and Smith, R.L. (1992), Pure Adaptive Search in Global Optimization, *Mathematical Programming*, 53, 323-338

Žilinskas, A. (1992), A review of statistical methods for global optimization, *Journal of Global Optimization*, 2, 145-153

Summary

In many research situations where mathematical models are used, researchers try to find parameter values such that a given performance criterion is at an optimum. If the parameters can be varied in a continuous way, this in general defines a so-called Nonlinear Programming Problem. Methods for Nonlinear Programming usually result in local optima. A local optimum is a solution (parameter values) which is the best with respect to values in the neighbourhood of that solution, not necessarily the best over the total admissible, feasible set of all possible parameter values, solutions.

For mathematicians this results in the research question: How to find the best, global optimum in situations where several local optima exist?, the field of Global Optimization (GLOP). Literature, books and a specific journal, has appeared during the last decades on the field. Main focus has been on the mathematical side, i.e. given assumptions on the structure of the problems to be solved and specific global optimization methods and properties are derived. Cooperation between mathematicians and researchers (in this book called 'the modeller' or 'the potential user'), who saw global optimization problems in practical problems has lead to application of GLOP algorithms to practical optimization problems. Some of those can be found in this book. In this book we started with the question:

Given a potential user with an arbitrary global optimization problem, what route can be taken in the GLOP forest to find solutions of the problem?

From this first question we proceed by raising new questions. In Chapter 1 we outline the target group of users we have in mind, i.e. agricultural and environmental engineers, designers and OR workers in agricultural science. These groups are not clearly defined, nor mutually exclusive, but have in common that mathematical modelling is used and there is knowledge of linear programming and possibly of combinatorial optimization.

In general, when modellers are confronted with optimization aspects, the first approach is to develop heuristics or to look for standard nonlinear programming codes to generate solutions of the optimization problem. During the search for solutions, multiple local optima may appear. We distinguished two major tracks for the path to be taken from there by the potential user to solve the problem. One track is called the deterministic track and is discussed in Chapters 2, 3 and 4. The other track is called the stochastic track and is discussed in Chapters 5 and 6. The two approaches are intended to reach a different goal. The deterministic track aims at:

The global optimum is approximated (found) with certainty in a finite number of steps.

The stochastic track is understood to contain some stochastic elements and aims at:

Approaching the optimum in a probabilistic sense as effort grows to infinity.

Both tracks are investigated in this book from the viewpoint of a potential user corresponding to the way of thinking in Popperian science. The final results are new challenging problems, questions for further research. A side question along the way

is: *How can the user influence the search process given the knowledge of the underlying problem and the information that becomes available during the search?*

The deterministic approach
When one starts looking into the deterministic track for a given problem, one runs into the requirements which determine a major difference in applicability of the two approaches.

Deterministic methods require the availability of explicit mathematical expressions of the functions to be optimized.

In many practical situations which are also discussed in this book, these expressions are not available and deterministic methods cannot be applied. The operations in deterministic methods are based on concepts such as Branch-and-Bound and Cutting which require bounding of functions and parameters based on so-called mathematical structures.

In Chapter 2 we describe these structures and distinguish between those which can be derived directly from the expressions, such as quadratic, bilinear and fractional functions and other structures which require analysis of the expressions such as concave and Lipschitz continuous functions. Examples are given of optimization problems revealing their structure. Moreover, we show that symmetry in the model formulation may cause models to have more than one extreme.

In Chapter 3 the relationship between GLOP and Integer Programming (IP) is highlighted for several reasons.

- Sometimes practical GLOP problems can be approximated by IP variants and solved by standard Mixed Integer Linear Programming (MILP) techniques.
- The algorithms of GLOP and IP can similarly be classified.
- The transformability of GLOP problems to IP problems and vice versa shows that difficult problems in one class will not become easier to solve in the other.
- Analysis of problems, which is common in Global Optimization, can be used to better understand the complexity of some IP problems.

