International Series in Operations Research & Management Science

Series Editor
Frederick S. Hillier
Stanford University

For further volumes:
http://www.springer.com/series/6161

Dylan Jones · Mehrdad Tamiz

Practical Goal Programming

Springer

Dylan Jones
Logistics and Management Mathematics Group
Department of Mathematics
University of Portsmouth
Lion Gate Building
Lion Terrace
Portsmouth
UK PO1 3HF
dylan.jones@port.ac.uk

Mehrdad Tamiz
College of Business Administration
Department of Quantitative Methods and Information Systems
Kuwait University
P.O. Box 5486
Safat 13055
Kuwait
mehrdad@cba.edu.kw

ISSN 0884-8289
ISBN 978-1-4419-5770-2 e-ISBN 978-1-4419-5771-9
DOI 10.1007/978-1-4419-5771-9
Springer New York Dordrecht Heidelberg London

Library of Congress Control Number: 2010921491

Dedicated to our parents

Rahmat Tamiz and Mahtalat Helmi
and
James Jones and Brenda Jones

Preface

The setting and attainment of goals is a fundamental aspect of human decision making, which is manifest in the modern discipline of operational research by the technique of goal programming. Influences from the fields of mathematical programming and multiple criteria decision making (MCDM) can be found in goal programming, and it is our view that in order to use goal programming most effectively, users should be aware of the basic aspects and concepts of both fields.

We are aware that this will not be the first book on goal programming and have ourselves learnt about the topic from the previous works of Charnes and Cooper (1961), Lee (1970), Ignizio (1976, 1982, 1985, 1994), and Romero (1991). We have noted, however, that most of these excellent books are now out of print, and practitioners and post-graduate students in the field are reporting increasing difficulty in obtaining a work that will allow them to grasp the fundamentals of goal programming and hence utilise the full power and flexibility of the technique. It is also our desire to see goal programming continue to develop and be applied in a practical and correct manner. We would also like users of the technique to have access to and the ability to choose from the full range of variants and extensions of goal programming and its analysis tools in order to build models that best reflect the preferences and desires of their decision maker(s).

The purpose of this book is therefore to empower academics and practitioners to be able to build effective goal programming models, as well as to detail the current state of the art in the topic and lay the foundation for its future development and continued application to new and varied fields of application as they arise.

The notation and terminology used in this book for investigating GP and its variants have been designed and refined in collaboration with the leading experts in the field. We believe they give the best description of the subject and would want them to become the standards.

This book is divided into nine chapters. Chapter 1 gives a brief history of goal programming and details the fundamental definitions arising from the fields of mathematical programming and multiple criteria decision making that are used throughout the text. A section on the underlying philosophies of goal programming is also included. Chapter 2 details the goal programming variants and defines them algebraically.

Chapter 3 is particularly important to readers unfamiliar with goal programming who wish to learn how to build effective goal programming models that avoid common modelling pitfalls and formulation errors. It details the step-by-step formulation of a basic goal programming model in the form of each of the three main variants, as well as discussing basic modelling techniques. Chapter 4 details more advanced modelling issues and highlights some recently proposed extensions as well as giving a new and pragmatic weight sensitivity algorithm.

Chapter 5 details the solution methodologies of goal programming, concentrating on computerised solution by the Excel Solver and LINGO packages for each of the three main variants. A discussion of the viability and use of specialised goal programming packages is also included. Chapter 6 discusses the linkages between Pareto efficiency and goal programming. The state of the art in detecting Pareto inefficiency and restoring Pareto efficiency for each of the major variants is given.

Chapters 3–6 are supported by a set of 10 exercises drawn from our two decades of experience of applying goal programming to practical decision-making situations. An Excel spreadsheet giving the basic solution of each example can be found on the accompanying website (www.mopgp.com).

Chapter 7 details the current state of the art in terms of the integration of goal programming with other techniques from operational research and artificial intelligence. This is a key area which we believe will be an important topic of future research.

The text concludes with two case studies which are chosen to demonstrate the application of goal programming in practice and to illustrate the principles developed in Chapters 1–7. Chapter 8 details an application in health care and Chapter 9 describes applications in portfolio selection.

We are indebted to the many researchers in the field whose works on goal programming, conversations, and presentations on the topic have helped shape this text. In particular we would like to acknowledge the seminal work of James Ignizio and Carlos Romero. We would also like to thank the many academic members, international visitors, and doctoral students of the Management Mathematics Group at the University of Portsmouth, who have contributed to our research on the theory and application of goal programming in the past two decades. These include Simon Mardle, Rishma Hasham, Keith Fargher, Bijan Hesni, Keyvan Mir-Razavi, Sita Patel, Zul Mohd-Nopiah, Richard Treloar, John-Paul Oddoye, Jon Large, Patrick Beullens, Reza Khorramshahgol, XiaoDong Li, Kevin Willis, Rania Azmi, Blanca Perez, Amelia Bilbao, Ali Foroughi, Mohammad Ali Yaghoobi, Mohammad Afzalinejad, and Ersilia Liguigli. We are also grateful to Dr Paul Schmidt of the Queen Alexandra Hospital in Portsmouth for his help in developing the case study described in Chapter 8. We would also like to thank the British Royal Society for providing Dr Jones with an international visiting fellowship to the University of Malaga, Spain, where writing of this book was started. Dr Jones would also like to thank Rafael Caballero and the members of the Department of Applied Economics at the University of Malaga; Maria-Victoria Rodriguez and the members of the Department of Quantitative Economics at the University of Oviedo; and Sydney Chu and the members of the Department of Mathematics, University of Hong Kong,

for their hospitality and useful discussions about goal programming during his stay at their respective institutions.

Last, but not least, we initiated the international series of bi-annual conferences in multi-objective and goal programming (MOPGP). The seventh conference was held in Portsmouth in September 2008. We would particularly like to thank Belaid Aouni and Ralph Steuer for their on-going commitment and help in ensuring the success of this conference series over the years. We also wish to thank all the delegates of these conferences for their participation, useful discussions, and indirect contributions to the completion of this book.

Finally, we would like to thank our families: Sherry, Yasaman, and Sam Rahmat Tamiz; and Katia, Thais, Isabella, and Owen Jones for their support and encouragement during the sometimes time-consuming process of writing this book.

University of Portsmouth, UK — Dylan Jones
Kuwait University, Kuwait — Mehrdad Tamiz
September 2009

Contents

Chapter 1
History and Philosophy of Goal Programming

Man is a goal seeking animal. His life only has meaning if he is reaching out and striving for his goals.

Aristotle 384–322 BC

The above quote demonstrates that goal-based behaviour and decision making has a long history. This goal-based philosophy has been formalised in the modern field of operational research and management science by the technique of goal programming. The earliest goal programming formulation was introduced by Charnes et al. (1955) in the context of executive compensation. At that point the term 'goal programming' was not used and the model was seen as an adaptation of linear programming. A more formal theory of goal programming is given by Charnes and Cooper (1961). Further development took place by Ijiri (1965), and seminal textbooks by Lee (1972) and Ignizio (1976) brought the technique into common usage as an operational research tool. This led to large number of applications being reported in the literature from the mid-1970 onwards. The relatively straightforward ease of which a goal programme could be formulated and the familiarity of practitioners and academics with linear programming methodology ensured that goal programming quickly rose to become the most popular technique within the field of multi-criteria decision making (MCDM). This also, however, brought problems. Although goal programming can correctly be viewed as a generalisation of linear programming, it is also a bona fide multi-criteria decision-making technique. Therefore, users of goal programming should be aware of the theories, methodologies, and pitfalls of multiple criteria decision making if they are to build 'effective' models. Thus goal programming came under criticism in the 1980s because of some basic errors caused, in our opinion, by lack of awareness of good MCDM practice. These included the generation of Pareto-inefficient solutions, the use of excessive numbers of priority levels leading to redundancy, lack of weight sensitivity analysis, direct comparison of incommensurable goals, and ineffectual elicitation and representation of decision maker preferences. This debate culminated in the publication of a key textbook by Romero (1991) in which good goal programming practice is detailed and the problems shown to be due more to poor modelling practice rather than any fundamental deficiency in goal programming.

D. Jones, M. Tamiz, *Practical Goal Programming*, International Series in Operations Research & Management Science 141, DOI 10.1007/978-1-4419-5771-9_1,

During the 1990s there were further theoretical developments and more complex and varied applications of goal programming. Greater awareness of the multiple criteria aspects of goal programming was in evidence in portions of the literature, with more care being taken to avoid the modelling pitfalls outlined by Romero (1991). There were advances in the user-friendliness of computer software, with the introduction of the GPSYS system which had automated checking for many common modelling errors as well as suggesting ways of overcoming them (Jones et al., 1997). In addition, modern mathematical programming modelling and solution systems made it easier to model and solve all variants of goal programming. Goal programming was combined with other techniques from MCDM and the wider field of operational research with symbiotic advantages. This has been termed the 'trend of integration and combination' and will be further explored in Chapter 7. In addition, the MOPGP: Multiple Objective and Goal Programming – Theory and Application International Conference Series (Tamiz, 2009) started in 1994 and has been running on bi-annual basis since then. This series has provided a dedicated forum for academics and practitioners to discuss and advance the state of multiple objective and goal programming. Figure 1.1 shows an increasing use of goal programming in the early to mid-2000s, with a wide variety of modern fields of application utilising the simplicity and power of the technique.

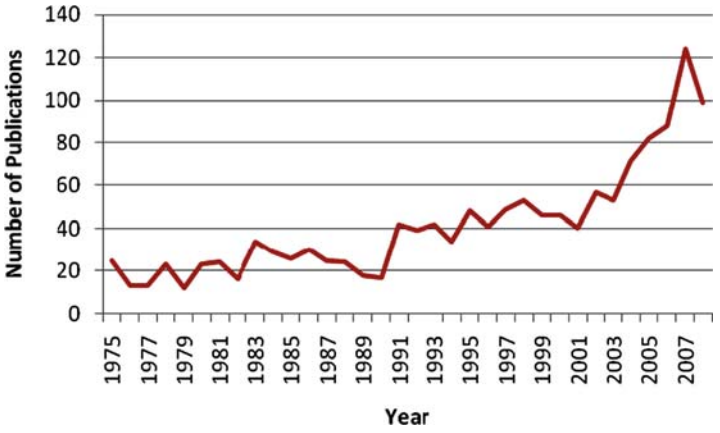

Fig. 1.1 Goal programming publications in the period 1975–2008. Source: ISI Web of Knowledge, published by Thomson Reuters

1.1 Terminology

The plethora of terminology used in the field of MCDM can be confusing, with Ehrgott (2005) listing eight different variants of the most fundamental definitions in the field. This section therefore details the basic definitions and concepts from the fields of MCDM and mathematical programming pertaining to goal programming

that will be used throughout this book. These concepts will be developed further and brought together algebraically in Chapter 2.

Definition 1.1 – Decision Maker(s): The decision maker(s) in this text refer to the person(s), organisation(s), or stakeholder(s) to whom the decision problem under consideration belongs. For a description of group and organisational decision-making processes in the case of multiple decision makers the reader is referred to French et al. (2009).

Definition 1.2 – Decision Variable: A decision variable is defined as a factor over which the decision maker has control. An example is a manufacturing company which has to decide how many of a certain product to make in the next month. The set of decision variables fully describe the problem and form the decision to be made. The purpose of the goal programming model can be viewed as a search of all the possible combinations of decision variable values (known as decision space) in order to determine the point which best satisfies the decision maker's goals and constraints. Figure 3.1 in Chapter 3 is an example of a graph drawn in decision space.

Definition 1.3 – Criterion: A criterion is a single measure by which the goodness of any solution to a decision problem can be measured. There are many possible criteria arising from different fields of application but some of the most commonly arising relate at the highest level to

- Cost
- Profit
- Time
- Distance
- Performance of a system
- Company or organisational strategy
- Personal preferences of the decision maker(s)
- Safety considerations

A decision problem which has more than one criterion is therefore referred to as a multi-criteria decision making (MCDM) or multi-criteria decision aid (MCDA) problem. The space formed by the set of criteria is known as criteria space.

Definition 1.4 – 'Objective': An objective in this book will be referred to as a criterion with the additional information of the direction (maximise or minimise) in which the decision maker(s) prefer on the criterion scale, for example minimise cost or maximise the performance of a system. A decision problem with a set of objectives to be maximised or minimised is referred to as a **multi-objective optimisation problem**. In practice, these objectives will be conflicting, that is they cannot reach their optimal values simultaneously. If they could, then the model can be solved as a single-objective problem for any of the objectives. The space formed by the values of the set of objectives is known as objective space. Figure 1.2 shows an example of objective space for a bi-objective decision problem.

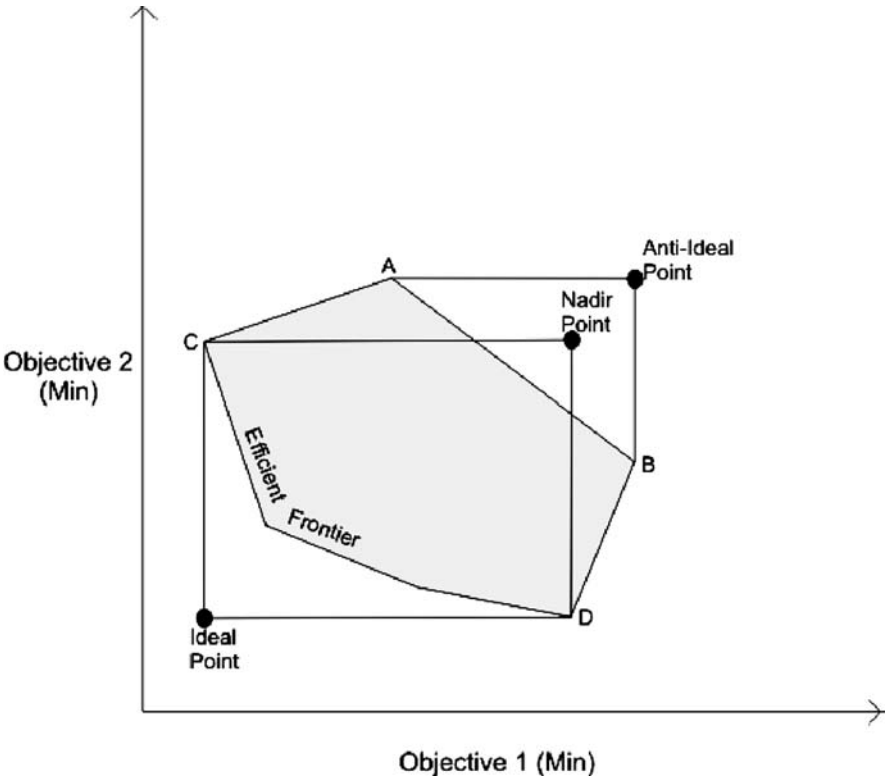

Fig. 1.2 Graphical representation of objective space for bi-objective model

Table 1.1 Three principal types of goals

Goal type	Significance	Example
1	Achieve at most the target level	Keep costs within a budget of $1 m
2	Achieve at least the target level	Aim to produce at least 20 items
3	Achieve the target level exactly	Aim to employ exactly 20 workers

Definition 1.5 – 'Goal': A goal in this book refers to a criterion and a numerical level, known as a **target level**, which the decision maker(s) desire to achieve on that criterion. There are three principal types of goals that can occur in a goal programming model, as listed in Table 1.1.

The relationship between these goals and the penalisation of deviations from the target level is given in Table 2.1.

Definition 1.7 – 'Deviational Variable': A deviational variable measures the difference between the target level on a criterion and the value that is able to be achieved in a given solution. If the achieved value is above the target level then

the difference is given by the value of the **positive deviational variable**. If the achieved value is below the target level then the difference is given by the value of the **negative deviational variable**.

The essence of goal programming is the minimisation of unwanted deviational variables. For goal type 1 or 'less is better', the positive deviational variable is said to be the unwanted deviational variable For goal type 2 or 'more is better', the negative deviational variable is said to be the unwanted deviational variable. For goal type 3, both positive and negative deviational variables are said to be unwanted deviational variables. Algebraic examples of deviational variables and their minimisation are given in Section 2.1.

Definition 1.8 – 'Constraint': A constraint is a restriction upon the decision variables that must be satisfied in order for the solution to be implementable in practice. This is distinct from the concept of a goal whose non-achievement does not automatically make the solution non-implementable. A constraint is normally a function of several decision variables and can be an equality or an inequality.

Definition 1.9 – 'Sign Restriction': A sign restriction limits a single decision or deviational variable to only take certain values within its range. The most common sign restriction is for the variable to be non-negative and continuous.

Definition 1.10 – 'Feasible Region': The set of solutions in decision space that satisfy all constraints and sign restrictions in a goal programming form the feasible region. Any solution that falls within the feasible region is deemed to be implementable in practice.

Definition 1.11 – 'Pareto-Efficient Solution' (also known as Pareto optimal or in objective space as non-dominated): A solution to a multi-objective problem is **Pareto efficient** if no other feasible solution exists that is at least as good with respect to all objectives and strictly better with respect to at least one objective.

Definition 1.12 – 'Pareto-Inefficient Solution' (also known as Pareto sub-optimal or in objective space as dominated): A solution to a multi-objective problem is **Pareto inefficient** if another feasible solution exists that is at least as good with respect to all objectives and strictly better with respect to at least one objective.

A fundamental law of decision making states that no rational decision maker will knowingly choose a Pareto-inefficient solution, if they have knowledge of a Pareto-efficient solution that dominates it.

Definition 1.13 – 'Ideal Point': The point in objective space at which each objective in a multi-objective optimisation problem takes its optimal value when optimised individually, within the feasible region, is known as the ideal point. If the objectives are conflicting then this point will be outside the feasible region in objective space and hence an infeasible point. Nevertheless, it provides a useful point of reference to measure the goodness of any solution against. In goal programming, the ideal point is used to calculate the normalisation constants when using zero–one

normalisation, as described in Section 3.5.2. An example of an ideal point is given in Fig. 1.2.

Definition 1.14 – 'Anti-ideal Point': To provide a correct scaling, a measure of the worst solution as well as the best for each objective is needed. The anti-ideal point is one possibility for this measure. It is the point in objective space formed from the anti-optimisation (i.e. maximise an objective the decision maker(s) wish to minimise or minimise an objective the decision maker(s) wish to maximise) of each individual objective within the feasible region. In goal programming, the anti-ideal point is often used to calculate the normalisation constants when using zero–one normalisation, as described in Section 3.5.2. An example of an anti-ideal point is given in Fig. 1.2.

Definition 1.15 – 'Nadir Point': The major drawback of the anti-ideal point is that it is calculated from solutions that are not Pareto efficient (e.g. from points A and B in Fig. 1.2). This means that the decision-making axiom known as 'independence from irrelevant alternatives' is broken. A better solution in theory would be to use the nadir point, which is defined as the worst value from amongst the Pareto frontier for each individual objective. The nadir point can be seen from Fig. 1.2 as being calculated from solutions C and D, which are both Pareto efficient. The problem is that there is no computationally efficient means of generating the nadir point exactly, although heuristics for its estimation such as the method of Korhonen et al. (1997) exist.

Definition 1.16 – 'Pareto Frontier' (also known as Pareto front, Pareto set, and in objective space as the non-dominated set): The set of all Pareto-efficient solutions to a decision problem is known as the Pareto frontier. An example of a Pareto frontier is shown in Fig. 1.2.

The relationship between goal programming and Pareto efficiency is the topic of Chapter 6 in this book.

1.2 Underlying Philosophies

In order to utilise goal programme fully, it is important to understand the philosophies and economic concepts that underpin the mathematics. This will ensure that the correct goal programming variants are chosen and the parameters are set appropriately. This section details the major underlying philosophies of goal programming.

1.2.1 Satisficing

Goal programming is often cited as being primarily a satisficing-based technique. The term satisficing and related verb 'to satisfice' were introduced by the American economist Herbert Simon (Simon, 1957). Satisficing comes from the words 'satisfy' and 'suffice'. It describes a behaviour in which decision makers aim to reach a set

of defined goals. If they reach those goals then this suffices for them in that decision situation and they are hence satisfied. This is an alternative to the philosophy of optimising outlined in Section 1.2.2, as Simon's argument is that human beings are more interested and able to reach goals than in the abstract concept of optimising each outcome of the decision problem.

The linkage between satisficing and goal programming is clear. All goal programming models contain a set of goal values to be reached. Meeting these goals as closely as possible is the main aim of the goal programme. Thus satisficing can be rightly thought of as the prime underlying philosophy of goal programming. However, as will be shown in the following sections, it is not necessarily the sole underlying philosophy.

1.2.2 Optimising

To optimise in the context of decision making means to find the decision which gives the best possible value of some measure from amongst the set of possible decisions. The theory of optimising in the presence of multiple objectives is defined by adapting a concept from the work of the Italian economist Vilfredo Pareto into the field of decision making. Pareto (1896) defined a society as being in an optimal state if no individual could become richer without making another individual poorer. This leads to the definition of Pareto optimality in a multi-objective model given by Definitions 1.11 and 1.12 and 6.1–6.8 in Chapter 6.

Goal programming contains elements of optimising. There are three situations to note where the optimising philosophy has particular importance:

(1) If the goal target levels are set very optimistically, such as at their ideal values, in the goal programme, then the dominant underlying philosophy has changed from satisficing to optimising.
(2) If Pareto optimality detection and restoration (as detailed in Chapter 6) take place then the goal programme has a mix of the satisficing and optimising philosophies.
(3) If the goals are two-sided (i.e. a particular value is optimal rather than a 'more is better' or 'less is better' situation) then the satisficing and optimising philosophies can be thought of as coinciding for those goals.

1.2.3 Ordering or Ranking

This underlying philosophy is important in the lexicographic variant (detailed in Section 2.2.1). In this variant, the minimisation of unwanted deviations from goals takes place according to a weak order. In lexicographic goal programme, it is assumed that the weak ranking of the goals in terms of importance exists and is known (or able to be estimated) by the decision maker. Romero (1991) as well as

Table 1.2 Possible ranking of goals for a hospital patient diet formation model

Priority level (rank)	Goal
1	Satisfy nutritional requirements
2	Stay with daily budget
3	Meet patient's food preferences

the discussion in Section 2.2.1 gives practical examples of situations when such a ranking of the goals exists.

A further situation is given by Table 1.2, that of formulation of a diet for a hospital patient following surgery. Here a clear priority ordering exists, with nutritional requirements taking precedent over financial requirements and then patient food preferences. It should also be noted that a natural ordering does not exist in many real-life situations and in these cases the decision maker will be more interested in exploring trade-offs or the balance between the goals. In these cases lexicographic goal programming should not be used and instead another goal programming variant should be chosen. Romero et al. (1998) demonstrates that there is no fundamental incompatibility between the ordering philosophy and the other philosophies utilised in goal programming.

1.2.4 Balancing

In many goal programming problems it is insufficient to consider solely the average level of achievement of the goals without looking at the balance between the achievement of the goals. In order to illustrate this fact, Table 1.3 shows two solutions to a simple two-goal programme where negative deviations from the target level are unwanted. The two solutions have the same average level of achievement but different levels of balance between goals.

Table 1.3 Example of two-goal model

	Goal 1 target	Goal 1 achieved	Goal 2 target	Goal 2 achieved
Solution 1	**100**	90	**100**	90
Solution 2	**100**	100	**100**	80

Clearly, if balancing the achievement of the goals is of any importance to the decision maker then they will prefer solution 2 over solution 1. However, if only the average level of achievement is considered when solving the goal programme then solutions 1 and 2 are shown to have the same value to the decision maker. Omitting the balancing element from a goal programme when it should be included is, in our view, one of the major reasons why unbalanced and hence unattractive and unimplementable solutions are sometimes incorrectly produced by poor goal

programming modelling. This phenomenon and how to avoid it by utilising good goal programming practice is discussed in Chapters 3 and 4.

The balancing philosophy is linked to the Rawlsian utility function after the principles of social justice introduced by Rawls (1973). This utility function utilises the L_∞ distance metric as defined by setting $\rho = \infty$ in the general distance metric equation given in Section 7.2.1. The L_∞ distance metric is sometimes known as the Minmax or Chebyshev distance metric. In the context of goal programming, it minimises the worst deviation from the set of goals and hence attempts to balance the level of achievement of set of goals. This forms the basis of the Chebyshev goal programming variant as detailed in Section 2.2.3. It also appears as part of the extended goal programming and meta-goal programming frameworks discussed in Chapter 4. The balance between minimising the average deviation and minimising the worst deviation from amongst the set of goals is sometimes referred to as the efficiency–equity trade-off. Advanced goal programming variants such as extended goal programming are able to effectively explore this trade-off.

Chapter 2
Goal Programming Variants

This chapter introduces the major goal programming variants. The purpose and underlying philosophy of each variant are given. The three major variants in terms of underlying distance metric (and hence philosophy) used are introduced first. These are lexicographic, weighted, and Chebyshev goal programming. Then the variants in terms of the mathematical nature of the decision variables and/or goals used are introduced. These are fuzzy, integer, binary, and fractional goal programming.

2.1 Generic Goal Programme

Algebraically speaking, we allow our generic goal programme to have Q goals, which we give the index $q = 1, ..., Q$. We also define n decision variables which we shall term $\underline{x} = x_1, x_2, ..., x_n$. These are the factors over which the decision maker(s) have control and define the decision to be made. Each goal has an achieved value, $f_q(\underline{x})$, on its underlying criterion. This is a function of the decision variables. Note that in this generic form no assumptions have yet been made about the nature of the decision variables of goals. The decision maker(s) sets a numeric target level for each goal which is denoted b_q. This then leads to the basic formulation of the qth goal:

$$f_q(\underline{x}) + n_q - p_q = b_q$$

where n_q is the **negative deviational variable** of the $_q$th goal. It represents the level by which the target level is under-achieved. For example if $b_q = 40$ and $f_q(\underline{x}) = 25$ then $n_q = 15$ p_q is the **positive deviational variable** of the qth goal. It represents the level by which the target level is over-achieved. For example if $b_q = 40$ and $f_q(\underline{x})$ then $p_q = 30$. The two deviational variables are constrained to take non-negative values and both cannot take a non-zero value simultaneously.

D. Jones, M. Tamiz, *Practical Goal Programming*, International Series in Operations Research & Management Science 141, DOI 10.1007/978-1-4419-5771-9_2,

The decision maker must then decide which deviational variables are unwanted. These are penalised in an achievement function. There are three basic types of penalisation, as discussed in Section 1.1. The relation to the deviational variables is given in Table 2.1.

Table 2.1 Algebraic significance of goal types

Type	Variables to be penalised
1	p_q
2	n_q
3	$n_q + p_q$

A typical type 1 goal would involve cost, where any positive deviation above the goal level would be penalised. A typical type 2 goal would involve profit, where any negative deviation below the goal level would be penalised. A typical type 3 goal would involve a workforce-level target, where any negative or positive deviation from the target level would be penalised.

We note that the set of goals are sometimes termed soft constraints. That is, the decision maker desires to meet each goal but if the goal is not achieved then this does not imply that the solution is infeasible. Goal programming also allows for an addition of a set of linear programming style hard constraints whose violation will make the solution infeasible. These are modelled by adding the condition

$$\underline{x} \in F$$

where F is the feasible region made up of points in decision space that satisfy all of the constraints and sign restrictions.

Finally, the unwanted deviational variables need to be brought together in the form of an **achievement function** whose purpose is to minimise them and thus ensure that a solution that is 'as close as possible' to the set of desired goals is found. The exact nature of the achievement is dependent on the goal programming variant used, so in our generic form it is simply represented by a generic function of the deviational variables: Min $a = h(\underline{n}, \underline{p})$ where $\underline{n}$ is the vector of q negative deviational variables and $\underline{p}$ is the vector of q positive deviational variables. Although this function is termed an achievement function in reality it measures the 'lack' of achievement of the goals. That is the distance from the target to the achieved level of the goals.

The above considerations lead to the generic algebraic form of the goal programme:

$$\text{Min a} = \text{h}(\underline{\text{n}}, \underline{\text{p}})$$

subject to

$$f_q(\underline{x}) + n_q - p_q = b_q \qquad q = 1, ..., Q$$
$$\underline{x} \in F$$
$$n_q, p_q \geq 0 \qquad q = 1, ..., Q$$

Note that the condition $n_q \times p_q = 0$ is also required to hold but this will automatically be satisfied due to the nature of the goal programming minimisation.

2.2 Distance Metric Based Variants

2.2.1 Lexicographic Goal Programming

The vast majority of the early goal programming formulations (e.g. Lee, 1972) used the lexicographic goal programming variant. This is also sometimes termed 'pre-emptive' goal programming in the literature. The distinguishing feature of lexicographic goal programming is the existence of a number of **priority levels**. Each priority level contains a number of unwanted deviations to be minimised. A practical example of how to formulate a lexicographic goal programme is given in Sections 3.1 and 3.2.

To formulate a generic lexicographic goal programme algebraically we define the number of priority levels as L with corresponding index $l = 1, ..., L$. Each priority level is now a function of a subset of unwanted deviational variables which we define as $h_l(\underline{n}\ \underline{p})$. This leads to the following formulation:

$$Lex\ \text{Min}\, a = \left[h_1(\underline{n}, \underline{p}),\ h_2(\underline{n}, \underline{p}), ..., h_L(\underline{n}, \underline{p})\right]$$

subject to

$$f_q(\underline{x}) + n_q - p_q = b_q \qquad q = 1, \ldots, Q$$
$$\underline{x} \in F$$
$$n_q, p_q \geq 0 \qquad q = 1, \ldots, Q$$

where each $h_l(\underline{n}, \underline{p})$ contains a number of unwanted deviational variables. The exact nature of $h_l(\underline{n}, \underline{p})$ depends on the nature of the goal programme to be formulated, but if we assume that it is linear and separable then it will assume the form

$$h_1(\underline{n}, \underline{p}) = \sum_{q=1}^{Q} \left(\frac{u_q^l n_q}{k_q} + \frac{v_q^l n_q}{k_q} \right)$$

where u_q^l is the preferential weight associated with the minimisation of n_q in the lth priority level. v_q^l is the preferential weight associated with the minimisation of p_q in the lth priority level. The preferential weights are used to model the relative importance of the minimisation of the associated deviational variable to the decision maker. The setting and control of the preferential weights is discussed in Section 3.6. Note that if a deviational variable is not included in a particular priority level then its preferential weight for that priority level is set equal to zero. Deviational variables whose minimisation is considered unimportant to the decision maker(s) (e.g. positive deviation from a profit goal) are assigned a preferential weight of zero in every priority level. k_q is the normalisation constant associated with the qth goal. These constants are necessary in order to scale all the goals onto the same units of measurement. A discussion of types of normalisation used in goal programming is given in Section 3.5.

The meaning of the lexicographic minimisation of the achievement function is that the minimisation of deviational variables placed in a higher priority level is regarded as infinitely more important than that of deviational variables placed in a lower priority level. This has the effect of producing a series of sequential optimisations, each of which has a reduced feasible region as the minimal values of the higher priority level optimisations must be maintained. A practical numerical example of this process is given in Section 3.3. This also has the effect of combining the ordering philosophy discussed in Section 1.2.3 with the underlying satisficing philosophy of goal programming. The lexicographic structure has been criticised by some authors on the grounds of its incompatibility with utility function theory (Min and Storbeck, 1991). However, Romero (1991) points out its practicality in modelling situations where the decision maker has a pre-defined ordering of the goals in mind and does not wish to make direct 'trade-off' comparisons between goals. This could be due to political sensitivities or ethical considerations, such as trade-offs between safety goals (measured in lives lost) and monetary goals (such as cost or profit). Furthermore, Tamiz et al. (1998) explore the relationship between lexicographic ordering and utility functions and conclude that the non-compensatory lexicographic model can be appropriate when modelling some real-life decisions. Therefore, whilst it is true that lexicographic goal programming will not be appropriate for every multi-objective situation, it can be seen that there is a class of situations in which it proves to be an effective and appropriate decision aiding tool.

It should also be noted that some authors do not include the constraint set $\underline{x} \in F$ in the generic formulation of a lexicographic goal programming. This is because each hard constraint can be converted into a goal and the first priority level can be reserved for the minimisation of the deviational variables from the hard constraints. This is a mathematically correct transformation; however, it is not the convention adopted in this book as we wish to emphasise and maintain the difference between hard constraints (whose non-satisfaction will render the solution infeasible) and goals (whose non-satisfaction is undesirable but will not render the solution infeasible).

2.2.2 Weighted Goal Programming

The weighted goal programme variant allows for direct trade-offs between all unwanted deviational variables by placing them in a weighted, normalised single achievement function. Weighted goal programming is sometimes termed non-pre-emptive goal programming in the literature. If we assume linearity of the achievement function then we can represent the linear weighted goal programme by the following formulation:

$$\text{Min a} = \sum_{q=1}^{\text{Q}} \left(\frac{\text{u}_q \text{n}_q}{\text{k}_q} + \frac{\text{v}_q \text{n}_q}{\text{k}_q} \right)$$

subject to

$$f_q(\underline{x}) + n_q - p_q = b_q \qquad q = 1, \ldots, Q$$
$$\underline{x} \in F$$
$$n_q, p_q \geq 0 \qquad q = 1, \ldots, Q$$

with the variable definitions identical to those introduced for the lexicographic goal programming variant, except that the preference weights u_q and v_q are no longer indexed by priority level.

As reported in Tamiz et al. (1995) the split between articles using lexicographic and weighed goal programming in pre-1990 application papers is 75% lexicographic and 25% weighted. During the period 1990–2000 Jones and Tamiz (2002) report this split as 59% lexicographic and 41% weighted. A similar survey for the period 2000–2010 is not yet available but a clear trend from using lexicographic towards using weighted goal programming can be seen. The reasons for this are due to the greater flexibility afforded by the weighted variant and the desire of decision makers to do more 'trade-off' analysis and direct comparison between goals. Increases in computing power and hence decreased solution times may have contributed to this effect as multiple weighting schemes and trade-offs can now be more easily compared even for larger scale goal programmes.

2.2.3 Chebyshev Goal Programming

The third major variant of goal programming was introduced by Flavell (1976). It is known as Chebyshev goal programming, because it uses the underlying Chebyshev (L_∞) means of measuring distance. That is, the maximal deviation from any goal, as opposed to the sum of all deviations, is minimised. For this reason Chebyshev goal programming is sometimes termed Minmax goal programming. As discussed in Section 1.2.4, the underlying philosophy when using the L_∞ distance metric is that

of balance. That is, the decision maker is trying to achieve a good balance between the achievement of the set of goals as opposed to the lexicographic approach which deliberately prioritises some goals over others or the weighted approach which chooses the set of decision variable values which together make the achievement function lowest, sometimes at the expense of a very poor value in one or two of the goals. Chebyshev goal programming is also the only major variant that can find optimal solutions for linear models that are not located at extreme points in decision space (such as point D in Fig. 3.4).

All of the above points lead to the conclusion that Chebyshev goal programming has the potential to give the most appropriate solution where a balance between the levels of satisfaction of the goals is needed. This should include a large number of application areas, especially those with multiple decision makers each of whom has a preference to their own subset of goals that they regard as most important. However, surveys of the literature (e.g. Jones and Tamiz, 2002) find little practical use of the Chebyshev goal programming variant. It is hoped that greater awareness and the development of methodologies such as extended goal programming (outlined in Section 4.3) which encompasses Chebyshev goal programming will lead to a greater use of this variant in practice.

If we let λ be the maximal deviation from amongst the set of goals then the Chebyshev goal programming has the following algebraic format:

$$\text{Min a} = \lambda$$

subject to

$$\begin{aligned}
&f_q(\underline{x}) + n_q - p_q = b_q \quad q = 1,...,Q \\
&\tfrac{u_q n_q}{k_q} + \tfrac{v_q p_q}{k_q} \leq \lambda \quad q = 1,...,Q \\
&\underline{x} \in F \\
&n_q, p_q \geq 0 \quad q = 1,...,Q
\end{aligned}$$

Provided all objective functions $f_q(\underline{x})$ are linear, this shares the advantage along with the other two major variants of being solvable by a linear programming solution package (as described in Chapter 5).

2.3 Decision Variable and Goal-Based Variants

There is a fundamental difference between the variants presented in this section and those presented in Section 2.2. In the previous section the factor that varied was the underlying distance metric. In this section the factor that varies is the mathematical nature of the goals and/or decision variables. Therefore, it is possible to formulate a goal programme that has a variant from both Sections 2.2 and 2.3. For example,

an integer weighted goal programme (see part (3) of Example 5 of Section 3.9 – Exercises) or a lexicographic goal programme with fuzzy priority levels (Akoz and Petrovic, 2007) can be formulated.

2.3.1 Fuzzy Goal Programming

Fuzzy goal programming utilises fuzzy set theory (Zadeh, 1965) to deal with a level of imprecision in the goal programming model. This imprecision normally relates to the goal target values (b_q) but could also relate to other aspects of the goal programme such as the priority structure. The early fuzzy goal programming models used both Chebyshev (Narasimhan, 1980; Hannan, 1981) and weighted (Hannan, 1981; Tiwari et al., 1987) distance metrics.

There are various possibilities for measuring the fuzziness around the target goals, each of which leads to a different fuzzy membership function. These functions model the drop in dissatisfaction from a state of total satisfaction (where the membership function takes the value 1) to a state of total dissatisfaction (where the membership function takes the value 0). There are many possible fuzzy membership functions, the algebraic structure of four of the most common linear fuzzy membership functions are outlined below:

1. Right-sided (positive deviations penalised) linear function – shown graphically by Fig. 2.1:

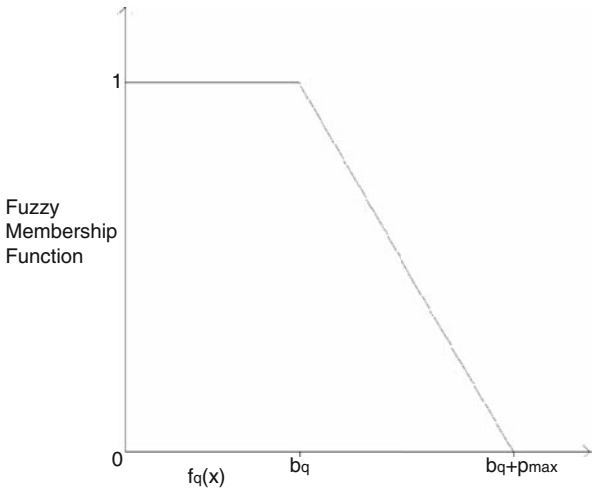

Fig. 2.1 Right-sided fuzzy membership function

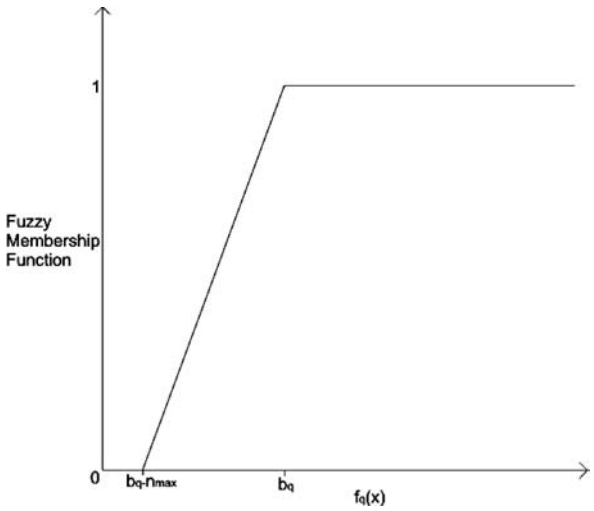

Fig. 2.2 Left-sided fuzzy membership function

$$\mu\left[f_q(x)\right] = \begin{cases} 1 & f_q(x) \le b_q \\ 1 - \frac{f_q(x) - b_q}{p_{\max}} & b_q \le f_q(x) \le b_q + p_{\max} \\ 0 & f_q(x) \ge b_q + p_{\max} \end{cases}$$

2. Left-sided (negative deviations penalised) linear function – shown graphically by Fig. 2.2:

$$\mu\left[f_q(x)\right] = \begin{cases} 1 & f_q(x) \ge b_q \\ 1 - \frac{b_q - f_q(x)}{n_{\max}} & b_q - n_{\max} \le f_q(x) \le b_q \\ 0 & f_q(x) \le b_q - n_{\max} \end{cases}$$

3. Triangular (both deviations penalised) linear function – shown graphically by Fig. 2.3:

$$\mu\left[f_q(x)\right] = \begin{cases} 0 & f_q(x) \le b_q - n_{\max} \text{ or } f_q(x) \ge b_q + p_{\max} \\ 1 - \frac{b_q - f_q(x)}{n_{\max}} & b_q - n_{\max} \le f_q(x) \le b_q \\ 1 - \frac{f_q(x) - b_q}{p_{\max}} & b_q \le f_q(x) \le b_q + p_{\max} \end{cases}$$

4. Trapezoidal (both deviations penalised with an interval of complete satisfaction) linear function – shown graphically by Fig. 2.4:

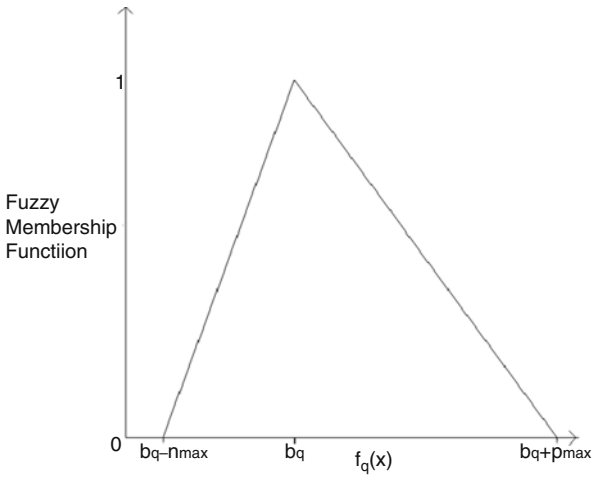

Fig. 2.3 Triangular fuzzy membership function

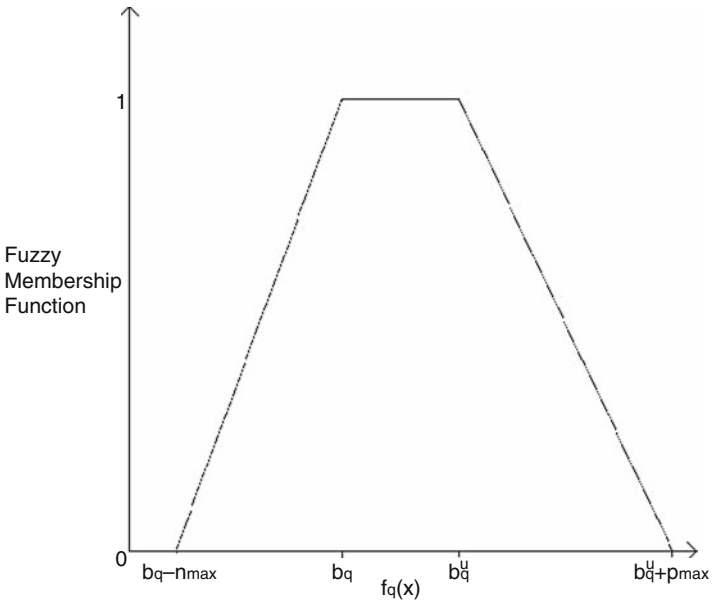

Fig. 2.4 Trapezoidal fuzzy membership function

$$\mu\left[f_q(x)\right] = \begin{cases} 0 & f_q(x) \le b_q^l - n_{\max} \text{ or } f_q(x) \ge b_q + p_{\max} \\ 1 - \frac{b_q^l - f_q(x)}{n_{\max}} & b_q^l - n_{\max} \le f_q(x) \le b_q^l \\ 1 & b_q \le f_q(x) \le b_q^u \\ 1 - \frac{f_q(x) - b_q^u}{p_{\max}} & b_q \le f_q(x) \le b_q^u + p_{\max} \end{cases}$$

where $\mu[f_q\ (x)]$ represents the fuzzy membership function associated with the qth goal. $p_{\max}$ represents the level of positive deviation beyond the goal at which total dissatisfaction occurs. $n_{\max}$ represents the level of negative deviation beyond the goal at which total dissatisfaction occurs. b_q^l and b_q^u give the lower and upper bounds of the interval of total satisfaction for the trapezoidal membership function.

