TITLE: Introduction (Chapter one), Applications and special types of mathematical programming model (Chapter 5) / from Model Building in Mathematical Programming
AUTHOR: Williams, H. P.
SOURCE: New York: Wiley, 1999 pp. 3-9, 59-92

Model Building in Mathematical Programming

Fourth Edition

H. PAUL WILLIAMS

Faculty of Mathematical Studies
University of Southampton

JOHN WILEY & SONS, LTD

Chichester · New York · Brisbane · Toronto · Singapore

Fourth edition first published 1999

Other Wiley Editorial Offices

John Wiley & Sons, Inc., 605 Third Avenue,
New York, NY 10158-0012, USA

Jacaranda Wiley Ltd, 33 Park Road, Milton,
Queensland 4064, Australia

John Wiley & Sons (Canada) Ltd, 22 Worcester Road,
Rexdale, Ontario M9W 1L1, Canada

John Wiley & Sons (SEA) Pte Ltd, 37 Jalan Pemimpin #05-04,
Block B, Union Industrial Building, Singapore 2057

Library of Congress Cataloging-in-Publication Data

Williams, H.P.
Model building in mathematical programming/H.P. Williams.–4th ed.
p. cm.
Includes bibliographical references and index.
ISBN 0-471-99788-9 (pbk. : alk. paper)
1. Programming (Mathematics). 2. Mathematical models. I. Title.
T57.7.W55 1999
519.7—dc21 99-32311
CIP

British Library Cataloguing in Publication Data

Williams, H.P. (Hilary Paul), *1943–*
Model building in mathematical programming.–4th ed. rev.
1. Mathematical programming. Applications of mathematical models
I. Title
519.7

ISBN 0 471 99788 9 (pbk)

Typeset by C.K.M. Typesetting, Salisbury, Wiltshire
Printed and bound in Great Britain by Bookcraft Ltd, Midsomer Norton, Somerset.

CHAPTER 1

Introduction

1.1 The Concept of a Model

Many applications of science make use of *models*. The term 'model' is usually used for a structure which has been built purposely to exhibit features and characteristics of some other objects. Generally only some of these features and characteristics will be retained in the model depending upon the use to which it is to be put. Sometimes such models are *concrete*, as is a model aircraft used for wind tunnel experiments. More often in operational research we will be concerned with *abstract* models. These models will usually be mathematical in that algebraic symbolism will be used to mirror the internal relationships in the object (often an organization) being modelled. Our attention will mainly be confined to such mathematical models, although the term 'model' is sometimes used more widely to include purely descriptive models.

The essential feature of a mathematical model in operational research is that it involves a set of *mathematical relationships* (such as equations, inequalities, logical dependencies, etc.) which correspond to some more down-to-earth relationships in the real world (such as technological relationships, physical laws, marketing constraints, etc.).

There are a number of motives for building such models:

(i) The actual exercise of building a model often reveals relationships which were not apparent to many people. As a result a greater understanding is achieved of the object being modelled.
(ii) Having built a model it is usually possible to analyse it mathematically to help suggest courses which might not otherwise be apparent.
(iii) Experimentation is possible with a model whereas it is often not possible or desirable to experiment with the object being modelled. It would clearly be politically difficult, as well as undesirable, to experiment with unconventional economic measures in a country if there was a high probability of disastrous failure. The pursuit of such courageous experiments would be more (though not perhaps totally) acceptable on a mathematical model.

It is important to realize that a model is really defined by the relationships which it incorporates. These relationships are, to a large extent, independent of the *data* in the model. A model may be used on many different occasions with differing data, e.g. costs, technological coefficients, resource availabilities, etc. We would usually still think of it as the same model even though some coefficients had changed. This distinction is not, of course, total. Radical changes in the data would usually be thought of as a change in the relationships and therefore the model.