In Chapter 4 we analyze the use of deterministic methods, demonstrating the application of the Branch-and-Bound concept. The following can be stated from the point of view of the potential user:

- Analysis of the expressions is required to find useful mathematical structures (Chapter 2). It should be noted that also interval arithmetic techniques can be applied directly on the expressions.
- The elegance of the techniques is the guarantee that we are certain about the global optimality of the optimum, when it has been discovered and verified.
- The methods are hard to implement. Thorough use should be made of special data structures to store the necessary information in memory.

Two cases are elaborated. The quadratic product design problem illustrates how the level of Decision Support Systems can be reached for low dimensional problems, i.e. the number of variables, components or ingredients, is less than 10. The other case, the nutrient problem, shows how by analysis of the problem many useful properties can be derived which help to cut away large areas of the feasible space where the optimum cannot be situated. However, it also demonstrates the so-called Curse of Dimensionality; the problem has so many variables in a realistic situation that it is impossible to traverse the complete Branch-and-Bound tree. Therefore it is good to see the relativity of the use of deterministic methods:

No global optimization method can guarantee to find and verify the global optimum for every practical situation, within a humans lifetime.

The stochastic approach

The stochastic approach is followed in practice for many optimization problems by combining the generation of random points with standard nonlinear optimization algorithms. The following can be said from the point of view of the potential user.

- The methods require no mathematical structure of the problem and are therefore more generally applicable.
- The methods are relatively easy to implement.
- The user is never completely certain that the global optimum has been reached.
- The optimum is approximated in a probabilistic sense when effort increases to infinity.

In Chapter 5 much attention is paid to the question what happens when a user wants to spend a limited (not infinite) amount of time to the search for the optimum, preferably less than a humans lifetime:

What to do when the time for solving the problem is finite?

First we looked at the information which becomes available during the search and the instruments with which the user can influence the search. It appeared that besides classical instruments which are also available in traditional nonlinear programming, the main instrument is to influence the trade-off between global (random) and local search (looking for a local optimum). This lead to a new question:

Is there a best way to rule the choice between global and local search, given the information which becomes available?

Analyzing in a mathematical way with extreme cases lead to the comfortable conclusion that a best method of choosing between global and local search -thus a best global optimization method- does not exist. This is valid for cases where further information (more than the information which becomes available during the search) on the function to be optimized is not available, called in literature the black-box case. The conclusion again shows that mathematical analysis with extreme cases is a powerful tool to demonstrate that so-called magic algorithms –algorithms which are said in scientific journals to be very promising, because they perform well on some test cases– can be analyzed and 'falsified' in the way of Popperian thinking. This leads to the conclusion that:

Magic algorithms which are going to solve all of your problems do not exist.

Several **side questions** derived from the main problem are investigated in this book. In Chapter 6 we place the optimization problem in the context of parameter estimation. One practical question is raised by the phenomenon

Every local search leads to a new local optimum.

We know from parameter estimation that this is a symptom in so called non-identifiable systems. The minimum is obtained at a lower dimensional surface or curve. Some (non-magic) heuristics are discussed to overcome this problem.
There are two side questions of users derived from the general remark:

"I am not interested in the best (GLOP) solution, but in good points".

The first question is that of Robust Solutions, introduced in Chapter 4, and the other is called Uniform Covering, concerning the generation of points which are nearly as good as the optimum, discussed in Chapter 6.

Robust solutions are discussed in the context of product design. The robustness is defined as a measure of the error one can make from the solution so that the solution (product) is still acceptable. Looking for the most robust product is looking for that point which is as far away as possible from the boundaries of the feasible (acceptable) area. For the solution procedures, we had a look at the appearance of the problem in practice, where boundaries are given by linear and quadratic surfaces, properties of the product.

- For linear boundaries, finding the most robust solution is an LP problem and thus rather easy.
- For quadratic properties the development of specific algorithms is required.