If we assume a weighted goal programme variant then the most recent and comprehensive modelling framework is given by Yaghoobi et al. (2008). Assuming that the Q goals are divided into q_1 right-sided membership functions, q_2 left-sided membership functions, q_3 triangular membership functions, and q_4 trapezoidal membership functions gives the following algebraic formulation:

$$\text{Min a} = \sum_{q=1}^{q_1} \frac{\text{v}_q \text{p}_q}{\text{p}_{\max}} + \sum_{q=q_1+1}^{q_1+q_2} \frac{\text{u}_q \text{n}_q}{\text{n}_{\max}} + \sum_{q=q_1+q_2+1}^{\text{Q}} \left(\frac{\text{u}_q \text{n}_q}{\text{n}_{\max}} + \frac{\text{v}_q \text{p}_q}{\text{p}_{\max}} \right)$$

subject to

$$\begin{aligned}
& f_q(x) - p_q \le b_q && q = 1,...,q_1 \\
& \mu_q + \tfrac{p_q}{p_{\max}} = 1 && q = 1,...,q_1 \\
& f_q(x) + n_q \ge b_q && q = q_1 + 1,...,q_1 + q_2 \\
& \mu_q + \tfrac{n_q}{n_{\max}} = 1 && q = q_1 + 1,...,q_1 + q_2 \\
& f_q(\underline{x}) + n_q - p_q = b_q && q = q_1 + q_2 + 1,...,q_1 + q_2 + q_3 \\
& \mu_q + \tfrac{n_q}{n_{\max}} + \tfrac{p_q}{p_{\max}} = 1 && q = q_1 + q_2 + 1,...,Q \\
& f_q(\underline{x}) - p_q \le b_q && q = q_1 + q_2 + q_3 + 1,...,Q \\
& f_q(\underline{x}) + n_q \le b_q && q = q_1 + q_2 + q_3 + 1,...,Q \\
& \underline{x} \in F \\
& n_q, p_q, \mu_q \ge 0 && q = 1,...,Q
\end{aligned}$$

where μ_q represents the achieved level of the fuzzy membership function for the qth goal. This formulation allows the solution of a weighted fuzzy goal programme via a single, linear model that can be solved via any standard linear programming software.

2.3.2 Integer and Binary Goal Programming

An integer goal programme has one or more decision variables that are restricted to take only a discrete (countable) number of values within their defined ranges. From a mathematical programming perspective this violates the 'divisibility assumption'

as defined in Chapter 4. This means that techniques from the field of integer programming rather than linear programming must be adapted when designing solution and Pareto efficiency detection and restoration tools detailed in Chapter 6. As integer programmes are a lot harder to solve than similarly sized linear programmes, this will result in longer solution times and more restrictions on the number of decision variables and constraints in the goal programme. A practical example of the issues raised in solving large-scale integer goal programmes is given in Chapter 8 in the context of a healthcare planning model. A discussion of the technical aspects of modelling and solving integer programmes is beyond the scope of this book but the reader is referred to the following seminal books on the topic: Williams (1993); Wolsey (1998).

As previously noted, an integer goal programming can be of either the lexicographic, weighted, or Chebyshev distance metric variants. If we assume the weighted variant and restrict all decision variables to take only positive integer variables then we have the following algebraic structure:

$$\text{Min a} = \sum_{q=1}^{Q} \left(\frac{u_q n_q}{k_q} + \frac{v_q p_q}{k_q} \right)$$

subject to

$$\begin{aligned} & f_q(\underline{x}) + n_q - p_q = b_q \qquad q = 1,...,Q \\ & \underline{x} \in F \quad \text{including} \quad x_i \geq 0 \ \text{and integer i} = 1,...,\text{n} \\ & n_q, p_q \geq 0 \qquad q = 1,...,Q \end{aligned}$$

If all of the integer variables are restricted to take one of two values (most commonly zero or one) then the resulting goal programme is termed a binary goal programme. This subset of integer goal programmes has special characteristics that may aid the solution process. The algebraic representation of a weighted binary goal programme is

$$\text{Min a} = \sum_{q=1}^{Q} \left(\frac{u_q n_q}{k_q} + \frac{v_q p_q}{k_q} \right)$$

subject to

$$\begin{aligned} & f_q(\underline{x}) + n_q - p_q = b_q \qquad q = 1, ..., Q \\ & \underline{x} \in F \quad \text{including} \quad x_i = 0 \ \textit{or} \ 1 \ \text{i} = 1, ..., \text{n} \\ & n_q, p_q \geq 0 \qquad q = 1, ..., Q \end{aligned}$$

Integer and binary goal programmes are of particular use when formulating many practical problems that have both logical conditions and multiple, conflicting goals. Typical application areas include multi-objective shortest path, assignment,

logistics, network flow, spanning tree, travelling salesperson, knapsack, scheduling, location, and set covering problems (Ehrgott and Gandibleux, 2002).

2.3.3 Fractional Goal Programming

A fractional goal programme has one or more goals of the form

$$\frac{f_q(\underline{x})}{g_q(\underline{x})} + n_q - p_q = b_q$$

where $g_q(\underline{x})$ is a generic function of the decision variables. This type of goal programme is mentioned by Romero (1991) as arising in the fields of financial planning, production planning, and engineering. This variant also occurs in some goal programming based methods for deriving weighting vectors from pairwise comparison matrices (Despotis, 1996). Romero also warns on the perils of simple linearisation to the following form:

$$f_q(\underline{x}) + n_q - p_q = b_q g_q(\underline{x})$$

This is not a valid mathematical transformation as the deviational variables should have been multiplied by the function $g_q(\underline{x})$ as well, giving

$$f_q(\underline{x}) + n_q g_q(\underline{x}) - p_q g_q(\underline{x}) = b_q g_q(\underline{x})$$

Hence the new terms $n_q g_q(\underline{x})$ and $p_q g_q(\underline{x})$ mean that linearisation has not been achieved.

The advent of more powerful non-linear programming technology and computing power; heuristics such as multi-objective evolutionary methods (Deb, 2001); and advances in exact methods for solving fractional goal programmes (Audet et al., 2004) have made the need for linearisation less urgent. In many cases the model can now be solved without linearisation. Fractional goal programmes also require their own adapted form of Pareto efficiency detection and restoration procedure (Caballero and Hernandez, 2006), as will be discussed in Chapter 6.

Chapter 3
Formulating Goal Programmes

This chapter concerns methods and concepts that need to be considered in order to formulate effective goal programming models. There are many issues unique to goal programming that even those who are experienced in building other types of operational research and/or mathematical programming models need to be aware of. This chapter will address the modelling issues sequentially and show how to employ good goal programming practice and avoid common pitfalls.

3.1 Formulating Goals and Setting Target Levels

Before arriving at this stage it is necessary to have employed good operational research practice to such issues as discovering the operation of the system, data collection and analysis, and liaison with the problem owner(s) and stakeholder groups in order to extract the criteria of relevance to the problem. For a good description of the holistic process of operational research modelling the reader is referred to the first chapter of the Winston (2004) textbook. Advice on liaising with decision maker(s) in order to extract criteria of relevance is given in Belton and Stewart (2002). For the remainder of this chapter, it is assumed for the purposes of explanation that all technical coefficient data is deterministic and known with certainty.

When a set of criteria has been determined it is necessary to distinguish between goals and hard constraints. Hard constraints are in variable space and, as described in Chapter 2, are conditions that must be satisfied in order for the solution to be implementable. Any condition that does not fulfil this requirement should be included as a goal and not a hard constraint. The modeller is cautioned against using more hard constraints than is necessary as this could cause infeasibility and also exclude solutions that may be of practical interest to the decision maker.

Once the set of goals has been determined then it is necessary to both form the algebraic structure of the goal and set its target level.

D. Jones, M. Tamiz, *Practical Goal Programming*, International Series in Operations Research & Management Science 141, DOI 10.1007/978-1-4419-5771-9_3,

3.1.1 Example

Suppose the weekly levels of two products to be manufactured are to be determined. Suppose our first criterion relates to the amount of hours labour per week used to manufacture the two products A and B. Assume our decision variables are

x_1: The number of products of type A manufactured per week

x_1: The number of products of type B manufactured per week

and each type A product takes 4 h to manufacture and each type B product takes 3 h to manufacture.

Our function for the amount of labour used is given by

$$f_1(x) = 4x_1 + 3x_2$$

Now in order to complete the goal formulation the two deviational variables must be added and a goal level set. Suppose that the desired level is to use no more than 120 h labour. This gives the first goal

$$4x_1 + 3x_1 + n_1 - \underline{p_1} = 120$$

The variable p_1 is underlined here as it is the unwanted deviation away from 120 h. This is an example of a type 1 goal. If we imagine that 100 h of labour is used in practice then the deviational variables will take the values $n_1 = 20$, $p_1 = 0$, showing that the solution is 20 units under the target. If, on the other hand, 150 h of labour is used then the deviational variables will take the values $n_1 = 0$, $p_1 = 30$, showing that 30 h more than the desired level of 120 h is used.

It can be seen that at most one deviational variable should take a positive value if this solution is to make sense, with both deviational variables taking the value zero if the goal target value of 120 h is achieved exactly. This is equivalent to imposing the condition $n_1 \times p_1 = 0$. Fortunately, this non-linear condition need not be modelled explicitly in the standard goal programming model due to the nature of the achievement function not making it profitable to allow both deviational variables to take positive values. Also, both deviational variables are constrained to take non-negative values:

$$n_1, p_1 \geq 0$$

Note that both deviational variables are included in the formulation. This is good practice, even if it is suspected that solutions below the 120 h limit will never occur, as if these are by chance possible and the variable n_1 has been omitted, then this will act as a hard constraint, forcing the solution to remain above 120 h and hence excluding some potentially good solutions, and even in some cases the true optimal solution. Therefore, **deviational variables should not be omitted from the formulation**. The one exception to this rule is where the ideal (best) value of the objective has been used as a target level, as detailed in the discussion in the following paragraph.

When setting the actual goal level, several factors must be considered. The first is that: are different degrees of certainty associated with the confidence with which the value can be set? At one extreme, the target level will be determined beforehand with certainty due to some technical requirement, natural occurrence, or policy of the problem-owning organisation. For instance, if a university has a strategy of recruiting 20% of its student intake from a certain social economic group then that is a goal that has a certain target level already established. A goal for the level of financial surplus desired at the same university could be a lot more difficult to set. It is important to note that the initial target level chosen in such a case should be regarded as **an initial estimate and not a definitive set-in-stone value**. Sensitivity analysis techniques, an interactive methodology, or a preference structure should be employed in order to explore the effects of different levels of the target level. These techniques will be outlined in Chapter 4.

There are also modelling and solution reasons why the target levels have to be set at appropriate levels, especially for type 1 or type 2 goals. If they are set too pessimistically then the resulting solution may be Pareto inefficient (Romero, 1991), and a Pareto efficiency restoration technique (Tamiz and Jones, 1996) will need to be employed in order to restore Pareto efficiency to the solution. Such techniques are detailed in Chapter 6. If on the other hand they are set too optimistically, especially when using the lexicographic variant, then the problem of lexicographic redundancy will occur (Romero, 1991), meaning that only a few (and in many cases in the literature, only one) of the goals will be taken into consideration. This will not give a true multi-objective analysis of the situation. Lexicographic redundancy is dealt with in Section 3.3.1.

One means of avoiding decision maker setting target levels as seen in the literature is to set the target level at its maximum possible value within the feasible set, known as the ideal value. This value can be relatively easily found by the following single-objective optimisation (assuming without loss of generality a first goal of type 2):

$$\text{Max } f_1(x)$$

subject to

$$x \in F$$

This assumes that this problem is not unbounded, which in most real-life situations is a fairly safe assumption. This is the ultimate in optimistic target-level setting and for reasons outlined above therefore **should not be used with the lexicographic variant**. With the Chebyshev variant ideal values could also cause scaling problems as this variant is very sensitive to the distance from target to ideals for certain goals and therefore stretching that distance by setting the target level to the ideal value could lead to an unintended bias. Ideal target levels have the potential to work for the weighted variant but it has been noted (Romero et al., 1998) that to do this for all goals represents a move away from the satisficing philosophy that underlies goal

programming and towards an optimising philosophy. In this case it has been shown (Romero et al., 1998) that what in fact is being used is a limited version of the compromise programming (Yu, 1973) technique that just considers a single distance metric. If this is the desired philosophy, it is suggested that the modeller move from using goal programming to a full use of compromise programming which allows for a complete analysis of such models.

In short, setting target levels to the ideal is neither in accordance with the philosophy of goal programming nor a panacea for the problem of setting target levels. It can also cause real problems for some variants. In our opinion, a better solution would be increased interaction with the decision maker, sensitivity analysis, interactive methods, or preference modelling inclusion in order to set realistic target levels.

3.1.2 Resumption of Example

Now suppose the other criteria in our example have been identified. These relate to the levels of desired profit and strategic production plans. The profit per item A is \$100 and for product B is \$150 per item. The company is somewhat unsure about the level of profit to set as a target but give an initial estimate of \$7000 per week. This leads to the following algebraic formulation of the second goal:

$$100x_1 + 150x_2 + \underline{n_2} - p_2 = 7000$$

This is an example of a type 2 goal as any negative deviation away from \$7000 is not desired but any positive deviation from \$7000 should certainly not be penalised.

Additionally, the company has some strategic aims for their weekly production. They wish to maintain production of at least 40 units of each of the products. This leads to the following two type 2 goals:

$$x_1 + \underline{n_3} - p_3 = 40$$

$$x_2 + \underline{n_4} - p_4 = 40$$

Also, the company has two hard constraints. The first relates to a minimum agreement for a type of material used in the manufacture of the products. The company has to purchase a minimum of 50 l of this product weekly. Each item of type A uses 2 l of the product and each item of type B uses 1 l of the product. Disposing of unused product is prohibitively expensive. The second constraint relates to machine time, which caps the maximum combined production of both products to 75 per week.

These considerations lead to the following two constraints:

$$2x_1 + x_2 \geq 50$$

$$x_1 + x_2 \leq 75$$

Putting all our goals and constraints together and adding sign restrictions to prohibit negative production leads to the following 'subject to' section of the goal programme:

$$4x_1 + 3x_2 + n_1 - \underline{p_1} = 120$$
$$100x_1 + 150x_2 + \underline{n_2} - p_2 = 7000$$
$$x_1 + \underline{n_3} - p_3 = 40$$
$$x_2 + \underline{n_4} - p_4 = 40$$
$$2x_1 + x_2 \geq 50$$
$$x_1 + x_2 \leq 75$$
$$x_1, x_2 \geq 0,\ n_q, p_q \geq 0\ \text{q} = 1,....,4$$

As this problem has just two decision variables, this formulation can also be expressed graphically as shown by Fig. 3.1.

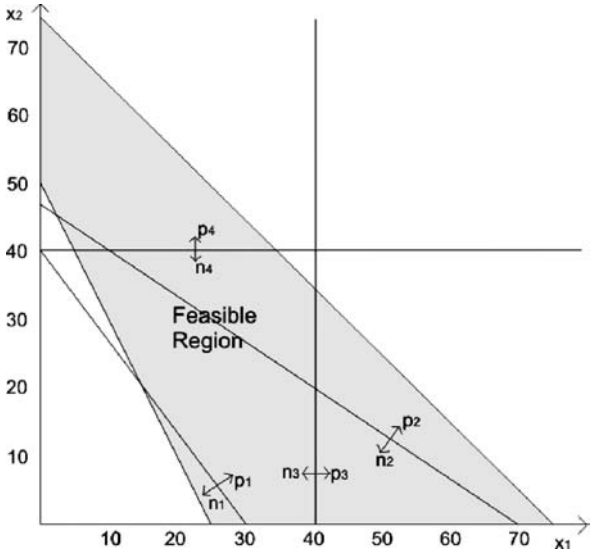

Fig. 3.1 Graphical representation of example

A quick inspection of Fig. 3.1 or of the goals and constraints shows that this set is conflicting. There is no feasible solution that simultaneously satisfies all the goals at their target values. Some of the conflicts can be seen immediately. For example if the company has a strategic aim to produce 40 of each of the two products but only machine time for 75 products then the strategic aim will never be met. Other conflicts are less obvious. For example Fig. 3.1 shows that no solution exists that simultaneously meets the target levels for labour and for profit. A level of conflict between goals is to be expected in any modern organisation due to the laudable aim to achieve good levels on all criteria.

3.2 Variant Choice

After the formulation of the goals, constraints, and sign restrictions the achievement function must now be formulated. As detailed in Chapter 2, the nature of the achievement function will be dependent on the variant or variants chosen. The following section will demonstrate how the above example can be formulated as a lexicographic, weighted, and Chebyshev goal programme and discuss good modelling practice within these variants.

3.3 Lexicographic Variant

Suppose that the company has a clear order in which they wish to see the goals satisfied. An example of such an order could be

Priority 1: Achieve the profit goal.
Priority 2: Achieve the strategic production goals.
Priority 3: Achieve the labour goal.

The algebraic formulation is given as a vector to be lexicographically minimised:

$$\text{Min}\, a = \left[(n_2),(n_3 + n_4),(p_1)\right].$$

The significance of this vector is that the satisfaction of goals placed in a higher priority level is strictly preferred to that of goals placed in lower priority levels. Thus the first step is to minimise the first priority level:

$$\text{Min}\, z = n_2$$

subject to

$$\begin{aligned}
&4x_1 + 3x_2 + n_1 - \underline{p_1} = 120\\
&100x_1 + 150x_2 + \underline{n_2} - p_2 = 7000\\
&x_1 + \underline{n_3} - p_3 = 40\\
&x_2 + \underline{n_4} - p_4 = 40\\
&2x_1 + x_2 \geq 50\\
&x_1 + x_2 \leq 75\\
&x_1, x_2 \geq 0,\ n_q, p_q \geq 0\ \text{q} = 1,....,4
\end{aligned}$$

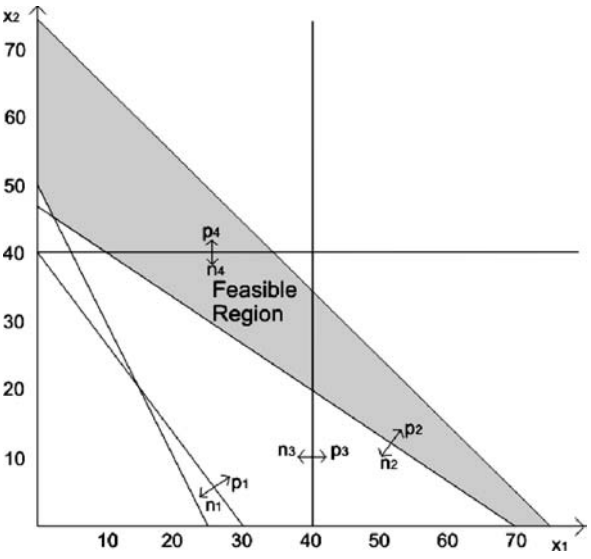

Fig. 3.2 Feasible region for second priority level

The optimal value is $n_2 = 0$. Figure 3.2 shows the set of feasible alternative optimal solutions to this optimisation. This set will form the feasible region for the optimisation of the second priority level.

The second priority level is now optimised. This optimisation has all the goals and constraints of the previous linear programme plus the additional constraint $n_2 = 0$.

Chapter 5 on goal programming solution will detail Ignizio's efficient algorithms (Ignizio and Cavalier, 1994) for performing this series of optimisations. The minimisation is that of the second priority level:

$$\text{Min } z = n_3 + n_4$$

subject to

$$\begin{aligned}
&4x_1 + 3x_2 + n_1 - \underline{p_1} = 120\\
&100x_1 + 150x_2 + \underline{n_2} - p_2 = 7000\\
&x_1 + \underline{n_3} - p_3 = 40\\
&x_2 + \underline{n_4} - p_4 = 40\\
&2x_1 + x_2 \geq 50\\
&x_1 + x_2 \leq 75\\
&n_2 = 0\\
&x_1, x_2 \geq 0,\ n_q, p_q \geq 0 \text{ q} = 1,....,4
\end{aligned}$$

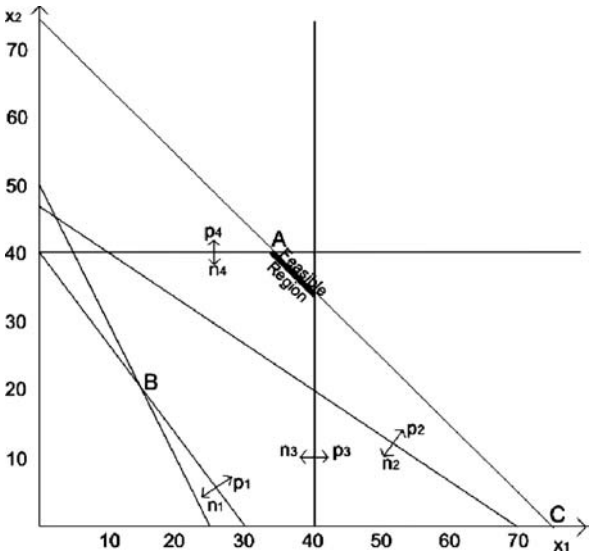

Fig. 3.3 Feasible region for third priority level and lexicographic solution points

The goals in the second priority level cannot be completely satisfied. The minimum value is $z = 5$. The feasible alternative optimal solutions to this optimisation are shown by Fig. 3.3. These form the feasible region for the minimisation of the third priority level.

$$\text{Min}\, z = p_1$$

subject to

$$4x_1 + 3x_2 + n_1 - \underline{p_1} = 120$$
$$100x_1 + 150x_2 + \underline{n_2} - p_2 = 7000$$
$$x_1 + \underline{n_3} - p_3 = 40$$
$$x_2 + \underline{n_4} - p_4 = 40$$
$$2x_1 + x_2 \geq 50$$
$$x_1 + x_2 \leq 75$$
$$n_2 = 0$$
$$n_3 + n_4 = 5$$
$$x_1, x_2 \geq 0,\ n_q, p_q \geq 0 \text{ q} = 1,....,4$$

Table 3.1 Solution of example using lexicographic variant

Goal number	Description	Target level	Sat	Achieved value
1	Labour	120	No	260
2	Profit	7000	Yes	9500
3	Type A produced	40	No	35
4	Type B produced	40	Yes	40

This optimisation has a single optimal solution given by point A on Fig. 3.3. This is the optimal solution to the lexicographic solution. The solution in decision space is

$$x_1 = 35, x_2 = 40$$

i.e. make 35 type A products and 40 type B products per week. In terms of satisfaction of the target levels, the data given in Table 3.1, calculated by substituting the optimal decision variable values into the goals, can be observed.

The effects of using the priority structure together with the L_1 distance metric which is not concerned with balance between goals can be observed. The goals placed in the top two priority levels have done relatively well but the labour goal, placed in the third priority level, is a long way from being achieved. If these indeed reflect the priorities of the company then this solution is appropriate and they will employ extra labour to make up for the shortfall. If they are not satisfied with this solution then it will probably be due to the labour goal and either a different variant or sensitivity analysis with the priority structure and goal target values should be employed to investigate other solutions. Also, recall that the company was uncertain about the level of the profit goal. The optimal solution shows an over-achievement by $2500. In fact the optimal solution for this variant turns out not to be sensitive to this target level.

If the priority structure is altered then a completely different solution will probably be obtained. For example suppose the company's priorities were

Priority 1: Achieve the labour goal.
Priority 2: Achieve the strategic production goals.
Priority 3: Achieve the profit goal.

The achievement function is now

$$\text{Min } a = \left[(p_1),(n_3 + n_4),(n_2)\right]$$

and solving gives the point B shown in Fig. 3.3. That is $x_1 = 15$ $x_2 = 20$. The goal satisfaction levels are as in Table 3.2.

Table 3.2 Solution of example with alternative priority structure

Goal number	Description	Target level	Sat	Achieved value
1	Labour	120	Yes	120
2	Profit	7000	No	3500
3	Type A produced	40	No	15
4	Type B produced	40	No	20

The labour goal is now satisfied, being the company's fist priority. This is, however, at the expense of considerable non-achievement of the profit and strategic production goals.

3.3.1 Good Modelling Practice for the Lexicographic Variant

It has been observed that the overall proportion of goal programming papers using the lexicographic variant has declined considerably in the past two decades (Jones and Tamiz, 2002). One reason for this decline could be the perceived lack of flexibility and suggested incompatibility with utility functions (Min and Storbeck, 1991). An inspection of the literature, however, reveals that in many cases the 'problems' are more to do with lack of good goal programming practice rather than a fundamental flaw in the technique or its underlying philosophy. This section gives some advice on the topic.

One major issue in lexicographic goal programming is that of priority level redundancy. Suppose for instance that the modeller decides to use the ideal values of 100 h for labour and $11,250 for profit. The model with priority scheme 1 is now

$$\text{Min } a = \left[(n_2),(n_3 + n_4),(p_1)\right]$$

subject to

$$\begin{aligned}
&4x_1 + 3x_2 + n_1 - \underline{p_1} = 100\\
&100x_1 + 150x_2 + \underline{n_2} - p_2 = 11{,}250\\
&x_1 + \underline{n_3} - p_3 = 40\\
&x_2 + \underline{n_4} - p_4 = 40\\
&2x_1 + x_2 \geq 50\\
&x_1 + x_2 \leq 75\\
&x_1, x_2 \geq 0,\ n_q, p_q \geq 0\ \text{q} = 1,....,4
\end{aligned}$$

The first priority level now has a single optimal solution, point C in Fig. 3.3 corresponding to $x_1 = 75\ x_2 = 0$. Therefore there is now no point in continuing with the

optimisations associated with priority levels 2 and 3 as they will not affect the final solution. The only goal to be considered is the one in the first priority level. One may as well have conducted a single-objective optimisation of that goal. There has therefore been no true consideration of the multi-objective nature of the problem. This is an extreme case of the problem of **lexicographic redundancy**(Romero, 1991). In reality it does not have to involve all but one of the priority levels and does not only occur when goals are set to their ideal values. There is some discussion on the gravity of this problem, with Schniederjans (1995) arguing that some redundancy is not necessarily a problem as goals are placed in lower priority levels by the decision maker for a reason, but nevertheless the consensus seems to be that redundancy is in general undesirable as it limits the number of goals considered. The following causal effects of redundancy are therefore identified:

- Setting of target levels that are too optimistic, or in the worst case, ideal
- Use of an excessive number of priority levels

The number of priority levels used depends on how many goals can be grouped together in a priority level. As one of the advantages of using the lexicographic variant is the avoidance of direct comparison of incommensurate goals, one would be wary of including goals in the same priority level that one did not wish to compare. However, in many cases there are goals which are in the same units or which can be compared if normalised (e.g. goals 3 and 4 in the example). These should be placed in the same priority level rather than creating separate priority level if they have similar levels of decision maker priority. Intra-priority weights can be used to allow for decision maker preference between them. The opposite approach would be to use one priority level for each goal. It can be appropriate for cases with few goals but great care should be taken over setting target levels if this approach is used.

The consensus in the goal programming literature is that **no more than five priority levels should be used in the lexicographic variant**. It would be unusual to find a practical situation with more than five mutually non-comparable priorities.

In alternative notation, seen in the goal programming literature is the 'Big P' notation. This uses P_i to represent the ith priority level. For example, the achievement function of the first priority scheme in the example would be

$$\text{Min}\, a = P_1 n_2 + P_2(n_3 + n_4) + P_3 p_1$$

where P_1, P_2, P_3 are pre-emptive constants such that $P_1 \gg P_2 \gg P_3$. This notation is not preferred in this book as it expresses the lexicographic minimisation as a summation when in fact it is not, neither from a theoretical nor a practical viewpoint. The preferred notation

$$\text{Lex Min}\, a = \left[(n_2),(n_3 + n_4),(p_1)\right]$$

expresses the problem as a lexicographic minimisation of a vector which matches both the theory and the practice of the variant.

To conclude, the lexicographic variant provides a very strong scheme for the decision maker to express their preferences. It can be the ideal choice when such a priority scheme exists in the mind of the decision maker. It can also be used to avoid unwanted direct comparisons between sensitive criteria.

3.4 Weighted Variant

When the decision maker wishes to compare the deviations and investigate the trade-offs between them then the weighted variant should be used. The example will now be formulated as a weighted goal programme. Critical to this process are the weights associated with the penalisation of the deviational variables. These give the means of obtaining the decision maker's preferences for this variant. The weight technically contains two parts. The first is a preferential weight in the numerator that represents the relative importance of the penalisation to the decision maker. Additionally, a normalisation factor in the denominator is needed that scales the deviations so they can be compared in the same units. It is vital that this second part is not omitted if the goals are not measured in compatible units. Otherwise, the summation will be meaningless due to the problem of incommensurability. As a demonstration of this consider that the following two goals express the same concept: minimisation of any negative deviation away from a target level of 5 m for the length of a roll of material, but in different units (millimetres and metres):

$$1000x_1 + 1000x_2 + \underline{n_1} - p_1 = 5000$$
$$x_1 + x_2 + \underline{n_1^*} - p_1^* = 5$$

As the optimal solution should be independent of the choice of units, these two versions of the same goal should give the same contribution to the achievement function, but if they are left as they are then the goal in metres will only return a thousandth of the deviation of the goal in millimetres. Therefore, some scaling is needed. The following section examines commonly used scaling, or normalisation, constants.

3.5 Normalisation

3.5.1 Percentage Normalisation

The first choice to consider is known as percentage normalisation. Here each deviation is turned into a percentage value away from its target level. Thus all deviations are measured in the same units. In our example, consider that the decision maker regards the penalisation of all unwanted deviations as equally important. The weighted goal programme with percentage normalisation is given as

$$\text{Min } a = \frac{p_1}{120} + \frac{n_2}{7000} + \frac{n_3}{40} + \frac{n_4}{40}$$

subject to

$$4x_1 + 3x_2 + n_1 - \underline{p_1} = 120$$
$$100x_1 + 150x_2 + \underline{n_2} - p_2 = 7000$$
$$x_1 + \underline{n_3} - p_3 = 40$$
$$x_2 + \underline{n_4} - p_4 = 40$$
$$2x_1 + x_2 \geq 50$$
$$x_1 + x_2 \leq 75$$
$$x_1, x_2 \geq 0,\ n_q, p_q \geq 0 \text{ q} = 1,....,4$$

Notice how the weights in the objective function have been divided by the target levels rather than the entire goal. The former gives the same effect as the latter but with less effort in reformulation. Also, the objective function contributions are actually proportions rather than percentages. That is they are one hundredth of the percentage value. The entire achievement function could be multiplied by 100 to give percentages, but this does not affect the optimal solution found. The achievement function value a now has meaning, the total sum of proportional deviations away from the goals. The solution is $x_1 = 10$, $x_2 = 40$, represented by point A in Fig. 3.4 and given in Table 3.3.

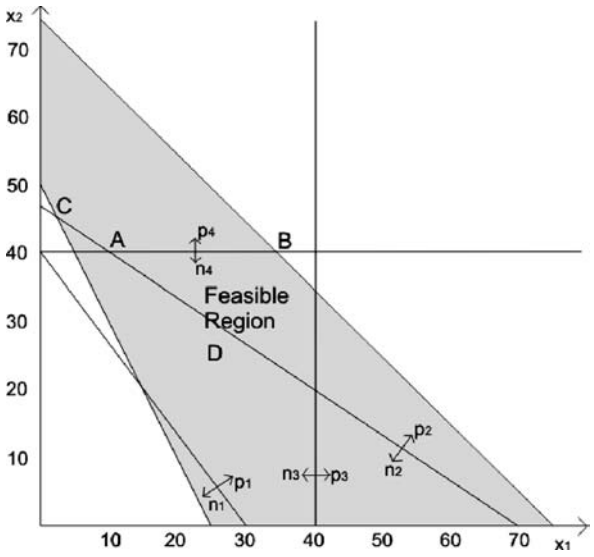

Fig. 3.4 Solution of example by weighted and Chebyshev variants

Table 3.3 Solution of example with percentage normalisation

Goal number	Description	Target level	Sat	Achieved value
1	Labour	120	No	160
2	Profit	7000	Yes	7000
3	Type A produced	40	No	10
4	Type B produced	40	Yes	40

The percentage deviation is both convenient to apply and simple to understand. It can, however, cause some distortion when a subset of the goals is measured in the same units. For instance consider the following two goals, both expressed in dollars that form part of a goal programme:

$$x_1 + 3x_2 + \underline{n_1} - p_1 = 10{,}000$$
$$2x_1 + x_2 + \underline{n_1} - p_2 = 5000$$

Applying percentage normalisation will give an achievement function of

$$\text{Min}\, a = \frac{n_1}{10000} + \frac{n_2}{5000}$$

which now means that 1 dollar shortfall from goal 1 has only half the importance of 1 dollar shortfall from goal 2. This is correct if the concern is about percentage shortfall of budgets but not correct if wishing to compare the shortfalls directly in dollars. Thus care should be taken as the correct choice is problem dependent. The other situation which will cause problems for percentage normalisation are goals with zero target values or target values that are not representative of the range of possible values for the criterion. In this case, a realistic value for the normalisation constant based on investigations of the range of possible values should be derived.

In any case, whichever normalisation scheme is used, for every goal a verification of the normalisation constant should take place in the context of the decision situation to ensure that it is having the desired effect.

3.5.2 Zero–One Normalisation

An alternative normalisation scheme is to scale all unwanted deviations onto a zero–one range. The value zero will represent a deviation of zero and the value one will represent the worst (highest) possible value of the deviation within the feasible set, which gives the anti-ideal value of the underlying objective. This latter value can be found by a single-objective maximisation, except in some cases where this optimisation is unbounded in which case a realistic worst practical value will have to be estimated.

In our example problem the worst possible values of our unwanted deviational variables are given in Table 3.4.

Table 3.4 Maximal values of unwanted deviational variables in example

Unwanted deviation	Maximum value
p_1	180
n_2	4500
n_3	40
n_4	40

This leads to the achievement function

$$\text{Min}\, a = \frac{p_1}{180} + \frac{n_2}{4500} + \frac{n_3}{40} + \frac{n_4}{40}$$

with the same set of constraints and sign restrictions as the percentage normalisation formulation. This leads to the optimal solution

$$x_1 = 35, x_2 = 40$$

which is represented by point B in Fig. 3.4 and whose goal attainment values are given in Table 3.5.

Table 3.5 Solution of example with zero–one normalisation

Goal number	Description	Target level	Sat	Achieved value
1	Labour	120	No	260
2	Profit	7000	Yes	9500
3	Type A produced	40	No	35
4	Type B produced	40	Yes	40

In particular it should be noted that the normalisation constant for n_2 is considerably lower when considering the range of values (zero–one normalisation) rather than percentage movement from the target value (percentage normalisation). This shows that the choice of normalisation method can indeed affect the optimal solution found and hence care should be taken to choose a method that reflects the preferences and interests of the decision maker.

The zero–one method is good in cases such as the example presented when each objective has clearly defined ranges and all portions of the feasible set are of potential interest to the decision maker. However, for examples with unbounded goals (i.e. no clear worst value) or with large portions of the feasible region that are unbounded it can suffer from the problem of irrelevant alternatives. That is, the worst value, and hence the normalisation constant, corresponds to an arbitrarily chosen or irrelevant solution. A theoretical alternative is to use the worst value from amongst the set of Pareto-efficient solutions (nadir value), but this value is computationally difficult to determine (Ehrgott, 2005, p. 34). The other point to note about zero–one normalisation is that it requires Q single-objective optimisations in order to determine the worst values for the objectives. This could be impractical for complex or large problems where the solution time for the problem is large.

Table 3.6 Euclidean normalisation constants

Unwanted deviation	Normalisation constant
p_1	$\sqrt{3^2 + 4^2} = 5$
n_2	$\sqrt{100^2 + 150^2} = 180.28$
n_3	$\sqrt{1^2} = 1$
n_4	$\sqrt{1^2} = 1$

3.5.3 Euclidean Normalisation

A third normalisation scheme found in the literature is that of Euclidean normalisation. Here the normalisation constant is the Euclidean mean of the technical coefficients of the goal. Table 3.6 illustrates the calculation of the normalisation constants in the example.

This leads to the optimal solution

$$x_1 = 2.5, x_2 = 45$$

which is represented by point C in Fig. 3.4 and whose goal attainment values are given in Table 3.7.

Table 3.7 Solution of example with Euclidean normalisation

Goal number	Description	Target level	Sat	Achieved value
1	Labour	120	No	145
2	Profit	7000	Yes	7000
3	Type A produced	40	No	2.5
4	Type B produced	40	Yes	45

Euclidean normalisation is a very computationally robust method; it works with all goals and target levels and does not require any optimisations or complex calculations in order to compute the normalisation constants. However, the two problems outlined by Romero (1991) with respect to Euclidean optimisation are evident from the example. The first is that a lack of consideration of the target value has led to relatively low values of the normalisation constants of goals 3 and 4. The second is that, unlike the other two normalisation methods, the optimal value of the achievement function does not have any obvious meaning. Hence, in our opinion, this normalisation scheme should be reserved for cases in which it is impractical to apply either percentage or zero–one normalisation.

A further normalisation scheme based on the summation of the absolute values of the technical coefficients of the goals is proposed by Jones (1995), as an automated means of measuring the level of incommensurability in a goal programme. It should be noted, however, that the ultimate choice of the normalisation scheme is dependent on the individual problem situation and preferences of the decision maker.

3.6 Preferential Weight Choice

The other part of the weight in the WGP model, the preferential weight, will now be considered. As detailed in Chapter 2, these weights should be non-negative for the basic WGP model and the following notation will be used:

u_i: preferential weight associated with the penalisation of n_i
v_i: preferential weight associated with the penalisation of p_i

An important consideration is that **only unwanted deviational variables should be given a positive weight**. Other deviations, for example excess profit in the example, p_2, should be given a zero weight. Giving deviations such as p_2 a positive weight will unnecessarily penalise good solutions and hence may lead to erroneous conclusions.

The derivation of a weight set that will accurately represent the preferences of the decision maker is an issue of key importance when using weighted goal programming. Many decision makers will not instantly be able to know or express weights with confidence, especially when implications of their weight choice are as yet unknown. Decision maker(s) naturally relate more to the implications of weighting strategies in terms of the levels of achievement of the goals or the actual decisions made. Therefore, just as for the target values, **it is important to regard weight determination as a process of interaction with the decision maker(s) rather than a single a priori declaration of a weighting scheme**. This interaction can take many forms. Formal interactive methods exist for goal programming and for the wider field of multiple objective programming (e.g. Gardiner and Steuer, 1994; Caballero et al., 2006), but these sometimes run the risk of further obscuring the situation by asking the decision maker(s) for complex information about derivatives or trade-offs. Otherwise, a good sensitivity analysis should be conducted, with the effects on goal satisfaction of changing individual weights or subsets of weights described and reported to the decision maker(s). This could result in the decision maker(s) moving to a different solution once this information is known. This process needs to start from an initial solution and there are several options seen in the literature for choosing an initial point:

- An initial estimate by the decision maker. This is a good option if the decision maker(s) is confident enough to produce one.
- The equal weighting scheme, with the preferential weights of all the unwanted deviations set equal to one. This is a valid starting point if no other information is available, but tragically it can be seen in the literature as the only point investigated in some cases.
- The solution found by the analytical hierarchy process (Saaty, 1981) after the decision maker(s) supplies pairwise information regarding the importance of the goals. This can supply a starting point, but again does not circumvent the need for good sensitivity analysis with respect to the set of weights.

It also should be recognised that decision maker(s) and decision-making committees often prefer the presentation of a relatively small number of sufficiently different solutions given in terms of their achievements of goals and decision variable values. They do not always have the time or inclination to engage in a complex or iterative weight elicitation process. In this case, the weighting space should be explored by the modeller and an appropriate set of solutions prepared for the decision maker(s). Figure 3.5 shows the weighting space of the example from which the set of solutions given in Table 3.8 is found. The weight space exploration heuristic used to produce Table 3.8 is outlined in Section 4.3. The input to the algorithm is the equal weights solution and the parameters are set to TMax=2 and max_level=2.

Figure 3.5 demonstrates the four-dimensional weighing space that is being searched in the sensitivity analysis. The three axes give the value of the first three weights and the value of the fourth weight can be ascertained by the fact that the

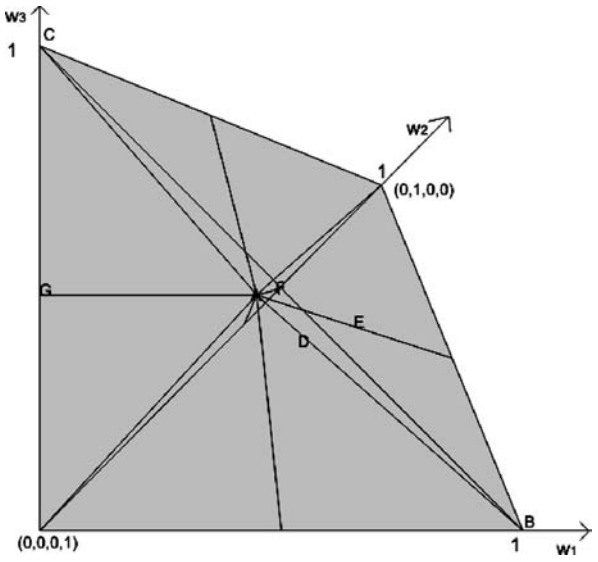

Fig. 3.5 Weight sensitivity analysis for example

Table 3.8 Weight sensitivity analysis for example

	w_1	w_2	w_3	w_4	x_1	x_2	Obj1	Obj2	Obj3	Obj4
A	0.25	0.25	0.25	0.25	10	40	160	7000	10	40
B	0.99	0.0033	0.0033	0.0033	15	20	120	4500	15	20
C	0.0033	0.0033	0.99	0.0033	40	20	220	7000	40	20
D	0.4375	0.1875	0.1875	0.1875	5	40	140	6500	5	40
E	0.495	0.495	0.005	0.005	2.5	45	145	7000	2.5	45
F	0.495	0.005	0.495	0.005	30	0	120	2500	30	0
G	0.005	0.005	0.495	0.495	35	40	260	9500	35	40

weighting vector has been normalised and hence sums to one. See Section 4.4 for a discussion of the theoretical issues relevant to this weight normalisation.