Many models used in operational research (and other areas such as engineering and economics) take standard forms. The mathematical programming type of model which we consider in this book is probably the most commonly used standard type of model. Other examples of some commonly used mathematical models are *simulation models, network planning models, econometric models*, and *time series models*. There are many other types of model, all of which arise sufficiently often in practice to make them areas worthy of study in their own right. It should be emphasized, however, that any such list of standard types of model is unlikely to be exhaustive or exclusive. There are always practical situations which cannot be modelled in a standard way. The building, analysing, and experimenting with such new types of model may still be a valuable activity. Often practical problems can be modelled in more than one standard way (as well as in non-standard ways). It has long been realized by operational research workers that the comparison and contrasting of results from different types of model can be extremely valuable.

Many misconceptions exist about the value of mathematical models, particularly when used for planning purposes. At one extreme there are people who deny that models have any value at all when put to such purposes. Their criticisms are often based on the impossibility of satisfactorily quantifying much of the required data, e.g. attaching a cost or utility to a social value. A less severe criticism surrounds the lack of precision of much of the data which may go into a mathematical model; e.g. if there is doubt surrounding 100 000 of the coefficients in a model, how can we have any confidence in an answer it produces? The first of these criticisms is a difficult one to counter and has been tackled at much greater length by many defenders of cost–benefit analysis. It seems undeniable, however, that many decisions concerning unquantifiable concepts, however they are made, involve an implicit quantification which cannot be avoided. Making such a quantification explicit by incorporating it in a mathematical model seems more honest as well as scientific. The second criticism concerning accuracy of the data should be considered in relation to each specific model. Although many coefficients in a model may be inaccurate it is still possible that the structure of the model results in little inaccuracy in the solution. This subject is mentioned in depth in Section 6.3.

At the opposite extreme to the people who utter the above criticisms are those who place an almost metaphysical faith in a mathematical model for decision making (particularly if it involves using a computer). The quality of the answers which a model produces obviously depends on the accuracy of the structure and data of the model. For mathematical programming models the definition of the objective clearly affects the answer as well. Uncritical faith in a model is obviously unwarranted and dangerous. Such an attitude results from a total misconception of how a model should be used. To accept the first answer produced by a mathematical model without further analysis and questioning should be very rare. A model should be used as one of a number of tools for decision making. The answer which a model produces should be subjected to close scrutiny. If it represents an unacceptable operating plan then the reasons for unacceptability should be spelled out and if possible incorporated in a modified model. Should the answer be acceptable it might be wise only to regard it as an *option*. The specification of another objective function (in the case of a mathematical programming model) might result in a different option. By successive questioning of the answers and altering the model (or its objective) it should be possible to clarify the options available and obtain a greater understanding of what is possible.

1.2 Mathematical Programming Models

It should be pointed out immediately that *mathematical programming* is very different from *computer programming*. Mathematical programming is 'programming' in the sense of 'planning'. As such it need have nothing to do with computers. The confusion over the use of the word 'programming' is widespread and unfortunate. Inevitably mathematical programming becomes involved with computing since practical problems almost always involve large quantities of data and arithmetic which can only reasonably be tackled by the calculating power of a computer. The correct relationship between computers and mathematical programming should, however, be understood.

The common feature which mathematical programming models have is that they all involve *optimization*. We wish to *maximize* something or *minimize* something. The quantity which we wish to maximize or minimize is known as an *objective function*. Unfortunately the realization that mathematical programming is concerned with optimizing an objective often leads people summarily to dismiss mathematical programming as being inapplicable in practical situations where there is no clear objective or there are a multiplicity of objectives. Such an attitude is often unwarranted since, as we shall see in Chapter 3, there is often value in optimizing some aspect of a model when in real life there is no clear-cut single objective.

In this book we confine our attention to some special sorts of mathematical programming model. These can most easily be classified as *linear programming models*, *non-linear programming models* and *integer programming models*. We begin by describing what a linear programming model is by means of two small examples.

Example 1. A Linear Programming (LP) Model (Product Mix)

An engineering factory can produce five types of product (PROD 1, PROD 2, . . . , PROD 5) by using two production processes: grinding and drilling.