The question of Uniform Covering concerns the desire to have a set of "suboptimal" points, i.e. points with low function value (given an upper level of the function value); the points are in a so-called level set. To generate "low" points, one could run a local search many times. However, we want the points not to be concentrated in one of the compartments or one sub-area of the level set, we want them to be equally, uniformly spread over the region. This is a very difficult problem for which we test and analyze several approaches in Chapter 6. The analysis taught us that:

It is unlikely that stochastic methods will be proposed which solve problems in an expected calculation time, which is polynomial in the number of variables of the problem.

Final result

Whether an arbitrary problem of a user can be solved by GLOP requires analysis. There are many optimization problems which can be solved satisfactorily. Besides the selection of algorithms the user has various instruments to steer the process. For stochastic methods it mainly concerns the trade-off between local and global search. For deterministic methods it includes setting bounds and influencing the selection rule in Branch-and-Bound. We hope with this book to have given a tool and a guidance to solution procedures. Moreover, it is an introduction to further literature on the subject of Global Optimization.

Samenvatting

In veel onderzoeken waarin gebruik wordt gemaakt van wiskundige modellen, proberen onderzoekers parameterwaarden te vinden waarbij een criterium een beste, optimale, waarde bereikt. Wanneer de waarden vrij gevarieerd kunnen worden, leidt dit al snel tot een zogenaamd Niet-Lineair Programmeringsprobleem. Methoden voor Niet-Lineaire Programmering (NLP) resulteren in het algemeen in oplossingen, parameterwaarden, die het beste zijn in de directe omgeving van de oplossing, maar niet noodzakelijkerwijze het beste op het gehele toegelaten gebied van mogelijke waarden, zogenaamde lokale optima. Voor wiskundigen geeft dit de onderzoeksvraag: *Hoe kan het beste, globale optimum worden gevonden, wanneer er sprake is van meerdere lokale optima?*, het onderwerp van de zogenaamde Globale Optimalisering (GLOP). Literatuur bestaande uit boeken en een specifiek tijdschrift op dit gebied zijn de laatste decennia verschenen. Nadruk lag daarin vooral op de wiskundige kant; gegeven aannamen ten aanzien van de structuur van op te lossen problemen en globale optimaliseringsalgoritmen, worden eigenschappen (theorema's) afgeleid. Samenwerking tussen wiskundigen en onderzoekers die tegen optimaliseringsproblemen uit de praktijk aanliepen, in dit boek de potentiële gebruikers of modelleurs genoemd, heeft geleid tot toepassing van GLOP algoritmen. Sommige van deze praktische problemen worden besproken in dit boek. Uitgangsvraag in dit boek is:

> *Gegeven een potentiële gebruiker met een willekeurig globaal optimaliseringsprobleem, welk pad kan worden bewandeld in het "GLOP bos" om te komen tot oplossingen van het probleem?*

Vanuit deze vraag ontwikkelt het onderzoek zich door het telkens stellen van nieuwe vragen. In hoofdstuk 1 wordt eerst de doelgroep geschetst waaraan we denken: Onderzoekers in de landbouw en milieu wetenschappen, ontwerpers en OR mensen. Deze groepen zijn niet duidelijk afgebakend en overlappen, maar hebben gemeen dat wiskundige modellen worden gebruikt in het onderzoek en dat er kennis is over Lineaire Programmering en Combinatorische Optimalisering.