3.7 Chebyshev Variant

A common feature in all of the solutions found by both the weighted and lexicographic variants can be found. That is, they are all to be found at extreme points (intersections of goals, constraints, and/or axes). This leads to a certain imbalance, with some goals doing very well but other goals being a long way from being achieved. This is due to the underlying Manhattan metric that has a 'ruthless optimisation' property. If a balance between the objectives is the dominant need, then the Chebyshev goal programming (CGP) variant (Flavell, 1976) should be applied. The Chebyshev version of the example, assuming equal preferential weights and a percentage normalisation scheme, is given as

$$\text{Min}\, a = \lambda$$

subject to

$$\begin{aligned}
&\tfrac{p_1}{120} \leq \lambda \\
&\tfrac{n_2}{7000} \leq \lambda \\
&\tfrac{n_3}{40} \leq \lambda \\
&\tfrac{n_4}{40} \leq \lambda \\
&4x_1 + 3x_2 + n_1 - \underline{p_1} = 120 \\
&100x_1 + 150x_2 + \underline{n_2} - p_2 = 7000 \\
&x_1 + \underline{n_3} - p_3 = 40 \\
&x_2 + \underline{n_4} - p_4 = 40 \\
&2x_1 + x_2 \geq 50 \\
&x_1 + x_2 \leq 75 \\
&x_1, x_2 \geq 0,\ n_q, p_q \geq 0 \text{ q} = 1,...,4
\end{aligned}$$

This model is a specific version of the generic CGP given in Section 2.2.3. Its solution is given as $x_1 = 24$, $x_2 = 24$. This is represented by point D in Fig. 3.4. The levels of the goals are given in Table 3.9.

Table 3.9 shows that although none of the goals is achieved, neither is there any goal that is under-achieved significantly more than the others. The worst level of achievement is for goals 1, 3, and 4 which all show a 40% deviation from their target levels. This compares with a worst deviation of 75% for goal 3 for the weighted variant with percentage normalisation.

Table 3.9 Solution of example by Chebyshev variant

Goal number	Description	Target level	Sat	Achieved value
1	Labour	120	No	168
2	Profit	7000	No	6250
3	Type A produced	40	No	24
4	Type B produced	40	No	24

3.8 Summary – Ten Rules for Avoiding Pitfalls in Goal Programming Formulations

1. Always include both deviational variables in the goal formulation except where the target level has been set to the ideal value.
2. Don't combine ideal target levels with the lexicographic variant.
3. Don't use an excessive number of priority levels with the lexicographic variant.
4. Regard the weights, priority schemes, and target levels as initial estimates and not set in stone values.
5. Always use an appropriate normalisation scheme with the weighted variant.
6. Remember that the Chebyshev variant and priority levels within the lexicographic variant need normalising too if they are not commensurable.
7. Always perform a validation of each normalisation constant in the context of the decision situation.
8. Always check the solution for Pareto efficiency and use a restoration scheme if required.
9. Only penalise unwanted deviations.
10. Always make an informed choice of variant based on the nature of the problem.

3.9 Exercises

Example 1 – Conversion from Linear to Goal Programming

Given the following linear programme

$$\text{Max}\, z = x_1 + x_2$$

subject to

$$x_1 + 2x_2 \leq 20$$
$$3x_1 + x_2 \leq 30$$

Table 3.10 Preference weights for Example 1

Goal	Weight
The objective function should be at least 40	0.25
Constraint 1 should not be violated	0.5
Constraint 2 should not be violated	0.25

(i) Convert this model into a weighted goal programme with percentage normalisation and the three (incompatible) goals and associated preference weights given in Table 3.10.
(ii) Reformulate the model as a Chebyshev goal programme using the same normalisation scheme and preference weights.
(iii) Reformulate the model as a lexicographic goal programme with the following two priority levels and the same preference weights within priority level 1 as used for the relevant goals in the weighted variant:

- *Priority Level 1*: Neither constraint should be violated.
- *Priority Level 2*: The objective function should achieve a level of at least 40.

Compare this formulation to that of the original linear programme. What does this lead you to conclude about the relationship between linear programming and lexicographic goal programming?

Example 2 – On-line Retailer

An on-line grocery retailer must decide on how many of two types of picker* to employ in order to process customer orders. Novice pickers are paid £6 per hour and can process 20 orders per hour. Experienced pickers are paid £10 per hour and can process 40 orders per hour. The retailer has the following goals:

Goal 1: The retailer wishes to employ exactly 15 pickers.
Goal 2: The retailer wishes to process at least 500 orders per hour.
Goal 3: The retailer wishes for the total wage bill not to exceed £95 per hour.

In addition, there are hard constraints that at least three experienced pickers must be employed and no more than 20 pickers can be employed in total.

(a) Represent this situation graphically, marking the feasible region, the goals, and the deviational variables on your graph.
(b) Formulate this decision problem as a lexicographic goal programme with the following two priority levels:

Priority Level 1: Achieve goal 3.
Priority Level 2: Achieve goals 1 and 2 (equally weighted).

Table 3.11 Preference weights for Example 2

Goal	Preference weight
1 (both sides)	0.1
2	0.3
3	0.6

(c) Formulate this decision problem as a weighted goal programme with percentage normalisation and the preference weighting scheme given in Table 3.11.
(d) Formulate this decision model as a Chebyshev goal programme with the same normalisation and preference weighted schemes as in part (c).

* A picker is responsible for assembling all the items in a customer's order once it has been received via the internet.

Example 3 – Production Planning

A company produces three types of cup, termed grade A, grade B, and grade C. Each grade A cup requires 2 h of furnace time and 3 h of finishing labour. Each grade B cup requires three hours of furnace time and 5 h of finishing labour. Each grade C cup requires 4 h of furnace time and 10 h of finishing labour. A grade A cups yields a profit of £1.00, a grade B cup a profit of £1.50, and a grade B cup a profit of £2.20. The company currently has 1000 h of furnace time and 2000 h of finishing labour per day. They have a high level of demand and therefore they have a strategic aim of increasing production to 1000 cups per day by increasing the level of furnace and finishing hours available.

(i) Represent this situation as a normalised weighted goal programme with the preference weights given in Table 3.12.
(ii) Represent this situation as a lexicographic goal programme with the following priority structure:

Priority 1: Achieve the strategic aim of 1000 cups per day.
Priority 2: Achieve a profit of at least £1250.
Priority 3: Minimise the extra finishing and furnace hours needed.
Priority 4: Ensure that at least 300 of each type of cup is manufactured.

Table 3.12 Preference weights for Example 3

Goal	Preference weight
(1) Achieve the strategic aim of 1000 cups per day	5.0
(2) Achieve a profit of at least £1250	3.0
(3) Minimise the extra finishing and furnace hours needed	2.0
(4) Ensure that at least 300 of each type of cup is manufactured	1.0

Example 4 – Employee Scheduling

A factory must determine how many workers to employ at different times of the day. The estimated demand by 3-hourly intervals is given in Table 3.13.

Table 3.13 Employees required by hour of day

Hours of day	Estimated number of workers
00:00–03:00	5
03:00–06:00	7
06:00–09:00	15
09:00–12:00	21
12:00–15:00	22
15:00–18:00	12
18:00–21:00	7
21:00–24:00	3

The workers can start at the beginning of any of the 3-hourly intervals and work a 6-h shift. Workers working at night (between 22:00 and 08:00) get paid £10 for each hour that falls into the night interval. In the daytime workers are paid £7 for each hour worked.

(i) Formulate a normalised weighted goal programme with the preference weights given in Table 3.14.
(ii) Formulate a lexicographic goal programme using the preference weights given in Table 3.14 where necessary as intra-priority level weights and with the following priorities:

Priority 1: Ensure that each interval has at least the estimated number of workers.
Priority 2: Ensure the total cost of the workers is no more than £280.
Priority 3: Minimise the number of workers employed above the estimate for each interval.
Priority 4: Minimise the number of workers starting their shift at 03:00 hours

Table 3.14 Preference weights for Example 4

Goal	Preference weight
Ensure that each interval has at least the estimated number of workers	0.65
Ensure the total cost of the workers is no more than £280	0.2
Minimise the number of workers employed above the estimate for each interval	0.1
Minimise the number of workers starting their shift at 03:00 hours	0.05

Table 3.15 Nutritional and cost data for foodstuffs

Food	Protein (g)	Carbohydrate (g)	Sat fat (g)	Vit B6 (mg)	Vit C (mg)	Calcium (mg)	Cost ($)
1	3.3	5.0	1.0	0.06	1	120	0.45
2	25.5	0.1	21.7	0.10	0	720	1.00
3	2.5	0	0.6	0.02	0	11	0.60
4	11.0	75.7	0.4	0.22	0	35	0.20
5	27.3	0	5.2	0.29	0	7	2.50
6	3.3	1.1	0.2	0.11	44	40	1.00
7	1.2	23.2	0.1	0.29	11	6	0.30
8	1.2	2.6	0.1	0.05	7	20	0.80
9	2.6	30.9	0.3	0.07	0	18	0.20
10	0.4	11.8	0.0	0.06	6	4	0.30
Daily min	40.0	100.0	2.0	1.0	50	800	–
Daily max	60.0	300.0	15.0	2.0	100	1500	£3.50

Example 5 – Diet Planning

A dietician is required to propose a diet for a special needs patient. The values of protein, carbohydrate, saturated fat, vitamin B6, vitamin C, and calcium should ideally fall between the bounds given in Table 3.15. The diet can be composed of an integer number of units of ten basic foodstuffs, termed FOOD1,. . ., FOOD10. The amount of nutrients in one unit of foodstuff and its cost is given in Table 3.15. Ideally no more than three units of any foodstuff should be consumed on a given day and the daily cost should be below £3.50. Also the patient desires that no more that 3 units of FOODS 1 through 3 be consumed daily.

Formulate

1. A normalised weighted goal programme that places ten times as much importance on medical needs as cost considerations or patient preferences
2. A lexicographic goal programme that considers medical needs, cost considerations, and patient preferences in that order
3. A lexicographic goal programme that considers medical needs, patient preferences, and cost considerations in that order

Example 6 – Travelling Salesperson

A salesperson has to visit four cities exactly once, returning to his/her point of origin. The distance (in kilometres), time (in hours), and cost (in €) of travelling between cities are given in Table 3.16.

The salesperson wishes to achieve goals of no more than 1000 km, 10 h, and €200 for the trip.

Table 3.16 Distance, time, and cost between cities

Dist/time/cost	Alphaville	Betatown	Gammaton	Delcester
Alphaville	0/0/0	248/4.25/80	237/3.10/56	344/4.50/95
Betatown	248/4.25/80	0/0/0	469/4.60/85	127/2.00/40
Gammaton	237/3.10/56	469/4.60/85	0/0/0	540/6.10/130
Delcester	344/4.50/95	127/2.00/40	540/6.10/130	0/0/0

(i) Represent this situation as a normalised weighted goal programme with equal weights on achieving the distance, time, and cost goals*.
(ii) Reformulate your model from part (i) as a normalised Chebyshev goal programme that provides a balance between the achievement of the distance, time, and cost goals*.

* Note that to solve a travelling salesperson problem via mathematical programming, subtour prevention techniques must be utilised. Refer to Applegate et al. (2007) for details.

Example 7 – Downstream Oil Industry

An oil company has to transport oil products from three sites (S1, S2, S3), through three refineries (R1, R2, R3), onto three depots (D1, D2, D3), and finally to three customers (C1, C2, C3). The supply and demand levels (in 1000 l per day) are given in Table 3.17.

Table 3.17 Site supply and customer demand levels

Site	Supply	Customer	Demand
S1	1000	C1	500
S2	500	C2	800
S3	800	C3	900

The costs in £ of transporting 1000 l of fuel between sites and refineries, refining 1000 l of fuel at a refinery, and maximum capacities of the refineries are given in Table 3.18.

The costs in £ of transporting 1000 l of fuel between refineries and depots and maximum capacities of the depots are given in Table 3.19.

The costs in £ of transporting 1000 l of fuel between depots and customers and maximum capacities of the depots are given in Table 3.20.

Given that all customer demands must be fulfilled, formulate a normalised weighted goal programme with the following goals:

Table 3.18 Site to refinery costs, refinery costs, and refinery capacities

From/to	R1	R2	R3
S1	20	7	8
S2	12	10	6
S3	15	8	11
Refining cost	7	6	5
Capacity	1000	1200	800

Table 3.19 Refinery to depot costs and depot capacities

From/to	D1	D2	D3
R1	6	8	5
R2	4	10	7
R3	5	9	8
Capacity	1400	900	900

Table 3.20 Depot to customer costs

From / to	C1	C2	C3
D1	6	8	5
D2	4	10	7
D3	5	9	8

- Goal 1: Keep costs to below £40,000 per day.
- Goal 2: Keep each refinery operating at at least 75% of its capacity.
- Goal 3: Keep each depot at a level of between 50 and 90% of its capacity.
- Goal 4: Assign any unused capacity to site three.

1. How would you go about determining weights for this goal programme and checking their validity?
2. Reformulate as a lexicographic goal programme with the following priority order: goals 2 and 3 (equally weighted), goal 1, goal 4. Does this solve the issue of commensurability or is normalisation required?
3. Reformulate as a normalised Chebyshev goal programme.

Example 8 – Macro Economics

A finance minister requires to raise an extra £200,000,000 through taxation. Five possible budgetary enhancing measures, termed actions A–E, are possible, each raising an extra £100,000,000. These actions affect the low-income, mid-income, and high-income members of the populace in different ways. Table 3.21 gives the per annum changes in pounds caused by implementing the actions.

The minister wishes to limit the total per annum loss of income to £1000 for high-income earners, £750 for mid-income earners, and £500 for low-income earners.

Table 3.21 Impact of actions

Action	Low income	Mid-income	High income
A	200	200	200
B	100	500	1500
C	200	700	2000
D	500	800	1000
E	800	100	–100

1. Assuming the three groups are of equal importance, formulate a weighted goal programme to represent this situation. Does this model require normalisation? Why or why not?
2. Again, assuming the three groups are of equal importance, formulate a Chebyshev goal programme of this situation.
3. Assuming the priority order: low income, mid-income, high income, formulate a lexicographic goal programme of this situation.

Example 9 – Budget Planning

A new budget has to be allocated to ten facilities based on their values in five key performance measures (F1–F5), given in Table 3.22. The current budgets (CB) in £1000 are given in the right-hand column of Table 3.22.

1. Formulate a goal programme whose solution will give the correct mix of the five measures so as to minimise the total change in budget (positive or negative) for the ten facilities in transitioning from the current to the new budget. Include a constraint to ensure that the total funds allocated increases by no more than 20%.

Table 3.22 Facility performance measures and current budgets

Facility	F1	F2	F3	F4	F5	CB
1	6	8	5	1	9	100
2	4	10	7	8	3	120
3	6	9	5	6	3	130
4	10	3	2	6	6	90
5	7	9	5	5	2	140
6	4	8	10	3	5	140
7	1	10	9	8	9	210
8	2	9	2	8	8	170
9	2	1	3	10	4	110
10	8	6	3	7	7	100

2. Formulate a goal programme whose solution will give the correct mix of the five measures so as to minimise the maximum change in budget (positive or negative) for any facility in transitioning from the current to the new budget. Include a constraint to ensure that the total funds allocated increases by no more than 20%.

Example 10 – Healthcare Planning

A hospital consists of six departments. The hospital manager wishes to achieve goals relating to the time of service and cost of service, for both individual departments and the hospital as a whole. A total of 280 beds must be divided amongst the six departments by the hospital manager. Each department has a minimum and maximum bound on the number of beds, as shown in Table 3.23, and has a relative importance weighting assigned by the hospital manager, also given in Table 3.23. Each department is given the opportunity to place its own relative preference weights on time and cost and these are given in Table 3.23.

Table 3.23 Departmental bed limits and preferential data

Department	Mininum beds	Maximum beds	Relative importance	Weight on time	Weight on cost
1	15	30	1	0.5	0.5
2	25	50	3	0.6	0.4
3	50	120	5	0.7	0.3
4	20	50	1	0.5	0.5
5	30	70	3	0.4	0.6
6	15	30	1	0.5	0.5

The relative performance of the departments on the two performance measures, time and cost, is found by simulation modelling to be a linear function of the number of beds (x_1) as shown in Table 3.24. The target cost and time levels for the departments are also given in Table 3.24.

The hospital manager also has an overall time target of 110 and cost target of 300. She weights these two goals equally at a hospital level.

Table 3.24 Departmental time and cost targets

Department	Time (t_i)	Cost (c_i)	Time target	Cost target
1	$t_1 = 40 - x_1$	$c_1 = 2x_1 + 10$	10	40
2	$t_2 = 100 - 2x_1$	$c_2 = x_2$	20	30
3	$t_3 = 500 - 4x_3$	$c_3 = 2x_3 - 20$	20	100
4	$t_4 = 150 - 3x_4$	$c_4 = 3x_4 - 40$	20	20
5	$t_5 = 230 - 3x_5$	$c_5 = 2x_5 - 10$	20	60
6	$t_6 = 30 - 2x_6$	$c_6 = x_6 - 10$	10	25

(i) Formulate a weighted goal programme that allows the hospital manager to assign beds to departments by placing equal weighting on achieving her hospital goals and achieving the departments' individual goals.

(ii) Formulate a lexicographic goal programme with the preferential weights given in Table 3.23 as intra-priority weights and the following priority structure:

Priority 1: Achieve the hospital-level goal with respect to time of service.
Priority 2: Achieve the individual departments' goals with respect to time of service.
Priority 3: Achieve the hospital-level goal with respect to cost of service.
Priority 4: Achieve the individual departments' goals with respect to cost of service.

(iii) Formulate a Chebyshev goal programme that just considers the goals of the individual departments and minimises the maximum deviation from the combined weighted time and cost goals.

Chapter 4
Advanced Topics in Goal Programming Formulation

The topic of how to formulate effective goal programmes using the major variants is covered in Chapter 3. This chapter expands on those concepts by introducing techniques developed to expand the flexibility of goal programming and enhance its use.

4.1 Axioms

The goal programming models formulated in Chapter 3 concerned situations where the basic axioms required for linear programming held. These axioms (Winston, 2004) are as follows:

- Proportionality: In the goal programming paradigm this axiom requires that the penalisation for an unwanted deviation from a target level is directly proportional to the distance away from the target level.
- Additivity: This axiom requires that the level of penalisation for an unwanted deviation from a target level is independent of the levels of unwanted deviations from the other goals.
- Divisibility: This axiom requires that all the decision variables should be free to take any value within their prescribed range (greater or equal to zero by default). Thus a decision variable cannot be forced to take an integer or a discrete value.
- Certainty: This axiom requires all the data coefficients to be known with certainty.

However, if one of the above axioms does not hold, the use of goal programming is not necessarily precluded. In the case of the additivity axiom not holding, a non-linear goal programme could be formulated or a meta-heuristic method described in Chapter 7 could be used. In the event of the divisibility axiom not holding, an integer or binary goal programming described in Section 2.3.2 could be formulated. When the certainty axiom does not hold then the method used will depend on the type of coefficients over which there exists uncertainty. As described in Chapter 3, a certain amount of uncertainty over weights and target values often exists and this can frequently be handled by good sensitivity or weight analysis techniques or

D. Jones, M. Tamiz, *Practical Goal Programming*, International Series in Operations Research & Management Science 141, DOI 10.1007/978-1-4419-5771-9_4,

an interactive method. An alternative is to use the fuzzy goal programming variant described in Section 2.3.1. If there is uncertainty over the technological coefficients then either the fuzzy goal programming variant could be utilised or a combination with a technique such as simulation could be attempted. The choice of method in this case is highly problem dependent. A range of real-life situations in which the proportionality axiom does not hold was identified in the 1980s and a technology known as penalty functions was developed to allow more accurate modelling in such cases. This has been expanded into the general case of non-standard preference function modelling in the 1990s. Section 4.2 describes this methodology with the help of a case study.

4.2 Non-standard Preference Function Modelling

The standard goal programming assumes a per unit penalty of either u_i or v_i for every unit of unwanted deviation from the target value. These lead to a linear relationship between the magnitude of the unwanted deviation and the penalty imposed. This type of relationship will be termed a preference structure in this book. The basic linear preference structure is graphically demonstrated by Fig. 4.1. The gradient of the line shown is u_i or v_i.

However, in many fields of application the penalties for exceeding a certain level of unwanted deviation could become more or even less severe. This change could for example take the form of a sudden increase in penalty to represent the cost of an event. An example of this would be if an extra vehicle or warehouse is required in a logistics model. Alternatively, there could be an increase in per unit penalty at

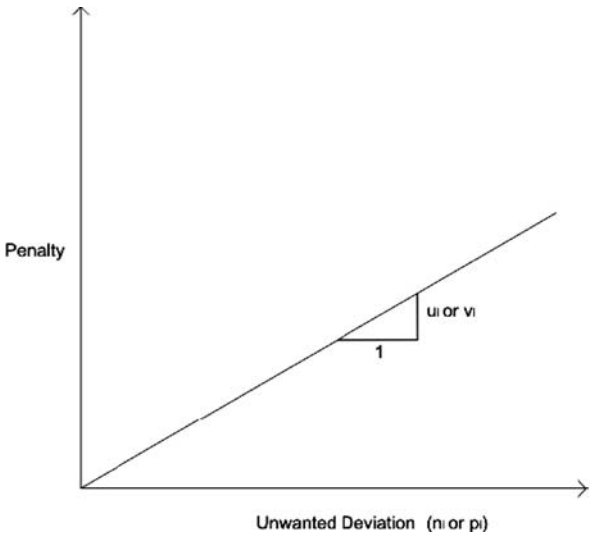

Fig. 4.1 Standard preference function

a point to represent more expensive goods or more severe consequences of missing the target by a long way. The opposite could also be true, as is the case if a bulk discount is offered. These considerations lead to the proposition of modelling any combination of the above circumstances as a series of four types of preference changes. These are proposed by Jones and Tamiz (1995) and are capable of modelling any type of monotonically increasing penalty structure.

Type 1: Increase in Per Unit Penalty (Penalty Function)

This type of preference change is used when the per unit penalty to be applied is increased beyond a certain level of deviation. For example, consider the second objective in the manufacturing example given in Chapter 3. This relates to profit and has the form

$$100x_1 + 150x_2 + \underline{n_2} - p_2 = 7000$$

with the penalisation of the term $\frac{n_2}{7000}$ in the achievement function. This leads to the relationship between profit made and penalty applied to the achievement function shown graphically by Fig. 4.1. This particular preference structure allows the proportionality axiom to hold but may not be a correct representation of the company's preferences. Suppose the company were to regard a profit level of below £5000 as twice as serious, below £2000 as four times as serious, and below £1000 as unacceptable. This leads to the new preference structure given by Fig. 4.2.

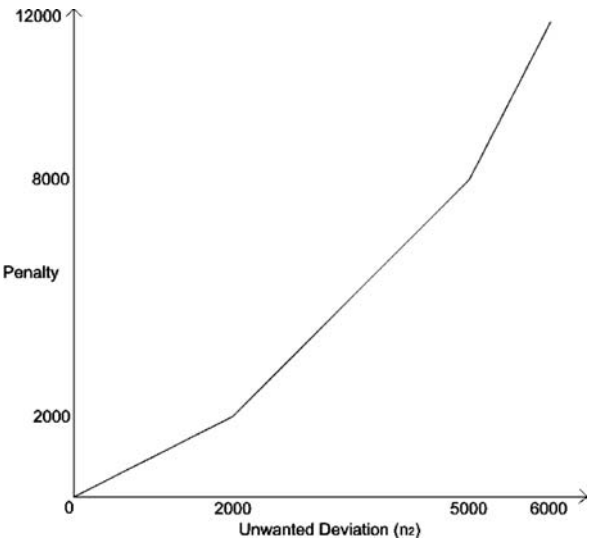

Fig. 4.2 Type 1 (penalty function) preference change

There are two major ways of modelling such a preference structure. The first is introduced by Can and Houck (1984) in the context of water resources planning. It is also used by Romero (1991), Martel and Aouni (1990), and Chang (2006). This method involves the introduction of bounded deviational variables to represent the different linear segments of the preference structure. The above example requires two extra deviation variables, which shall be termed $n_2^{(2)}$ and $n_2^{(3)}$. The original deviation variable n_2 will be re-termed $n_2^{(1)}$. The relevance of these deviational variables is shown in the preference structure given by Fig. 4.3.

The corresponding algebraic formulation for the second objective is given by

$$\text{Min } z = \ldots + \frac{n_2^{(1)} + 2n_2^2 + 4n_2^3}{7000} \ldots$$

subject to

$$100x_1 + 150x_2 + \underline{n_2^{(1)}} + \underline{n_2^{(2)}} + \underline{n_2^{(3)}} - p_2 = 7000$$

$$0 \leq n_2^{(1)} \leq 2000,\ 0 \leq n_2^{(2)} \leq 3000,\ 0 \leq n_2^{(3)} \leq 1000$$

As the per unit penalty is increasing with distance from the original goal, the variables $n_2^{(1)}$, $n_2^{(2)}$, $n_2^{(3)}$ will, if necessary, be introduced in that order by the

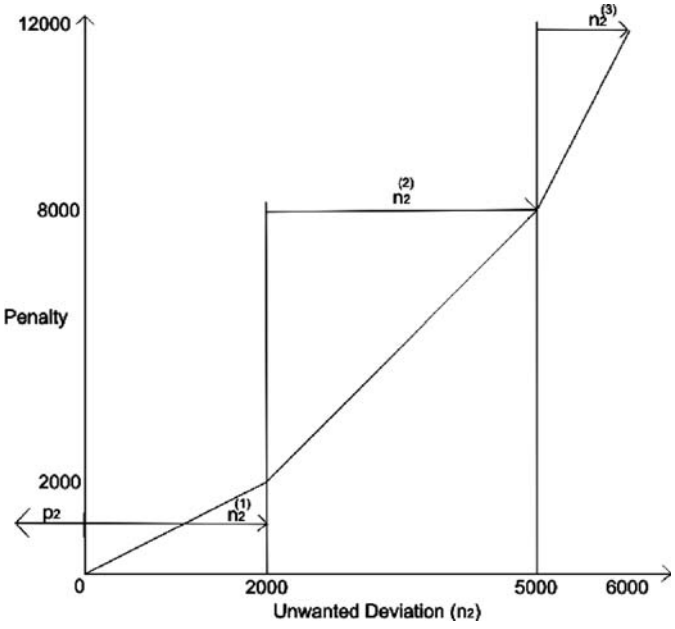

Fig. 4.3 Can and Houck's representation of type 1 preference change

optimisation routine used. It should also be noted that the same normalisation constant, based on the original goal, is used to normalise all the deviational variables. This is the case because they are all measured in the same unit.

The advantage of this type of formulation is that it is able to represent the new preference structure without adding any extra constraints to the model. The disadvantage is the lack of transparency in reading the results. This can be overcome by a simple piece of computer code to add the deviations together or by using a graphical representation of the achievement level of the preference function such as Fig. 4.3. Another point to consider when using this type of formulation is that it could interfere with some of the goal programming speed-up and analysis techniques (Jones, 1995).

The second method, introduced by Jones and Tamiz (1995), is to define a secondary objective at each of the points where the preference increases, and a hard constraint, known as an **objective bound** at the point beyond which the objective becomes so badly achieved as to make the resulting solution unimplementable. Again considering the second objective of the manufacturing examples, the original deviational variables n_2 and p_2 are re-termed $n_2^{(1)}$ and $p_2^{(1)}$. New deviational variables $n_2^{(2)}$ and $p_2^{(2)}$ are formed around the secondary target value of £5000 and $n_2^{(3)}$ and $p_2^{(3)}$ are formed around the secondary target value of £2000. The significance of the new deviational variables is shown graphically by Fig. 4.4.

The corresponding algebraic formulation for the second objective is given as

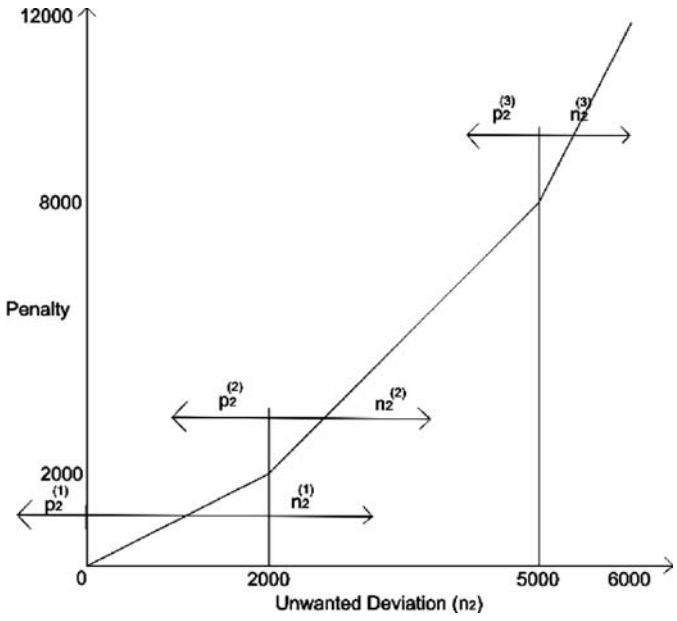

Fig. 4.4 Jones and Tamiz's representation of type 1 preference change

$$\text{Min } z = \ldots + \frac{n_2^{(1)} + n_2^{(2)} + 2n_2^{(3)}}{7000} + \ldots$$

subject to

$$\begin{aligned}
100x_1 + 150x_2 + \underline{n_2^{(1)}} - p_2^{(1)} &= 7000 \\
100x_1 + 150x_2 + \underline{n_2^{(2)}} - p_2^{(2)} &= 5000 \\
100x_1 + 150x_2 + \underline{n_2^{(3)}} - p_2^{(3)} &= 2000 \\
100x_1 + 150x_2 &\geq 1000
\end{aligned}$$

The achievement function coefficients of the deviational variables now refer to the increase in per unit penalty at the corresponding target value rather than the absolute per unit penalty. This formulation has added more objectives and deviational variables but is more transparent and also has the advantage of being more easily extendable to the other types of preference structure change discussed in this section.

Type 2: Decrease in Per Unit Penalty (Reverse Penalty Function)

A less common but still realistic situation is to see a decrease in per unit penalty beyond a certain level of achievement. For instance, consider the first objective in the manufacturing example given in Chapter 3. This pertains to the level of labour employed and has the algebraic format

$$4x_1 + 3x_2 + n_1 - \underline{p_1} = 120$$

with an achievement function contribution of $\frac{p_1}{120}$. Now suppose that any labour between 120 and 150 h should be made up by overtime at a high cost but over 150 h would result in the hiring of permanent labour at a lower cost and greater strategic benefit for the company. Therefore labour beyond 150 h, whilst still being undesirable, has a lower per unit cost than for between 120 and 150 h. Suppose the company estimates that the overall per unit penalty should be reduced by 20% beyond 150 h. Using the second (Jones and Tamiz, 1995) system as for an increase in penalty, only subtracting the change term in the achievement function leads to the following algebraic representation of the preference structure:

$$\text{Min } z = \frac{p_1^{(1)} - 0.2p_1^{(2)}}{120} + \ldots$$

subject to

$$4x_1 + 3x_2 + n_1^{(1)} - \underline{p_1^{(1)}} = 120$$
$$4x_1 + 3x_2 + n_1^{(2)} - \underline{p_1^{(2)}} = 150$$

There is an additional problem encountered by this type of preference change. The original goal programming models were designed to have deviational variables with only positive or zero coefficients in the achievement function. This implicitly ensured that both deviational variables of an objective would never take non-zero values simultaneously. This is an implicit modelling of the non-linear condition:

$$n_i \times p_i = 0$$

However, in the above formulation, $p_1^{(2)}$ has a negative achievement function coefficient. If solved as a linear programme on a standard solver the resultant goal programme would be incorrectly classed as unbounded with a violation of the implicit $n_1^{(2)} \times p_1^{(2)} = 0$ condition taking place. Thus one of two ways to explicitly take this condition into account must be utilised. If access to the solver code is available, a basis restriction routine (Jones 1995) can be implemented to ensure that these two variables are not allowed to take non-zero values simultaneously. Alternatively, a binary variable, say $s_1^{(1)}$, could be added to the formulation along with the following two constraints:

$$n_1^{(2)} - Ms_1^{(1)} \leq 0$$
$$p_1^{(2)} - M(1 - s_1^{(1)}) \leq 0$$

where M is a large positive constant. These two constraints will together enforce the required condition. A graphical example of a type 2 preference change is given by Fig. 4.5.

Type 3: Single Increase in Penalty (Discontinuity in Preference)

There may exist situations where sudden consequences occur at a certain level of deviation from the target. Examples of this are in logistics where items have to be sent in lorries and to exceed the target beyond a certain point would require an extra lorry. Figure 4.6 illustrates such a case with a per unit penalty combined with a discontinuous increase in penalty to represent the requirement of an extra transportation vehicle or storage warehouse. This discontinuous increase occurs with every new lorry that is required. Figure 4.7 illustrates another possibility, the case in which there is no per unit penalty, just a penalty at regular intervals to represent the cost of vehicle or warehouse hire.

Consider the third and fourth strategic production objectives in the manufacturing example given in Chapter 3. They have the algebraic form

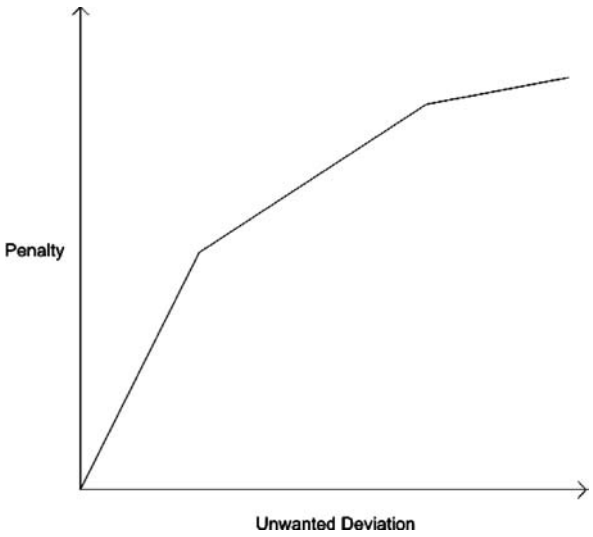

Fig. 4.5 Type 2 preference change – decrease in penalty

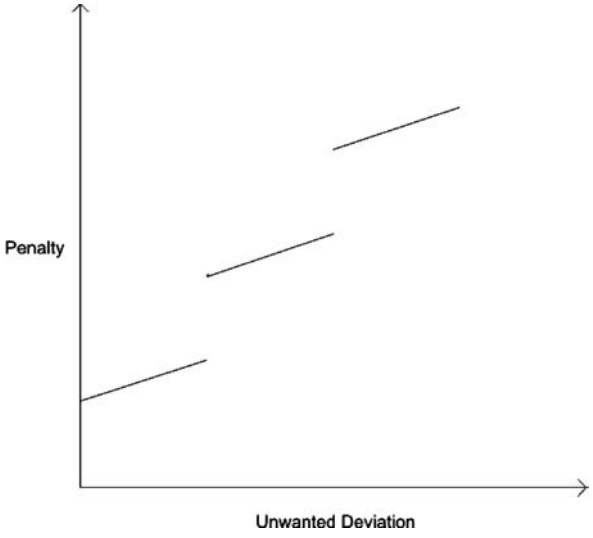

Fig. 4.6 Type 3 preference change (discontinuity) combined with per unit penalty

$$x_1 + n_3 - \underline{p_3} = 40$$
$$x_2 + n_4 - \underline{p_4} = 40$$

with achievement function contribution $\frac{n_3}{40} + \frac{n_4}{40}$.

Suppose the company now wishes to refine their strategic aims to give the preference structure of the type shown graphically by Fig. 4.7. That is no per unit penalty but an immediate dissatisfaction level of 50% if the target level of 40 units is not met

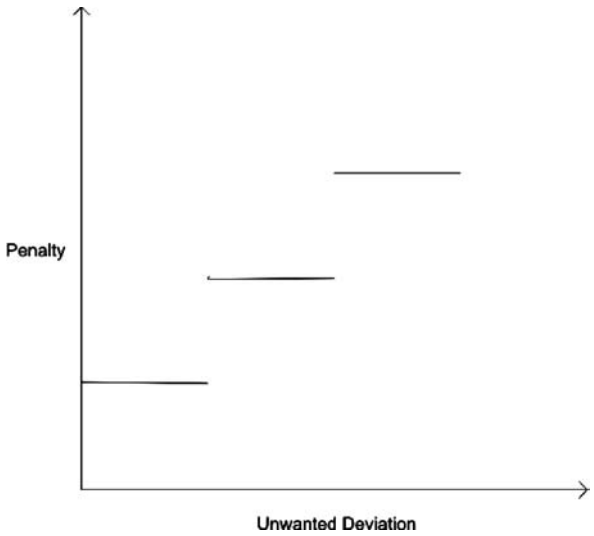

Fig. 4.7 Type 3 preference change (discontinuity) with no per unit penalty

increasing to 100% if less than 30 units are made. This requires the use of binary variables and will result in a binary goal programme. Introducing the variables

$$s_3^{(1)} = \begin{cases} 0 & if\ x_1 \geq 40 \\ 1 & if\ x_1 < 40 \end{cases} \quad s_3^{(2)} = \begin{cases} 0 & if\ x_1 \geq 30 \\ 1 & if\ x_1 < 30 \end{cases}$$
$$s_4^{(1)} = \begin{cases} 0 & if\ x_2 \geq 40 \\ 1 & if\ x_2 < 40 \end{cases} \cdots s_4^{(2)} = \begin{cases} 0 & if\ x_2 \geq 30 \\ 1 & if\ x_2 < 30 \end{cases}$$

allows the replacement of the two goals by the following constraints:

$$x_1 + Ms_3^{(1)} \geq 40$$
$$x_1 + Ms_3^{(2)} \geq 30$$
$$x_2 + Ms_4^{(1)} \geq 40$$
$$x_2 + Ms_4^{(2)} \geq 30$$

where M is a sufficiently large positive constant (in this case $M = 40$ will suffice). The term $\frac{n_3}{40} + \frac{n_4}{40}$ should be replaced in the achievement function by $\frac{20s_3^{(1)}+20s_3^{(2)}}{40} + \frac{20s_4^{(1)}+20s_4^{(2)}}{40}$. This allows for the scaling of the dissatisfaction of a range between 0 and 1 unit with 0.5 units being added by the forced setting of $s_3^{(1)}$ or $s_4^{(1)}$ to 1 for less than 40 units per day made and a further 0.5 being added by the forced setting of $s_3^{(2)}$ or $s_4^{(2)}$ to 1 for less than 30 units per day made. Notice that just as in the Jones−Tamiz system of increase of penalty, the effect of the penalty increases is cumulative.

If the company's preference scheme would have been to include a per unit penalty as well as the sudden increase in penalty then both the goals and the constraints must be included. The achievement function contribution (assuming the wish to keep the scaling between zero and one) becomes $\frac{n_3}{80} + \frac{n_4}{80} + \frac{20s_3^{(1)}+20s_3^{(2)}}{80} + \frac{20s_4^{(1)}+20s_4^{(2)}}{80}$.

Type 4: Non-linearity

The final case in the Jones−Tamiz framework differs from the first three in that it applies to a range rather than a single point in the criterion scale. The objective function $f_i(x)$ is non-linear over this range and therefore linear programming based techniques cannot be immediately applied. Jones and Tamiz recommend that this situation is handled by applying a piecewise linear approximation (Williams, 1993) and then applying a series of preference changes of types 1–3 to convert the problem into an integer goal programme.

The greater the level of accuracy required in the approximation, the more preference changes are required and hence the larger the size of the resulting goal programme.

Model Growth

Adding non-standard preferences causes the size of the goal programme to grow as they require additional constraints, variables, and integer variables. This can result in longer computational solution times, particularly if large numbers of preference types 2, 3, and 4 are added because they require binary variables. Table 4.1 gives the dimensions of the model growth for the different preference types.

Table 4.1 Model growth under Jones–Tamiz system

Preference (n of)	Constraints added	Variables added	Binary variables added
Type 1	n	$2n$	0
Type 2	$4n$	$4n$	$2n$
Type 3 ($n \leq 2$)	$2n + 1$	$n + 1$	$n + 1$
Type 3 ($n \geq 3$)	$4n - 3$	$n + 1$	$n + 1$

Objective Bounds

If, on a criterion, the deviation exceeding a certain value will cause the decision maker to reject the final solution this is treated as a rigid constraint. This case is termed an objective bound. An appropriate upper bound is then added to the relevant deviational variable. In the case where no deviational variables exist for the criterion

the extra constraint $f_i(x) \leq b_{\text{HIGH}}$ or $f_i(x) \geq b_{\text{LOW}}$ is added to the model, depending on whether we are concerned with negative or positive deviations. For discontinuous criteria, these bounds can often be implicitly specified in the choice of value M.

4.2.1 Interval Goal Programming

The method of interval programming is one of the earliest means of expanding the range of available preference structures in goal programming. Charnes and Collomb (1972) proposed to relax the condition that a single goal target value should be specified. Instead they allow the decision maker to choose an interval which is satisfactory and penalise deviations from either end of this interval. This approach is termed goal interval programming. In algebraic terms it involves converting the single target point b_i. into the target range $[b_{\text{LOWER}}, b_{\text{UPPER}}]$. This involves replacing the two-sided goal

$$f_i(x) + \underline{n_i} - \underline{p_i} = b_i$$

with the following two single-sided goals:

$$f_i(x) + \underline{n_i^L} - p_i^L = b_{\text{LOWER}}$$
$$f_i(x) + n_i^U - \underline{p_i^U} = b_{\text{UPPER}}$$

4.2.2 Other Paradigms for Modelling Non-standard Preferences

Since the interval goal programming model of Charnes and Cooper, there have been various suggestions for modelling non-standard preferences. Most of the frameworks prior to the Jones–Tamiz framework described above concentrated on the type 1 (penalty function) preference change, the earliest example of which is formulated by Charnes et al. (1976) in the context of resource allocation for a marine environmental problem. Martel and Aouni (1990) provide the first global preference change framework in 1990 with their adaptation of the Promethee discrete multi-criteria method (Brans et al., 1985) to the goal programming format. This framework is valuable in providing linkages between the discrete multi-criteria approaches and goal programming but does require more binary variables and linearisations than the Jones–Tamiz approach.

More recently, Chang (2006) proposed a method which is claimed to improve on the efficiency of the Jones–Tamiz approach. However, the proposed method only deals with type 1 (increasing penalty) preference changes. The two formulations given in the paper deal with an increasing penalty case for a left-sided goal (penalise negative deviations) and a right-sided goal (penalise positive deviations) rather than the type 1 (increasing penalty) and type 2 (decreasing penalty) claimed. Type 3

preference changes (discontinuities) are not mentioned and so it is assumed that the author intends to deal with these in the same manner as Jones–Tamiz. The method proposed by Chang for handling penalty functions imposes bounds on the deviational variables. In this sense it is similar to that of Can and Houck's approach in the context of reservoir planning (Can and Houck, 1984, as shown by Fig. 4.3). In fact, Romero (1991, p. 83) proposes a Can and Houck formulation of the same underlying goal programming as used by Chang (2006). The differences are that Romero's version uses negative deviational variables rather than slack and surplus variables, has a goal at 140 rather than 130, and slightly different weights and objective bounds. The (Can and Houck, 1984, Chang 2006) method is certainly more efficient in handling models with solely type 1 preference changes, especially with regard to constraints added; however, it has not yet been shown to be extendable to the full range of preference structures covered by the Jones and Tamiz (1995) framework.