After deducting raw material costs each unit of each product yields the following contributions to profit:

PROD 1	PROD 2	PROD 3	PROD 4	PROD 5
£550	£600	£350	£400	£200

Each unit requires a certain time on each process. These are given below (in hours). A dash indicates when a process is not needed.

	PROD 1	PROD 2	PROD 3	PROD 4	PROD 5
Grinding	12	20	—	25	15
Drilling	10	8	16	—	—

In addition the final assembly of each unit of each product uses 20 hours of an employee's time.

The factory has three grinding machines and two drilling machines and works a six-day week with two shifts of 8 hours on each day. Eight workers are employed in assembly, each working one shift a day.

The problem is to find how much to make of each product so as to maximize the total profit contribution.

This is a very simple example of the so-called 'product mix' application of linear programming.

In order to create a *mathematical model* we introduce variables $x_1, x_2, \ldots, x_5$ representing the numbers of PROD 1, PROD 2,..., PROD 5 which should be made in a week. Since each unit of PROD 1 yields £550 contribution to profit and each unit of PROD 2 yields £600 contribution to profit, etc., our total profit contribution will be represented by the expression:

$$550x_1 + 600x_2 + 350x_3 + 400x_4 + 200x_5. \tag{1}$$

The *objective* of the factory is to choose $x_1, x_2, \ldots, x_5$ so as to make this expression as big as possible, i.e. (1) is the *objective function* which we wish to *maximize* (in this case).

Clearly our processing and labour capacities, to some extent, limit the values which the x_j can take. Given that we have only three grinding machines working for a total of 96 hours a week each, we have 288 hours of grinding capacity available. Each unit of PROD 1 uses 12 hours grinding. x_1 units will therefore use $12x_1$ hours. Similarly x_2 units of PROD 2 will use $20x_2$ hours. The total amount of grinding capacity which we use in a week is given by the expression on the left-hand side of (2) below:

$$12x_1 + 20x_2 \qquad + 25x_4 + 15x_5 \leqslant 288. \tag{2}$$

(2) is a mathematical way of saying that we cannot use up more than the 288 hours of grinding available per week. (2) is known as a *constraint*. It restricts (or constrains) the possible values which the variables x_j can take.

The drilling capacity is 192 hours a week. This gives rise to the following constraint:

$$10x_1 + 8x_2 + 16x_3 \leqslant 192. \tag{3}$$

Finally, the fact that we have only a total of eight assembly workers each working 48 hours a week gives us a labour capacity of 384 hours. Since each unit of each product uses 20 hours of this capacity we have the constraint

$$20x_1 + 20x_2 + 20x_3 + 20x_4 + 20x_5 \leqslant 384. \tag{4}$$

We have now expressed our original practical problem as a mathematical model. The particular form which this model takes is that of a *linear programming (LP) model.* This model is now a well-defined mathematical problem. We wish to find values for the variables $x_1, x_2, \ldots, x_5$ which make the expression (1) (the objective function) as large as possible but still satisfy the constraints (2), (3), and (4). You should be aware of why the term 'linear' is applied to this particular type of problem. Expression (1) and the left-hand sides of constraints (2), (3), and (4) are all linear. Nowhere do we get terms like x_1^2, x_1x_2 or $\log x$ appearing.

There are a number of implicit assumptions in this model which we should be aware of. Firstly, we must obviously assume that the variables $x_1, x_2, \ldots, x_5$ are not allowed to be negative, i.e. we do not make negative quantities of any product. We might explicitly state these conditions by the extra constraints

$$x_1, x_2, \ldots, x_5 \geqslant 0. \tag{5}$$

In most linear programming models the non-negativity constraints (5) are implicitly assumed to apply unless we state otherwise. Secondly, we have assumed that the variables $x_1, x_2, \ldots, x_5$ can take fractional values, e.g. it is meaningful to make 2·36 units of PROD 1. This assumption may or may not be entirely warranted. If, for example, PROD 1 represented gallons of beer, fractional quantities would be acceptable. On the other hand, if it represented numbers of motor cars, it would not be meaningful. In practice the assumption that the variables can be fractional is perfectly acceptable in this type of model, if the errors involved in rounding to the nearest integer are not great. If this is not the case we have to resort to *integer programming.*

The model above illustrates some of the essential features of an LP model:

(i) There is a single linear expression (the *objective function*) to be maximized or minimized.