Wanneer een modelleur optimaliseringsaspecten tegenkomt, is in het algemeen de eerste aanpak het ontwikkelen van heuristieken of het zoeken van NLP codes voor het genereren van (goede) oplossingen van het optimaliseringsprobleem. Tijdens het zoeken kan men tot de ontdekking komen dat er meerdere lokale optima zijn. Voor de weg die de potentiële gebruiker kan volgen vanaf dit moment worden in dit boek twee aanpakken beschreven. Een aanpak is de deterministische aanpak gedoopt en wordt besproken in hoofdstukken 2,3 en 4. De andere wordt de stochastische aanpak genoemd en wordt besproken in hoofdstukken 5 en 6. De twee aanpakken verschillen in uitgangspunt. De deterministische aanpak streeft naar:

> *Het globale optimum wordt met zekerheid bereikt (benaderd) in een eindig aantal stappen.*

Stochastische methoden bevatten stochastische aspekten en mikken op:

> *Het globale optimum wordt op een probabilistische wijze met toenemende inspanning benaderd.*

Beide aanpakken worden onderzocht in dit boek vanuit het gezichtspunt van een potentiële gebruiker volgens de gedachte van Popper. Het uiteindelijke resultaat zijn nieuwe uitdagingen, vragen voor verder onderzoek. Een nevenvraag is:

> *Hoe kan de gebruiker gedurende het zoekproces gebruik maken van informatie over het probleem en van informatie die vrijkomt tijdens het zoeken?*

De deterministische aanpak

Wanneer de deterministische aanpak wordt bekeken voor een gegeven probleem, loopt men al snel tegen een vereiste aan die het verschil in toepasbaarheid van de twee aanpakken bepaalt.

> *Voor deterministische methoden zijn expliciete wiskundige uitdrukkingen nodig van de functie die wordt geoptimaliseerd.*

In vele praktische situaties die men ook in dit boek tegenkomt, zijn deze uitdrukkingen niet beschikbaar. Deterministische methoden kunnen dan niet worden toegepast. De methoden zijn gebaseerd op concepten zoals het genereren van sneden en het toepassen van Branch-and-Bound welke afschattingen van functies of parameterwaarden vereisen die weer zijn gebaseerd op wiskundige structuren.

In hoofdstuk 2 worden deze structuren beschreven, waarbij onderscheid wordt gemaakt tussen structuren die direct te herkennen zijn uit de formules, zoals kwadratische, bilineaire en fractionele functies en structuren die verdere analyse vragen zoals concave en Lipschitz continue functies. Voorbeeldproblemen worden besproken met een herkenbare structuur. Verder wordt aangetoond dat symmetrie in de modelformulering verantwoordelijk kan zijn voor het bestaan van meerdere optima.

In hoofdstuk 3 wordt de relatie tussen GLOP en Geheeltallige Programmering (IP) besproken. Daar zijn verscheidene redenen voor.

- Benadering van praktische GLOP problemen is soms mogelijk met IP formuleringen en de oplossing ervan met standaard Gemengd Geheeltallige Linaire Programmering technieken.
- Indeling van GLOP en IP algoritmen kan op eenzelfde wijze.
- Het omzetten van problemen uit de GLOP naar de IP klasse en vice-versa laat zien dat moeilijke problemen uit een klasse niet tot eenvoudig op te lossen problemen reduceren in de andere klasse.
- Analyse van problemen zoals we die kennen uit de GLOP kan worden gebruikt om complexiteit van verschillende IP problemen beter te begrijpen.

In hoofdstuk 4 wordt de werking van deterministische methoden gedemonstreerd. Vanuit het oogpunt van de potentiële gebruiker kan worden opgemerkt:

- Analyse is nodig voor het ontdekken van bruikbare wiskundige structuren (hoofdstuk 2). Overigens kunnen ook zogenaamde Interval Methoden worden toegepast op de wiskundige uitdrukkingen.
- Voordeel van de aanpak is dat we zeker zijn van de globale optimaliteit van het optimum, wanneer dit is gevonden.
- De methoden zijn moeilijk te implementeren en vereisen handig gebruik van data structuren om de nodige informatie op te slaan.