4.3 Extended Lexicographic Goal Programming

The extended lexicographic goal programming (ELGP) was introduced by Romero (2001) with the purpose of providing a general framework which covers and allows the combination of the most common goal programming variants. It also encompasses several other distance-based MCDM techniques. This work is further extended by Romero (2004) who provides a more generalised form of the achievement function and by Arenas et al. (2004) who extend the framework to include fuzzy models.

The ELGP model is given in its general algebraic form as

$$\text{Min } a = \begin{bmatrix} \left(\alpha_1\lambda_1 + (1-\alpha_1)\left\{\sum_{i=1}^{q}(u_i^1 n_i^1 + v_i^1 p_i^1)\right\}\right),\ldots, \\ \left(\alpha_l\lambda_l + (1-\alpha_l)\left\{\sum_{i=1}^{q}(u_i^l n_i^l + v_i^l p_i^l)\right\}\right),\ldots, \\ \left(\alpha_L\lambda_L + (1-\alpha_L)\left\{\sum_{i=1}^{q}(u_i^L n_i^L + v_i^L p_i^L)\right\}\right) \end{bmatrix}$$

subject to

$$\begin{aligned} &\alpha_l(u_i^l n_i^l + v_i^l p_i^l) \le \lambda_l \quad && l = 1,\ldots,L \\ &f_i(x) + n_i - p_i = b_i && i = 1,\ldots,q \\ &n_i, p_i \ge 0 && i = 1,\ldots,q \end{aligned}$$

where, in common with the notation introduced in Chapter 2, unwanted deviations are given a positive weight and deviations which are not desired to be minimised are given a zero weight in the achievement function.

The ELGP formulation allows for the inclusion and combination of all of the underlying philosophies discussed in Chapter 2. The lexicographic ordering philosophy is available via the priority structure of the achievement function. The satisfying philosophy is evident in the set of goals. The optimising philosophy can be achieved through the use of Pareto efficiency detection and restoration techniques discussed in Chapter 6 or, in the case of a single priority level ($L = 1$), through the setting of sufficiently high target goal values. The balancing philosophy is achieved through the inclusion of the maximal deviation terms (λ_l) in each priority level. Furthermore, the balance between optimisation (efficiency) and balance (equity) can be controlled at each priority level through the parameter α_l which can be varied between complete emphasis on optimisation ($\alpha = 0$) and complete emphasis on balance ($\alpha = 1$). The ELGP framework is therefore a comprehensive tool for the inclusion of all relevant types of underlying philosophies into the goal programming framework.

Considering the example from Chapter 3, suppose that, having examined the weighted and Chebyshev goal programming solutions given in Tables 3.3 and 3.9, we now wish to investigate the balance of efficiency versus equity in this model. We form the following single priority level extended goal programme:

$$\text{Min } a = a\lambda + (1-\alpha)\left(\frac{p_1}{120} + \frac{n_2}{7000} + \frac{p_3}{40} + \frac{p_4}{40}\right)$$

subject to

$$\frac{p_1}{120} \leq \lambda$$
$$\frac{n_2}{7000} \leq \lambda$$
$$\frac{n_3}{40} \leq \lambda$$
$$\frac{n_4}{40} \leq \lambda$$
$$4x_1 + 3x_2 + n_1 - \underline{p_1} = 120$$
$$100x_1 + 150x_2 + \underline{n_2} - p_2 = 7000$$
$$x_1 + \underline{n_3} - p_3 = 40$$
$$x_2 + \underline{n_4} - p_4 = 40$$
$$2x_1 + x_2 \geq 50$$
$$x_1 + x_2 \leq 75$$
$$x_1, x_2 \geq 0,\ n_q, p_q \geq 0 \text{ q} = 1,...,4$$

Solving this model for intervals of 0.2 between the end-points of $\alpha = 0$ and $\alpha = 1$ leads to the solutions given in Table 4.2. These show the trade-off between efficiency and equity for the non-lexicographic variants in this example.

Table 4.2 Solutions of the extended goal programming model

Point	α	x_1	x_2	Obj1	Obj2	Obj3	Obj4
A	1.0	24	24	168	6250	24	24
A	0.8	24	24	168	6250	24	24
B	0.6	20	33.33	180	7000	20	33.33
B	0.4	20	33.33	180	7000	20	33.33
C	0.2	10	40	160	7000	10	40
C	0.0	10	40	160	7000	10	40

In this small example the analysis at intervals of $\alpha = 0.2$ has produced a single intermediate point B at $x_1 = 20$, $x_2 = 33.33$. The points found in the efficiency–equity trade-off analysis are graphically illustrated by Fig. 4.8.

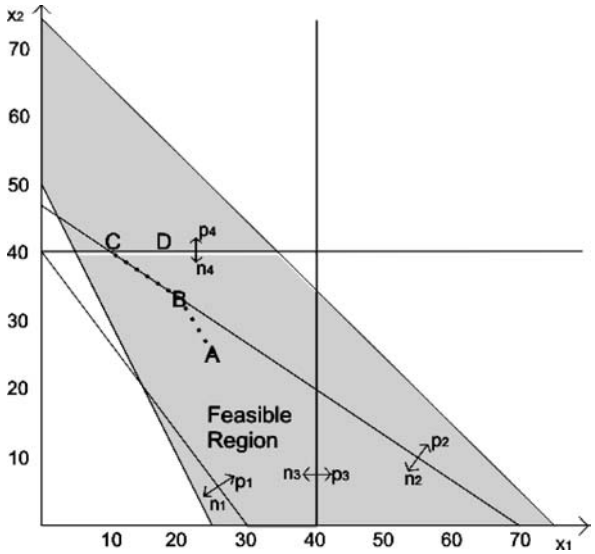

Fig. 4.8 ELGP efficiency–equity trade-off analysis for example

4.4 Meta-goal Programming

The meta-goal programming framework is proposed by Rodríguez et al. (2002) as a means of allowing the decision maker more flexibility in expressing their preferences by means of setting **meta-goals**. These can be thought of as secondary goals derived from the original set of goals. They can be achieved by means of a lexicographic or a weighted structure as deemed appropriate by the decision maker. The three types of meta-goals proposed are

Type 1: a meta-goal relating to the percentage sum of unwanted deviations
Type 2: a meta-goal relating to the maximum percentage deviation
Type 3: a meta-goal relating to the percentage of unachieved goals.

Considering these three types of meta-goals from the perspective of underlying distance metrics allows for an understanding of the philosophy of this method. The type 1 meta-goal has an L_1 underlying metric whereas the type 2 meta-goal has an L_∞ underlying metric. The type 3 meta-goal has an L_0 underlying metric of the type sometimes found in classification models, with a binary value of 1 if the goal is satisfied and 0 otherwise. Thus the meta-goal programming framework is valuable in allowing decision maker(s) to explore their preference structure without having to commit beforehand to a specific distance metric or philosophy. The following example illustrates the use of meta-goal programming for the production planning model developed in Chapter 3.

Example

Consider the weighted percentage model of the manufacturing model given in Chapter 3. Now suppose the company has the following three meta-goals:

- MG1: The percentage maximum deviation from all goals should be at most 50%.
- MG2: The maximum percentage deviation from any goal should be at most 30%.
- MG3: At most two of the four goals should be unsatisfied.

The negative and positive deviations from the ith meta-goal will be denoted by α_i and β_i, respectively. The total percentage deviation is given by $\frac{p_1}{120} + \frac{n_2}{7000} + \frac{n_3}{40} + \frac{n_4}{40}$ and therefore the first meta-goal takes the format

$$\frac{p_1}{120} + \frac{n_2}{7000} + \frac{n_3}{40} + \frac{n_4}{40} + \alpha_1 - \underline{\beta_1} = 0.5$$

The second meta-goal follows a Chebyshev type of Minmax optimisation. It is therefore necessary to denote the maximum deviation from any of the four objectives by λ. The following constraints must be added to the model:

$$\frac{p_1}{120} \leq \lambda$$
$$\frac{n_2}{7000} \leq \lambda$$
$$\frac{n_3}{40} \leq \lambda$$
$$\frac{n_4}{40} \leq \lambda$$

Our second meta-goal is therefore expressed as

$$\lambda + \alpha_2 - \underline{\beta_2} = 0.3$$

To model our third meta-goal, techniques from integer programming are needed (Williams, 2009). The following binary variables are defined:

$$y_i = \begin{cases} 1 & \text{If goal i not satisfied} \\ 0 & \text{otherwise} \end{cases} \quad i = 1,...,4$$

The following constraints impose the required conditions

$$\begin{aligned} p_1 - My_1 &\leq 0 \\ n_2 - My_2 &\leq 0 \\ n_3 - My_3 &\leq 0 \\ n_4 - My_4 &\leq 0 \end{aligned}$$

where M is a large enough positive constant (in this example the value $M = 9999$ will suffice). A quick inspection of these goals shows that if the goal is not satisfied the penalised deviational variable will take a positive value. According to the constraints this can only happen if the associated binary variable (y_i) takes the value one. Otherwise if $y_i = 0$ then the penalised deviational variable is constrained to take the value zero and the goal must be satisfied.

The number of goals not satisfied is therefore given by $\sum_{i=1}^{4} y_i$ and the third meta-goal can be expressed as

$$\sum_{i=1}^{4} y_i + \alpha_3 - \underline{\beta_3} = 2$$

Having formulated the three meta-goals a meta-achievement function must now be designed to combine the unwanted deviations from these goals. Suppose that a weighted formulation is desired and the second meta-goal is estimated to be of twice as important as the other two goals. Again, percentage normalisation will be used in order to scale the deviations from the meta-goals. These considerations lead to the following meta-goal programme:

$$\text{Min } a = \frac{\beta_1}{0.5} + \frac{2\beta_1}{0.3} + \frac{\beta_3}{2}$$

subject to

$$4x_1 + 3x_2 + n_1 - \underline{p_1} = 120$$
$$100x_1 + 150x_2 + \underline{n_2} - p_2 = 7000$$
$$x_1 + n_3 - \underline{p_3} = 40$$
$$x_2 + n_4 - \underline{p_4} = 40$$
$$2x_1 + x_2 \geq 50$$
$$x_1 + x_2 \leq 75$$
$$\frac{p_1}{120} + \frac{n_2}{7000} + \frac{n_3}{40} + \frac{n_4}{40} + \alpha_1 - \underline{\beta_1} = 0.5$$
$$\lambda + \alpha_2 - \underline{\beta_2} = 0.3$$
$$\sum_{i=1}^{4} y_i + \alpha_3 - \underline{\beta_3} = 2$$
$$p_1 - My_1 \leq 0$$
$$n_2 - My_2 \leq 0$$
$$n_3 - My_3 \leq 0$$
$$n_4 - My_4 \leq 0$$
$$\frac{p_1}{120} \leq \lambda$$
$$\frac{n_2}{7000} \leq \lambda$$
$$\frac{n_3}{40} \leq \lambda$$
$$\frac{n_4}{40} \leq \lambda$$
$$x_1, x_2 \geq 0;\ n_i, p_i \geq 0\ \ i = 1,...,4;\ y_i = 0\, or\, 1\ i = 1,...,4;$$
$$\alpha_i, \beta_i \geq 0\ \ i = 1,...,3,\ \lambda \geq 0$$

which leads to solution point D in Fig. 4.8, given as $x_1 = 17.14$, $x_2 = 40$. This point combines aspects of all three meta-goals. It is close to the weighted solution (point C) which corresponds to meta-goal 1. It has, however, moved towards a balance between the two unsatisfied goals (1 and 3) due to the influence of meta-goal 2. It has also remained on the $x_2 = 40$ line, ensuring that two goals are satisfied, due to the influence of meta-goal 3. The more importance given to the second meta-goal, the more the solution will be of a balanced Chebyshev nature. The more importance given to the first and third meta-goals, the more the solution will tend to an optimising approach.

Thus the meta-goal programming model can be seen as providing a new dimension to goal programming, that of combination of and trade-off between underlying distance metrics and hence philosophies. This, however, does come at the expense of a more complex algebraic model and in the case of the type 3 meta-goal, the addition of a number of binary variables equal to the number of goals, which may add to the solution time for problems with a large number of goals.

A further development of meta-goal programming is the concept of interactive meta-goal programming (Caballero et al., 2006). This allows the decision maker to interactively explore payoff information regarding the meta-goals and change priority structures in order to learn more of the nature of the problem and find a solution that best satisfies their preferences.

4.5 Weight Space Analysis

Critical to providing solutions to goal programmes that are of practical use to the problem owner is the accurate elicitation and exploration of their preferences. In the weighted and Chebyshev variants this is controlled by the set of preference weights assigned to the penalisation of unwanted deviations. The basics of this weight assignment are described in Section 3.6. In that section it is emphasised that the initial weight set chosen is just the start of the weight elicitation process rather than a single fait accompli declaration. This involves an exploration of the t-dimensional **weight space** (where t is the number of unwanted deviations to be penalised). Each potential weighting scheme $(w_1, w_2, ..., w_t) \in R^t$ is a point in weight space. It is also important to note that each weight is relative rather than absolute. That is, the weights only have meaning when considering their relative magnitude with respect to the other weights. For this reason, many authors choose to impose the condition $w_1 + w_2 + ... + w_t = 1$, which has the effect of normalising the weight vector and allowing it to be viewed in $R^{t\text{-}1}$ space. Figures 3.5 and 4.9 give examples of graphical illustration of weight space viewed in $R^{t\text{-}1}$.

In order to progress with the weighting process we first must consider what information the decision maker is looking for from the goal programming process. There are several possibilities that are listed below:

1. A complete analysis of the weight space assuming no a priori knowledge
2. A report on a restricted portion of the weight space after obtaining information about portions of the decision, objective, or parameter space that are of interest to the decision maker(s)
3. A report on the sensitivity of a particular solution chosen with regard to the decision maker's preferences
4. A small set of stable, sufficiently different (in decision or objective space) solutions from which a decision-making committee or group could make a decision
5. A formal or informal interactive process allowing decision maker(s) to move between neighbouring solutions in order to investigate the parameter space and hence arrive at a solution in line with their preferences

All of the above points have the common factor that some type of systematic exploration of the weight space must be conducted. This section proposes a general heuristic that will perform such an exploration. It is suitable for generating sets of

solutions which can be used to provide information for points 1–4 above. Point 5 type analysis is analysed in the section on interactive methods in Chapter 7.

The heuristic developed in its *t*-unwanted deviational variable form is outlined below. The input is an initial set of weights which could be the equal weight solution (as in the example), a decision maker estimate, or a weight vector generated by a pairwise comparison method such as the AHP. The output is a set of solutions S each of which has an associated set of weights, decision variable values, and objective values. The parameters to be set are TMax which dictates the maximum number of deviation weights to be varied simultaneously and max_level which dictates the level of granularity in the exploration of the weighting space.

```
Select initial starting point
Let S=ø
Let w=Initial set of weights
Add [Solve WGP(w)] to S
For n=1 to TMax
For each subset T* of deviations of cardinality n
    Form new vector w* by:
    Share 1-ε of the weight of w* equally amongst the deviations in T*
    Share ε of the weight of w* equally amongst the deviations not in T*
    If [Solve WGP(w)] is not in S then add [Solve WGP(w)] to S
    Let w=w_low, w*=w_up
    Examine_Weight_Line (w_low, w_up, 1)
End_For
Next n
End
Subroutine Examine_Weight_Line (w_low, w_ up, level)
    If [Solve WGP(w_low)]=[Solve WGP(w_up)] then EXIT
    If level>max_level then EXIT
    Form a new weight vector w_mid by setting the
    weight of each deviation to (w_low + w_up)/2
    If x* of [Solve WGP(w_mid)] is not in S then add [Solve WGP(w_mid)] to S
    Examine Weight Line (w_low, w_mid, level+1)
    Examine Weight Line (w_mid, w_up,level+1)
End
```

where [Solve WGP(w)] is a sub-procedure that solves the underlying weighted goal programme applying a weight set w to the unwanted deviations in the achievement function. Advice on solving weighted goal programmes is given in Chapter 5.

An example of three-dimensional weight space and the 25 possible points investigated by the above heuristic given the initial input of the equal weights solution (0.333, 0.333, 0.333) and the parameters TMax=2 and max_level=2 is given by Fig. 4.9.

Note that in practice it is not a good idea when conducting weight sensitivity analysis to set the weight of an unwanted deviational variable to zero because of

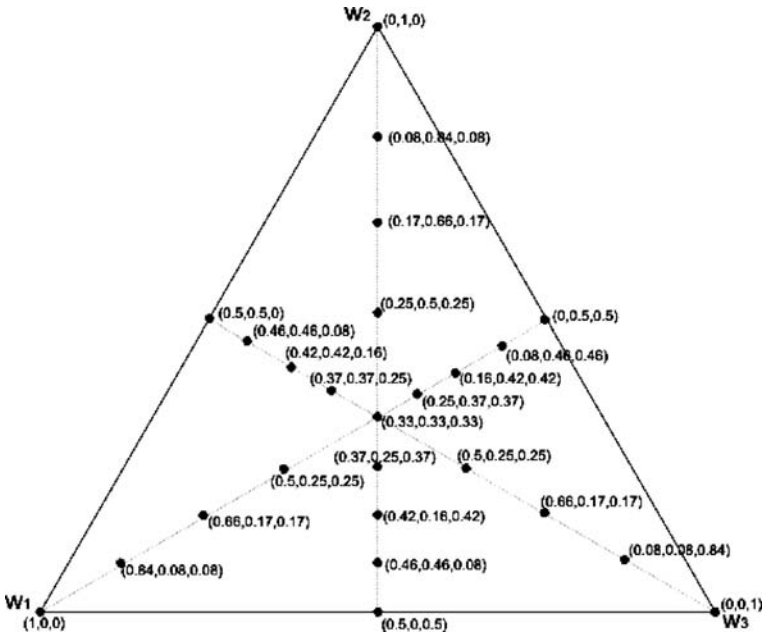

Fig. 4.9 Points in three-dimensional weight space generated by heuristic

the possibility of generating Pareto-inefficient solutions, as discussed in Chapter 6. Therefore any such weighting schemes (e.g. those on the outer edge of the weight space triangle in Fig. 4.9) are given a small weight on those deviations that would otherwise have zero weight. For example the weighting vector $\mathbf{w} = (0,1,0)$ would be modified to $\mathbf{w} = (0.005,0.99,0.005)$. This is evident from Table 3.8 of solutions to the example in Chapter 3.

4.6 Exercises

The following refer to extensions to the exercises given in Chapter 3 to include the topics of non-standard preferences, meta-goal programming, extended lexicographic goal programming, and weight space analysis:

(1) For Example 1 (Conversion from Linear to Goal Programme) from Chapter 3,

   (a) Graphically represent the weight space as an equilateral triangle and add the weighting scheme given in part (i) ($w_1 = 0.25$, $w_2 = 0.5$, $w_3 = 0.25$).
   (b) Add all the points that will be potentially investigated by the weight space analysis heuristic given in Section 4.5, with the starting weight point ($w_1 = 0.25$, $w_2 = 0.5$, $w_3 = 0.25$) and the parameters max_level=2 and TMax=2 to your graph.

(2) For Example 2 (On-line Retailer) from Chapter 3, extend the lexicographic goal programme formulated in part (b) into an extended goal programme with a parameter to control the mix between efficiency and equity in the second priority level.
(3) For Example 3 (Production Planning) from Chapter 3, formulate an algebraic model of this example with a revised, non-standard preference structure given by Table 4.3.

Table 4.3 Preference structure for Example 3

Goal	Target level	Per unit penalty*	One-off penalty**
1	1000	–	500
2	1250	3	–
	1000	6	–
	500	–	∞ (i.e. an objective bound)
3	0	2	100
	500	4	200
4	300	1	–
	200	2	
	100	1	

*Indicates the non-cumulative per-unit of deviation penalty to be applied to the unwanted deviation.
**Indicates a single penalty to be applied if the goal is not achieved at the stated target value.

(4) For Example 4 (Employee Scheduling) from Chapter 3, use the weight space analysis algorithm given in Section 4.4 to produce a table of solutions defining the weight space. Use the solution to part (ii) of the original example as the starting solution, and set the parameters to TMax=3, max_level=3.
(5) For Example 5 (Diet Planning) from Chapter 3, use the weight space analysis algorithm given in Section 4.4 to produce a table of solutions defining the weight space. Use the solution to part (3) of the original example as the starting solution, and set the parameters to TMax=2, max_level=2.

   (a) Comment on the performance of the weight search algorithm in exploring the weight space for this model.
   (b) If you had to recommend a subset of three possible diets to the patient which ones would you recommend and why?

(6) For Example 6 (Travelling Salesperson) from Chapter 3, extend the weighted goal programming into a meta-goal programme with the following three equally weighted meta-goals:

- MG1: The maximum percentage deviation from all goals should be at most 50%.
- MG2: The maximum percentage deviation from any goal should be at most 15%.
- MG3: At most one goal should be unsatisfied.

(7) For Example 7 (Downstream Oil Industry) from Chapter 3, introduce the non-standard preferences given in Table 4.4 into the weighted model:

Table 4.4 Preference structure for Example 7

Goal	Target level	Per unit penalty*	One-off penalty**
1	1,000,000	1	–
	1,250,000	2	–
	1,500,000	4	–
2	75%	1	10
	60%	2	30
	50%	4	100
3 (upper)	90%	1	50
	95%	1	100
3 (lower)	50%	1	
	40%	2	
	25%	0.5	

*Indicates the non-cumulative per unit of deviation penalty to be applied to the unwanted deviation.
**Indicates a single penalty to be applied if the goal is not achieved at the stated target value.

Goal 4 has a standard preference structure. Give the full algebraic formulation of the resulting integer goal programme.

(8) For Example 8 (Macro Economics) from Chapter 3, extend the weighted goal programming into a meta-goal programme with the following three meta-goals:

- MG1: The percentage maximum deviation from all goals should be at most 40%.
- MG2: The maximum percentage deviation from any goal should be at most 20%.
- MG3: At most one goal should be unsatisfied.

Their relative importance is given in Table 4.5.

Table 4.5 Meta-goal weights for Example 8

Meta-goal	Weight
MG1	4.0
MG2	2.0
MG3	1.0

(9) For Example 9 (Budget Planning) from Chapter 3,

(a) Formulate algebraically a (non-preemptive) extended goal programming model of this situation.
(b) Investigate the trade-off between equity and efficiency in this example by varying the parameter α in steps of 0.1 between $\alpha = 0$ and $\alpha = 1$.

(10) For the lexicographic model formulated in part (ii) of Example 10 (Healthcare Planning) from Chapter 3,

(a) Give the algebraic form of the extended lexicographic goal programme that allows for a mix of equity and efficiency to be investigated at each priority level.
(b) Analyse this model in the context of the four axioms needed for mathematical programming to be applied. What action should be taken in the case of each of the axioms not holding?

Chapter 5
Solving and Analysing Goal Programming Models

The purpose of this chapter is to explain how to 'effectively' solve and analyse goal programming models. The majority of the chapter will consider the solution and analysis of the three major goal programming variants using computerised software. The theory and methods of goal programming solution will also be detailed.

5.1 Computerised Solution of Weighted Goal Programming Example

Algebraically speaking, the linear weighted goal programming is the simplest variant to solve as it can be solved as a linear programme. This section will detail solution of the weighted example formulated in Chapter 3 via the widely used Microsoft Excel Solver add-on and the LINGO (LINDO, 2009) package as an example of a mathematical programming modelling and solution language.

5.1.1 Solution via Excel Solver

The spreadsheet for use with Excel Solver is labelled 'Book Example Weighted.xlsx' in the accompanying website (www.mopgp.com). Figure 5.1 presents the spreadsheet for the weighted version with percentage normalisation applied and an equal weighting scheme.

The preferential weights are given in the 'Pref Weight' row and then the Excel formulae in the 'norm weight' row convert them into normalised weights. The target level b_q and achieved ($f_q(x)$) values can be read from the columns on the right of the spreadsheet. This format allows a quick comprehension of the level of achievement over the set of goals to take place. The values of the decision and deviational variables can be read from the ''variables' row at the bottom of the spreadsheet.

The full coefficient matrix of the problem is manually inputted in the large box in the centre of the spreadsheet. This is not a problem in this case as the model is small. For larger models, the power of Excel in handling data can be useful, with data being able to be stored in other spreadsheets and accessed from other

D. Jones, M. Tamiz, *Practical Goal Programming*, International Series in Operations Research & Management Science 141, DOI 10.1007/978-1-4419-5771-9_5,

	Pref weight			1	1	1	1						
	x1	x2	n1	n2	n3	n4		p2	p3	p4			
Norm weight			0	0.000143	0.025	0.025	0.00833	0	0	0			
												Target	Achieved
Hours	4	3	1				−1				120	= 120	160
Profit	100	150		1				−1			7000	= 7000	7000
Product A	1				1				−1		40	= 40	10
Product B		1				1				−1	40	= 40	40
Const 1	2	1									60	>= 50	60
Const 2	1	1									50	<= 75	50
												Achievement Fn	
Variables	10	40	0	0	30	0	40	0	0	0		1.08	

Fig. 5.1 Excel spreadsheet of weighted goal programming variant

sources. The key pieces of data from the solution are also able to be brought together elsewhere to form a summary report that the decision maker(s) will find easier to comprehend. The widespread familiarity of practitioners with the Excel package is a further advantage when using Excel solver.

The access to Visual Basic via Excel Macros allows for the possibility of modelling assistance such as those described in Chapters 3, 4, and 6 and techniques such as automated weight sensitivity analysis to be encoded and accessed by the decision maker(s).

5.1.2 Solution via LINGO

The model file for use with LINGO is labelled 'Book Example Weighted' in the accompanying website (www.mopgp.com). Figure 5.2 gives the LINGO code of the model. This also uses percentage normalisation and is the equal weights solution. Figure 5.3 gives the corresponding LINGO results file.

The advantage of using a modern mathematical programming language such as LINGO is the ability to generate multiple sets of decision variables and constraints (such as those in the SETS section of Fig. 5.2). This allows larger models to be able to be expressed succinctly. Mathematical programming languages also include or interface with the powerful state-of-the-art solution packages that allow for the solution and analysis of larger scale models. Most mathematical programming modelling packages are now able to be called by or interface with other packages so automation of modelling techniques and weight sensitivity analysis is also a possibility. The solution can also easily be exported to a spreadsheet package for easier analysis. The learning curve for practitioners in programming in a mathematical programming language may, however, be greater than that of using Excel solver if they are unfamiliar with the syntax.

Fig. 5.2 LINGO model of weighted goal programming variant

```
MODEL:
SETS:
  DECISION_SET / 1..2 /: X;
  DEVIATION_SET / 1..4 / : N,P;
ENDSETS

DATA:

w1=1.0;
w2=1.0;
w3=1.0;
w4=1.0;

ENDDATA

MIN = w1*(1/120)*P(1) +
w2*(1/7000)*N(2)+w3*(1/40)*N(3)+w4*(1/40)*N(4);

4*X(1) + 3*X(2) + N(1) - P(1) = 120;
100*X(1) + 150*X(2) + N(2) - P(2) = 7000;
X(1) + N(3) - P(3) = 40;
X(2) + N(4) - P(4) = 40;
2*X(1) + X(2)>=50;
X(1) + X(2) <= 75;

END
```

5.2 Computerised Solution of Chebyshev Goal Programming Example

As described in Section 3.7, Chebyshev goal programmes can be solved as linear programmes provided the constraints and goals are linear and all the linear programming axioms outlined in Chapter 4 apply. This section details solution of the Chebyshev variant of the manufacturing model presented in Section 3.7 by the Excel Solver and LINGO packages. It should be noted that the right-hand sides of the four introduced constraints are all zero. This implies that there is degeneracy present in the linear programming model which may make it harder to solve than the corresponding weighted variant model. However, modern linear programming solution packages have advanced mechanisms for handling degeneracy.

5.2.1 Solution via Excel Solver

The spreadsheet for using Solver is labelled ‘Book Example Chebyshev.xlsx’ in the accompanying website (www.mopgp.com). Figure 5.3 gives the spreadsheet of the model’s solution, with percentage normalisation and the equal preferential weight solution. This corresponds to point D in Fig. 3.4.

Fig. 5.3 LINGO results file for weighted variant

Global optimal solution found at step: 9
Objective value: 1.083333

Variable	Value	Reduced Cost
W1	1.000000	0.0000000
W2	1.000000	0.0000000
W3	1.000000	0.0000000
W4	1.000000	0.0000000
X(1)	10.00000	0.0000000
X(2)	40.00000	0.0000000
N(1)	0.0000000	0.8333334E-02
N(2)	0.0000000	0.5952381E-04
N(3)	30.00000	0.0000000
N(4)	0.0000000	0.1250000E-01
P(1)	40.00000	0.0000000
P(2)	0.0000000	0.8333333E-04
P(3)	0.0000000	0.2500000E-01
P(4)	0.0000000	0.1250000E-01

Row	Slack or Surplus	Dual Price
1	1.083333	1.000000
2	0.0000000	0.8333334E-02
3	0.0000000	-0.8333333E-04
4	0.0000000	-0.2500000E-01
5	0.0000000	-0.1250000E-01
6	10.00000	0.0000000
7	25.00000	0.0000000

The minimum achieved value of λ is 0.4, which as percentage normalisation is used implies that the furthest deviations are 40% away from their targets. These are the unwanted deviational variables that have binding 'lambda' constraints. Figure 5.4 shows these to be p_1, n_3, and n_4. These are the critical deviational variables in terms of this solution. n_2 does not have a binding lambda constraint and hence is less critical. In fact it is only 14% below its target and hence could be reduced by a further 26% before it becomes critical, as shown by the '–0.26' value in its 'lambda' row.

5.2.2 Solution via LINGO

The model file for use with LINGO is labelled 'Book Example Chebyshev' in the accompanying website (www.mopgp.com). Its contents are given by Fig. 5.5 and the corresponding LINGO result file by Fig. 5.6. It also shows the solution with equal weights and percentage normalisation.

The same conclusions can be drawn as from the Excel Solver spreadsheet. The 40% maximum deviation from any target can be seen from the optimal value, which is the reported value of LAMBDA. The extra 26% from its target value by which n_2

	Pref weight			1	1	1	1								
	x1	x2	n1	n2	n3	n4	p1	p2	p3	p4	Lambda			Target	Achieved
Norm weight			0	0.000143	0.025	0.025	0.00833	0	0	0	1				
Hours	4	3	1				−1					120	=	120	168
Profit	100	150		1				−1				7000	=	7000	6000
Product A	1				1				−1			40	=	40	24
Product B		1				1				−1		40	=	40	24
Constraint 1	2	1										72	>=	50	72
Constraint 2	1	1										48	<=	75	48
Lambda			1				1				−1	0.00	<=	0	
Lambda				1				1			−1	-0.26	<=	0	
Lambda					1				1		−1	0.00	<=	0	
Lambda						1				1	−1	0.00	<=	0	
	24	24	0	1000	16	16	48	0	0	0	0.4		Min lambda		
													0.4		

Fig. 5.4 Excel spreadsheet of weighted goal programming variant

could be worsening before reaching the maximum deviation level of 40% is given by slack value of row 9. The slack values of 22 and 27 in rows 6 and 7 correspond to the fact that the two hard constraints do not bind the optimal solution.

5.3 Computerised Solution of Lexicographic Goal Programming Examples

5.3.1 Theory of Solving Lexicographic Goal Programmes

A lexicographic goal programme can be modelled as a series of mathematical programming models, with each model representing a priority level. Extra constraints on the unwanted deviational variables can be added at each priority level to ensure that the optimal values obtained at higher priority levels are not worsened. Using the notation introduced in Chapter 2 this constraint has the form

$$h_l\left(\underline{n}, \underline{p}\right) = l^*$$

where l^* is the optimal value achieved in the minimisation of the lth priority level. An example of this process in practice is given in Section 3.3 in the context of the manufacturing example presented in that chapter. This leads to a series of mathematical programming models that are larger in terms of number of constraints but smaller in terms of the feasible region. It is computationally efficient for this type of model to start each mathematical programming minimisation at the point found by the previous optimisation as this solution will already be feasible.

Fig. 5.5 LINGO model of Chebyshev variant

```
MODEL:
SETS:
  DECISION_SET / 1..2 /: X;
  DEVIATION_SET / 1..4 / : N,P;
ENDSETS

DATA:

w1=1;
w2=1;
w3=1;
w4=1;

ENDDATA

MIN = LAMBDA;


4*X(1) + 3*X(2) + N(1) - P(1) = 120;
100*X(1) + 150*X(2) + N(2) - P(2) = 7000;
X(1) + N(3) - P(3) = 40;
X(2) + N(4) - P(4) = 40;
2*X(1) + X(2)>=50;
X(1) + X(2) <= 75;
w1*(1/120)*P(1) - LAMBDA <=0;
w2*(1/7000)*N(2) - LAMBDA <=0;
w3*(1/40)*N(3) - LAMBDA <=0;
w4*(1/40)*N(4) - LAMBDA <=0;

END
```

A faster mathematical programming based solution method is given by Ignizio (1982) in which the variables with positive reduced cost at the optimal solution of each priority level are fixed to prevent them from taking a value above their lower bound (normally zero) in future priority level optimisations. This has the same effect as imposing constraints on the unwanted deviational variables but also reduces the problem in terms of number of variables and size of feasible region whilst keeping the number of constraints constant.

A third solution method introduced by Ignizio (1985) is to solve the multi-dimensional dual problem (Ignizio 1982) of the lexicographic goal programme. In this case, non-binding constraints from the dual model rather than variables from the primal model are removed on finding an optimal solution to each priority level. This has the effect of reducing the number of constraints rather than the number of variables in the model being solved and hence has the possibility of providing faster computational solution times.

Whatever solution method is used, it is easy to detect whether lexicographic redundancy, as discussed in Section 3.3.1, is occurring. This will be shown by the feasible region collapsing to a single point or by all the decision and deviational

Fig. 5.6 LINGO results file for Chebyshev variant

Global optimal solution found at step: 18
Objective value: 0.4000000

Variable	Value	Reduced Cost
W1	1.000000	0.0000000
W2	1.000000	0.0000000
W3	1.000000	0.0000000
W4	1.000000	0.0000000
LAMBDA	0.4000000	0.0000000
X(1)	24.00000	0.0000000
X(2)	24.00000	0.0000000
N(1)	0.0000000	0.2500000E-02
N(2)	1000.000	0.0000000
N(3)	16.00000	0.0000000
N(4)	16.00000	0.0000000
P(1)	48.00000	0.0000000
P(2)	0.0000000	0.0000000
P(3)	0.0000000	0.1000000E-01
P(4)	0.0000000	0.7500000E-02

Row	Slack or Surplus	Dual Price
1	0.4000000	1.000000
2	0.0000000	0.2500000E-02
3	0.0000000	0.0000000
4	0.0000000	-0.1000000E-01
5	0.0000000	-0.7500000E-02
6	22.00000	0.0000000
7	27.00000	0.0000000
8	0.0000000	0.3000000
9	0.2571429	0.0000000
10	0.0000000	0.4000000
11	0.0000000	0.3000000

variables being excluded from consideration if using Ignizio's primal algorithm. In this case all future priority levels should be marked as redundant.

5.3.2 Solution via Excel Solver

The spreadsheet for using Solver is labelled 'Book Example Lexicographic.xlsx' in the accompanying website (www.mopgp.com). Figure 5.7 gives the solution to the lexicographic model with priority scheme 1 in Section 3.3. That is with the achievement function

$$\text{Min } a = \left[(n_2), (n_3 + n_4), (p_1) \right]$$

	Pref	Weight		1	1	1	1							
	x1	x2	n1	n2	n3	n4	p1	p2	p3	p4				
Norm Weight			0	1	1	1	1	0	0	0			Target	Achieved
Hours	4	3	1				−1				120	=	120	260
Profit	100	150		1				−1			7000	=	7000	9500
Product A	1				1				−1		40	=	40	35
Product B		1				1				−1	40	=	40	40
Constraint 1	2	1									110	>=	50	110
Constraint 2	1	1									75	<=	75	75
Variables	35	40	0	0	5	0	140	2500	0	0				
	P1			1							0	=	0	
	P2				1	1					5	=	5	

P1 value	0.00
P2 value	5.00
P3 value	140.00

Fig. 5.7 Excel spreadsheet of lexicographic goal programming variant

The optimal values of the three priority levels can be read from the 'P Value' cells at the bottom right of the spreadsheet. Note that this spreadsheet can be used in a number of ways dependent on the size of the model and on whether the purpose of solution is didactic or not:

(1) The model can be solved a priority level at a time by adding a constraint of the form

$$h_l\left(\underline{n}, \underline{p}\right) = l^*$$

after each priority level and adjusting the target cell and constraint settings in the Excel Solver Parameters box. This equates to a semi-automated version of the first method listed in Section 5.3.1. It is best used only for smaller models for teaching purposes where a step-by-step demonstration of the process is required.

(2) The model can be solved in a semi-automated process by using Ignizio's primal algorithm instead. In this case the reduced costs could be found using the sensitivity analysis section for each priority level and those with positive reduced cost could be excluded from the decision and deviational variable (by changing cells) section in the Excel Solvers Parameters box. Care should be taken if this also

includes slack or surplus variables from hard constraints; those binding constraints with shadow prices should be prohibited from becoming non-binding in future priority levels by converting the constraint to equality form in the constraint settings in the Excel Solver Parameters box. In this example that implies excluding the variable n_2 from the second priority level optimisation and additionally excluding the variables p_3 and p_4 as well as changing constraint 2 to an equality ($x_1 + 2x_2 - 75$) for the third priority level optimisation. This approach is more computationally efficient and hence can be used for larger models. Again, it is semi-automated and hence should be used for teaching purposes.

(3) The model solution process for method (1) can be fully automated by adding an Excel Macro. This piece of Visual Basic code will need to use the SolverOK function at each priority level to point the target cell to be minimised at the appropriate priority level. The actual minimisation can be executed by use of the SolverSolve function. The Solveradd function should be used at the end of each priority level minimisation to add the constraint

$$h_l\left(\underline{n}, \underline{p}\right) = l^*$$

to future priority level minimisations.

(4) The primal algorithm of Ignizio can also be automated by using an Excel Macro. Here the SolverOK function should again be used to set the priority level and the SolverSolve function to solve but the ReportArray argument of the SolverFinish must be set to produce the sensitivity report. This can then be examined in order to find the positive reduced costs and shadow costs which can then be fixed using the SolverOK and SolverChange functions, respectively.

5.3.3 Solution via LINGO

Figure 5.8 gives the LINGO model used to solve the lexicographic model from Chapter 3 with the following priority structure:

$$\text{Min } a = \left[(n_2), (n_3 + n_4), (p_1)\right]$$

This uses Ignizio's primal algorithm with the priority levels being solved by manual selection of the priority level and restriction of variables. On solution of the first priority level it is found that variable n_2 has a positive reduced cost and hence it is prohibited from entering the basis by the addition of the @BND(0,X(2),0) statement. Upon selection of the second priority level the variables P_3, P_4 and the slack variable for the constraint $x_1 + x_2 \leq 75$ all have positive reduced costs. These are then restricted by the addition of the statements @BND(0,P(3),0), @BND(0,P(4),0) and the conversion of the constraint to an equality ($x_1 + x_2 = 75$). An intuitive understanding of how the restriction of these variables leads to the formation of the

```
MODEL:
SETS:
  XSET / 1..2 /: X ;
  DSET / 1..4 / : N,P;
ENDSETS

![PRI1] MIN = N(2);
![PRI2] MIN = N(3) + N(4);
[PRI3] MIN = P(1);

4*X(1) + 3*X(2) + N(1) - P(1) = 120;
100*X(1) + 150*X(2) + N(2) - P(2) = 7000;
X(1) + N(3) - P(3) = 40;
X(2) + N(4) - P(4) = 40;
2*X(1) + X(2)  >=50;
X(1) + X(2) = 75;
@BND(0,N(2),0);
@BND(0,P(3),0);
@BND(0,P(4),0);

END
```

Fig. 5.8 LINGO model of lexicographic variant

feasible region for the third priority level minimisation can be gained by referring to Fig. 3.3. The third priority level is then selected, leading to the LINGO file given by Fig. 5.8. The optimal solution to this priority level and hence the goal programme is given by the LINGO solution file to this optimisation file, which is shown by Fig. 5.9.

It can also be seen from the above process that not all variables were fixed in the process of solving the first two priority levels. Hence a non-singular feasible region

```
Global optimal solution found at step:          2
Objective value:                        140.0000

                    Variable           Value        Reduced Cost
                        X( 1)        35.00000           0.0000000
                        X( 2)        40.00000           0.0000000
                        N( 1)       0.0000000           1.000000
                        N( 2)       0.0000000          0.0000000
                        N( 3)        5.000000          0.0000000
                        N( 4)       0.0000000           1.000000
                        P( 1)        140.0000          0.0000000
                        P( 2)        2500.000          0.0000000
                        P( 3)       0.0000000          0.0000000
                        P( 4)       0.0000000          -1.000000

                         Row    Slack or Surplus      Dual Price
                        PRI3        140.0000           1.000000
                           2       0.0000000           1.000000
                           3       0.0000000          0.0000000
                           4       0.0000000          0.0000000
                           5       0.0000000           1.000000
                           6        60.00000          0.0000000
                           7       0.0000000          -4.000000
```

Fig. 5.9 LINGO results file for lexicographic variant

is available for the third and final priority level and thus lexicographic redundancy has not occurred in this example. The values of the first two priority levels are not recorded in this results file but they can be easily found by recording them as separate variables, recording them at the end of each priority level minimisation, or calculating them from the values of the deviational variables.

If many lexicographic goal programmes are to be solved then the user may wish to automate the process of priority level minimisation. This is possible via the LINGO DLL facility from a global control program written in, say, Visual Basic. Readers are referred to the LINGO users' guide (LINGO, 2009) for further details.

5.4 Solution of Other Goal Programming Variants

5.4.1 Fuzzy Goal Programmes

The fuzzy goal formulations introduced in Section 2.3.1 can be modelled and solved as linear programmes providing the standard assumptions for linear programming hold. This implies that the solution should be similar as for the standard weighted goal programme. However, it should be noted that extra constraints are added and this will lead to longer solution times, which will not be an issue for small- or medium-scale models on modern LP solvers but may become an issue for large-scale models, particularly if the model contains other factors that lead to computational challenges, such as integer variables or non-linearities.

5.4.2 Integer and Binary Goal Programmes

The additionally computational challenges of adding integer or binary variables to a goal programme are comparable to those of converting a linear programme to a binary or integer programme. Most of the modelling practices and solution methodologies that have been developed for integer and binary programmes apply to integer and binary goal programmes as well. A full description of these techniques is beyond the scope of this book but can be found in Williams (2009). It should be noted that adding integer or binary variables to a goal programming model can result in substantially increased solution times. This increase is somewhat unpredictable and obviously depends in part on the number of binary or integer variables added to the model and the number and type of goals and constraints. Williams (2009) gives advice on detecting the potential level of difficulty in solving the resulting model. Both the Excel Solver and LINGO packages detailed in this chapter have the capacity to solve binary and integer programming models.

If an integer goal programming solution cannot be found by mathematical programming methods in reasonable computational time then a further possibility is to resort to a heuristic or meta-heuristic method of solution. Common heuristics that

have been used for goal programming are detailed in Section 7.3.3. It should be noted that genetic algorithms work naturally with binary and integer variables and hence can be regarded as a 'good fit' for solving difficult binary or integer goal programmes, although the amount of work involved in setting up a problem-specific genetic algorithm and fine-tuning its parameters to produce good solutions should not be underestimated (Mirrazavi et al., 2001).

5.4.3 Non-linear Goal Programmes

The extra level of difficulty of solving a non-linear goal programme over a linear goal programme can be thought of as similar to the extra level of difficulty in solving a non-linear programme over a linear programme. Non-linearity can occur in the achievement function, goals, and/or constraints. Many specialised algorithms are available to solve non-linear programmes. A treatment of these is beyond the scope of this book but details from a multi-objective perspective can be found in Miettinen (1999). Both the Excel Solver and LINGO packages detailed in this chapter have the capacity to solve non-linear programmes but the computational time taken will depend on the nature of the non-linearity and size of the model. This may make obtaining an exact optimal solution in reasonable computational time impossible.

Thus linearisation techniques are one possibility to obtain an approximate solution if the degree of non-linearity is not too great. A further possibility is to use one of the heuristic or meta-heuristic techniques detailed in Section 7.3.3, although this will require greater investment of time in setting up a bespoke model and fine-tuning the model parameters.

5.4.4 Meta and Extended Lexicographic Goal Programmes

The meta-goal programming introduced in Section 4.4 is a binary goal programme from a computational point of view if type 3 meta-goals are present. Hence, the comments made regarding binary goal programmes in Section 5.4.2 apply. The other consideration is that a number of variables and constraints have to be added to the original model, which could be an issue for large-scale models as these additions will effect the solution time. Hence, meta-goal programmes are solvable by any mathematical programming package that handles binary variables.

The extended lexicographic goal programming model introduced in Section 4.3 is a combination of the lexicographic and Chebyshev goal programming variants from a computational perspective. The formulation at each priority level is still solvable as a linear programming, provided the linear programming assumptions hold. Hence, the solution methods discussed in Section 5.3 also apply to extended lexicographic goal programming models.

5.5 Analysis of Goal Programming Results

A key skill in the modelling and solution process for goal programming is that of extracting and interpreting information from the goal programming results files. This information can be read directly from output such as given in Fig. 5.1 or Fig. 5.3 or it can be processed into a separate Excel spreadsheet file. The exact information required of any goal programme is application dependent, but there are some common types of information available that are discussed in this section. It should also be noted that the analysis is rarely of a single goal programming solution. The results of many goal programme optimisations relating to different variants, weights, priority schemes, target levels, and constraint sets will need to be compared and contrasted in order to advise the problem owner of the nature of their problem and propose solution(s).

Some of the key generic measures of goal programming output are as follows:

(i) *The value of the decision variables*. These give the decision to be taken in terms of the actual decision space of the problem. They do not directly give information about the achievement of the goals but they are fundamental in helping the decision maker visualise the solution, raise potential difficulties, and ensure that it is implementable in practice.

(ii) *The achievement levels of the goals*. A key set of information is how close the achieved values are in comparison to the target values. Are the target values under-achieved, over-achieved, or met exactly? This information can be presented directly or as the **values of the deviational variables**. Table 3.1 is an example of a comparison of target and achieved values for a set of goals in tabular form. The deviations from the goals where targets are not met can be expressed either as a percentage or as an absolute value. The achievement level of the goals can be thought of as giving a more strategic view of the solution rather than the practical view given by the decision variable values.

(iii) *The relative balance between goals*. Section 1.2.4 details the balancing philosophy as important in goal programming. As well as looking at the absolute value of the goals, the problem owner may require a report on the relative balance amongst the goals. This is particularly important in multi-stakeholder situations where stakeholders have conflicting views as to the importance of the goals. It is often particularly useful to compare balance across the solutions obtained by different goal programming variants, normally to see how much extra balance is given by the use of the Chebyshev variant.

(iv) *The value of the achievement function*. This will be a single-valued function for the weighted and Chebyshev variants and a vector of priority level optimal values for the lexicographic variant. This can give information about the total level of deviation of goals, particularly when percentage normalisation has been employed. In the lexicographic case, it will show if all the goals in a priority level have been reached (in which case the optimal value will be zero).

(v) *The status of the constraints*. Similar to the linear programming case, the solution file will show the level of slack or surplus in any inequality constraints and

indicate which constraints are critical because they are binding, that is where there is no slack or surplus because the underlying resource is being fully used.

(vi) *Technical modelling information*. Information regarding the lexicographic redundancy of priority levels or the Pareto efficiency of the solution can also be extracted if such techniques have been coded via Excel Macros or by the use of specialist goal programming package as discussed in Section 5.6. This information will help guard against goal programming modelling errors.

5.6 Specialist Goal Programming Packages – Past and Future

An alternative to using a general mathematical programming package for solving goal programmes is to use a specialist goal programming package. Dedicated goal programming codes have been available at various times in the development of GP. Early algorithmic codes include those of Jääskeläinen (1976), Ignizio and Perlis (1979), and Ignizio (1985). Later works concentrated on bringing modelling and solution tools together in a single package and include the GPSYS system (Jones et al., 1998) and the MOPEN system (Caballero et al., 2005).

The advantages of a dedicated goal programming system include the opportunity to automate modelling processes such as normalisation, Pareto efficiency detection and restoration, lexicographic redundancy checking, and interactive methodologies. In addition, procedures can be implemented that allow automated solution by multiple variants and comparison of solutions for analysis. This has the potential of improving the uptake of these techniques and hence promoting good modelling practice. Also, dedicated goal programming packages are more likely to use specialist algorithms such as the Ignizio (1982, 1985) primal and dual algorithms for lexicographic models detailed in Section 5.3.1. Specialist goal programming speed-ups such as advanced basis creation and the NPSWAP algorithm (Jones, 1995) may also be present.

The disadvantage with dedicated goal programming systems to date is that they have mainly been academic enterprises developed by university staff rather than commercial software companies. Hence, it has been impossible for them to keep pace with the general rapid improvements in modelling and solution technologies that have taken place in the commercial mathematical programming software packages. Additionally, the OLE nature of most commercial packages means that the automation of modelling and analysis techniques can take place within the software environment itself or via an object-orientated language such as Visual Basic. Hence, it is envisioned that future dedicated goal programming systems will be based more around the implementation of automated analysis tools, multiple variant solution and comparison techniques, and automated weight search algorithms. These will then make multiple calls to a commercial mathematical programming package to provide a state-of-the-art solution tool. They should also be able to facilitate the 'trend of integration and combination' detailed in Chapter 8 by allowing an easy interface between goal programming and other computerised packages from other parts of the operational research and artificial intelligence disciplines.

5.7 Exercises

(1) Consider Example 1 (Conversion from Linear to Goal Programming) from Section 3.9 – *Exercises*. The Excel spreadsheet solution template is given by file Example 1.xls on the accompanying website (www.mopgp.com).

(i) The weighted goal programming spreadsheet currently implements solution with percentage normalisation. Modify the spreadsheet to implement solution by

- Euclidean normalisation
- Zero–one normalisation

Compare the three results obtained by the three different normalisation methods. What conclusions can be drawn about normalisation for this example?

(ii) Suppose the original linear programming constraints had to hold with equality. Modify the lexicographic goal programming spreadsheet to implement this change. Is the second priority level now redundant? Explain why/why not.

(2) Consider Example 2 (On-line Retailer) from Section 3.9. The Excel spreadsheet solution template is given by file Example 2.xls on the accompanying website (www.mopgp.com).

(i) Consider the weighted goal programming spreadsheet. Re-solve the model under the three following target-level and constraint-related scenarios:

- The need to reduce the number of pickers employed to 12
- A future increase in orders per hour to 600
- The need to employ at least five experienced pickers

(ii) Repeat the exercise of part (i) for the Chebyshev goal programming spreadsheet.

(iii) Write a report to the management of the supermarket explaining the effects of the scenarios and comparing and contrasting the solutions obtained by weighted and Chebyshev goal programming.

(3) Consider Example 3 (Production Planning) from Section 3.9. The Excel spreadsheet solution template is given by file Example 3.xls on the accompanying website (www.mopgp.com).

(i) Consider the lexicographic goal programming spreadsheet. Modify the priority structure to investigate the effect of the following proposed priority schemes on the solution:

- Priority 1, Priority 3, Priority 2, Priority 4
- Priority 1, Priority 2, Priority 4, Priority 3
- Priority 2, Priority 1, Priority 3, Priority 4

Analyse the effects of these priority changes from a managerial perspective.

(ii) Consider the Chebyshev goal programme spreadsheet. Analyse the balance of achievement amongst the four goals, comparing it with the level of balance achieved by the weighted goal programming variant. Which goals have seen the most improvement when changing from the weighted to the Chebyshev variant?

(4) Consider Example 4 (Production Planning) from Section 3.9. The Excel spreadsheet solution template is given by file Example 4.xls on the accompanying website (www.mopgp.com).

(i) Consider the weighted goal programming spreadsheet. Investigate the alternative weighting schemes given in Table 5.1.
Show graphically the effect that the four weighting schemes have on the achievement of the four goals.

(ii) Repeat the exercise of part (i) for the Chebyshev variant.

(5) Consider Example 5 (Diet Planning) from Section 3.9. The Excel spreadsheet solution template is given by file Example 5.xls on the accompanying website (www.mopgp.com).

(i) Consider the first lexicographic formulation (given by sheet LGP1). Perform sensitivity analysis on the daily cost by changing the target value in intervals of 50 cents from $2 to $4.50. Record the differences in the actual diet and the extent to which the patients' preferences are met as the allowable cost increases.

(ii) Consider the second lexicographic formulation (given by sheet LGP2). Add a further constraint to the formulation that ensures that a maximum of three portions of any food should be eaten by the patient daily. Investigate how it affects the achievement of the goals.

Table 5.1 Alternative weighting schemes for Example 4

Goal	Original	Weight 1	Weight 2	Weight 3
Ensure that each interval has at least the estimated number of workers	0.65	0.45	0.3	0.3
Ensure the total cost of the workers is no more than £280	0.2	0.15	0.3	0.5
Minimise the number of workers employed above the estimate for each interval	0.1	0.15	0.2	0.1
Minimise the number of workers starting their shift at 03:00 hours	0.05	0.25	0.2	0.1

(6) Consider Example 6 (Travelling Salesperson) from Section 3.9. The Excel spreadsheet solution template is given by file Example 6.xls on the accompanying website (www.mopgp.com).

(i) Consider the weighted goal programming spreadsheet. Percentage normalisation is currently implemented. Modify the spreadsheet to implement

- Euclidean normalisation
- Zero–one normalisation

Comment on the results obtained and which method of normalisation is most appropriate for this type of example.

(ii) Suppose that the cost, time, and, distance goals are now all set to the infeasibly low level of zero. Re-solve the model from part (i) with Euclidean normalisation. Have the results changed? Why/why not?

(7) Consider Example 7 (Downstream Oil Industry) from Section 3.9. The LINGO model files are given by file Example 7_weighted.lg4 and Example 7_Chebyshev.lg4 on the accompanying website (www.mopgp.com).

(i) Consider the weighted goal programming file. This has the weight spread equally between the four sets of goals. Investigate the following changes in target levels:

- A reduction in target costs to £800,000 per day.
- A relaxation of the refinery goals to operate at least 65% of the day.
- A tightening of the depot goals to operate between 70 and 90% of their capacities.
- The unused capacity can be assigned to either site 2 or site 3.

Analyse the effect of each of these potential changes and write a report to the management of the oil company explaining their effect on the set of goals.

(ii) Repeat the exercise of part (i) for the Chebyshev variant.

(8) Consider Example 8 (Macro Economics) from Section 3.9. The Excel spreadsheet solution template is given by file Example 8.xls on the accompanying website (www.mopgp.com).

(i) Consider the weighted and Chebyshev goal programming spreadsheets, which currently implement percentage normalisation. Modify these spreadsheets so that they do not implement any normalisation and compare the results obtained. Explain why solving this model without normalisation is a valid option.

(ii) Consider the Chebyshev goal programming spreadsheet. Modify the spreadsheet to allow any number of tax changes to take place. Analyse the difference this makes to the achievement of the goals.

(9) Consider Example 9 (Budget Planning) from Section 3.9. The Excel spreadsheet solution template is given by file Example 9.xls on the accompanying website (www.mopgp.com).

(i) Consider the weighted goal programming spreadsheet. Modify the spreadsheet so that the effect of factor 1 (f1) must be positive and the combined effect of factors 3 and 4 (f3 and f4) is at most 10 units. Analyse what effect these changes have on the solution.

(ii) Modify the Chebyshev goal programming model so that the extra funds allocated becomes a goal rather than a constraint, with a target value of no more than 1% of the current budget. Analyse the effect this change has on the solution.

(10) Consider Example 10 (Healthcare Planning) from Section 3.9. The Excel spreadsheet solution template is given by file Example 10.xls on the accompanying website (www.mopgp.com).

(i) Consider the weighted goal programming example. Perform sensitivity analysis on the managers weighting between time and cost by considering the following weight allocations:

- 20% on time, 80% on cost
- 40% on time, 60% on cost
- 60% on time, 40% on cost
- 80% on time, 20% on cost

(ii) Repeat the analysis from part (i) for the Chebyshev model.

(iii) Consider the lexicographic model. Perform sensitivity analysis on the priority levels by investigating the following priority schemes:

- Priority 1, Priority 3, Priority 2, Priority 4
- Priority 2, Priority 4, Priority 1, Priority 3
- Priority 3, Priority 1, Priority 4, Priority 2
- Priority 4, Priority 2, Priority 3, Priority 1

(iv) Write up the results of the sensitivity analyses performed in parts (i)–(iii) in the form of a report to the general manager of the hospital.

Chapter 6
Detection and Restoration of Pareto Inefficiency

The greatest danger for most of us is not that our aim is too high and we miss it, but that it is too low and we reach it.
Michelangelo (1475–1564)

As discussed in Chapter 1, the main philosophy underlying goal programming is the Simonian theory of satisficing (Simon, 1957). This philosophy is implemented by the selection of a set of target levels by the decision maker to be achieved as closely as possible. However, goal programming, arising from the field of optimisation, also encompasses elements from the optimising philosophy. The solution to a goal programming is, mathematically speaking, either a single optimisation or a series of linked optimisations. The similarities and differences between the economic theories of optimising and satisficing are well documented (Byron, 2004).

The practical implication for goal programming of combining the two philosophies is that solutions produced by a satisficing decision maker may be Pareto inefficient. That is, according to Definition 1.12, the value of one or more of the underlying objectives can be improved without any of the others being worsened. This situation occurs when the decision maker is too pessimistic when setting his/her target levels. He/she produces a set of target levels that are within the feasible region. This situation is illustrated by the following weighted goal programme that is given graphically by Fig. 6.1:

$$\text{Min } a = \tfrac{n_1}{4} + \tfrac{n_2}{3} + \tfrac{n_3}{8}$$

subject to

$$\begin{aligned}
&x_1 + \underline{n_1} - p_1 = 4\\
&x_2 + \underline{n_2} - p_2 = 3\\
&x_1 + 2x_2 + \underline{n_3} - p_3 = 8\\
&x_1 + 4x_2 \leq 20\\
&2x_1 + x_2 \leq 12\\
&x_1, x_2, n_1, n_2, n_3, p_1, p_2, p_3 \geq 0
\end{aligned}$$

D. Jones, M. Tamiz, *Practical Goal Programming*, International Series in Operations Research & Management Science 141, DOI 10.1007/978-1-4419-5771-9_6,

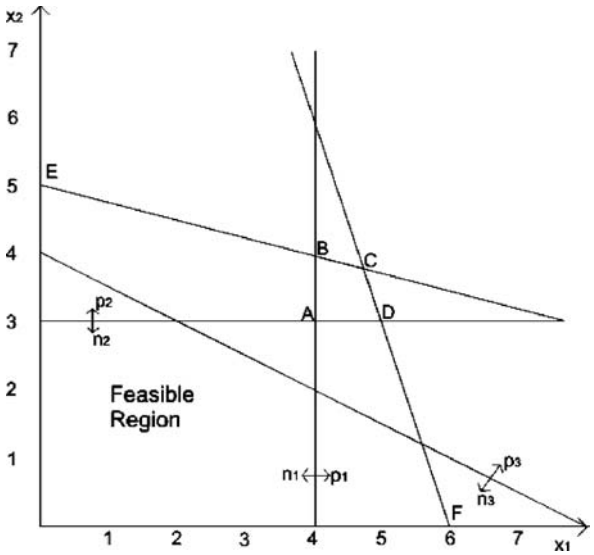

Fig. 6.1 Example of a Pareto-inefficient goal programme

Most computerised solvers will return the point A as the optimal solution to this goal programme. However, it can be seen that any point within the trapezium ABCD is an alternative optimal solution as each has $a = 0$, $n_1 = n_2 = n_3 = 0$. However, the efficient set for the underlying three objectives

$$\max z_1 = x_1, \max z_2 = x_2, \max z_3 = x_1 + 2x_2$$

is, in accordance with the definition of Pareto efficiency given in Chapter 1, given by the edges EC and CF. The line segments BC and CD give the subset of efficient solutions that dominate the original goal programme solution point A. Thus, if the decision maker is interested in going beyond the goals he/she has set, it is illogical to stop at point A and a solution from the line segments BC or CD should be preferred instead. The decision maker has been too pessimistic in his/her goal setting.

This phenomenon has been seen by some authors as a fundamental flaw in the goal programming methodology as its initial optimal solution (point A) can potentially violate a fundamental theory of decision analysis – that no rational decision maker will knowingly choose a Pareto-inefficient solution. From the opposing point of view it can be argued that goal programming is primarily a satisficing technique and therefore its solutions should be judged solely on how well they meet the goals of the decision maker and whether they produce a practical solution to the decision problem. A third, pragmatic, viewpoint is to say that although goal programming can produce Pareto-inefficient solutions, this is mainly due to poor elicitation and modelling of the decision maker's preferences and target levels by the analyst building the goal programme. In any case, techniques have been

developed from Hannan (1980) onwards that allow for the detection of Pareto inefficiency in a goal programming solution. These techniques also provide means of restoring Pareto efficiency by calculating a Pareto-efficient solution that dominates the goal programming solution. The remainder of this chapter describes Pareto detection and restoration techniques for different goal programming variants. Section 6.1 gives some definitions of Pareto optimality in the context of the objectives and priority levels of a goal programme.

6.1 Pareto Definitions

In order to better identify the nature of Pareto inefficiency in a goal programming, Tamiz and Jones (1996) give the following definitions:

Definition 6.1: An objective is **Pareto inefficient** if it can be improved without worsening the value of any other objective.

Definition 6.2: An objective is **Pareto efficient** if it cannot be improved without worsening the value of another objective.

Definition 6.3: An inefficient objective is **Pareto unbounded** if it can be increased to an arbitrarily high value without worsening any other objective.

Definitions 6.1, 6.2, and 6.3 apply to the underlying objective $f_q(x)$ of the goal. Note that we are mainly considering one-sided goals with a single deviational variable to be penalised. Two-sided goals with both deviational variables to be penalised are by their nature Pareto efficient at optimality.

The concept of Pareto efficiency can then be extended to a priority level in a lexicographic goal programme via the following three definitions.

Definition 6.4: In a lexicographic goal programme, a priority level is **Pareto inefficient** if one or more objectives within that priority level is Pareto inefficient

Definition 6.5: In a lexicographic goal programme, a priority level is **Pareto unbounded**if one of its objectives is Pareto unbounded.

Definition 6.6: In a lexicographic goal programme, a priority level is **Pareto efficient** if every objective within that level is Pareto efficient.

We can link these definitions into the more usual definitions of Pareto efficiency at the level of the whole goal programme by the following three definitions:

Definition 6.7: A goal programme is **Pareto efficient** if every objective is Pareto efficient.

Definition 6.8: A goal programme is **Pareto inefficient** if one or more objectives is Pareto inefficient.

Definition 6.9: A goal programme is **Pareto unbounded** if one or more objectives is Pareto unbounded.

For the objective set the Pareto-efficient and Pareto-inefficient subsets form a mutually exclusive and exhaustive set. The Pareto unbounded objectives form a further subset of the Pareto-inefficient subset.

6.2 Pareto Inefficiency Detection

6.2.1 Continuous Weighted and Lexicographic Variants

The initial algorithms by Hannan (1981) and Romero (1991) were based on the lexicographic and weighted variants. These combined the tasks of detecting and restoring Pareto efficiency. We shall term the unwanted deviational variables in the original goal programme as *WV*s and their corresponding deviational variable that is not penalised in the achievement function *NWV*. Each *NWV* is by implication to be welcomed (such as positive deviation from a profit goal). The Hannan and Romero methods perform a maximisation of some function of the *NWV*s, which, if it results in a change in the optimal solution from that of the original goal programme, demonstrated that the original goal programming solution was Pareto inefficient. Jones and Tamiz (1996) separate the tasks of detection and restoration and show by the following five tests that Pareto detection can be performed in many cases without resorting to optimisation:

Test A. Two-sided goals (both deviational variables penalised) can be classified as Pareto efficient.

By their nature they must always be Pareto efficient as any possible improvement towards the goal without worsening any other objective will improve the achievement function value and hence must be included in the solution of the goal programme.

One-sided objectives can belong to any of the three Pareto states. As stated previously, we refer to the deviational variable to be penalised as the weighted variable *WV*and the variable which does not have to be penalised as the non-weighted variable *NWV* for that objective.

Test B. Any objective with a basic non-zero *WV* at optimal solution is Pareto efficient.

The optimal value of a *WV* can never be decreased without worsening another objective as it would have already been decreased as part of the solution of the goal programme. Hence any objective with a basic *WV* is Pareto efficient.

As the modeller wishes to decrease the value of an objective's *WV*, we assume that they would also wish to increase the value of that objective's *NWV*, if that increase did not worsen any other objective. Hence any objective whose *NWV*can meet that criterion is considered to be Pareto inefficient.

The next two tests use an alternative method based on the examination of the optimal simplex tableau of the goal programme. Each objective not yet classified can fall into one of two categories.

Test C. Any objective with a non-basic *NWV*which can be entered into the basis without lowering the values of any basic *NWV* or increasing the value of any basic *WV* is Pareto inefficient.

In this case the optimal solution shows the target level being met exactly. Any increase beyond the target level corresponds to the *NWV* entering the basis. We therefore examine the *NWV* as a candidate for basis entry. If the *NWV* has a positive entry in the achievement function row then increasing the *NWV* would increase another objective's *WV* and hence the *NWV* is unsuitable for entry. Further, if any *NWV*s for other objectives are basic, we examine their row entries in the *NWV*column. If it is positive then increasing the *NWV* decreases another objective's *NWV*. Hence the objective is unsuitable for entry. If neither of the above cases occur, we can increase the *NWV*without worsening any other objective and the objective is Pareto inefficient. Further, if no suitable pivot element is found in the *NWV* column, then the objective is Pareto unbounded.

Test D. For any objective with a basic *NWV*, if that *NWV* can be increased by entry of any variable to the basis without causing the increase of another objective's *WV* or decrease in another objective's *NWV* then that objective is Pareto inefficient.

In this case the optimal solution shows the target level is already exceeded but there may still be the possibility of increasing the value of the *NWV* further without worsening other objectives. We investigate each column in turn for suitable incoming variables. The first criterion is that, for the same reasons as in the *NWV* non-basic case, the achievement function row coefficient must equal zero. In order for the basic *NWV* to increase if this variable is introduced, the coefficient in its row must be negative. Further, in order for no other basic *NWV*s to decrease, the coefficient in the incoming variable′s column associated with their basic rows must also be negative. If a column is found which satisfies the above criteria, then that column may be introduced into the basis without worsening any other objective, and the objective under scrutiny is Pareto inefficient. Further, if a column is found which satisfies the above criteria and has no suitable choice of pivot then the objective is Pareto unbounded.

The above tests will determine the Pareto states of many objectives. Neither test C nor test D requires any simplex iterations as both tests are based on the examination of the optimal simplex tableau. A final test is required for objectives not yet classified.

Test E. For the *t*th objective not yet classified attempt to increase NWV_t without worsening any other objective. This is achieved by the following sequence of steps:

(1) Impose the following bounds:

$$WV_q \leq WV_q^* \; q = 1, \ldots, Q$$

$$NWV_q \geq NWV_q^* \; q = 1, \ldots, Q \; q \neq t$$

where WV^*_q, NWV^*_q represent the values of the deviational variables of the *q*th objective in the optimal solution. This procedure can be achieved by a combination

of fixing non-basic variables to their bounds and setting the right-hand sides of basic variables equal to zero.

If NWV_t is a non-basic variable then attempt to enter that column into the basis. If NWV_t can enter at a non-zero value then objective t is Pareto inefficient. If objective t is now classified, exit procedure without performing simplex iteration. If no suitable pivot can be found then objective t is Pareto unbounded; the objective is now classified, so exit procedure.

NWV_t is now a basic variable at zero value due to the bounds set. Attempt to increase the value of NWV_t from zero. This process terminates with one of two conditions:

(i) A pivot is found which raises the value of NWV_t. That is, a non-degenerate iteration can take place, then objective t is Pareto inefficient, and so exit without performing iteration.
(ii) No pivot can be found to perform a non-degenerate iteration. Objective t is Pareto efficient. Exit.

In all cases the algorithm terminates with only degenerate iterations having taken place. Hence the solution point is still unchanged from that of the original goal programme optimum. This method therefore requires no saving of or return to past bases.

Since test E will always classify the objective in question, performing the test for each remaining unclassified objective will guarantee the partitioning of the objective set into its efficient and inefficient subsets. As test E is the most computationally expensive test the following algorithm for Pareto detection is proposed by Tamiz and Jones (1996):

(i) For each objective, conduct efficiency tests A and B.
(ii) For each unclassified objective with a non-basic *NWV*, conduct inefficiency test C.
(iii) For each non-basic variable able to enter basis, conduct inefficiency test D for every unclassified objective with a basic *NWV*.
(iv) For each unclassified objective, conduct efficiency test E.

The Tamiz and Jones (1996) framework remains the most computationally efficient means of detecting Pareto inefficiency. It is recommended that all goal programmes formulated for practical situations should be tested for Pareto efficiency and the results reported to the decision maker.

6.2.2 Integer and Binary Variants

The integer or binary restrictions on some of the decision variables mean that the algorithm outlined in Section 3.2.1 cannot be directly applied to integer or binary

goal programmes. A valid dominating solution has to be found that satisfies all integer or binary restrictions, sign restrictions, and constraints in order to demonstrate Pareto inefficiency. Therefore Tamiz et al. (1999) give a modified series of tests in order to classify the objectives in an integer or binary goal programme as Pareto inefficient. Tests A and B can still be carried out as before but tests C and D are no longer valid as inspection of the optimal simplex tableau is no longer sufficient to classify an objective as Pareto efficient or inefficient. Instead the following modified sequential version of test E is applied:

Test E-Int Algorithm

(1) Impose the following bounds:

$$WV_q \leq WV_q^* \; q = 1, \ldots, Q$$
$$NWV_q \geq NWV_q^* \; q = 1, \ldots, Q \; q \neq t$$

where $WV^*{}_q$, $NWV^*{}_q$ represent the values of the deviational variables of the qth objective in the integer optimal solution. This procedure can be achieved by a combination of fixing non-basic variables to their bounds and setting the right-hand sides of basic variables equal to zero.

(2) Solve the following integer programme, taking the original integer goal programming solution as the starting point:

$$\text{Max } z = \sum_{\text{q}\in\text{Q}^{\text{u}}} \text{NWV}_{\text{q}}$$

subject to

$$f_q\left(\underline{x}\right) + n_q - p_q = b_q \quad q = 1, \ldots Q$$
$$WV_q \leq WV_q^* \quad q = 1, \ldots, Q$$
$$NWV_q \geq NWV_q^* \quad q = 1, \ldots, Q$$
$$\underline{x} \in F$$
$$n_q, p_q \geq 0 \quad q = 1, \ldots, Q$$

where Q^{u} is the set of not-yet classified objectives.

(3) Inspect the optimal solution of the integer programme solved. Classify any objectives with a larger NWV_q value than $NWV^*{}_q$ as Pareto inefficient. If no objectives have larger NWV_q values then classify all remaining not-yet classified objectives as Pareto efficient and terminate the algorithm. Otherwise update the set Q^{u} to remove the objectives classified as Pareto inefficient and go to step 2.

The Tamiz et al. (1999) algorithm will classify the objectives of any integer or binary goal programme into Pareto-inefficient and Pareto-efficient subsets. The classification of objectives and hence the goal programme as Pareto unbounded is not mentioned but this can be seen to occur when the integer programme in step 2 is unbounded. The set of the *NWV*s of not-yet classified objectives can then be maximised individually to see which are causing the Pareto unboundedness to occur.

6.3 Restoration of Pareto Efficiency

Once Pareto inefficiency has been detected, the following questions should be asked:

(i) Should Pareto efficiency be restored?
(ii) If so, which Pareto-efficient point should be recommended to the decision maker?

As discussed at the beginning of this chapter, the answer to question (i) for most academics is 'yes – Pareto efficiency should be restored' because otherwise a fundamental axiom of decision theory – that no rational decision maker will knowingly choose a Pareto-inefficient solution – is violated. However, it should be good practice to inform the decision maker that their original goal estimates were too pessimistic and that Pareto restoration is taking place. Such a disclosure could cause the decision maker to decide to re-examine their objective set and goal target levels as an alternative means of solving the problem of the Pareto inefficiency.

Given that Pareto efficiency is to be restored, there is normally more than one possible candidate Pareto-efficient solution that can be chosen as the suggested solution to present to the decision maker. For example, from Fig. 6.1 it can be seen that any point on the line segments BC and CD is a potential efficient solution that could be presented to the decision maker.

The method of Hannan (1980) suggests various ways in which to calculate a Pareto-efficient solution. These include

(i) A possible weighting scheme to give relative importance to the improvement of the objectives
(ii) A priority order in which the objectives should be improved
(iii) A vector maximisation multiple objective programming model to find a set of efficient solutions that dominate the original goal programme's solution

However, Romero (1991) points out that the Hannan (1980) formulations do not adequately restrict the values of deviational variables which are not penalised in the original formulation (*NWV*s), and thus a proposed solution that leads to a worst solution for such an objective, and hence does not dominate the original goal programme's solution, could be generated. Therefore Romero proposes that any missing constraints from the constraint set

$$WV_q \leq WV_q^* \quad q = 1, \ldots, Q$$
$$NWV_q \geq NWV_q^* \quad q = 1, \ldots, Q$$

be added to the Hannan models.

Tamiz and Jones (1996) suggest the following three ways in which Pareto efficiency can be restored:

(1) Straight Restoration
This method simply assumes the decision maker will be satisfied if Pareto efficiency is restored. An extra priority level is formed consisting of all *NWV*s, each with a unity coefficient. This sum is then maximised.

This results in the following optimisation:

$$\text{Max } z = \sum_{q \in Q^C} NWV_q$$

subject to

$$f_q(\underline{x}) + n_q - p_q = b_q \quad q = 1, \ldots Q$$
$$WV_q \leq WV_q^* \quad q = 1, \ldots, Q$$
$$NWV_q \geq NWV_q^* \quad q = 1, \ldots, Q$$
$$\underline{x} \in F$$
$$n_q, p_q \geq 0 \quad q = 1, \ldots, Q$$

where Q^c is the complete set of *NWV*s. However, if the Pareto detection process outlined in the previous section has been implemented prior to restoration then only the *NWV*s of objectives classified as inefficient need be included in the final priority level.

Romero (1991) points out that the process of the original goal programming solution and Pareto restoration can in fact be conceptually modelled as a single lexicographic process.

A further complication occurs in the case where one or more objectives are Pareto unbounded. In this case the restoration optimisation will become unbounded and an efficient solution is not possible. In this case the objectives found to be unbounded should be removed from the final priority level and the optimisation continued from that point. This process may continue iteratively as not all the unbounded objectives may be detected at that point. The method will terminate with all objectives classified as either Pareto efficient or unbounded. This is the closest to efficiency that can be reached for a Pareto-unbounded model.

The straight restoration method has the drawbacks of making the following assumptions:

(i) The modeller wishes to maximise every *NWV*.
(ii) The decision maker treats each *NWV* to be maximised with equal importance.

(2) Preference-Based Restoration
In this method the preference information given by the modeller in the initial goal programme is used. The deviational variables are maximised with the weighting system used in the goal programme. For weighted goal programmes this entails maximising the *NWV*s associated with each *WV* in the objective function with the same weights, as a second priority level.

For weighted goal programming this leads to the following mathematical programme:

$$\text{Max } z = \sum_{\mathrm{q}\in\mathrm{Q^C}} \frac{\mathrm{W_q\,NWV_q}}{\mathrm{k_q}}$$

subject to

$$\begin{aligned}
&f_q\left(\underline{x}\right) + n_q - p_q = b_q \quad q = 1, \ldots Q\\
&WV_q \le WV_q^* \quad q = 1, \ldots, Q\\
&NWV_q \ge NWV_q^* \quad q = 1, \ldots, Q\\
&\underline{x} \in F\\
&n_q, p_q \ge 0 \quad q = 1, \ldots, Q
\end{aligned}$$

where w_q is the weight attached to WV_q in the original goal programming formulation.

For lexicographic goal programmes preference-based restoration entails a lexicographic maximisation model with the *NWV*s placed in identical priority levels to their corresponding *WV*s and with the same weight. If we reformulate the *l*th priority level of the original goal programme in the algebraic form

$$h_l\left(WV, n, p\right) = \sum_{q=Q^2}\left(\frac{u_q^l n_q}{k_q} + \frac{v_q^l p_q}{k_q}\right) + \sum_{q=Q^1}\left(\frac{w_q^l WV_q}{k_q}\right)$$

where Q^2 is the set of two-sided goals (both deviations penalised) and Q^1 is the set of one-sided goals (with WV_q penalised with weight $w_q{}^1$), then the following lexicographic optimisation will restore Pareto efficiency in accordance with the decision maker's original preference structure:

$$\text{Lex Max } z = \left[\mathrm{g_1}\left(\underline{\mathrm{NWV}}\right), \mathrm{g_2}\left(\underline{\mathrm{NWV}}\right), \ldots, \mathrm{g_L}\left(\underline{\mathrm{NWV}}\right)\right]$$

subject to

$$f_q\left(\underline{x}\right) + n_q - p_q = b_q \quad q = 1, \ldots, Q$$
$$\underline{x} \in F$$
$$n_q, p_q \geq 0 \quad q = 1, \ldots, Q$$

Where

$$g_l\left(\underline{NWV}\right) = \sum_{q=Q^1} \left(\frac{w_q^l \, NWV_q}{k_q} \right)$$

As in the case of straight restoration, a priority level only need contain inefficient *NWV*s if detection is performed beforehand and unbounded priority levels are dealt with on an iterative basis.

The preference-based method acknowledges the differing importance of objectives and uses the preference information available for the model. It still, however, assumes that the weighting scheme for the improvement of target levels is the same as for the achievement of targets.

(3) Interactive Restoration

The preference-based restoration method gives, in our opinion, the best possible way of restoring Pareto optimality without returning to the problem owner once Pareto inefficiency has been detected. However, it makes the large assumption that the decision maker′s preferences remain the same below and above the target level. This may be true in some cases where the target level has been set at a low level to facilitate the achievement of other target levels (e.g. a profit goal) but it may be the case that, although the achievement of the goal is important, there is total decision maker indifference above the goal (e.g. a legal requirement that is to be met).

In the interactive restoration approach the decision maker is presented with the set of inefficient objectives given by the detection process together with the optimal solution.

They then choose the objective from this set which they would most like to see raised beyond its target level. The *NWV* of this objective is then maximised in a new (lowest) priority level whilst not worsening the optimal value of any other objective. The detection process is then carried out at this new solution. If any objectives are found to be inefficient (and not unbounded) the decision maker is again asked to choose the objective most preferred for increase beyond its optimal value. Hence an iterative cycle is set up which stops only when all bounded objectives are made Pareto efficient.

If any variable selected to be made efficient is Pareto unbounded this will be detected in the maximisation of its *NWV*. That objective will then be marked Pareto unbounded and will not be able to become efficient. It will not, however, stop other objectives from being made efficient in subsequent iterations.

This method is strictly convergent as the set of objectives presented to the decision maker is reduced by at least one at each iteration. The theoretical maximum number of iterations is given by the number of objectives, but in practice very few iterations are required.

6.4 Detection and Restoration of Chebyshev Goal Programmes

When dealing with detecting and restoring efficiency in Chebyshev goal programmes, the third underlying philosophy of balancing must be taken into account. This can lead to a conflict between the two philosophies of balancing and optimising. For instance, consider the following goal programme:

$$\text{Min } a = \lambda$$

subject to

$$\begin{aligned}
&x_1 + n_1 - p_1 = 3\\
&x_2 + \underline{n_2} - p_2 = 3\\
&n_1 \leq \lambda\\
&n_2 \leq \lambda\\
&2x_1 - x_2 \leq 4\\
&2x_1 - 3x_2 \leq 24\\
&x_1, x_2, n_1, n_2, p_1, p_2, \lambda \geq 0
\end{aligned}$$

The most common optimal solution to this Chebyshev goal programming would be at the point x_1=3,x_2=3. This is shown as point A in Fig. 6.2. Any attempt to maximise p_1 or p_2 or attempt to pivot them into the basis will show that this solution is Pareto inefficient, with both objectives being classified as Pareto inefficient.

We notice that this solution is also perfectly equilibrated (Romero et al., 1998), which is perfect from the balancing philosophy point of view.

Maximising the sum of *NWV*s (straight restoration) whilst not worsening any other objective will lead to the following formulation:

$$\text{Max } a = p_1 + p_2$$

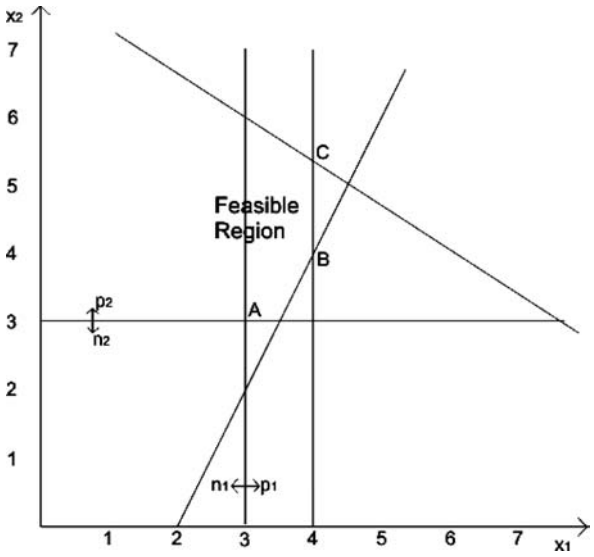

Fig. 6.2 Example of restoration in a Chebyshev goal programme

subject to

$$x_1 + n_1 - p_1 = 3$$
$$x_2 + n_2 - p_2 = 3$$
$$n_1 = 0$$
$$n_2 = 0$$
$$x_1 \leq 4$$
$$2x_1 - x_2 \leq 4$$
$$2x_1 - 3x_2 \leq 24$$
$$x_1, x_2, n_1, n_2, p_1, p_2, \geq 0,$$

which will lead to the solution point C ($x_1 = 4$, $x_2 = 5.33$) in Fig. 6.2. Point C is Pareto efficient but does not maintain the property of perfect equilibration as $p_1 = 1$, $p_2 = 2.33$. This two-phase process of detection and solution is shown by Romero et al. (1998) as equivalent to a type of reference point method (Wierzbicki, 1982) formulation.

If, on the other hand, we were to continue the L_∞ distance metric concept into restoration and solve the following linear programme

$$\max a = \mu$$

subject to

$$
\begin{aligned}
& x_1 + n_1 - p_1 = 3 \\
& x_2 + n_2 - p_2 = 3 \\
& n_1 = 0 \\
& n_2 = 0 \\
& p_1 \geq \mu \\
& p_2 \geq \mu \\
& 2x_1 - x_2 \leq 4 \\
& 2x_1 - 3x_2 \leq 24 \\
& x_1, x_2, n_1, n_2, p_1, p_2, \lambda \geq 0
\end{aligned}
$$

then the likely result from a linear programming optimiser is point B ($x_1 = 4$, $x_2 = 4$). This point is perfectly equilibrated but not Pareto efficient. We also note that point C is an alternative optimal solution to this linear programme, but probably would not be the solution found by a linear programming solver.

This phenomenon is discussed by Romero et al. (1998) who propose a three-phased approach that first solves the Chebyshev goal programme, then maximises an L_∞ metric of the *NWV*s, and finally maximises a weighted sum of the *NWV*s. This formulation attempts to ensure the maximum degree of balance before attempting to restore efficiency. It is equivalent to solving the following lexicographic optimisation:

$$\text{Lex Min}\left[\lambda, -\mu, -\sum\nolimits_{q\in Q^C} \frac{w_q\, NWV_q}{k_q}\right]$$

subject to

$$
\begin{aligned}
& f_q\left(\underline{x}\right) + n_q - p_q = b_q \; q = 1, \ldots, Q \\
& \tfrac{w_q WV_q}{k_q} \leq \lambda \; q \in Q^c \\
& \tfrac{w_q WV_q}{k_q} \leq \mu \; q \in Q^c \\
& \underline{x} \in F \\
& n_q, p_q \geq 0 \; q = 1, \ldots, Q, \lambda, \mu \geq 0
\end{aligned}
$$

However, Ogryczak (2001a, b) provides examples in which the Romero et al. (1998) formulation does not provide efficient solutions and suggests a four-phase solution process which adds an additional priority level to ensure that all the *WV*s are adequately minimised by weighted sum before moving onto the maximisation of the *NWV*s. This leads to the following formulation:

$$\text{Lex Min}\left[\lambda, \sum_{q\in Q^C} \frac{w_q\, WV_q}{k_q}, -\mu, -\sum_{q\in Q^C} \frac{w_q NWV_q}{k_q}\right]$$

subject to

$$f_q\left(\underline{x}\right) + n_q - p_q = b_q \; q = 1, \ldots, Q$$
$$\frac{w_q WV_q}{k_q} \leq \lambda \; q \in Q^c$$
$$\frac{w_q NWV_q}{k_q} \leq \mu \; q \in Q^c$$
$$\underline{x} \in F$$
$$n_q, p_q \geq 0 \; q = 1, \ldots, Q, \lambda, \mu \geq 0$$

The range of problems for which the Romero et al. (1998) model returned an inefficient solution was a debate in the literature at the time, detailed by Ogryczak (2001a–c), Romero et al. (2001), and Tamiz et al. (2001).

Thus models are available that will restore Pareto efficiency to a Pareto-inefficient goal programme, whilst taking into account the desire to maintain as high a level of balance as possible. The example in this section does, however, show that there is an inbuilt conflict between balance and efficiency in some Chebyshev goal programmes. In such cases there is a need to find out the decision maker's preferences and explore the trade-off between balance and efficiency, in a similar way as is done in the extended goal programming framework outlined in Section 4.3. This is a topic for future research in the goal programming area.