Twee cases worden uitgewerkt. Het kwadratische ontwerp probleem laat zien dat het niveau van Beslissing Ondersteunende Systemen kan worden bereikt voor relatief laag-dimensionale problemen d.w.z. het aantal variabelen is niet groter dan 10. De andere case, het nutriënten probleem, laat zien dat een analyse veel nuttige eigenschappen afleidt waarmee grote delen van het toegelaten gebied kunnen worden geschrapt, omdat het optimum daar niet kan liggen. Het laat echter ook de zogenaamde "Curse of Dimensionality" zien: Het probleem heeft in een realistisch model zoveel variabelen, dat een methode niet in staat zal zijn de volledige Branch-and-Bound boom te doorlopen. Daarom is het goed om de relativiteit van deterministische methoden te zien:

Geen enkel algoritme kan garanderen dat het optimum van een willekeurig GLOP probleem is te vinden en te verifiëren binnen een mensenleven.

De stochastische aanpak

De stochastische aanpak wordt in de praktijk vaak gevolgd door de combinatie van het genereren van toevalsgetallen en standaard niet-lineaire optimaliseringsalgoritmen. Het volgende kan worden gezegd vanuit het oogpunt van de potentiële gebruiker:

- De methoden vragen niet om een bijzondere wiskundige structuur en zijn daardoor meer algemeen toepasbaar.
- De methoden kunnen relatief eenvoudig worden geïmplementeerd.
- Het is nooit absoluut zeker dat het globale optimum is bereikt.
- Het optimum wordt bijna zeker benaderd wanneer de inspanning toeneemt tot oneindig.

In hoofdstuk 5 wordt veel aandacht besteed aan de vraag wat een gebruiker het beste kan doen wanneer hij een eindige hoeveelheid tijd (liefst minder dan een mensenleven) wil besteden aan het zoeken naar het optimum:

Wat is het beste om te doen wanneer er een beperkte tijd is voor het oplossen van het probleem?

Allereerst is er gekeken naar de informatie die vrijkomt gedurende het zoekproces en welke knoppen een gebruiker heeft om de zoektocht te beïnvloeden. Er bleek dat naast de klassieke middelen die ook gebruikt worden bij traditionele NLP methoden de hoofdknop vooral bestaat uit de afweging tussen globaal (met toevalsgetallen) en lokaal zoeken. Dit geeft de volgende vraag:

Bestaat er een beste keuzeregel om de keuze tussen globaal en lokaal zoeken te sturen met de informatie die vrijkomt?

Wiskundige analyse met extreme gevallen leidde tot de rustgevende conclusie dat zo'n beste keuzeregel niet kan bestaan. Let wel, dit geldt dan voor gevallen waarbij er niet meer informatie beschikbaar is over het probleem dan hetgeen wat vrijkomt tijdens het zoeken. In de literatuur worden dit de orakel of black-box gevallen genoemd. Dit toont aan dat wiskundige analyse met extreme gevallen een sterk instrument is om magische algoritmen, methoden waarvan in wetenschappelijke tijdschriften de indruk wordt gewekt dat het een wondermiddel is omdat ze op een aantal testvoorbeelden goed scoren, te ontmaskeren ofwel 'falsifiëren' volgens de Popperiaanse denkwijze. Wiskunde geeft ons dus een instrument om aan te tonen:

Wonderalgoritmen die al uw optimaliseringsproblemen oplossen bestaan niet.

Verschillende nevenvragen afgeleid van de hoofdvraag zijn in dit boek onderzocht. In hoofdstuk 6 is het optimaliseringsprobleem in de context bekeken van het schatten van parameters. Een praktische vraag ontstaat door het verschijnsel

Elke lokale minimalisatie leidt tot een nieuw lokaal minimum.

Bij het schatten van parameters kennen we dit verschijnsel wanneer er sprake is van zogenaamde niet-identificeerbare systemen. Het minimum wordt dan eenvoudigweg aangenomen door alle punten op een lager dimensionaal oppervlak of een curve. Enkele heuristieken (geen wondermiddel!) worden besproken om dit probleem aan te pakken.

Twee andere vragen zijn afgeleid van de algemene opmerking:

"Ik ben niet geïnteresseerd in de beste (GLOP) oplossing, maar in goede punten".