6.5 Detection and Restoration of Non-linear Goal Programmes

Goal programmes that are not linear in nature require dedicated techniques to detect and restore Pareto efficiency. Caballero et al. (1998) deal with the case of detection and restoration of convex non-linear models and interested readers are referred to this text for specialist algorithms. Cabellero and Hernandez (2006) deal with the detection and restoration of fractional goal programmes. They show that the simplification of the goal

$$\frac{f_q\left(\underline{x}\right)}{g_q\left(\underline{x}\right)} + n_q - p_q = b_q$$

into the form

$$f_q\left(\underline{x}\right) + n_q - p_q = b_q g_q\left(\underline{x}\right)$$

whilst not being a valid mathematical transformation for solution is actually sufficient for detection of Pareto inefficiency in all cases. They also propose a new

restoration technique designed specifically for fractional goal programmes, which is termed minimax straight restoration and is based on the Chebyshev distance metric.

6.6 Conclusion

It can be seen that there are effective ways of detecting and restoring Pareto efficiency for all major goal programming variants. This is a comprehensive answer to critics who suggest that goal programming is incompatible with the concept of Pareto optimality. At the very least, it is suggested that every goal programming built is checked for Pareto inefficiency. If the goal programme is Pareto efficient, as the majority of well-constructed goal programmes should be, then the decision maker has the added assurance of knowing this. If the goal programme is inefficient, then a Pareto restoration technique can be employed in accordance with the stated or interactive preferences of the decision maker to find an appropriate Pareto-efficient solution.

6.7 Exercises

(1) In Example 1 from Section 3.9 assume that the first goal has now been pessimistically set at 'The objective function should be at least 20'.

   (i) Reformulate the weighted goal programme from part (i) and apply Pareto efficiency detection to show the goal programme is now Pareto inefficient.
   (ii) Apply preference-based restoration to find a Pareto-efficient solution.

(2) In Example 2 from Section 3.9 assume that the first goal remains the same but the second and third goals have the following modified form:
Goal 2′:The retailer wishes to process at least 300 orders per hour.
Goal 3′:The retailer wishes for the total wage bill not to exceed £145 per hour.

   (i) Apply Pareto efficiency detection to the weighted goal programme formulated in part (a) to see if the solution is Pareto efficient or not.
   (ii) If the goal programme is Pareto inefficient then apply interactive restoration considering first goal 2′ to be improved.

(3) In Example 3 from Section 3.9, assume that the lexicographic structure in part (ii) has been reformulated as

Priority 1: Achieve the strategic aim of 500 cups per day.
Priority 2: Achieve a profit of at least £600.
Priority 3: Minimise the extra finishing furnishing and finishing hours needed.
Priority 4: Ensure that at least 100 of each type of cup is manufactured.

(i) Apply Pareto efficiency detection to the lexicographic goal programme formulated in part (ii) to see if the solution is Pareto efficient or not.
(ii) If the goal programme is Pareto inefficient then apply preference-based restoration to find a Pareto-efficient solution.

(4) In Example 4 from Section 3.9 use Pareto detection techniques to prove that the lexicographic model formulated in part (i) is Pareto efficient.
(5) In Example 5 from Section 3.9 assume the daily target cost for the diet has risen to £6.00.

(i) Apply Pareto efficiency detection to the lexicographic goal programme formulated in part (1) to see if the solution is Pareto efficient or not.
(ii) If the goal programme is Pareto inefficient then apply preference-based restoration to find a Pareto-efficient solution.

(6) In Example 6 from Section 3.9 assume that the target goals have quadrupled to no more than 4000 km, 40 h, and €800 for the trip.

(i) Apply the Pareto efficiency restoration algorithms given in Section 3.4 to the Chebyshev goal programme formulated in part (ii). Comment on the results.
(ii) Is a perfectly equilibrated solution possible in this problem. Why/why not? If yes, then is it Pareto efficient?

(7) In Example 7 from Section 3.9 assume that the target goals have been relaxed to

- Goal 1: Keep costs to below £45,000 per day.
- Goal 2: Keep each refinery operating at at least 55% of its capacity.
- Goal 3: Keep each depot at a level of between 30 and 95% of its capacity.
- Goal 4: Assign any unused capacity to site three.

(i) Apply Pareto efficiency detection to the weighted goal programme to see if the solution is Pareto efficient or not.
(ii) If the goal programme is Pareto inefficient then apply interactive restoration considering the goal order to be improved as goal 1, goal 2, goal 3, goal 4.

(8) In Example 8 from Section 3.9 assume that the target goals have been relaxed to limit the total per annum loss of income to £1000 for high-income earners, £750 for mid-income earners, and £500 for low-income earners.

(i) Apply the *Test E-Int Algorithm* detailed in Section 6.3.2 to the weighted goal programme formulated in part (1) to see if the optimal solution is Pareto efficient or not.
(ii) If the goal programme is Pareto inefficient then apply straight restoration to give a Pareto optimal solution.

(9) Explain why the solution produced to part (1) of Example 9 from Section 3.9 will always be Pareto efficient.

(10) In Example 10 from Section 3.9, check the Pareto efficiency status of the lexicographic model formed in part (ii). In the case of Pareto inefficiency, apply preference-based restoration to find a Pareto optimal solution.

Chapter 7
Trend of Integration and Combination of Goal Programming

This chapter is concerned with the place of goal programming within the wider fields of multi-criteria decision making, operational research, and artificial intelligence. Modern operational research methodology no longer regards its constituent techniques as stand-alone modelling and solution tools but looks to combine them. Hence there is a growing body of research in fields such as systems and hyper-heuristics (Ozcan et al., 2008), which seek to intelligently select and mix techniques for symbiotic advantage.

The consequence for goal programming is that the 'stand-alone' goal programme which provides a solution to a single model without reference or utilisation of other techniques is becoming rarer in the literature. Instead, a range of papers that incorporate goal programming into a wider system consisting of multiple analysis tools can be found. This trend is referred to by Jones and Tamiz (2002) as the 'trend of integration and combination'. Additionally, some articles use goal programming without specifically referencing it as a technique or categorising the model as belonging to the connected fields of linear programming, multi-objective programming, or multi-criteria decision making.

This chapter details the way in which goal programming has been combined with differing techniques in statistics, artificial intelligence, operational research, and multi-criteria decision analysis.

7.1 Goal Programming as a Statistical Tool

The most frequent use of goal programming is as a decision-making tool where the goals correspond to a range of diverse, conflicting criteria. However, the situation where the target levels correspond to a measured value on a single criterion presents a different type of goal programming model. The x_j values now represent coefficients of an equation corresponding to the other criteria and the (weighted) achievement function aims to provide a best possible fit to the set of observed target values by minimising the sum of deviations from the target values. The technological (a_{ij}) coefficients now represent the score of the ith observation on the jth criteria. This leads to the base model

D. Jones, M. Tamiz, *Practical Goal Programming*, International Series in Operations Research & Management Science 141, DOI 10.1007/978-1-4419-5771-9_7,

$$\text{Min } a = \sum_{i=1}^{q} (u_i n_i + v_i p_i)$$

subject to

$$\sum_{j=1}^{m} a_{ij} x_j + n_i - p_i = b_i \quad i = 1,...,q$$
$$x \in F$$
$$\text{x}_j \textit{ free} \quad j = 1,...,m \; n_i, p_i \geq 0 \quad i = 1,...,q$$

This model leads to goal programming becoming a useful tool for the fields of statistics and of data mining. The above equation is in fact a least-absolute value (LAV) regression model, which is based on the L_1 distance metric. This is used as an alternative to the standard least-squares regression model which has an underlying L_2 distance metric. This has the advantage of being less influenced by outlying observations. The other advantage gained by using this type of mathematical programming based model is that it is more flexible in that it requires fewer assumptions than standard least-squares regression. It is also more flexible in that it allows extra constraints on the combinations of the x_{ij} values and weighting of the observations according to their importance to the decision maker. The original goal programming model of Charnes et al. (1955) was this type of model for calculating executive compensation packages and took advantage of the ability to add extra constraints to ensure implementable results. A further significant application of goal programming as a regression tool is given by Charnes et al. (1988) in the context of corporate policy planning. The theory and statistical properties of the goal programming based L_1 regression are detailed in Sueyoshi (1997).

Recent applications of goal programming as a regression tool are given by Bhattacharya (2006) who takes advantage of the fewer assumptions required in order to provide improved pre-harvest forecasts in an agricultural planning model. Da Silva et al. (2006) use the technique for forestry management. They conclude that goal programming produces similar results to least-squares regression.

Other recent uses of goal programming in relation to statistical methods include its use to model user preferences in small-area statistics (Sedeno-Noda et al., 2009) and the use of fuzzy goal programming to optimise multi-response problems (Amiri and Salehi-Sadaghiani, 2008).

7.2 Goal Programming as a Multi-criteria Decision Analysis Tool

As stated in Chapter 1, goal programming belongs to the family of multiple criteria decision-making (MCDM/A) techniques. In fact, Fig. 7.1 demonstrates its popularity from amongst the distance-based MCDM techniques, and Fig. 1.1 in Chapter 1 demonstrates its popularity is continuing to grow in recent years. As would be expected from such an influential technique, goal programming has many linkages

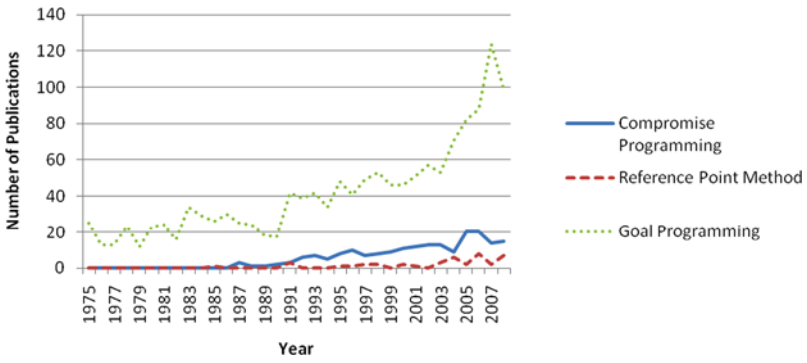

Fig. 7.1 Number of publications of distance metric based techniques in period 1975–2008. *Source*: 'Web of Knowledge' published by Thomson Reuters

and connections with other techniques within the wider MCDM/A field. This section explores these linkages.

7.2.1 Goal Programming and Other Distance Metric Based Approaches

The distance metric based approaches in multi-criteria decision making are defined as those techniques that minimise a function of the distance between the desired and achieved solutions. The desired solution could be a decision maker set of goals for each objective or it could be the ideal point. The principal distance metric based approaches are goal programming, compromise programming (Zeleny, 1982), and the reference point method (Wierzbicki, 1982), along with the emergence of goal-based multi-objective evolutionary algorithms, as detailed in Section 7.3.3.

The key difference between the distance metric techniques is mainly that of underlying philosophy and distance metric used. The measure of distance can be summarised by the L_ρ set of distance metrics which have the following algebraic form:

$$\text{Min } L_\rho = \left[\sum_{q=1}^{Q} \left(\frac{\left| f_q(x) - b_q \right|}{k_q} \right)^\rho \right]^{\frac{1}{\rho}}$$

Table 7.1 gives the distance metrics used by the major distance metric based techniques.

Romero et al. (1998) investigate linkages between the distance metric based techniques. They show that the compromise programming with the L_1 distance metric is equivalent to a weighted goal programme with the target values set at ideal levels. Also, the compromise programming with the L_∞ distance metric is equivalent to a

Table 7.1 Distance metrics used in MCDM distance-based techniques

Technique	Distance metrics used	Nature of b_q
Weighted goal programming	L_1	Decision maker set target values
Chebyshev goal programming	L_∞	Decision maker set target values
Compromise programming	Set of L_1,. . ., L_p,. . ., L_∞	Ideal values
Reference point method	L_∞ followed by L_1	Decision maker set reference values

Chebyshev goal programme with the target values set at ideal levels. It is noted that the original formulation of compromise programming is intended to use a (compromise) set of distance metrics between $\rho=1$ and $\rho=\infty$ to give a scale of solutions between ruthless optimisation and balance. In fact, many applications just concentrate in practice on these two end-points of the compromise set, in which case there is a strong connection between the process of compromise programme and the process of solving weighted and Chebyshev goal programmes with optimistic target values. The other distance metric sometimes used is $\rho=2$ which corresponds to a non-linear (quadratic) goal programme with optimistic target values. These theoretical linkages led to the development of the extended goal programming framework (Romero, 2004) as described in Section 4.3, which encompasses both techniques.

A further linkage is given between the reference point method and goal programming. The reference point method is shown to be able to set into a goal programming framework as an initial Chebyshev goal programme followed by an L_1 Pareto restoration phase by Romero et al. (1998) and Ogryczak (2001c).

7.2.2 Goal Programming and Pairwise Comparison Techniques

Under some circumstances it is easier for the decision maker to make 'pairwise comparison' judgments between pairs of objectives rather than directly give a weight set as is required for conventional goal programming. Such pairwise comparisons could take the following forms:

- I regard objectives A and B to be of equal importance.
- I regard objective A to be of moderately more importance than objective B.
- I regard objective B to be very strongly more important than objective A.

The most well-known pairwise comparison technique is the analytical hierarchy process (AHP) (Saaty, 1981). The synergy between goal programming and the AHP can take two forms, which are outlined separately below.

7.2.2.1 Using the AHP to Determine Goal Programming Preferential Weights

As mentioned in Section 3.6, there is a natural combination whereby the AHP can be used to elicit a set of preferential weights for a weighted or Chebyshev goal programme. This methodology was first used by Gass (1986) in the context of military planning. It has since been applied to a variety of application areas including recently in project selection (Kahraman and Buyukozkan, 2008), transportation resource allocation (Wey and Wu, 2007), and health care (Li et al. 2009).

This can be an effective approach to goal programming weight determination, especially if the nature of the application means the decision maker finds it easier to give his/her preferences using pairwise comparisons. However, there is no formal proof that the weight vector derived by AHP leads to a solution that is 'optimal' within the set of possible weighting vectors when it is used as the preferential weighting vector in a goal programme. Rather, it should be regarded in the same way as one would regard any initial decision maker estimate of preferential weights. That is, as a good starting point for exploring the decision maker's preferences via some formal or informal weight sensitivity analysis, such as the algorithm proposed in Section 4.5.

7.2.2.2 Using Goal Programming as a Technique to Derive the Weighting Vector in AHP

The other major combination of AHP and goal programming can be viewed of as a reverse of the first combination. In this case, goal programming is used as an alternative method for deriving the AHP weights rather than the other way around. The standard means of deriving AHP weights is the eigenvector method, as detailed by Saaty (1981). However there has been a lot of discussion in the literature about the advantages and disadvantages of the eigenvalue methods, and other methods have been proposed for deriving the weighting vector from the pairwise comparison matrix (Saaty and Hu, 1998). These include the logarithmic weighted goal programming approach of Bryson (1995) and the Chebyshev goal programming approach of Despotis (1996).

Jones and Mardle (2004) give a generic distance metric framework that encompasses the works of Bryson and Despotis. The following generic distance metric formulation is used:

$$\text{Min } a = \left[\sum_{i=1}^{n} \sum_{j=1, i \neq j}^{n} \left(\frac{n_{ij} + p_{ij}}{a_{ij}} \right)^{\rho} \right]^{\frac{1}{\rho}}$$

subject to

(either

$$v_i - a_{ij}\, v_j + n_{ij} - p_{ij} = 0 \quad i = 1,...,q, \quad j = 1,...,q \;\; i \neq j$$

or

$$\log(v_i) - \log(v_j) + n_{ij} - p_{ij} = \log(a_{ij}) \;\; i = 1,...,q, \;\; j = 1,...,q \;\; i \neq j$$

)

$$\sum_{i=1}^{n} v_i = 1$$

$$v_i > 0 \;\; i = 1,...,q \;\; n_i, p_i \geq 0 \quad i = 1,...,q$$

where a_{ij} is the numerical value of the decision maker's pairwise comparison between objectives i and j. v_i is the resulting weight of objective i. The most common distance metrics are the Manhattan ($\rho = 1$), the Euclidean ($\rho = 2$), and the Chebyshev ($\rho = \infty$). The least-squares (LSM) and logarithmic least-squares (LLSM) methods are examples of Euclidean metric based methods that can be represented by quadratic goal programmes. Table 7.2 classifies all of the methods mentioned above within the single framework proposed by Jones and Mardle.

Table 7.2 Classification of distance metric based methods to determine AHP weighting vectors

Distance metric	$\rho = 1$	$\rho = 2$	$\rho = \infty$
Logarithmic	Bryson (1995)	LLSM	–
Non-logarithmic	–	LSM	Despotis (1996)

The two non-classified combinations are theoretically possible but have not yet been utilised. In any case, Jones and Mardle compare the results of the three distance metrics against the classic eigenvalue method over the criteria of

- Maximum deviation from the set of pairwise comparison:
- (Max d_{ij} $i = 1,...,q$ $j = 1,...,q$ $i \neq j$)
- Average deviation from the set of pairwise comparisons:
- $\left(\sum_{i=1}^{q} \sum_{j=1, i \neq j}^{q} \frac{d_{ij}}{q(q-1)} \right)$
- The maximum pairwise comparison produced from a ratio of the derived weights
- An approximate rank preservation statistic

Jones and Mardle conclude that methods based around the Manhattan or Chebyshev distance metrics fare worse across the above set of distance metrics, with both Chebyshev- and Manhattan-based methods struggling to differentiate between objectives and the Chebyshev additionally violating the preference order. They therefore recommend using a distance metric close to $\rho = 2$ if a goal programming based method is to be used. They note that the main benefit of using a goal

programming based approach is the ability to control the mix of minimisation of the average and maximum d_{ij} values by changing the distance metric ρ. This is particularly important when there are a number of decision makers, each with different views on the importance of the objectives and a consensus has to be reached. Jones and Mardle give a fisheries economics example (Jones and Mardle, 2004) where this is the case. This results in non-linear programming models but fortunately they are not large in size and hence should be solvable by a non-linear programming solver.

The process described above deals with crisp pairwise judgements producing crisp weighting vectors. A methodology for producing interval weighting vectors from interval pairwise comparisons is given by Wang (2007).

7.2.3 Goal Programming and Other MCDM/A Techniques

Although Sections 7.2.1 and 7.2.2 describe two of the closest connections and synergies of goal programming within the other techniques with the wider field of multi-criteria decision making/aid (MCDM/A), there are many other linkages and commonalities to be explored. This section looks at the current research and future potential of these topics.

MCDM techniques are often categorised (Hwang and Masud, 1979) into three subsets according to when the decision maker preferences are elicited:

- A priori techniques in which all decision maker preferences are specified before the solution process
- Interactive techniques in which the decision maker preferences are elicited during the solution technique, mainly in response to their opinion of solutions generated to that point
- A posteriori techniques where the solution process takes place first and the decision maker preferences are then elicited from the generated set of solutions

Goal programming is often cited as an example of an a priori technique, and this can be linked to a potential weakness as it is correctly concluded that it is difficult for the decision maker to specify all their preferences before seeing any generated solutions. However, as detailed in Chapter 3, it is our opinion that goal programming should not be used as a stand-alone 'one-off' solution tool. Rather, it is correctly used as part of a formal or informal interactive process of developing and refining the model, exploring the weighting space, exploring different variants, and hence providing an accurate modelling and elicitation of the decision maker preferences. Such a view of goal programming places it more towards the 'interactive techniques' category than as a pure a priori technique.

7.2.3.1 Goal Programming and Interactive Methods

There have been several works that look at the formal intersection of interactive techniques with goal programming dating back to Dyer (1973). Gardiner and Steuer

(1994) classify the principal interactive methods from the wider field of MCDM into a unified framework. Tamiz and Jones (1997) present an interactive method for weighted and lexicographic goal programmes and discuss the design of interactive methods for goal programming. The following is an expanded, updated list based on key parameters identified:

- *Choice of the initial solution*. Does the algorithm start from an infeasibly good starting point and relax the objectives; from a Pareto-inefficient starting point and improve objectives; or from a decision maker specified starting point?
- *Information presented to the decision maker*. Does this include target and achieved values for objectives; ideal, anti-ideal, or nadir values for objectives; deviational variable weights; matrices of trade-off values; priority level information; or Pareto state of objectives?
- *Information requested from the decision maker*. Are quantitative or qualitative questions asked? Does the information relate directly to the parameters of the goal programme or not? Does it require simple or more complex information? What level of understanding of goal programming/quantitative skills does the decision maker need?
- *Terminating conditions*. Is the algorithm search based, learning based, or has it distinct learning and search phases? When does the algorithm terminate? How many solutions does it produce? How much decision maker time and effort does it require?

There are also specialist formal interactive goal programming methods for several variants. Cabellero et al. (2006) give an interactive meta-goal programming approach. Interactive methods are particularly used in the fuzzy goal programming variant, with Arora and Gupta (2008) in the context of bi-level programming; Selim and Ozkarahan (2008) and Liang (2007) in the context of supply change management; and Abd El-Wahed and Lee (2006) in the context of transportation being recent examples. In addition, there are many articles which use goal programming in an iterative or repeated manner, sometimes within a larger multi-technique modelling and solution system. This leads to a more informal, but nevertheless effective, form of interactive use of goal programming. Recent examples of this type of interaction include within a land use planning decision and support system (Janssen et al., 2008) and within an environmental management tool to identify best available techniques (Mavrotas et al., 2009).

7.2.3.2 Goal Programming and A Posteriori Techniques

The a posteriori techniques in MCDM are concerned with generating Pareto-efficient solutions before eliciting the decision maker(s) preferences. This could either be by classical methods, as detailed by Steuer (1986), or by evolutionary methods, as discussed in Section 7.3.3. The major use of these techniques for goal programming is detailed in Chapter 6 on Pareto detection and restoration. This chapter describes how points on, or a portion of, the efficient set that dominates a goal

programming solution that is Pareto inefficient can be generated. A goal programming type of philosophy can also be used to limit the portion of the Pareto set being generated to within an area of interest to the decision maker as defined by an expressed set of goals. A third combination to note is where goal programming and efficient set generation solutions to a model can be generated by the same computer package and compared. An example of this is given by the MOPEN package (Caballero et al., 2005).

7.2.3.3 Goal Programming and Discrete Choice/Outranking Methods

The discrete choice and outranking methods are used to choose between or provide a ranking of a discrete number of alternative solutions in the presence of multiple criteria. Although there is potential for combination with goal programming there are not as yet many examples in the literature. One possible combination is to use an outranking method to provide preference information for a goal programming. Martel and Aouni (1990) use the Promethee method for this purpose. Another possibility is to use the discrete choice method to choose between or the outranking method to rank a number of goal programme solutions produced by different weighting schemes or variants. Perez Gladish et al. (2007) apply this methodology to a mutual fund portfolio selection problem using the ELECTRE I method as the outranking technique.

7.3 Goal Programming and Artificial Intelligence/Soft Computing

7.3.1 Goal Programming and Pattern Recognition

Pattern recognition models classify a set of observations into a number of groups based on their characteristics. The most common situation and easily analysable is the two-group classification problem where items have to be classified into one of two distinct groups by consideration of a number of attributes of the item. Two-group classification models arise in fields such as finance (creditworthy of non-creditworthy), medicine (benign or malignant), and defence (friendly or hostile). A related field is that of discriminant analysis, which concentrates on the values and nature of the factors used to discriminate the different classes. Normally, a training set of observations whose class and attributes are known is available in order to train or form the model in its classification. An overview of different methods used to solve pattern classification models is given by Zhai et al. (2006). The use of mathematical programming techniques for pattern classification is described by Baek and Ignizio (1993). Freed and Glover (1981, 1986) describe and analyse the use of linear and goal programming methods for pattern classification and discriminant analysis. Nakayama and Kagaku (1998) give a goal programming model that uses a piecewise linear discriminant line for the purposes of classification. Nakayama et al.

(2005) examine the use of goal programming to assist the support vector machine (SVM) approach to pattern classification.

Jones et al. (2007) give a framework for modelling two-group pattern classification and discriminant analysis using different goal programming variants. The algebraic format of the generalised pattern classification model is given as

$$\text{Min } z = \left[w_a^{\rho} \sum_{k=1}^{k} \left[\sum_{i=1}^{n_1} u_k n_{ik}^{(a)} + \mu_k s_{ik} \right]^{\rho} + w_b^{\rho} \sum_{k=1}^{k} \left[\sum_{i=1}^{n_2} v_k p_{ik}^{(b)} + v_k t_{ik} \right]^{\rho} \right]^{\frac{1}{\rho}}$$

subject to

$$\begin{aligned}
&\sum_{j=1}^{m} a_{ij} x_j + n_i^{(a)} - p_i^{(a)} = x_0 + \beta_k \quad i = 1,...,n_1, \quad k = 1,..., K \\
&\sum_{j=1}^{m} a_{ij} x_j + M s_{ik} \geq x_0 + \beta_k \quad i = 1,...,n_1, \quad k = 1,..., K \\
&\sum_{j=1}^{m} b_{ij} x_j + n_i^{(b)} - p_i^{(b)} = x_0 - \beta_k \quad i = 1,...,n_2, \quad k = 1,..., K \\
&\sum_{j=1}^{m} b_{ij} x_j - M t_{ik} \leq x_0 - \beta_k \quad i = 1,...,n_2, \quad k = 1,..., K \\
&\\
&\sum_{j=1}^{m} x_j = 1 \\
&-\alpha \leq x_j \leq a \quad j = 1,... , m
\end{aligned}$$

which produces the discriminant line

$$x_1 y_1 + x_2 y_2 + \cdots + x_m y_m = x_0$$

where x_j $j = 1,..., m$ are the set of attributes whose coefficients in the discriminant line are the decision variables of the goal programming. These are bounded within a sufficiently large range $-\alpha \leq x_j \leq \alpha$ in order to aid the solution process, as bounded decision variables are generally easier to handle in a mathematical programming framework than unrestricted ones. y_j $j = 1,..., m$ are the attribute values of a new observation to be classified.

The Jones et al. (2007) model allows the use of all distance metrics via the parameter ρ and the inclusion of binary variables s_{ik} and t_{ik}. The binary variables allow for the inclusion of the L_0 distance metric, which is a binary measure of distance, in this case taking the value zero if an observation is correctly classified and one if it is incorrectly classified. The L_0 distance metric is rarely used in goal programming, with the only other usage being in the meta-goal programming formulation in Section 4.4.

Table 7.3 Three-level classification scheme (two groups, A and B, problem)

Achieved value ($f(\underline{x})$)	Classification
$f(\underline{x}) < x_0 - \beta_2$	Certain B
$x_0 - \beta_1 < f(\underline{x}) \leq x_0 - \beta_2$	Intermediate B
$x_0 < f(\underline{x}) \leq x_0 - \beta_1$	Probable B
$f(\underline{x}) = x_0$	**Unknown**
$x_0 < f(\underline{x}) \leq x_0 + \beta_2$	**Probable A**
$x_0 + \beta_1 < f(\underline{x}) \leq x_0 + \beta_2$	**Intermediate A**
$f(\underline{x}) > x_0 + \beta_2$	**Certain A**

The training set of observations is grouped into n_1 observations of the first class and n_2 observations of the second class. The class weights w_a and w_b ensure that each group is represented with the required decision maker weighting (often equal weight for both classes is desired) even though the classes may have different numbers of observations. Multiple (k) levels of classification are also allowed, with a three-level (k=3) classification scheme demonstrated in Table 7.3. The unwanted deviations from the classification boundaries ($n_{ik}^{(a)}$, $p_{ik}^{(b)}$) are penalised with weights u_k and v_k controlling the importance and hence power of the classification at each boundary. The weights μ_k and v_k control the importance of the L_0 distance metric in the formulation. a_{ij} and b_{ij} represent the score of the ith observation on the jth attribute and M represents an arbitrarily large positive constant.

Jones, Collins, and Hand investigate different parameter settings in the context of a cinema (movie-theatre) viewing model. They conclude that the use of the non-linear distance metrics $1 < \rho < \infty$ is not worthwhile given the extra computational effort of solving non-linear, and in many cases for this type of application, large-scale models. The L_0 metric offered interesting possibilities but was not practical on this occasion due to the large number of binary variables introduced; however, investigating the solution of this type of model via a heuristic method is an avenue for future research. The L_∞ metric was adversely affected by 'outlier' observations. More useful was the addition of the multi-levels of classification which improved the granularity of the classification. The inclusion of the weights at the classification boundaries was also useful as it helped the decision maker control the power of the classification and produce different strategies with respect to percentage of correct classifications versus avoidance of total misclassification.

7.3.2 Goal Programming and Fuzzy Logic

The major combination between the Zadeh (1965) theory of fuzzy numbers and goal programming is that of the fuzzy goal programming variant, which is detailed in Section 2.3.1. However, there is also benefit to be gained from combining goal

programming with the wider field of fuzzy logic in order to configure decision maker systems that can both process fuzzy information and make satisficing decisions based on the outcome of the fuzzy logic process. A recent example of this combination is found in Famuyiwa et al. (2008) who use fuzzy logic to identify and classify potential suppliers in the supplier selection problem and goal programming to subsequently choose suppliers according to the manufacturer's preferences. Similarly, Nepal et al. (2006) use a combination of fuzzy logic and the Chebyshev goal programming variant for a quality design problem.

7.3.3 Goal Programming and Meta-heuristic Methods

A heuristic method is defined as 'a technique that produces good solutions to a model in reasonable computational time without being able to guarantee optimality, or in many cases, a measure of distance from optimality' (Reeves, 1995). A meta-heuristic method has the ability to go beyond local optima in its search for the global optimum solution to a problem. Well-known meta-heuristics include genetic algorithms, simulated annealing, and tabu search.

As heuristics do not guarantee an optimal solution their use should be reserved for models which do not fit the underlying assumptions for, or are not able to be solved in reasonable computational time by, the mathematical programming based solution methods described in Chapter 5. These could include large-scale non-linear models, models with large number of integer variables, models with ill-defined objectives and constraints, and models with stochastic or dynamic elements. Some recent examples of using genetic algorithms as a substitute for mathematical programming methods in order to provide solutions to hard-to-solve goal programmes for the major meta-heuristic methods are listed below:

- *Genetic Algorithms*. Wang and Chang (2008) in the context of network topology design; Leung (2007) in the context of transportation planning; and Stewart et al. (2004) in the context of land use planning. Mishra et al. (2006) combine concepts from genetic algorithms and simulated annealing to solve a fuzzy goal programming model relating to machine-tool selection and operation allocation.
- *Simulated Annealing*. Baykasoglu (2005) presents a general method for solving lexicographic goal programmes using simulated annealing. Mishra et al. (2006) combine concepts from genetic algorithms and simulated annealing to solve a fuzzy goal programming model relating to machine-tool selection and operation allocation. Zolfaghari et al. (2004) use simulated annealing to solve a goal programme for the multi-period task assignment model.
- *Tabu Search*. Yang and Feng (2007) present a tabu search approach to solve a stochastic goal programme related to solid transportation.
- *Ant Colony Optimisation*. Chan and Swarnkar (2006) use ant colony optimisation to solve a fuzzy goal programme for machine-tool selection and operation allocation.

7.3.3.1 Multi-objective Evolutionary Algorithms

A type of heuristic that has become prominent within the MCDM community in the past decade is that of multi-objective evolutionary algorithms (MOEAs) (Deb, 2001). Evolutionary algorithms are population-based methods that use concepts from the field of genetics to breed successively better generations of solutions to a problem. A genetic algorithm is a specific example of an evolutionary algorithm. There is a particular synergy between the Pareto set generating a posteriori methods and MOEAs. Because a MOEA is population based, it can provide an estimation of the entire Pareto set with its population, rather than having to generate it a point at a time. This, together with the fact that MOEAs can handle a large range of ill-defined models means that the MOEA is proving useful for producing a choice of Pareto-efficient solutions in fields such as engineering design (Dufo-Lopez and Bernal-Agustin, 2008). The current standard MOEA algorithms are the non-dominated sorting algorithm NSGAII (Deb et al., 2002), the strength hyper-volume-based algorithm (Zitzler and Thiele, 1999), and the epsilon-MOEA algorithm (Deb et al., 2003).

However, a couple of provisos should be added about MOEAs. First, they are a meta-heuristic method and hence are unable to guarantee an optimal solution. Hence, their use in simple models where the efficient set can be generated in reasonable computational time by conventional methods seems unwarranted. Second, it should be noted that calculating the Pareto set is not the same as solving the decision problem. A choice still needs to be made from amongst the, sometimes very large number of, Pareto-efficient solutions found.

This is where combination with a satisficing technique such as goal programming is potentially very useful. Goal programming methodology could, in theory, be used to guide the MOEA to concentrate on certain areas of the Pareto set that are of more interest to the decision maker. This type of methodology is used by Park et al. (2007) who present a goal-based version of NSGAII and use it to find optimal configurations of radio systems.

7.4 Goal Programming and Other Operational Research Techniques

Goal programming has been used extensively to solve classical models arising in operational research. A non-exhaustive list includes goal programming versions, applications, or combinations with the following:

- transportation model (Kwak and Schniederjans, 1985)
- trans-shipment model (Hemaida and Kwak, 1994)
- inventory planning model (Panda et al., 2005)
- newsvendor problem (Taleizadeh et al., 2009)
- portfolio selection model (Perez Gladish et al., 2007)
- forecasting model (Love and Lam, 1994)

- supply chain management configuration (Ho and Emrouznejad, 2009)
- loading and scheduling (Petrovic and Akoz, 2008)
- nurse rostering (Azaiez and Al Sharif, 2005)
- queuing theory (Li et al., 2009)
- personnel scheduling (Ignizio, 2004)
- production planning (Leung and Chan, 2009)
- tour scheduling (Mathirajan and Ramanathan, 2007)
- vehicle routing (Calvete et al., 2007)
- knapsack problem (Ghomi and Ashjari, 2002)

This section looks at some of the most important synergies in this area, where goal programming is actively combined or integrated with another operational research technique rather than just used as solution tool for that technique.

7.4.1 Goal Programming and Data Envelopment Analysis

The technique of data envelopment analysis originated from two of the three authors of goal programming. In 1978 Charnes, Cooper, and Rhodes (Charnes et al., 1978) gave the first DEA formulation, with the purpose of measuring the efficiency of decision-making units. Glover and Sueyoshi (2009) describe DEA as having its roots in the goal programming (Charnes et al., 1955; Charnes and Cooper, 1961) and fractional programming (Charnes and Cooper, 1962) work of Charnes and Cooper.

The similarities between DEA and goal programming are well documented. The technical details of the mathematical similarities between the weighted goal programming and additive DEA techniques are given in Cooper (2005) and the historical context of the two techniques are given by Glover and Sueyoshi (2009). The philosophical differences between the two techniques are also emphasised by Cooper (2005), who states the goal programming is primarily a planning technique whereas DEA is primarily a control and evaluation technique.

Given the different purposes of the two techniques, there is potential benefit in their combination to form integrated planning and control systems. A recent example is given by Dharmapala et al. (2007) in the context of academic salary planning. Goal programming is used as a planning tool to set the average salaries and then DEA is used as a measurement tool to set the merit payments for individual staff. An integrated GP/DEA approach with both planning and performance measurement aspects is given in the context of transportation planning by Sheth et al. (2007). Another possibility is to use DEA to first eliminate some inefficient units before using goal programming as a selection tool amongst the efficient units. This approach is applied by Ekinci (2007) in the context of supplier selection.

7.4.2 Goal Programming and Simulation

The combination of goal programming and simulation is potentially very powerful due to the different strengths and uses of the two techniques. Simulation

(Law and Kelton, 2000) is very good at capturing stochastic flow of entities through a system and hence has a large body of application in diverse fields such as health care, transportation, and manufacturing. It does not, however, optimise the multiple, sometimes conflicting, performance measures contained in its output per se as it is a simulating rather than a decision-making technique. Goal programming, being a multiple objective decision maker technique, is good at reconciling multiple conflicting goals and recommending pragmatic decisions. Being a technique based on mathematical programming and hence subject to the certainty assumption, it is not strong at capturing stochastic flow. Therefore there exists potential to combine the two techniques so as to use their respective strengths and cover the other's weaknesses.

There are three ways in which the techniques can be combined:

- Pre-goal programme: Illustrated by Fig. 7.2. Here the goal programme is used before the simulation in order to determine good values of the parameters to investigate and/or to influence the design of the simulation model.
- Integrated GP: Illustrated by Fig. 7.3. Here the goal programme is embedded in the simulation model and may be repeatedly solved to determine the path of entities through the system, the resource levels available at a given time, or the stopping criteria of the simulation.
- Post GP: Illustrated by Fig. 7.4. Here the goal programme is used after the simulation in order to analyse the output and make decisions as to the scenario(s) which have most closely met the decision maker's goals.

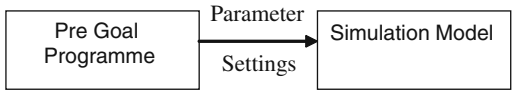

Fig. 7.2 Pre-goal programme

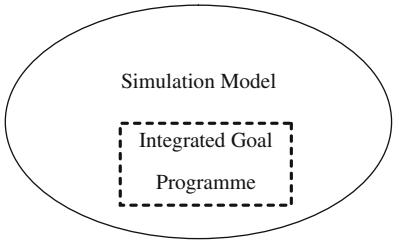

Fig. 7.3 Integrated goal programme

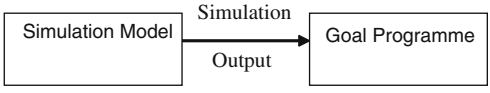

Fig. 7.4 Post-goal programme

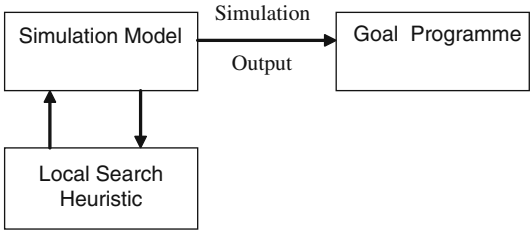

Fig. 7.5 Post-goal programme with added local search heuristic

Additionally it is possible to add a local search or heuristic technique to control the search path of the simulation through the large number of combinations of simulation inputs. This type of system is illustrated graphically for the post-goal programme case by Fig. 7.5 and described by Willis and Jones (2008). The local search technique could be either single or multiple objective in nature.

Chapter 8 gives an example of an integration of goal programming and simulation in the context of healthcare planning.

Chapter 8
Case Study: Application of Goal Programming in Health Care

The purpose of Chapter 8 and 9 is to present two practical applications of goal programming that illustrate the methodology detailed in Chapter 1–7. The first, application, detailed in this chapter, relates to the field of healthcare planning, more specifically to the planning of a medical assessment unit in a primary care hospital. It is a specific example of the use of integer goal programming and an illustration of the trend of integration and combination detailed in Chapter 7, with goal programming being combined with simulation. The technical details of the research are detailed in Oddoye et al. (2007, 2009). However, the purpose of this chapter is to look at the application in the context of the wider field of goal programming and identify the modelling techniques and paradigms used.

8.1 Context of Application Area

The setting for this research is the Medical Assessment Unit (MAU) at Queen Alexandra Hospital (QAH). The QAH is an acute primary care hospital serving the population of Portsmouth, UK, and its surrounding areas (around 250,000 people). The QAH is a public-service hospital belonging to the British National Health Service (NHS).

There has been a lot of pressure on the Accident and Emergency Departments (equivalent to Emergency Rooms in the United States) in UK hospitals in recent years, with a 3–7% annual rise since the early 1990s. The UK government has also introduced a range of performance targets relating to various aspects of hospital performance including maximum permissible waiting times. MAUs arose as the trusts that manage NHS hospitals attempted to reorganise acute care delivery in a more efficient and effective manner. The MAU acts a buffer between the Accident and Emergency Department and the rest of the hospital. Patients were able to be observed, assessed, and treated for a period of up to 48 h on the MAU after arriving from the Accident and Emergency Department or being referred by a general practitioner. They are then either discharged or transferred to a specialist department in the hospital.

D. Jones, M. Tamiz, *Practical Goal Programming*, International Series in Operations Research & Management Science 141, DOI 10.1007/978-1-4419-5771-9_8,

The MAU certainly can relieve pressure on the Accident and Emergency Department, but unless it is properly staffed and organised this could turn into a case of 'bottleneck transfer' from the Accident and Emergency Department to the MAU. The research was initiated by clinicians from the QAH MAU who were motivated to the perception that, although the MAU was improving patient service in the QAH, it may ultimately be unable to cope with a continued rise in emergency admissions. Their concerns and questions can be categorised into several areas, some of which became clearer as the study proceeded. These were as follows:

- Questions on resource levels, such as numbers of types of doctors, nurses and beds to employ at different hours of the day
- Questions on optimal procedures and layout of the MAU. These include many What/If type scenarios
- Questions on how to control and improve patient flow through the MAU.
- A desire to understand the amount and nature of doctor and nurse workload within the MAU

8.2 Initial Goal Programming Models

The initial meetings with the MAU clinicians concentrated on the first question from the list above: How many nurses, doctors, and beds were required at different hours of the day? The following four objectives were identified:

- Minimise patient delay
- Minimise extra number of doctors required by hour of day
- Minimise extra number of nurses required by hour of day
- Minimise extra number of beds required by hours of day

A weighted goal programming was chosen as the initial variant as the clinicians did not have an existing priority order of their objectives and were interested in trade-off information between the objectives.

8.2.1 Data Collection

For the model to be tested to ascertain its reliability and effectiveness in finding solutions for resource levels that are required for patients to pass through the MAU with minimum delay, reliable data was needed. The model was developed using data and clinical prioritising rules from a short stay MAU in use at the QAH between 1998 and 2001. This was the only data available on the MAU when this research commenced in 2002 and was obtained from the MAU database. The eight-bedded MAU within this period treated 8000 patients annually, an average of 22 patients per day within an average length of stay of less than 5 h.

The data was obtained from the database system of the MAU having 20 patient attributes collected from their arrival through to their discharge. Information about individual patients is governed by UK Laws on Data Protection, Human Rights, and the common law on privacy and confidentiality. Thus to ensure confidentiality of the patients, their names and type of illness were omitted from the data used. The severity of the illness is, however, determined by their given triage. The only bio-social information included in the data related to the patient's date of birth and sex.

For the purposes of the model, four parameters were important. These were as follows:

- Start date – which gives the date and time of a patients' admission onto the MAU;
- Method of admission – which indicates where the patients' were coming from, either from the Accidents and Emergency departments or general practitioner referrals;
- Triage level – the Manchester triage system is in use on the MAU to identify patients that require preferential access to assessment;
- End date – which gives the date and time when patients are discharged from the MAU either to their homes or to other hospital beds.

The data was stored in an Access database after which it was cleaned to eliminate errors. The errors stem from the incompleteness of some of the parameters collected. For example some patients had the start date and end date recorded, however with no triage level recorded. Graphs of the data were drawn to look for unusual patterns and outliers. For the purpose of the model, the parameters stated above needed to be extracted from the database to test the model. To do this queries were written with structured query language (SQL) within Access. SQL is syntax for executing queries. But the SQL language also includes syntax to update, insert, and delete records.