De eerste vraag betreft de kwestie van zogenaamde robuuste oplossingen (hoofdstuk 4) en de tweede betreft het genereren van bijna-optimale punten, uniform overdekken genoemd (hoofdstuk 6).

Robuustheid wordt in de context van het ontwerpen van produkten bekeken en gedefinieerd als de fout (afwijking t.o.v. het ontwerp) die tijdens de produktie kan worden gemaakt zonder dat het produkt afgekeurd wordt. Het bepalen van het meest robuuste produkt, betekent dus zover mogelijk van de randen van het acceptatie gebied af gaan zitten. Voor het bedenken van goede methoden hebben we gekeken naar praktijk (uit de smeermiddelen en elektronica industrie) problemen waarbij de grenzen van het gebied worden bepaald door lineaire en kwadratische oppervlakten, eigenschappen van het produkt. Conclusie:

- Voor lineaire eigenschappen bleek het bepalen van het meest robuuste probleem een LP probleem te zijn en dus eenvoudig op te lossen.
- Voor kwadratische problemen dienen specifieke algoritmen te worden ontwikkeld. Deze zijn besproken in hoofdstuk 4.

De kwestie van uniform overdekken vloeit voort uit de wens om een aantal sub-optimale punten te willen hebben, d.w.z. een aantal punten met lage functiewaarden. De punten liggen in een zogenaamde level set, een verzameling met een gegeven bovengrens op de functiewaarde. Om deze punten te genereren zou men natuurlijk een aantal keer een lokale zoekmethode (NLP) kunnen uitvoeren. We willen echter ook dat de punten netjes (uniform) verspreid liggen over de compartimenten van de level set, als het ware deze verzameling overdekken en niet dat alle punten op een hoopje liggen in een deelgebied. Dit bleek een extreem moeilijk probleem waarvoor we in hoofdstuk 6 diverse methoden hebben bekeken. De analyse heeft ons ook het volgende geleerd:

Het is onwaarschijnlijk dat er in de toekomst stochastische methoden zullen worden ontwikkeld met een verwachtte oplossingstijd die polynomiaal is in het aantal variabelen van het op te lossen probleem.

Uiteindelijk resultaat

Het oplossen van een willekeurig probleem door globale optimaliseringsmethoden vereist analyse. Zoals geïllustreerd zijn er vele praktische problemen die met behulp van globale optimalisering kunnen worden opgelost. Naast de keuze van algoritmen heeft een gebruiker ook een aantal instrumenten om het zoekproces te sturen. Bij stochastische methoden is dit voornamelijk de keuze tussen lokaal en globaal zoeken. Bij deterministische methoden bestaat dit uit het aanleveren van grenzen en het beïnvloeden van de keuzeregel bij Branch-and-Bound. We hopen dat dit boek een leidraad vormt in de richting van oplossingsmethoden en verdere literatuur over globale optimalisering.

About the author

Eligius Maria Theodorus Hendrix was born December 20th, 1962 in Roosendaal as number 4½ in the family of an employee of the Catholic Metal Workers Union St. Eligius. He completed his secondary education at Gymnasium Bernrode, Heeswijk, in 1981 and in the same year started to study Econometrics at the Catholic University of Tilburg. In 1984 he obtained his Bachelor degree in Econometrics and started to work as a teaching assistant first for the section Mathematics and later for the section Operational Research until 1987.
In 1986 he added subjects on Development Planning at Erasmus University in Rotterdam to his studies. In 1987 he graduated in Tilburg with a master's thesis written at the United Nations Industrial Development Organisation in Vienna. In the same year he moved to the Department of Mathematics at the Agricultural University in Wageningen as a Universitary Lecturer. Eligius Hendrix is a member of the Mathematical Programming Society, the Dutch Society for Statistics and Operational Research, the Dutch MZ Club and the Society of Informatics in the Agricultural Sector.