After the extraction of the relevant parameters, the time of patient admission and discharge within the start date and end date were also retrieved to use as part of the data for the model. Data on time spent attending to patients by doctors and nurses was also obtained. Generally there is a variation in the time spent by doctors and nurses with patients during admission, discharge, and in-between care periods. The data on these times was given by a clinician in the MAU.

Also patients can be delayed on the MAU for an amount of time in certain periods when doctors and nurses are all occupied. The maximum acceptable delay times are based on the triage levels of the patients involved.

8.2.2 Model Description

8.2.2.1 Assumptions

Patients must be treated between their arrival and discharge including any delay that may occur as a result of their triage levels. This model is written for a day and a half;

a suggestion is given at the end for dealing with a longer period of time. In the data used for testing the model, it is assumed that patient number i is seen by a doctor or nurse during his/her arrival time a_i hour and an ideal discharge time d_i hour.

Therefore it is assumed that patient i needs $ds_i = d_i - a_i$ hours to stay in hospital for his/her treatment. The number of patients is i and each one assigned an arrival time, an ideal discharge time, and a triage level T_i.

Triage levels range from 1 to 5. A patient with a triage of 1 has an emergency situation and should be seen immediately. When triage increases the patient's situation has a lower emergency. Patients with triage level 1 should not be delayed while triage levels 2, 3, 4 and 5 could be delayed for t_i hours, 15 min, 1 h, 2 h and 4 h, respectively. It is worth noting that the MAU nurses triage patients being admitted. The triage levels are assigned based on the presenting case as defined by the clinical definitions given in Table 8.1.

Table 8.1 Definition of triage levels

Triage level	Maximum delay	Clinical definition
1	None	Immediate resuscitation required
2	15 min	Serious condition and abnormal vital signs
3	1 h	Serious condition and normal vital signs
4	2 h	Minor condition – worrying symptoms
5	4 h	Minor condition – few/no conditions

A patient with a triage level T_i needs $\alpha(A, T_i)$, $\alpha(B, T_i)$, and $\alpha(C, T_i)$ hours of doctors' time per hour during admission, care period, and discharge, respectively. A patient with a triage level T_i needs $\beta(A, T)_i$, $\beta(B, T)_i$, and $\beta(C, T)_i$ hours of nurses' time per hour during admission, care period, and discharge, respectively. The triage therefore provides a clinical indicator of the acute work load. The values of α and β coefficients are given in Table 8.2.

Table 8.2 Doctor and nurse time required per hour by triage level

Triage level (T_i)	$\alpha(A, T_i)$	$\alpha(B, T_i)$	$\alpha(C, T_i)$	$\beta(A, T_i)$	$\beta(B, T_i)$	$\beta(C, T_i)$
1	0.33	0.33	0.33	0.50	0.50	0.50
2	0.25	0.25	0.20	0.33	0.33	0.33
3	0.25	0.08	0.16	0.33	0.08	0.33
4	0	0	0	0.25	0	0.25
5	0	0	0	0.25	0	0.25

8.2.2.2 Decision Variables

The main decision variables for the goal programme relate to whether patients are seen by doctors and/or nurses at a given hour. These are binary in nature and are divided into first contact, on-going contact, and discharge. These key variables then allow other variables such as free and extra doctors, nurses, and beds to be determined:

$$\mathrm{PT}_{ih} = \begin{cases} 1 & \text{if patient } i \text{ is treated by a doctor or nurse in hour h} \\ 0 & \text{otherwise} \end{cases}$$

$$Y_{ih} = \begin{cases} 1 & \text{if patient } i \text{ is first seen by a doctor or nurse in hour h} \\ 0 & \text{otherwise} \end{cases}$$

$$Z_{ih} = \begin{cases} 1 & \text{if patient } i \text{ is discharged in hour h} \\ 0 & \text{otherwise} \end{cases}$$

DA_h	The doctors time available but not used by patients in hour h
NA_h	The nurses time available but not used by patients in hour h
$\mathrm{Bed}A_h$	The number of free beds at the beginning of hour h
$\mathrm{Extra}D_h$	The amount of extra doctors' time needed in hour h
$\mathrm{Extra}N_h$	The amount of extra nurses' time needed in hour h
$\mathrm{ExtraBed}_h$	The number of extra beds needed in hour h
PD_i	The amount of time patient i has been delayed
SA_i	The starting hour of treatment process for patient i

where h=1,. . .,36 and the number of patients (n) vary according to the size of the MAU.

8.2.2.3 Achievement Function

The following achievement function is formulated to represent the four goals detailed at the start of this section:

$$\text{Min } a = w_1 \sum_{i=1}^{n} \mathrm{PD}_i + w_2 \sum_{h=1}^{36} \mathrm{ExtraD}_h + w_3 \sum_{h=1}^{36} \mathrm{ExtraN}_h + w_4 \sum_{h=1}^{36} \mathrm{ExtraBed}_h$$

The initial weights were set, after consultation with the clinicians of the MAU, at

$$w_1 = 0.3,\ w_2 = 0.1,\ w_3 = 0.1,\ w_4 = 0.5$$

This weight set represents the cost and difficulty (practical and political) of obtaining extra beds and also an emphasis on minimising patient delay. These weights contain an implicit normalisation component as well as preferential information.

8.2.2.4 Goals

Goal Set 1: Patient Delay

The following goals are considered for calculating the delay of each patient. Delays on the MAU are normally due to lack of appropriate resource levels and may occur

anytime from arrival through to discharge, i.e. delay before admission, delay before discharge, or delay between care periods. It is important to add that delayed ward transfer can cause great disruption to the functioning of the MAU:

$$\sum_{h=a_i}^{d_i+\tau_i} (h+1)Z_{ih} - PD_i = a_i + ds_i \qquad i = 1,..., n$$

where the first term of the objective obtains the discharge hour for patient i and $a_i + ds_i$ is the expected discharge hour without any delay. The negative deviational variable in this formulation is omitted as the target level is set at the ideal (no delay) value. The best situation is when the actual discharge hour is the same as the expected discharge hour without any delay and hence $PD_i = 0$.

Goal Set 2: Extra Doctor Usage

This goal set represents a measurement and control on the use of doctors by patients. It states that the amount of doctors' time needed by patients must be equal to the total available amount of doctors' time for patients:

$$\sum_{i=1}^{n} \sum_{a_1 \le h}^{k} [\alpha(A, T_i)\, Y_{ih} + +\alpha(B, T_i)\, PT_{ih} + \alpha(C, T_i)\, Z_{ih}] + DA_h$$
$$-\underline{\text{ExtraD}_h} = \text{TADT} \qquad h = 1,..., 36$$

where TADT is the total amount of doctors' time available without using extra doctors.

Goal Set 3: Extra Nurse Usage

This goal set represents a measurement and control on the use of nurses by patients. It states that the amount of nurses' time needed by patients must be equal to the total available amount of nurses' time for patients:

$$\sum_{i=1}^{n} \sum_{a_1 \le h}^{k} [\beta(A, T_i)Y_{ih} + +\beta(B, T_i)PT_{ih} + \beta(C, T_i)Z_{ih}] + NA_h$$
$$-\underline{\text{ExtraN}_h} = \text{TANT} \qquad h = 1,...,36$$

where TANT is the total amount of nurses' time available without using extra nurses.

Goal Set 4: Extra Bed Usage

The following goal controls the number of extra beds needed for each hour:

$$\sum_{i=1}^{n} \sum_{k \geq h, a_1 \leq h}^{d_i+\tau_i} Y_{ik} + n_h - \underline{\text{ExtraBed}_h} = \text{BedA}_h \quad h = 1,...,36$$

Here the goal is to keep within the available bed limit BedA_h.

8.2.2.5 Constraints

Constraint set 1. This states that patient i requires an average resource use of ds_i hours of treatment:

$$\sum_{h=a_1}^{d_i+\tau_i} \text{PT}_{ih} = \text{ds}_i \qquad i = 1,..., n$$

Constraint set 2. Due to the possibility of patients waiting to be attended to after their admission onto the MAU for lack of available doctors and nurses, constraint set 2 determines which hour patients are seen for the first time by a doctor or nurse after their arrival onto the MAU:

$$\sum_{h=a_1}^{d_i+\tau_i} Y_{ih} = 1 \qquad i = 1,..., n$$

Constraint set 3. The starting hour of treatment (SA_i for each patient after their arrival time is calculated by this constraint set:

$$\sum_{h=a_1}^{d_i+\tau_i} hy_{ih} = Sa_i \qquad i = 1,..., n$$

Constraint set 4. The following constraints state the priorities for the starting hour of treatment process of patients. The constraint can be imposed if priorities such as 'first come first served' are important within a triage level:

$$\text{SA}_i \leq \text{SA}_j \quad i, j = 1,..., n, \quad i < j, \quad T_i = T_j$$

$$\text{SA}_i \leq \text{SA}_j \quad i, j = 1,..., n \quad a_i, = a_j, \quad T_i < T_j$$

Constraint set 5. This constraint set states that the hour in which a patient is seen for the first time should be accounted for within the patient's treatment process or treatment duration:

$$\mathrm{PT}_{ih} \geq \mathrm{Y}_{ih}\ i = 1,..., n\ h = a_i,..., d_i + t_i$$

Constraint set 6. This states the starting time of the treatment process of each patient. This constraint states that each patient cannot be treated before the first time of their treatment:

$$\sum_{h=a_i}^{k-1} \mathrm{PT}_{ih} \leq \mathrm{M}_1(1 - \mathrm{Y}_{\mathrm{ik}}) \quad i = 1,..., n, \quad k = a_i + 1,..., d_i + t_i - 1$$

where M_1 is a sufficiently large positive constant.
Constraint set 7. This ensures an hour in which doctors or nurses see patients for the last time during their treatment process before patients leave the MAU:

$$\sum_{h=a_1}^{d_i+\tau_i} Y_{ih} = 1 \qquad i = 1,..., n$$

Constraint set 8. This ensures that the hour when a patient will be seen for the last time should be accounted for in the patient's treatment process:

$$\mathrm{PT}_{ih} \geq \mathrm{Z}_{ih} \quad i = 1,..., n \quad h = a_i,..., d_i + t_i$$

Constraint set 9. This is added for finishing time of the treatment process of each patient. This constraint states that patients cannot be treated after the last time of their treatment:

$$\sum_{h=a_i}^{d_i+\tau_i} \mathrm{PT}_{ih} \leq M_2(1 - \mathrm{Z}_{ik}) \quad i = 1,..., n \quad k = a_i + 1,..., d_i + t_i - 1$$

where M_2 is a sufficiently large positive constant.
Constraint set 10. This states that the last hour of patient's treatment process cannot be before the first time (starting time) of treatment:

$$\sum_{h=a_i}^{k} Z_{ih} \leq M_3(1 - Y_{ik})\ i = 1,..., n\ k = a_i + 1,..., d_i + t_i$$

where M_3 is a sufficiently large positive constant.
Constraint set 11. The number of free beds in each hour, which are not used by patients, is determined in the following constraint set. TB is the number of total beds in the MAU:

$$\sum_{i=1}^{n} \sum_{k=1, a_1 \leq h}^{h-1} (Y_{ik} + Z_{ik}) + \mathrm{BedA}_h = \mathrm{TB} \quad h = 1,..., 36$$

Constraint set 12. When there is not a free bed, new patients must wait until at least one bed becomes free. Therefore the number of patients which can be seen for the first time after their arrival should be less than or equal to the number of free beds. The following constraint states this restriction:

$$\sum_{i=1}^{n} \sum_{k=1,a_1 \leq h}^{h-1} Y_{ih} \leq \text{BedA}_h \quad h = 1,..., 36$$

8.2.2.6 Sign Restrictions

The variables *PT, Y, Z* are defined as binary, all other variables are defined as the standard case of continuous and non-negative.

8.2.3 Solution and Analysis

The integer weighted goal programming model formulated in Section 8.3.1 was used to solve a test-set of three models containing 22, 27, and 33 patient admissions onto an 8-bedded, 3-nurse, 2-doctor MAU as existed between 1998 and 2001. The MPL mathematical programming package with CPLEX solver was used as a modelling and solution tool. Full details of the results obtained are available in Oddoye et al. (2007). Table 8.3 shows, as a sample, the extra beds and nurses needed by hour of day for the 27-patient model. The number of doctors was adequate at all hours of the day for this model. All hours not included in Table 8.3 did not require any extra resources.

Table 8.3 Extra resources required by hour of day for 27-patient model

Hour of day	Extra beds required	Extra nurses required
14:00 – 15:00	2	0.71
15:00 – 16:00	5	1.21
16:00 – 17:00	2	0.29
17:00 – 18:00	0	0
18:00 – 19:00	1	0
19:00 – 20:00	0	0
20:00 – 21:00	3	0
21:00 – 22:00	1	0
22:00 – 23:00	1	0
23:00 – 00:00	1	0

Figure 8.1 gives a typical output as presented to the MAU clinicians. This graph relates to the number of free nurses by hour of day.

This model confirmed some of the prior hypotheses and observations of the MAU clinicians, such as the peak demand for resources being in the afternoon and early evening. It also helped them isolate the demand shortfalls by showing that the shortfall of resources was predicted to be in the beds and nurses rather than doctors.

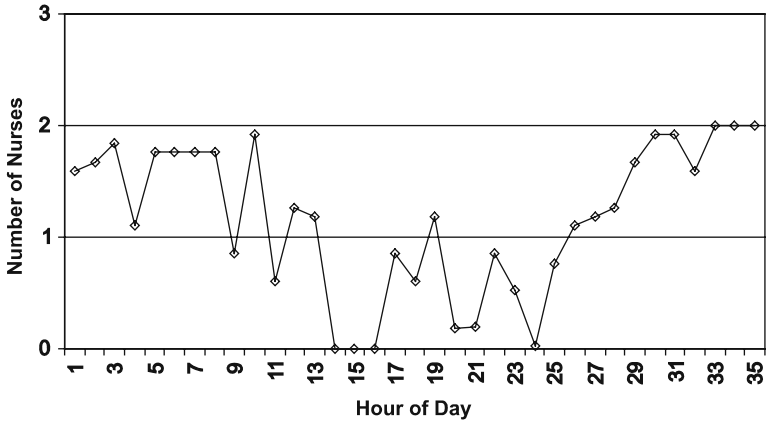

Fig. 8.1 Number of nurses free by hour of day

The model was effective in answering the first question of the clinicians about appropriate resource levels. However, a number of limitations of the model also emerged. Questions 2–4 from the list in Section 8.1 relating to patient flow, MAU layout, sensitivity analysis, and a deeper understanding of the doctor and nurse workload started to emerge from the MAU clinicians. The algebraic structure of the model was already fairly complex and difficult for the stakeholders of the MAU to comprehend. Additionally, by the time of the study in 2004 the MAU had 58 beds and a larger patient throughput. Given that the 33-patient model already contained large number of binary variables which meant solution times of several hours, an increase to the scale of hundred of patients per day seemed computationally infeasible. The amount of sensitivity and/or weight analysis that could be performed was also limited by the long solution times. A solely mathematical programming based model also seemed ill-equipped to solve the emerging questions relating to patient flow, optimal set-up of the MAU, and deeper understanding of the doctor and nurse workload. Hence, the decision was made to switch to a combined simulation and goal programming framework as detailed in Section 8.3.

8.3 Combined Simulation and Goal Programming Model

The combined simulation and goal programming model is an example of the methodology detailed in Section 7.4.2. The results of the initial goal programming model built in Section 8.2 did not directly feed into the simulation model. However, they did prove useful in helping to set resource levels by hour of day in the simulation. In this sense there is an aspect of the pre-goal programme methodology to this model. However, the major and more direct combination is the post-goal programme which is built to investigate the output of the simulation and hence inform the MAU clinicians.

There has been an extensive amount of simulation applied to the healthcare sector in recent years. For instance, Eldabi et al. (2007) quote Royston (2005) who finds over 3500 Google Scholar hits pertaining to the term 'Health Simulation' in the period 2000–2004. This is an approximate threefold increase on the previous 5-year period. Despite this growth Eldadi et al. use the literature and expert opinion to make the claim that 'the immense potential of simulation has not yet been realized in practice'. Whilst analysing the expert opinion to understand why this is the case, they highlight the need to mix modelling techniques more readily. This conclusion is in line with the focus of Chapter 7, and given the underlying multi-objective and goal-based nature of many healthcare decision situations shows a real potential of goal programming to be used in combination with other modelling techniques in this application area.

A major consideration of the clinicians in the MAU is the efficient management of patient flow through their department. The criticality of an efficient post A&E patient flow framework is highlighted by Proudlove et al. (2007) who call for 'clear analysis and effective modelling... to disentangle flows to establish root causes of problems and demonstrate the effects of better management'.

There is also a general need to develop models to examine and understand the resource requirements of MAUs and to provide a framework or develop standards that hospital developers and clinical managers could consult. Delays within MAUs stem from the unmatched number of patients discharged to the number of incoming patients. There is an increase in the workload of doctors and nurses possibly caused by insufficient numbers to meet the demand for assessment. The number of unoccupied beds has a significant impact on patient delay. Patients waiting to be transferred to other hospital wards after their assessment cause bed blockage which in turn prevents new patients being admitted. For patients referred by general practitioners this can lead to ill and unstable patients waiting in their homes. Thus the work of this case study aimed to work towards the (Proudlove et al., 2007) agenda in helping the hospital management to understand their patient flow and hence make more effective decisions.

It also aimed to unleash the potential of simulation in this situation by combining it with goal programming. This allows the relative strengths of simulation (capturing and modelling patient flow) and goal programming (meeting multiple targets over a range of conflicting criteria) to be brought together in order to aid the MAU clinicians in their decision making.

8.3.1 Further Data collection for the Simulation Model

Data collection is one of the most important and difficult problems in simulation and it can determine the success of the simulation model in representing the reality of the situation modelled. Unlike the goal programming model built in Section 8.2, the data used for the simulation could not be obtained from the database system of the MAU since it lacked detailed and specific information required.

The interest was not solely in the type and number of resources on the MAU but also the various activities doctors and nurses are involved in and the average length of specific activities. This also provides answers to the fourth general category of questions posed by the MAU clinicians related to better understanding of the doctor and nurse activities that make up their workload. Thus doctors and nurses were observed while performing their duties. Robinson (2004) describes three categories of data differentiated by two factors: availability and collectability. The first category is a data that is available, second there is some data that is not available but collectable and finally some data is not available and not collectable. The type of data required for the simulation fell under the second category. It needed to be collected by observing doctors and nurses on the MAU and monitoring the time activities take on average to perform. For data collection by observation, approval was obtained from the Local Research Ethics Committee. A 50-page application form was submitted to the Isle of Wight, Portsmouth and South East Hampshire Local Research Ethics Committee for consideration.

8.3.2 Simulation Model Description

The simulation model is developed in Micro Saint, a visual interactive modelling system. Through Micro Saint, useful information about processes that might be too expensive or time-consuming to test in the real world is gained. A version of the top-level model network developed is shown in Fig. 8.2.

The rectangular tasks are networks within which are various other tasks to be performed. The first network is a 'patient generator' network that generates two types of patients that are admitted to the MAU (A&E and general practitioner referral patients), all of whom go to the reception. The generation of patients from either A&E or general practitioner follows an exponential distribution and there is usually a wait for a bed to become available in the MAU before patients are admitted.

For the patients that are generated and moved to the reception, those requiring beds for the MAU length of stay (LoS) form about 95% of the total population, while those with no bed requirement except chairs make up the remaining 5%. Patients with bed requirements are assigned a bed before going through the reception, and the available bed numbers are decreased according to the numbers assigned. On completion of a patient's LoS or treatment, the patient is either transferred to another hospital ward or discharged home and the bed becomes available.

Patients can follow different paths in the model depending on where they are being routed to. Patients can either see a nurse or a doctor, a tactical decision based on the availability of doctors and nurses. However, for 95% of the time patients are likely to see a nurse first. The stacks in front of each task are queues and show when patients wait for that task. Data from these waiting points is important and will be analysed to evaluate the system performance.

Patients that follow to see the nurse enter into the nurse clerking network, where the nurses clerk the patient and document the diagnosis ready for the doctor. After

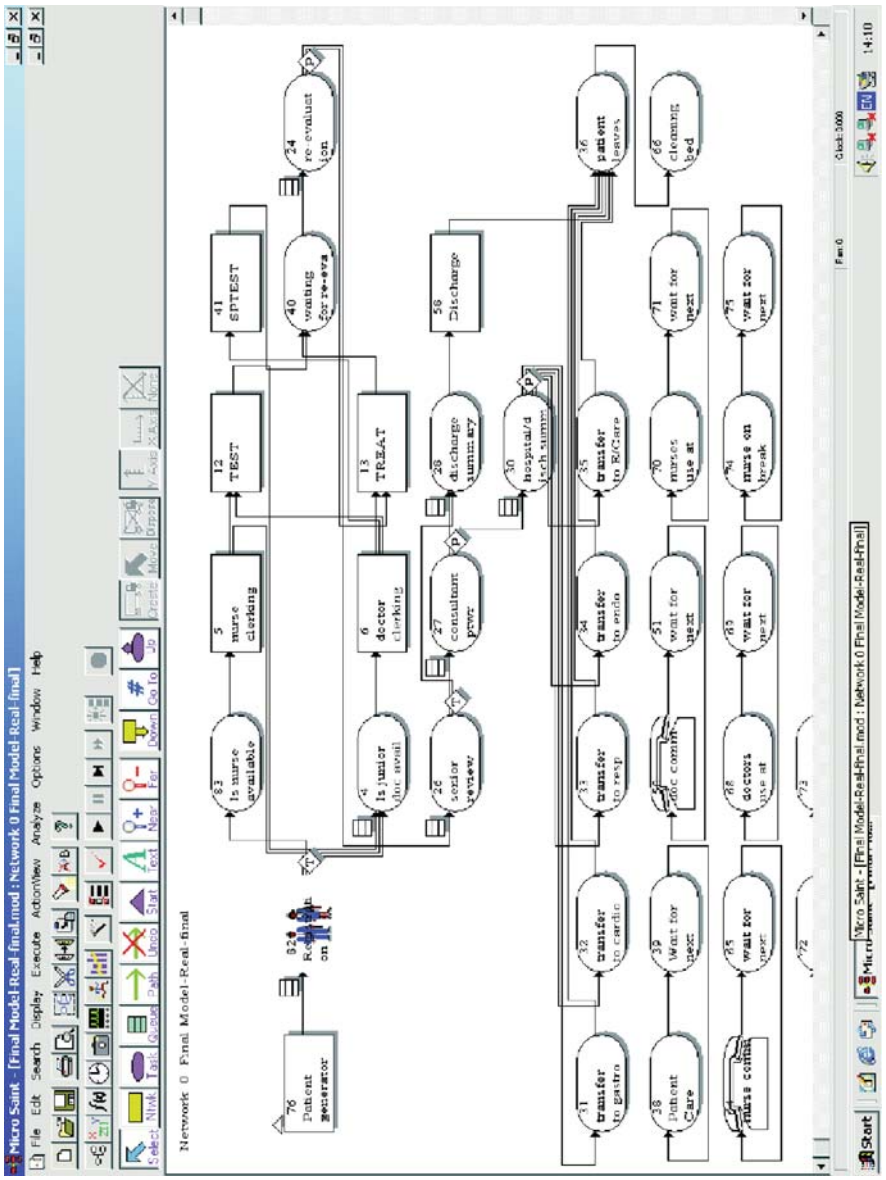

Fig. 8.2 Top-level simulation network

the nurse clerking when it is determined that preliminary tests are needed immediately before the doctor sees the patient, tests are done on the patient. The patient then follows through to the TEST network. These tests generally involve taking blood samples from patients for investigation or performing ECG.

After patients have had their blood investigations requested and their ECGs done, they are then seen by any available junior doctor. Usually by the time the junior doctor is ready to attend to the patient, the results from the investigations would be ready and this aids in the diagnosis of the patient's illness. Patients then enter the 'doctor clerking' network where the junior doctor takes the history and examines the patients to fully understand their problem.

All patients receive some form of treatment and some of them have their treatment initiated immediately. All these tasks could happen concurrently. In the TREAT network all patients have their observations checked regularly; this involves checking the temperature, pressure, and other vital signs. For patients that need airway management, circulation management, fluid management, or medication after their observations, they are routed by the tactical decision node to the required treatment.

If after clerking the patient, the doctor sees the need to have a specialised test done on the patient so as to have a better and more precise understanding of the patient's illness, the patient is routed through to the Specialised test (SPTEST) network.

The specialised tests include detailed blood investigations, X-rays, ultra sound, CT scan, endoscopy, and echocardiogram. Patients could have one or more specialised tests done depending on the request of the doctor. When a patient has completed all required tests, they follow on and wait for re-evaluation. After re-evaluation, about 20% of the patients are routed back for further treatment while the remaining 80% follow on for a senior doctor review.

The senior doctor review takes 16 min on average to complete. The senior doctor after reviewing decides the next line of action for the patient. About 23% of all the patients are discharged by the senior doctors while the remaining 77% wait to see a consultant.

The consultant post-take ward round is conducted between 8.30 and 11.30 am every morning. Consultants from different specialties are usually assigned to patients depending on the type of illness. For example a consultant cardiologist will be assigned to a patient with a cardio problem. After the post-take ward round, the consultant makes a decision on which patients need to be discharged home and those that need to be transferred to other hospital wards. In all 40% of patients are usually discharged home and the remaining 60% are transferred to other hospital specialised wards.

Patients referred by consultants to other hospital specialist wards have their discharge summary written by the junior doctors and wait for their transfer to one of the five specialised wards being gastroenterology, cardiology, respiratory, endocrine, and elderly care. Two main factors influence this long wait. The first being that before the patient can be transferred from the MAU, there must be a bed available in the destination ward. Second the ward transfer teams are usually busy with other patient transfers and thus it takes some time before they become available.

This delay in transfer causes problems for the MAU thus preventing other waiting patients from being admitted onto the unit. It is important to note that some patients wait longer before they are transferred to other hospital wards although this seldom happens. On completion of a patient's LoS or treatment, the bed becomes available after the patient is discharged home or transferred to other hospital wards.

Once the first version of the simulation model was fully built, the different types of resources were entered into the model. These resources include registration clerks, nurses, doctors, consultants, and beds. These numbers can be changed weekly, daily, or hourly to represent the step-up and step-down of requirements on the MAU. The developed model takes individual patients through time as they arrive and pass through the MAU.

8.3.3 Model Refinement, Verification, and Validation

The process of building the simulation model shown in Fig. 8.2 was one of interactive refinement through a series of meetings with the MAU clinicians. The current flow of patients at each stage was demonstrated and matched with the reality of the MAU. Hence refinements were made to the model to more closely represent the reality of the MAU at each stage. Amongst the significant refinements the modelling of administrative duties by doctors and nurses and a more accurate patient discharge network were included. Once the model had been built, it was also verified by observing how the model was running and checking to see that the distributions of times generated by the model matched the empirical data collected from the MAU. Patients generated were tracked to ensure that they are following the correct route, going to the right places, and treated by the appropriate personnel in the proper order through the model, and these were done by a combination of tag checking and observation of the visual display.

It is observed that the LoS from the simulation matched the patient LoS in the MAU. The model was tested to ensure that the actual system length of stay times is mirrored by the simulation model. The model is run for 4 weeks with a warm-up period of 24 h. Due to the stochastic nature of the results obtained from the simulation it is important therefore to ensure that the results accurately represent the MAU.

In doing so, the model is run for a number of times on each of the scenarios tested, ensuring that each run has a different random number seed. The results from running the model with different random number seeds are then added together. The average of the results is then used in the discussion of the results.

8.3.4 What/If Scenario Investigation

The base model for the simulation is the expanded MAU which at every hour of the day the MAU has 13 nurses, 23 doctors, 7 senior doctors, 2 registration clerks, 58 beds, and 3 consultants from different specialties come in every morning to

conduct the post-take ward round. This provided a more realistic model of the current MAU than the smaller version used in Section 8.2. The simulation is designed to be flexible and capable of testing a wide range of scenarios. Among the investigations requested by the MAU clinicians were the impact of the following future potential changes on the MAU:

- A decrease in the number of nurses, junior, and senior doctors as a result of sickness;
- An increase or decrease in the number of beds;
- A second consultant post-take ward round within 24 h.

The five key simulation outputs identified by the MAU clinicians for measuring the performance of the MAU are

- Patient queue lengths for nurses. This is directly proportional to nurses stress levels.
- Patient queue lengths for doctors. This is directly proportional to doctors stress levels.
- The total length of queues in the system.
- The total waiting time within the queues.
- The number of beds used.

Full details of all the scenario investigations are given in Oddoye et al. (2009). For the purposes of this case study, the scenario of a possible second post-take ward round is chosen for illustrative purposes. Presently the consultants' post-take ward round (ptwr) takes place once every morning on the MAU between the hours of 8.30 and 11.30 am, during which time three consultants from different specialties are assigned to patients. Such patients would have been admitted on the unit already and so if a patient is admitted after the ptwr or his/her name does not appear on the consultant's list of patients to be seen, that patient will have to wait till the following ptwr to be seen by a consultant before discharged home or transferred elsewhere. It is worth noting here that not all MAU patients are assigned to consultants. Currently it takes on average about 18 h from the time of admission before a patient gets to see a consultant on their ptwr. Overall, 77% of all MAU patients will see a consultant while the remaining 23% could be discharged by a senior doctor.

Table 8.4 and Fig. 8.3 show the base-model situation for the ptwr queue. It can be seen from Table 8.4 that on average about 27 patients will be waiting for the consultant ptwr in the morning. These are patients that have been admitted either during a previous ptwr or after. The queue length could increase to a maximum of 50 depending on how many patients are admitted in between post-take ward rounds. It is also possible that no patient will be in a consultant queue although this can only be brief. Figure 8.3 shows the daily build-up of patients waiting for a ptwr.

It is worth noting also that consultant ptwr has a significant impact on beds. Of all the patients who see the consultant, their discharge or transfers to other hospital wards can only take place after the ptwr and so the longer the consultant queue and

Table 8.4 Consultant waiting queue length with a single and twice daily post-take ward round

ptwr frequency	Minimum	Maximum	Mean	Standard deviation
Once daily	0	50	27	13.4
Twice daily	0	33	11	8.9

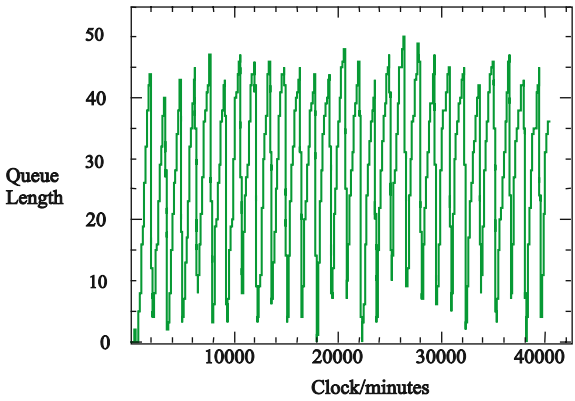

Fig. 8.3 Consultant queue with a single daily post-take ward round

waiting time, the lesser the number of patients leaving the MAU which leads to bed blockage and thus fewer patients being admitted. In an ideal situation, patients should be transferred or discharged home just after the ptwr. However, due to delays in finding beds in other hospital wards and the late arrival of the transport team or relatives, this is not possible. When this situation arises, queues build up among general practitioner referral and A&E referral patients waiting to be admitted onto the MAU. It was found that patients wait on average about 14 h with a standard deviation of 12 h to be transferred to other hospital wards. This can potentially cause severe disruptions to the operations of the MAU. With the consultant ptwr currently happening only once a day, it was important to investigate the effect and impact on MAU patient LoS if a second consultant ptwr is introduced on the MAU between the hours of 5 and 7 pm.

8.3.4.1 Second Consultant Post-Take Ward Round

The introduction of the second consultant ptwr affects the MAU to a great extent and the queues are reduced drastically, as shown in Table 8.5 and Fig. 8.4 compared with Fig. 8.3. The second ptwr is modelled to take place between 5 and 7 pm. There is a general reduction in the consultant queues as well as the waiting times.

The second ptwr results in a significant decrease in queue length from a mean of 27 to 11 patients in the queue. Likewise the waiting times within the queues reduced considerably from an average of about 11–6 h. This significant reduction is much appreciated by MAU management. The reduction in consultant queue lengths means that more patients are seen more rapidly and discharged home thus beds are

Table 8.5 Effect of increasing the weight on beds

Weighting scheme	n_1	n_2	n_3	n_4	n_5	p_1	p_2	p_3	p_4	p_5	B
0.2/0.2/0.2/0.2/0.2	1	2	3	258	0	0	0	0	0	8	63
0.15/0.15/0.15/0.15/0.4	1	2	3	258	0	0	0	0	0	8	63
0.13/0.13/0.13/0.13/0.5	1	2	3	258	0	0	0	0	0	8	63
0.12/0.12/0.12/0.12/0.52	1	2	3	258	0	0	0	0	0	8	63
0.11/0.11/0.11/0.11/0.53	0	0	0	0	0	0	2	2	318	2	57
0.003/0.003/0.003/0.003/0.99	0	0	0	0	0	1	2	20	2037	0	55

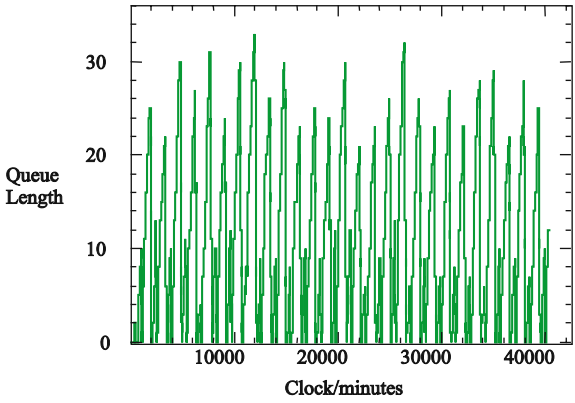

Fig. 8.4 Consultant queue with a twice daily consultant post-take ward round

freed up quickly for other patients to be admitted; however, patient transfer to other specialised wards proved to be largely dependent on availability of beds in those wards and the transfer team.

With beds becoming available for other admissions to be made, it is therefore not surprising that there were no queues whatsoever for beds among general practitioner referral and A&E referral patients. This effect is quite significant and will aid in optimal flow of patients and to an extent minimise the bottlenecks in the system.

Additionally, there was a significant reduction also on the average LoS from a LoS of 21 h to an average of 14 h. Also majority of patients stayed not longer than 24 h with a small minority staying between 24 and 48 h. At most one patient stayed longer than 72 h. A second consultant ptwr was hence a strong recommendation if the MAU is wished to further improve its operations for optimal patient flow and reduce waiting times and queues. This was recommended to the MAU management for implementation.

8.3.5 Post-goal Programme

Scenario analysis proved useful to answer some what/if questions such as the effect of a second daily post-take ward round detailed in the previous section. However,

for the question regarding the correct number of beds on the MAU a more formal multi-objective analysis was needed. Hence, a post-goal programming to analyse the simulation results was constructed.

Results from the simulation output were input into a weighted goal programming model with five objectives based on the key performance measures. These objectives (as listed in the previous section) are

- Patient queue lengths for nurses. Indicative of nurse stress levels.
- Patient queue lengths for doctors. Indicative of doctor stress levels.
- The total length of queues in the system. Indicative of patient satisfaction in the MAU.
- The total waiting time within the queues. Indicative of patient satisfaction in the MAU.
- The number of beds. A key resource and indicative of the cost and acceptability of the proposed solution.

8.3.5.1 Decision Variables

The sole set of decision variables in the goal programme relate to the number of beds allocated to the MAU, which is the resource under investigation. These are algebraically defined as

$$x_i = \begin{cases} 1 & \textit{if the} \text{ MAU } \textit{has } 54 + i \text{ beds} \\ 0 & \text{otherwise} \end{cases} \quad i = 1,...,14$$

This allows the range of 55–68 beds to be investigated, as compared to the current 58 beds on the MAU.

8.3.5.2 Data

The corresponding data taken from an average of the runs from the simulation model with the corresponding notation is notated for the range i=1,. . .,14 as

- N_i: the number of patients queuing to be seen by nurses in a 54 + i bed MAU
- D_i: the number of patients queuing to be seen by doctors in a 54 + i bed MAU
- TQ_i: the total number of patients queuing in a 54 + i bed MAU
- TW_i: the total patient waiting time in a 54 + i bed MAU
- B_i: the number of beds in a 54 + i bed MAU (given by the equation B_i=54 + i)

8.3.5.3 Algebraic Goal Programming Model

The following set of targets were determined in consultation with the MAU clinicians:

- A maximum of 1 patient on average queuing for a nurse
- A maximum of 13 patients on average queuing for a doctor
- A maximum of 35 patients on average queuing in total
- A maximum average total patient queuing time of 1260 min
- A maximum of 55 beds (in fact this was set artificially low in order to ensure Pareto efficiency of the solution and to produce trade-off information)

This led to the following binary weighted goal programming model:

$$Min\ a = \frac{w_1 p_1}{1} + \frac{w_2 p_2}{13} + \frac{w_3 p_3}{35} + \frac{w_4 p_4}{1260} + \frac{w_5 p_5}{55}$$

subject to

$$\sum_{i=1}^{14} N_i x_i + n_1 - \underline{p_1} = 1$$

$$\sum_{i=1}^{14} D_i x_i + n_2 - \underline{p_2} = 13$$

$$\sum_{i=1}^{14} TQ_i x_i + n_3 - \underline{p_3} = 35$$

$$\sum_{i=1}^{14} TW_i x_i + n_4 - \underline{p_4} = 1260$$

$$\sum_{i=1}^{14} B_i x_i + n_5 - \underline{p_5} = 55$$

$$\sum_{i=1}^{14} x_i = 1$$

$$x_i = 0 \text{ or } 1 \text{ i} = 1,...,14,\ n_q, p_q \geq 0\ q = 1,..., 5$$

The hard constraint ensures along with the binary restriction on the decision variables that only one bed level in the MAU is chosen. In fact, this is a use of goal programming for a multi-objective discrete choice problem.

The weighted goal programming variant was chosen for the same reason as for its use in the pre-goal programme detailed in Section 8.2; the MAU clinicians were interested in the trade-off between the different objectives and did not have a priority structure in mind or express a particular need to balance the goals. They did not have a clear a priori set of weights in mind and hence a weight sensitivity analysis was conducted starting with the set of equal weights $w_1 = w_2 = w_3 = w_4 = w_5 = 0.2$. This was not as comprehensive as the formal weight sensitivity algorithm presented in Section 4.5 but nevertheless did investigate on the first two levels, raising a single weight and raising key pairs of weights.

The equal weight model suggested a 63-bed MAU, an increase of five beds on the current 58-bed MAU, which as shown in the first line of Table 8.5 meets the

other four targets. Key results from a sample of a single weight increase are given by Table 8.5. Raising the weight on beds does indeed produce lower bed solutions, with 57- and 55-bed solutions found. However, the trade-off costs are shown to be large, with three of the other four objectives now missing their target in the 57-bed solution. The 55-bed solution is even more extreme and is basically akin to a two-priority model that concentrates on beds first and then considers the other four objectives at a lower priority. The point at which a 63- to 57-bed solution change occurs is around the point in weight space of $w_1 = w_2 = w_3 = w_4 = 0.11$, $w_5 = 0.53$. which indicates that the MAU clinicians should prefer this solution only if they regard the beds objective as being at least $\frac{0.53}{0.11} = 4.82$ as important as any other single objective.

The key results from an example of a two-weight variation are given in Table 8.6, where the weights on total queue length and beds are increased in different proportions. This analysis elicited a new, 58- and 62-bed solutions. These solutions were not visible when just raising the bed weight alone.

Table 8.6 Effect of increasing weights on total queue length and beds in different proportions

Weighting scheme	n_1	n_2	n_3	n_4	n_5	p_1	p_2	p_3	p_4	p_5	B
0.003/0.003/0.495/0.003/0.495	0	0	0	0	0	2	1	316	0	3	58
0.03/0.03/0.55/0.03/0.35	0	0	0	0	0	2	1	316	0	3	58
0.03/0.03/0.56/0.03/0.34	1	0	0	0	0	0	1	0	81	7	62
0.05/0.05/0.65/0.05/0.20	1	2	3	258	0	0	0	0	0	8	63

8.4 Conclusions

This case study has demonstrated a number of aspects of the methodology outlined in Chapter 1–7. It has demonstrated some of the practical issues related to data collection, model building, and liaising with decision makers (in this case the MAU clinicians). Some of the issues involved with building and solving a complex integer goal programme were discussed in Section 8.2. It has also been shown how goal programming alone was not sufficient to answer all of the questions posed by the decision makers, and two of the combinations of goal programming and simulation have been demonstrated. This proved particularly effective in analysing the MAU set-up and patient flow from a multi-objective point of view. The issue of how to handle unknown weights via weight sensitivity analysis has also been demonstrated in the post-goal programme in Section 8.3.4.

The combined goal programming and simulation framework has proved to be a valuable aid to the MAU clinicians in helping them understand the dynamics of their department and for present evidence to argue for appropriate resource levels and changes in organisational policy.

Chapter 9
Case Study: Application of Goal Programming in Portfolio Selection

A classic problem in operational research and finance is the selection of a portfolio of shares, investments, or financial products. The set of shares chosen and their percentage of the total portfolio are the decision variables of this model. The classic models tend to consider the minimisation or some measure of risk and the maximisation of return, with an 'efficient frontier' of possible portfolios being considered in the resulting bi-objective space. Markowitz (1952) introduced the mean–variance model for portfolio selection. In this model, the mean represents average return and variance represents a measure of risk. It can be seen that this problem is a multiple objective one by definition. It can also be expanded to contain further objectives related to issues such as liquidity, portfolio size or composition, social responsibility, or size of dividends (Steuer et al., 2007). If the decision makers (i.e. the portfolio manager or owners) have set goals for levels of return and risk or other measures of portfolio performance then the goal programming framework is a natural choice for modelling and solving this problem.

One of the earliest goal programming portfolio selection models is given by Lee (1972) who uses lexicographic goal programming to solve a portfolio selection from a selection of 53 candidate shares considering the objectives of return, risk, and dividends. This is converted into a weighted goal programming model by Tamiz et al. (1996). Goal programming based portfolio selection models from the 1980s include those of Levary and Avery (1984) and Lee and Chesser (1980). This chapter reviews methods in the past two decades and presents a multi-stage framework for modelling portfolio selection models by goal programming.

9.1 Overview of Issues and Objectives in Multi-objective Portfolio Selection

Some researchers involved in the mean–variance model of Markowitz for portfolio selection have only focused on portfolio selection as risk-adjusted return with little or no effort being directed to the inclusion of other essential factors. Therefore, the usual portfolio analysis assumes that investors are interested only with returns attached to specific levels of risk when selecting their portfolios. In a wide variety of

D. Jones, M. Tamiz, *Practical Goal Programming*, International Series in Operations Research & Management Science 141, DOI 10.1007/978-1-4419-5771-9_9,

applications, neither part of this restriction is desirable or important. Consequently, a portfolio analysis model that includes more essential factors in the analysis of portfolio problems is a more realistic approach. Some of these factors include liquidity, asset class, asset region, micro economics, macro economics, and market dynamics.

Markowitz (1952) suggests that investors should consider risk and return together and determine the allocation of funds among investment alternatives on the basis of the trade-off between them. Later on the recognition that many investors evaluate performance relative to a benchmark led to the idea of portfolio selection based on return and relative risk (Cremers et al., 2005). For many investors, both approaches fail to yield satisfactory results. Chow (1995) emphasises that the portfolio optimisation techniques can assist in the search of portfolio that best suits each investor's particular objectives.

An alternative to Markowitz model is the mean absolute deviation (MAD) model, proposed by Konno and Yamazaki (1991). While the Markowitz model assumes normality of stock returns, the MAD model does not make this assumption. The MAD model also minimises a measure of risk, where the measure is the mean absolute deviation (Kim et al., 2005; Konno and Koshizuka, 2005). Konno and Yamazaki (1991) further developed the MAD model into an equivalent goal programming model with n shares and T time periods:

$$\text{Min} \sum_{t=1}^{T} (n_t + p_t)$$

subject to

$$\sum_{j=1}^{n} (r_{jt} - \bar{r}_j)x_j + \underline{n_t} - \underline{p_t} = 0 \qquad t = 1,...,T$$

$$\sum_{j=1}^{n} \bar{r}_j x_j \geq \rho$$

$$\sum_{j=1}^{n} x_j = 1$$

$$x_j \geq 0 \quad j = 1,...,n \quad n_t, p_t \geq 0 \quad t = 1,...,T$$

where r_{jt} is the return of share j in period t, $\bar{r}_j$ is the average return of share j, and ρ is a desired level of return for the portfolio. The main advantage of this model over the mean–variance model is that it does not require a matrix of co-variances and hence some inaccuracies and non-linearity are eliminated. It only requires historical data on share's returns and their expected returns.

Figure 9.1 illustrates the number of publication of research papers in the area of portfolio selection using goal programming.

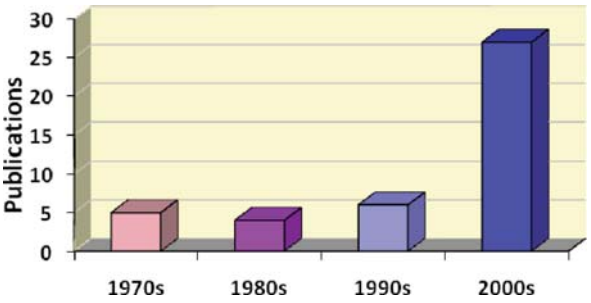

Fig. 9.1 Number of publications using goal programming for portfolio selection in the period 1970–2009

9.1.1 Lexicographic Goal Programming in Portfolio Selection Models

Lexicographic goal programming could deal with different priority levels in the portfolio selection problem, in which goals are included according to their importance of achievement in the model. For example, a portfolio selection model using lexicographic goal programming may include the following priority structure:

1. Maximising the portfolio's expected return, while minimising some measurement of portfolio's risk
2. Minimising other portfolios's risks (e.g. the systematic risk as measured by beta coefficient)
3. Minimising the portfolio's cost of rebalancing
4. Some other factors

Many authors developed lexicographic goal programming models for portfolio selection, particularly during the 1970s and 1980s (for example, Lee and Lerro, 1973; Kumar et al., 1978; Levary and Avery, 1984). Other applications of lexicographic goal programming for portfolio selection are developed recently within the mutual funds industry (Sharma and Sharma, 2006).

9.1.2 Chebyshev Goal Programming in Portfolio Selection Models

The model of Chebyshev goal programming portfolio selection usually seeks the minimisation of the maximum deviation from any single goal in portfolio selection. In other words, it seeks the solution that minimises the worst unwanted deviation from any single goal. Some authors focus historically on developing portfolio selection models using Minmax goal programming variant. Deng et al. (2005), amongst others, develop Minmax goal programming model for portfolio selection.

9.1.3 Fuzzy Goal Programming in Portfolio Selection Models

In a fuzzy goal programming portfolio selection model, the decision maker is required to specify an aspiration level for each objective in the model in which aspiration levels are not known precisely. In this case, an objective with an imprecise level can be treated as a fuzzy goal (Yaghoobi and Tamiz, 2006).

In this context, Watada (1997) argues that Markowitz's approach to portfolio selection has difficulty in resolving the situation where the aspiration level and utility given by the decision makers cannot be defined exactly. Therefore, a fuzzy portfolio selection to overcome such difficulty is proposed. The fuzzy portfolio selection model enables a solution to be obtained which realises the best within a vague aspiration level and the goal given as a fuzzy number, which is obtained from the expertise of the decision makers.

As detailed in Section 9.2.2, Perez et al. (2007) argue that portfolio selection problem is characterised by imprecision and/or vagueness inherent in the required data, in which they proposed a three-stage model, in order to mitigate such problems, based on a multi-index model and considering several market scenarios described in an imprecise way by an expert.

Perez et al. (2007) discuss how the proposed fuzzy model allowed the decision maker to select a suitable portfolio taking into account the uncertainty related to the market scenarios and the imprecision and/or vagueness associated with the model data.

Parra et al. (2001) deal with the optimum portfolio for a private investor with emphasis on three criteria, which are expected return of the portfolio, the variance return of the portfolio, and the portfolio's liquidity measured as the possibility of converting an investment into cash without any significant loss in value. They formulated these three objectives as a goal programming problem using fuzzy terms since they cannot be defined exactly from the point of view of the investors. Parra et al. (2001) propose a method to determine portfolios with fuzzy attributes that are set equal to fuzzy target levels. Their solution is based on the investor's preferences and on goal programming techniques.

Allen et al. (2003) investigate the notion of fuzziness with respect to funds allocation. They found that the boundary between the preference sets of an individual investor, for funds allocation between a risk-free asset and the risky market portfolio, tends to be rather fuzzy as the investors continually evaluate and shift their positions, unless it is a passive buy-and-hold kind of portfolio.

Inuiguchi and Ramik (2000) emphasise in their paper that the real-world problems are not usually so easily formulated as mathematical models or fuzzy models. Sometimes qualitative constraints and/or objectives are almost impossible to represent in mathematical forms. In such a situation, a fuzzy solution satisfying the given mathematical requirements is very useful in a sense of weak focus in the feasible area. Inuiguchi and Ramik (2000) apply fuzzy programming in portfolio selection problems and they found that decision maker can select the final solution from the fuzzy solution considering implicit and mathematically weak requirements.

9.2 Multi-phase Portfolio Models

The multi-phase goal programming models use goal programming as a portfolio selection tool in one phase of the modelling and solution process. They also use goal programming or other operational research techniques to produce parameters or data for the goal programme or to analyse the results of the goal programme in other phases of the modelling and solution process. They fall into the category of methods that combine and integrate goal programming discussed in Chapter 7.

9.2.1 The Two-Phase Model of Tamiz et al.

The multi-phase approach was first proposed by Tamiz et al. (1996). They propose a two-phase approach with both phases based on goal programming. The example used is a selection of shares from the British FTSE 100 index with data taken from the period 1998–1992. The 97 ($K = 1, ..., 97$) shares that remained in the FTSE 100 index throughout that period were considered as candidates for inclusion in the portfolio. The idea is to break down the risk measure into sensitivity against movement in key economic indices.

Phase 1

The first phase of the model involved using goal programming as a statistical tool, as described in Section 7.1. A goal programme was solved for each share to work out its sensitivity to each of the following ($j = 1, ..., 12$) key economic indices based on data from the collection period:

- The UK interest rate
- The US Interest rate
- The German Interest rate
- The US Inflation rate
- The German Inflation rate
- The Dow Jones Index
- The Nikkei Average
- The Hang Seng Index
- The Oil Price
- The Gold Price
- The UK House Price index
- The Sterling Index

The data was then separated into 19 time periods ($i = 1, ..., 19$) and the following variables and data:

Data

C_{ij} = Change in factor j in period i
R_i = Change in price of share under consideration in period i

Decision Variables

y_j = predicted sensitivity of share under consideration to factor j

The following goal programming model was then solved 97 times, once for each share:

$$\text{Min } a = \sum_{i=1}^{19} (n_i + p_i)$$

subject to

$$\sum_{i=1}^{12} c_{ij} y_j + \underline{n_i} - \underline{p_i} = R_i \qquad i = 1,...,19$$
$$y_j \text{ free } j = 1,...,12$$

A further goal programming model was also solved to give the sensitivity of the FTSE 100 index itself to the 12 economic indices:

$$\text{Min } a = \sum_{i=1}^{19} (n_i + p_i)$$

subject to

$$\sum_{i=1}^{12} c_{ij} y_j + \underline{n_i} - \underline{p_i} = F_i \qquad i = 1,...,19$$
$$y_j \text{ free } j = 1,...,12$$

where F_i = Change in FTSE 100 index in period i.

The data collected from this set of L_1 regressions is inputted into the phase two model as the data:

S_{kj} = Sensitivity of share k to factor j
FTSE_j = Sensitivity of FTSE 100 index to factor j

Phase 2

A goal programming model is now constructed to select a portfolio of shares. A number of future economic scenarios ($l = 1,..., N$) are constructed and the goals of the model are as follows:

- Aim for the portfolio to have the same level of sensitivity as the FTSE 100 index to each of the economic factors. This can be through a surrogate for controlling the systematic risk of the portfolio.

- Ensure that the portfolio makes a profit of at least T_l under scenario l. This can be thought of as a surrogate for ensuring a sufficient level of return.
- Ideally, no more than 5% of the portfolio should be invested in any single share. Hard constraints of no more than 9% in any single share and no more than 25% in any sector of the market are also imposed. Together these act to control the unsystematic risk of the portfolio.

These objectives are implemented via the following weighted goal programming formulation:

$$\text{Min}\, a = W_1 \sum_{j=1}^{12} (nf_j + pf_j) + W_2 \sum_{k=1}^{N} ns_l + W_3 \sum_{k=1}^{97} pp_k$$

subject to

$$\sum_{k=1}^{97} \frac{S_{kj} x_k U_{\text{FTSE}}}{\text{FTSE}_j \; U_k} + \underline{nf_j} - \underline{pf_j} = 1 \quad j = 1,...,12$$

$$\sum_{j=1}^{12} M_{ji} \sum_{k=1}^{97} \frac{100 S_{kj} x_k}{U_k} + \underline{ns_l} - ps_l = T_l \quad l = 1,...,N$$

$$x_k + np_k - \underline{pp_k} = 0.05 \quad k = 1,...,97$$

$$\sum_{k=1}^{97} x_k = 1$$

$$\text{Sector Constraints} \leq 0.25$$

$$0 \leq \text{x}_\text{k} \leq 0.09 \quad k = 1,...,97$$

with the following new data, decision variables, and deviational variables introduced:

Data

W_1, W_2, W_3 form the weighting scheme, each of which have a preferential and a normalisation component.

U_{FTSE} is the value of the FTSE 100 index at the end of the data collection period.

U_k is the value of share k at the end of the data collection period.

The previous two constants are included to ensure internal scaling amongst the shares.

M_{jl} is the movement of factor j under economic scenario l.

Decision and Deviation Variables

x_k is the proportion of the portfolio invested in share k.

nf_j is the negative deviation from the matching FTSE sensitivity target level for factor j.

pf_j is the positive deviation from the matching FTSE sensitivity target level for factor j.
ns_l is the negative deviation from the return target level under scenario l.
ps_l is the negative deviation from the return target level under scenario l.
np_k is the negative deviation from the 5% target level of investment in a single share.
pp_k is the positive deviation from the 5% target level of investment in a single share.

This model is tested in Tamiz et al. (1996) against four quarters in 1993 under a three-scenario ($N = 3$) model. It is shown to outperform the FTSE 100 index by values of between 53.6 and 556.7% over those four quarters. The Tamiz et al. (1996) model served as the basis for future works on multi-phase goal programming models for portfolio selection.

9.2.2 The Three-Phase Model of Perez et al.

The model of Perez et al. (2007) is a three-phase model that refines the process of Tamiz et al. (1996). The methodology is applied to the problem of selecting a portfolio of Spanish mutual funds from amongst 26 candidates. A set of 11 possible economic indices that could affect the mutual funds performance were investigated. These are given in Table 9.1.

Principal component analysis is then performed and it is concluded that the first three factors (IGBM, IBEX35, and FTSE) are the three factors which correlate to the return from the mutual funds above the threshold level set by the modeller.

Table 9.1 Economic indices that potentially affect the performance of Spanish mutual funds – from Perez et al. (2007)

Index		Description	
IGBM	I_1	General stock market index return	Quarterly index (31 December 1985=100)
IBEX35	I_2	Stock market index return	Quarterly index (29 December 1989=3000)
FTSE	I_3	Stock market index: FTSE all share price index return	Quarterly. Original series. Index base 04/1962=100
DAX	I_4	Stock market index: DAX all share price index return	Quarterly index. December 1997=1000
INTDEPUB10	I_5	Government debt rate: 10 years	Quarterly. Percentage
INTDEPUB3M	I_6	Government debt rate: 3–6 months	Quarterly. Percentage
INTERBAN1	I_7	12 months interbank rate	Quarterly. Percentage
INTOFICI	I_8	Official reference rate	Quarterly. Percentage
LETRAS1	I_9	Debt market: treasure bills rates 1 year	Quarterly. Percentage
BONOS3	I_{10}	Debt market: bonds rates 3 years	Quarterly. Percentage
OBLIGS10	I_{11}	Debt market: liabilities rates 10 years	Quarterly. Percentage

Phase 1

The first phase uses goal programming as an L_1 regression tool as described in Section 3.1. The purpose of each of the 26 goal programming models ($i = 1, ..., 26$) is to determine the sensitivity of each of the possible mutual funds to the three economic factors ($f = 1, ..., 3$) identified by the principal component analysis. The data used is taken over 20 quarters ($t = 1, ..., 20$) from the period 1996–2000. The generic mathematical formulation of the ith goal programming model is given as

$$\text{Min } a = \sum_{t=1}^{20} (n_t + p_t)$$

subject to

$$A_i + \sum_{f=1}^{3} \beta_{if} F_{tf} + \underline{n_t} - \underline{p_t} = R_{it} \qquad t = 1,..., 20$$
$$A_i \quad \text{free}, \ \beta_{if} \ \text{free} \ \ i = 1,...,26 \quad f = 1,..., 3$$

where the decision variables are A_i which is the expected value of returns not related to any index and β_{if} which gives the sensitivity of mutual fund i to factor f. R_{it} is the return of mutual fund i over period t. n_t and p_t are the deviational variables which are both penalised in the achievement function.

Phase 2

The second phase, similarly to Tamiz et al. (1996), makes use of scenarios over the factors. However, these are now fuzzy scenarios which are interactively constructed with the help of an economic analyst. Instead of a single goal programming model at this stage, a possibilistic multi-objective programme (Arenas et al., 2005) is constructed for each fuzzy scenario. This is in keeping with the recent trend for goal programming based portfolio selection models to use the fuzzy variant. Each possibilistic multi-objective programme is classical in the sense that it has two objectives – to maximise the expected return and to minimise the level of systematic risk. Unsystematic risk is also controlled by bounds on the amount of the portfolio that can be invested in any single mutual fund.

Phase 3

The above second phase formulation is not a solution to the decision problem, as in fact it chooses one portfolio for each of the fuzzy scenarios. Therefore a discrete choice now needs to be solved in order to choose the most preferred of these portfolios of mutual funds. Perez et al. (2007) use the ELECTRE I method with the additional consideration of regret in order to solve this problem and choose a portfolio most in line with decision maker preferences.

9.3 Summary

This chapter has shown that goal programming is a versatile and appropriate tool for the portfolio selection problem. Due to the nature of the problem, several variants of goal programming have been utilised by different authors. The most popular variant in recent years has been fuzzy goal programming, due to the fuzzy nature of the data and target levels required by the decision maker(s). It is hoped that goal programming continues to be used as a portfolio analysis and selection tool. In particular, it is hoped that its flexibility and ability to handle multiple objectives will encourage modellers to go beyond the standard risk return bi-objective model and include other criteria in their models.

References

Abd El-Wahed WF, Lee SM (2006) Interactive fuzzy goal programming for multi-objective transportation problems, *Omega-International Journal of Management Science*, **34**, 158–166.

Allen J, Bhattacharya S, Smarandache F (2003) Fuzziness and funds allocation in portfolio optimisation, *International Journal of Social Economics*, **30**, 619–632.

Akoz O, Petrovic D (2007) A fuzzy goal programming method with imprecise goal hierarchy, *European Journal of Operational Research*, **181**, 1427–1433.

Amiri M, Salehi-Sadaghiani J (2008) A methodology for optimizing statistical multi-response problems using fuzzy goal programming, *Scientia Iranica,* **15**, 389–397.

Applegate DL, Bixby RE, Chvatel V, Cook WJ (2007) *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics*, Princeton University Press, Princeton, NJ.

Arenas M, Bilbao A, Perez B, Rodriguez MV (2004) Fuzzy extended lexicographic Goal Programming, in *Soft Methodology and Random Information Systems*, LopezDiaz M, Gil MA, Grzegorzewski P, Hryniewicz O, Lawry J (Eds.), *Advances in Soft Computing*, Springer-Verlag, Berlin, 543–550.

Arenas M, Bilbao A, Perez B, Rodriguez MV (2005) Solving a multiobjective possibilistic program through compromise programming, *European Journal of Operational Research*, **164**, 748–759.

Arora SR, Gupta R (2008) Interactive fuzzy goal programming approach for bilevel programming problem, *European Journal of Operational Research*, **194**, 368–376.

Audet C, Carrizosa E, Hansen P (2004) An exact method for fractional goal programming, *Journal of Global Optimization*, **29**, 113–120.

Azaiez MN, Al Sharif SS (2005) A 0–1 goal programming model for nurse scheduling, *Computers & Operations Research*, **32**, 491–507.

Baek W, Ignizio JP (1993) Pattern-classification via linear-programming, *Computers and Industrial Engineering*, **25**, 393–396.

Baykasoglu A (2005) Preemptive goal programming using simulated annealing, *Engineering Optimization*, **37**, 49–63.

Belton V, Stewart TJ (2002) *Multiple Criteria Decision Analysis: An Integrated Approach*, Kluwer Academic Publishers, Boston, MA.

Bhattacharya A (2006) A goal programming approach for developing pre-harvest forecasts of crop yield, *Journal of the Operational Research Society*, **57**, 1014–1017.

Brans JP, Vincke P, Mareschal B (1985) A preference ranking organization method, *Management Science*, **31**, 647–656.

Bryson N (1995) A goal programming method for generating priority vectors, *Journal of the Operational Research Society*, **46**, 641–648.

Byron M (2004) *Satisficing and Maximizing: Moral Theorists on Practical Reason*, Cambridge University Press, Cambridge.

Caballero R, Hernandez M (2006) Restoration of efficiency in a goal programming problem with linear fractional criteria, *European Journal of Operational Research*, **172**, 31–39.

D. Jones, M. Tamiz, *Practical Goal Programming*, International Series in Operations Research & Management Science 141, DOI 10.1007/978-1-4419-5771-9,

Caballero R, Luque M, Molina J, Ruiz F (2005) MOPEN: A computational package for linear multiobjective and goal programming problems, *Decision Support Systems*, **41**, 160–175.

Caballero R, Rey L, Ruiz F (1998) Lexicographic improvement of the target values in convex goal programming, *European Journal of Operational Research*, **107**, 644–655.

Caballero R, Ruiz F, Uria MV, Romero C (2006) Interactive meta-goal programming, *European Journal of Operational Research*, **175**, 135–154.

Calvete HI, Gale C, Oliveros MJ, Sanchez-Valverde B (2007) A goal programming approach to vehicle routing problems with soft time windows, *European Journal of Operational Research*, **177**, 1720–1733.

Can EK, Houck MH (1984) Real time reservoir operations by goal programming, *Journal of Water Resources Planning and Management*, **110**, 297–309.

Chan FTS, Swarnkar R (2006) Ant colony optimization approach to a fuzzy goal programming model for a machine tool selection and operation allocation problem in an FMS, *Robotics and Computer-Integrated Manufacturing*, **22**, 353–362.

Chang CT (2006) Mixed binary interval goal programming, *Journal of the Operational Research Society*, **57**, 469–473.

Charnes A, Collomb B (1972) Optimal economic stabilization policy: Linear goal-interval programming models, *Socio-Economic Planning Sciences*, **6**, 431–435.

Charnes A, Cooper WW (1961) *Management Models and Industrial Applications of Linear Programming*, John Wiley & Sons, New York.

Charnes A, Cooper WW (1962) Programming with linear fractional functionals, *Naval Research Logistics Quarterly*, **9**, 181–186.

Charnes A, Cooper WW, Ferguson R (1955) Optimal estimation of executive compensation by linear programming, *Management Science*, **1**, 138–151.

Charnes A, Cooper WW, Harrald J, Karwan K, Wallace W (1976) A goal interval programming model for resource allocation in a marine environmental protection problem, *Journal of Environmental Economics and Management*, **3**, 347–362.

Charnes A, Cooper WW, Rhodes E (1978) Measuring the efficiency of decision making units, *European Journal of Operational Research*, **2**, 429–444.

Charnes A, Cooper WW, Sueyoshi T (1988) A goal programming constrained regression review of the bell system breakup, *Management Science*, **34**, 1–26.

Chow G (1995) Portfolio selection based on return, risk, and relative performance, *Financial Analysts Journal*, March–April, 54–60.

Cooper WW (2005) Origins, uses of, and relations between goal programming and data envelopment analysis, *Journal of Multi-Criteria Decision Analysis*, **15**, 3–11.

Cremers J-H, Kritzman M, Page S (2005) Optimal hedge fund allocations, *Journal of Portfolio Management*, Spring, 70–81.

Da Silva LMS, Rodriguez LCE, Caixeta JV, Bauch SC (2006) Fitting a taper function to minimize the sum of absolute deviations, *Scientia Agricola*, **63**, 460–470.

Deb K (2001) *Multi-Objective Optimization Using Evolutionary Algorithms*, J Wiley & Sons, New York.

Deb K, Mohan M, Mishra S (2003) Towards a quick computation of well-spread Pareto-optimal solutions, in *Evolutionary Multi-Criterion Optimization*, *Proceedings*, Fonseca CM, Fleming PJ, Zitzler E, Deb K (Eds.), Thiele L, *Lecture Notes in Computer Science*, **2632**, 222–236.

Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Transactions On Evolutionary Computation*, **6**, 182–197.

Deng X-T, Li Z-F, Wang S-Y (2005) A minimax portfolio selection strategy with equilibrium, *European Journal of Operational Research*, **166**, 278–292.

Despotis DK (1996) Fractional minmax goal programming: A unified approach to priority estimation and preference analysis in MCDM, *Journal of the Operational Research Society*, **47**, 989–999.

Dharmapala PS, Ghosh JB, Saber HM (2007) Market- and merit-based adjustment of faculty salaries, *Asia-Pacific Journal of Operational Research*, **24**, 1–19.

Dufo-Lopez R, Bernal-Agustin JL (2008) Multi-objective design of PV-wind-diesel-hydrogen-battery systems, *Renewable Energy*, **33**, 2559–2572.

Dyer JS (1973) Interactive goal programming, *Management Science*, **19**, 62–70.

Ehrgott M (2005) *MultiCriteria Optimization*, Springer, Berlin.

Ehrgott M, Gandibleux X (2002) Multiobjective combinatorial optimization – theory, methodology, and applications, *Multi-Criteria Optimization – State of the Art Annotated Bibliographic Surveys*, Kluwer Academic Publishers, Dordrecht, 369–444.

Ekinci Y (2007) Demand assignment: A DEA and goal programming approach, in *Applied Mathematics for Science and Engineering*, Demiralp M, Udriste C, Bognar G, Soni R, Nassar H (Eds.), *World Scientific and Engineering Acad and Soc*, Athens, Greece, 394–397.

Eldabi T, Paul RJ, Young T (2007) Simulation modelling in healthcare: Reviewing legacies and investigating futures, *Journal of the Operational Research Society*, **58**, 262–270.

Famuyiwa O, Monplaisir L, Nepal B (2008) An integrated fuzzy-goal-programming-based framework for selecting suppliers in strategic alliance formation, *International Journal of Production Economics*, **113**, 862–875.

Flavell RB (1976) A new goal programming formulation, *Omega*, **4**, 731–732.

Freed N, Glover F (1981) Simple but powerful goal programming-models for discriminant problems, *European Journal of Operational Research*, **7**, 44–60.

Freed N, Glover F (1986) Resolving certain difficulties and improving the classification power of LP discriminant-analysis formulations, *Decision Sciences*, **17**, 589–595.

French S, Maule J, Papamichail N (2009) *Decision Behaviour*, *Analysis, and Support*, Cambridge University Press, Cambridge.

Gardiner L, Steuer R (1994) Unified interactive multiple objective programming, *European Journal of Operational Research*, **74**, 391–406.

Gass SI (1986) A process for determining priorities and weights for large scale linear goal programmes, *Journal of the Operational Research Society,* **37**, 779–785.

Ghomi SMTF, Ashjari B (2002) A heuristic method for resource constrained allocation in multi-project scheduling, *Iranian Journal of Science and Technology*, **26**, 111–116.

Glover F, Sueyoshi T (2009) Contributions of Professor William W. Cooper in operations research and management science, *European Journal of Operational Research*, **197**, 1–16.

Hannan EL (1980) Nondominance in goal programming, *INFOR*, **18**, 300–309.

Hannan EL (1981) On fuzzy goal programming, *Decision Sciences*, **12**, 522–531.

Hemaida RS, Kwak NK (1994) A linear goal programming-model for transshipment problems with flexible supply-and-demand constraints, *Journal of the Operational Research Society*, **45**, 215–224.

Ho W, Emrouznejad A (2009) Multi-criteria logistics distribution network design using SAS/OR, *Expert Systems with Applications*, **36**, 7288–7298.

Hwang CL, Masud ASM (1979) *Multiple Objective Decision Making – Methods and Applications*. Springer, New York.

Ignizio JP (1976) *Goal Programming and Extensions*, Lexington Books, Lexington, MA.

Ignizio JP (1982) *Linear Programming in Single and Multiple Objective Systems*, Prentice-Hall, Upper Saddle River, NJ.

Ignizio JP (1985) An algorithm for solving the linear goal programming problem by solving its dual, *Journal of the Operational Research Society*, **36**, 507–515.

Ignizio JP (2004) Optimal maintenance headcount allocation: An application of Chebyshev goal programming, *International Journal Of Production Research*, **42**, 201–210.

Ignizio JP, Cavalier T (1994) *Linear Programming*, Prentice-Hall, Upper Saddle River, NJ.

Ignizio JP, Perlis JH (1979) Sequential linear goal programming: Implementation via MPSX. *Computers and Operations Research*, **6**, 141–145.

Ijiri Y (1965) *Management Goals and Accounting for Control*, North Holland, Amsterdam.

Inuiguchi M, Ramik J (2000) Possibilistic linear programming: A brief review of fuzzy mathematical programming and a comparison with stochastic programming in portfolio selection problem, *Journal of Fuzzy Sets and Systems*, **111**, 3–28.

Jääskeläinen V (1976) *Linear Programming and Budgeting*, Petrocelli-Charter, New York.
Janssen R, van Herwijnen M, Stewart TJ, Aerts JCJH (2008) Multiobjective decision support for land-use planning, *Environment and Planning B-Planning & Design*, **35**, 740–756.
Jones DF (1995) *The Design and Development of an Intelligent Goal Programming System*, Ph.D. thesis, University of Portsmouth, UK.
Jones DF, Collins A, Hand C (2007) A classification model based on goal programming with non-standard preference functions with application to prediction of cinema-going behaviour, *European Journal of Operational Research*, **177**, 515–524.
Jones DF, Mardle SJ (2004) A distance-metric methodology for the derivation of weights from a pairwise comparison matrix, *Journal of the Operational Research Society,* **55**, 869–875.
Jones DF, Tamiz M (1995) Improving the flexibility of goal programming via preference modelling techniques, *Omega*, **23**, 41–48.
Jones DF, Tamiz M (1997) An example of good modelling practice in goal programming: Means for overcoming incommensurability, in *Lecture Notes in Economics and Mathematical Systems*, Caballero R, Ruiz F (Eds.), Vol. 455, Springer, Berlin, 29–37.
Jones DF, Tamiz M (2002) Goal Programming in the period 1990–2000, in *Multi-Criteria Optimization: State of the Art Annotated Bibliographic Surveys*, Ehrgott M, Gandibleux X (Eds.), Kluwer, Dordrecht, 129–170.
Jones DF, Tamiz M, Mirrazavi SK (1998) Intelligent solution and analysis of goal programmes: The GPSYS system, *Decision Support Systems*, **23**, 329–332.
Kahraman C, Buyukozkan G (2008) A combined fuzzy AHP and fuzzy goal programming approach for effective six-sigma project selection, *Journal of Multiple-Valued Logic and Soft Computing*, **14**, 599–615.
Kim J, Kim Y, Shin K (2005) An algorithm for portfolio optimization problem, *INFORMATICA*, *Institute of Mathematics and Informatics*, **16**, 93–106.
Konno H, Koshizuka T (2005) Mean-absolute deviation model, *IIE Transactions*, **37**, 893–900.
Konno H, Yamazaki H (1991) Mean-absolute deviation portfolio optimization and its applications to Tokyo stock market, *Journal of Management Science*, **37**, 519–531.
Korhonen P, Seppo S, Steuer R (1997) A heuristic for estimating nadir criterion values in multiple objective linear programming, *Operations Research*, **45**, 751–757.
Kumar P, Philippatos G, Ezzell J (1978) Goal programming and selection of portfolio by dual-purpose funds, *Journal of Finance*, **33**, 303–310.
Kwak NK, Schniederjans MJ (1985) Goal programming solutions to transportation problems with variable supply and demand, *Socio-Economic Planning Sciences*, **19**, 95–100.
Law AM, Kelton WD (2000) *Simulation Modeling and Analysis*, 3rd Ed., McGraw Hill, New York.
Lee SM (1972) *Goal Programming for Decision Analysis*, Auerback, Philadelphia, PA.
Lee SM, Chesser DL (1980) Goal programming for portfolio selection, *Journal of Portfolio Management*, **6**, 23–25.
Lee S, Lerro A (1973) Optimizing the portfolio selection for mutual funds, *Journal of Finance*, **28**, 1086–1101.
Leung SCH (2007) A non-linear goal programming model and solution method for the multi-objective trip distribution problem in transportation engineering, *Optimization and Engineering*, **8**, 277–298.
Leung SCH, Chan SSW (2009) A goal programming model for aggregate production planning with resource utilization constraint, *Computers & Industrial Engineering*, 1053–1064.
Levary RR, Avery ML (1984) On the practical application of weighting equities in a portfolio via goal programming, *Opsearch*, **21**, 246–261.
Li X, Beullens P, Jones D, Tamiz M (2009) An integrated queuing and multi-objective bed allocation model with application to a hospital in China, *Journal of the Operational Research Society*, **60**, 330–338.
Liang TF (2007) Applying fuzzy goal programming to production/transportation planning decisions in a supply chain, *International Journal of Systems Science*, **38**, 293–304.
LINDO (2009) Lindo Systems Website, accessed 26th August 2009.

LINGO (2009) Lingo optimization modeling software for linear, nonlinear, and integer programming. Available from www.lindo.com, accessed 10 February 2010.

Love CE, Lam KF (1994) Classifying and controlling errors in forecasting using multiple criteria goal programming, *Computers & Operations Research*, **21**, 979–989.

Markowitz H (1952) Portfolio selection, *Journal of Finance*, **7**, 77–91.

Martel JM, Aouni B (1990) Incorporating the decision-makers preferences in the goal-programming model, *Journal of the Operational Research Society*, **41**, 1121–1132.

Mathirajan M, Ramanathan R (2007) A (0–1) goal programming model for scheduling the tour of a marketing executive, *European Journal of Operational Research*, **179**, 554–566.

Mavrotas G, Georgopoulou E, Mirasgedis S, Sarafidis Y, Lalas D, Hontou V, Gakis N (2009) Multi-objective combinatorial optimisation for selecting best available techniques (BAT) in the industrial sector: The COMBAT tool, *Journal of the Operational Research Society*, **60**, 906–920.

Miettinen K (1999) *Non-Linear Multiobjective Optimization*, Kluwer Academic Publishers, Dordrecht.

Min H, Storbeck J (1991) On the origin and persistence of misconceptions in goal programming, *Journal of the Operational Research Society*, **42**, 301–312.

Mirrazavi SK, Jones DF, Tamiz M (2001) A comparison of genetic and conventional methods for the solution of integer goal programmes, *European Journal of Operational Research*, **132**, 594–602.

Mishra S, Prakesh, Tiwari MK, Lashkari RS (2006) A fuzzy goal-programming model of machine-tool selection and operation allocation problem in FMS: A quick converging simulated annealing-based approach, *International Journal of Production Research*, **44**, 43–76.

Nakayama H, Kagaku N (1998) Pattern classification by linear goal programming and its extensions, *Journal of Global Optimization*, **12**, 111–126.

Nakayama H, Yun YB, Asada T, Yoon M (2005) MOP/GP models for machine learning, *European Journal of Operational Research*, **166**, 756–768.

Narasimhan R (1980) Goal programming in a fuzzy environment, *Decision Sciences*, **11**, 325–336.

Nepal B, Monplaisir L, Singh N (2006) A methodology for integrating design for quality in modular product design, *Journal of Engineering Design*, **17**, 387–409.

Oddoye JP, Tamiz M, Jones DF, Schmidt P (2009) Combining simulation and goal programming for healthcare planning in a medical assessment unit, *European Journal of Operational Research*, **193**, 250–261.

Oddoye JP, Yaghoobi MA, Tamiz M, Jones DF, Schmidt P (2007) A multi-objective model to determine efficient resource levels in a medical assessment unit, *Journal of the Operational Research Society*, **58**, 1563–1573.

Orgyczak W (2001a) Comments on Romero C, Tamiz M, Jones DF (1998) Goal programming, compromise programming and reference point method formulations: Linkages and utility interpretations, *Journal of the Operational Research Society*, **52**, 960–962.

Ogryczak W (2001b) Comments on Romero C, Tamiz M, Jones DF (1998) Goal programming, compromise programming and reference point method formulations: Linkages and utility interpretations – Comments on reply of Romero et al., *Journal Of The Operational Research Society*, **52**, 963–964.

Ogryczak W (2001c) On goal programming formulations of the reference point method, *Journal of the Operational Research Society*, **52**, 691–698.

Ozcan E, Bilgin B, Korkmaz, E (2008) A comprehensive analysis of hyper-heuristics, *Intelligent Data Analysis*, **12**, 3–23.

Panda S, Banerjee K, Basu M (2005) Determination of EOQ of multi-item inventory problems through nonlinear goal programming with penalty function, *Asia-Pacific Journal Of Operational Research*, **22**, 539–553.

Pareto V (1896) *Course d'Economie Politique*, Rouge, Lausannes.

Park SK, Shin Y, Lee WC (2007) Goal-Pareto based NSGA for optimal reconfiguration of cognitive radio systems, *Proceedings of 2nd IEEE International Conference on Cognitive Radio Oriented Wireless Networks and Communications*, 147–153.

Parra M, Terol A, UriaM (2001) A fuzzy goal programming approach to portfolio selection, *European Journal of Operational Research*, **133**, 287–297.

Perez Gladish B, Jones DF, Tamiz M, Bilbao Terol, A (2007) An interactive three stage model for mutual fund portfolio selection, *Omega-International Journal of Management Science*, **35**, 75–88.

Petrovic D, Akoz O (2008) A fuzzy goal programming approach to integrated loading and scheduling of a batch processing machine, *Journal of the Operational Research Society*, **59**, 1211–1219.

Proudlove NC, Black C, Fletcher A (2007) OR and the challenge to improve the NHS: Modelling for insight and improvement in in-patient flows, *Journal of the Operational Research Society*, **58**, 145–158.

Rawls J (1973) *A Theory of Justice*, Oxford University Press, Oxford.

Reeves CR (1995) *Modern heuristic techniques for combinatorial problems*, McGraw Hill.

Robinson S (2004) *Simulation: The Practice of Model Development and Use* (2nd ed.), John Wiley and Sons Ltd, New York.

Rodriguez MV, Caballero R, Ruiz F, Romero C (2002) Meta-goal programming, *European Journal of Operational Research*, **136**, 422–429.

Romero C (1991) *A Handbook of Critical Issues in Goal Programming*, Pergamon Press, Oxford.

Romero C (2001) Extended lexicographic goal programming: A unifying approach, *Omega*, **29**, 63–71.

Romero C (2004) A general structure of achievement function for a goal programming model, *European Journal of Operational Research*, **153**, 675–686.

Romero C, Tamiz M, Jones DF (1998) Goal programming, compromise programming and reference point method formulations: Linkages and utility interpretations, *Journal of the Operational Research Society*, **49**(9), 986–991.

Romero C, Tamiz M, Jones DF (2001) Comments on Romero C, Tamiz M, Jones DF (1998) Goal programming, compromise programming and reference point method formulations: Linkages and utility interpretations – Reply to Professor Ogryczak, *Journal of the Operational Research Society*, **52**, 962–963.

Royston G (2005) Modelling and simulation in health-potential achievement and challenge Presentation for MASHnet launch Available from http://www.pms.ac.uk/mashnet/accessed 24 July 2009.

Saaty TL (1981) *The Analytical Hierarchy Process*, McGraw-Hill, New York.

Schniederjans MJ (1995) *Goal Programming*, *Methodology and Applications*, Kluwer Publishers, Boston.

Sedeno-Noda A, Gonzalez-Davila E, Gonzalez-Martin C, Gonzalez-Yanes A (2009) Preemptive benchmarking problem: An approach for official statistics in small areas, *European Journal of Operational Research*, **196**, 360–369.

Selim H, Ozkarahan I (2008) A supply chain distribution network design model: An interactive fuzzy goal programming-based solution approach, *International Journal of Advanced Manufacturing Technology*, **36**, 401–418.

Simon HA (1957) *Models of Man*, J. Wiley & Sons, New York.

Sharma H, Sharma D (2006) A multi-objective decision-making approach for mutual fund portfolio, *Journal of Business & Economics Research*, **4**, 13–24.

Steuer RE (1986) *Multiple Criteria Optimization: Theory, Computation, and Application*, John Wiley & Sons.

Sheth C, Triantis K, Teodorovic D (2007) Performance evaluation of bus routes: A provider and passenger perspective, *Transportation Research Part E-Logistics and Transportation Review*, **43**, 453–478.

Steuer R, Qi Y, Hirschberger M (2007) Suitable-portfolio investors, nondominated frontier sensitivity, and the effect of multiple objectives on standard portfolio selection, *Annals of Operations Research*, **152**, 297–317.

Stewart TJ, Janssen R, van Herwijnen M (2004) A genetic algorithm approach to multiobjective land use planning, *Computers & Operations Research*, **31**, 2293–2313.

Sueyoshi T (1997) Least absolute value estimation, *Journal of the Operations Research Society of Japan*, **40**, 261–275.

Taleizadeh, AA, Niaki, STA, Hoseini, V (2009) Optimizing the multi-product, multi-constraint, bi-objective newsboy problem with discount by a hybrid method of goal programming and genetic algorithm, *Engineering Optimization*, **41**, 437–457.

Tamiz M (2009) www.mopgp.com , accessed 8th September 2009.

Tamiz M, Hasham R, Jones DF, Hesni B, Fargher EK (1996) A two staged goal programming model for portfolio selection, in *Multi-Objective Programming and Goal Programming*, *Springer Lecture Notes in Economics and Mathematical Systems*, **432**, Tamiz M (Ed.), 286–299.

Tamiz M, Jones DF (1996) Goal programming and Pareto efficiency, *Journal of Information and Optimization Sciences*, **17**, 291–307.

Tamiz M, Jones DF (1997) Interactive frameworks for investigation of goal programming models: Theory and practice, *Journal of Multi-Criteria Decision Analysis,* **6**, 52–60.

Tamiz M, Jones DF, El-Darzi E (1995) A review of goal programming and its applications, *Annals of Operations Research*, **58**, 39–53.

Tamiz M, Jones DF, Romero C (1998) Goal programming for decision making: An overview of the current state-of-the-art, *European Journal of Operational Research*, **111**, 569–581.

Tamiz M, Jones DF, Romero C (2001) Comments on Romero C, Tamiz M, Jones DF (1998) Goal programming, compromise programming and reference point method formulations: Linkages and utility interpretations – Final reply to the comments of Professor Ogryczak, *Journal of the Operational Research Society*, **52**, 964–965.

Tamiz M, Mirrazavi SK, Jones DF (1999) Extensions of Pareto efficiency analysis to integer goal programming, *Omega,* **27**, 178–188.

Tiwari RN, Dhahmar S, Rao JR (1987) Fuzzy goal programming: An additive model, *Fuzzy Sets and Systems*, **24**, 27–34.

Wang CS, Chang CT (2008) Integrated genetic algorithm and goal programming for network topology design problem with multiple objectives and multiple criteria, *IEEE-ACM Transactions on Networking*, **16**, 680–690.

Wang YM (2007) A goal programming method for obtaining interval weights from an interval comparison matrix, *European Journal of Operational Research*, **177**, 458–471.

Watada, J (1997) Fuzzy portfolio selection and its application to decision making, *Tatra Mountains Mathematical Publications*, **13**, 219–248.

Wey WM, Wu KY (2007) Using ANP priorities with goal programming in resource allocation in transportation, *Mathematical and Computer Modelling*, **46**, 985–1000.

Wierzbicki AP (1982) A mathematical basis for satisficing decision making, *Mathematical Modelling*, **3**, 391–405.

Williams HP (1993) *Model Solving in Mathematical Programming*, J Wiley and Sons, New York.

Williams HP (2009) *Logic and Integer Programming*, Springer, New York.

Willis K, Jones DF (2008) Multi-objective simulation optimization through search heuristics and relational database analysis, *Decision Support Systems*, **46**, 277–286.

Winston W (2004) *Operations Research: Applications and Algorithms*, Duxbury Press, Pacific Grove, CA.

Wolsey LA (1998) *Integer Programming*, J Wiley and Sons, New York.

Yaghoobi MA, Jones DF, Tamiz M (2008) Weighted additive models for solving fuzzy goal programming problems, *Asia-Pacific Journal of Operational Research*, **25**, 715–733.

Yaghoobi M, Tamiz M (2006) On improving a weighted additive model for fuzzy goal programming problems, *International Review of Fuzzy Mathematics*, **1**, 115–129.

Yang LX, Feng Y(2007) A bicriteria solid transportation problem with fixed charge under stochastic environment, *Applied Mathematical Modelling*, **31**, 2668–2683.

Yu PL (1973) A class of solutions for group decision problems, *Management Science*, **19**, 936–946.

Zadeh LA (1965) Fuzzy sets, *Information and Control*, **8**, 338–353.

Zeleny M (1982) *Multi Criteria Decision Making*, McGraw Hill, New York.

Zhai, JH, Zhang, SF, Wang, XZ (2006) An overview of pattern classification methodologies, *Proceedings of 2006 International Conference on Machine Learning and Cybernetics*, 1–7, 3222–3227.

Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach, *IEEE Transactions on Evolutionary Computation*, **3**, 257–271.

Zolfaghari S, Jaber MY, Hamoudi H (2004) A goal programming approach for a multi-period task assignment problem, *INFOR*, **42**, 299–309.

Index