(ii) There is a series of *constraints* in the form of linear expressions which must not exceed ($\leqslant$) some specified value. Linear programming constraints can also be of the form ' $\geqslant$ ' and '=', indicating that the value of certain linear expressions must not fall below a specified value or must exactly equal a specified value.

(iii) The set of coefficients 288, 192, 384, on the right-hand sides of the constraints (2), (3), and (4), is generally known as the *right-hand side* column.

Practical models will, of course, be much bigger (more variables and constraints) and more complicated but they must always have the above three essential features. The optimal solution to the above model is included in Section 6.3.

In order to give a wider picture of how linear programming models can arise, we give a second small example of a practical problem.

Example 2. A Linear Programming Model (Blending)

A food is manufactured by refining raw oils and blending them together. The raw oils come in two categories:

Vegetable oils	VEG 1
	VEG 2
Non-vegetable oils	OIL 1
	OIL 2
	OIL 3

Vegetable oils and non-vegetable oils require different production lines for refining. In any month it is not possible to refine more than 200 tons of vegetable oil and more than 250 tons of non-vegetable oils. There is no loss of weight in the refining process and the cost of refining may be ignored.

There is a technological restriction of hardness in the final product. In the units in which hardness is measured this must lie between 3 and 6. It is assumed that hardness blends linearly. The costs (per ton) and hardness of the raw oils are:

	VEG 1	VEG 2	OIL 1	OIL 2	OIL 3
Cost	£110	£120	£130	£110	£115
Hardness	8·8	6·1	2·0	4·2	5·0

The final product sells for £150 per ton.

How should the food manufacturer make their product in order to maximize their net profit?

This is another very common type of application of linear programming although, of course, practical problems will be, generally, much bigger.

Variables are introduced to represent the unknown quantities. $x_1, x_2, \ldots, x_5$ represent the quantities (tons) of VEG 1, VEG 2, OIL 1, OIL 2, and OIL 3 which should be bought, refined and blended in a month. y represents the quantity of the product which should be made. Our objective is to maximize the net profit:

$$-110x_1 - 120x_2 - 130x_3 - 110x_4 - 115x_5 + 150y. \qquad (6)$$

The refining capacities give the following two constraints:

$$x_1 + x_2 \leqslant 200, \tag{7}$$

$$x_3 + x_4 + x_5 \leqslant 250. \tag{8}$$

The hardness limitations on the final product are imposed by the following two constraints:

$$8{\cdot}8x_1 + 6{\cdot}1x_2 + 2x_3 + 4{\cdot}2x_4 + 5x_5 - 6y \leqslant 0, \tag{9}$$

$$8{\cdot}8x_1 + 6{\cdot}1x_2 + 2x_3 + 4{\cdot}2x_4 + 5x_5 - 3y \geqslant 0. \tag{10}$$

Finally it is necessary to make sure that the weight of the final product is equal to the weight of the ingredients. This is done by a continuity constraint:

$$x_1 + x_2 + x_3 + x_4 + x_5 - y = 0. \tag{11}$$

The objective function (6) (to be maximized) together with the constraints (7), (8), (9), (10), and (11) make up our LP model.

The linearity assumption of LP is not always warranted in a practical problem, although it makes any model computationally much easier to solve. When we have to incorporate non-linear terms in a model (either in the objective function or the constraints) we obtain a *non-linear programming (NLP) model.* In Chapter 7 we will see how such models may arise and a method of modelling a wide class of such problems using *separable programming*. Nevertheless, such models are usually far more difficult to solve.

Finally the assumption that variables can be allowed to take fractional values is not always warranted. When we insist that some or all of the variables in an LP model must take integer (whole number) values we obtain an *integer programming (IP) model.* Such models are again much more difficult to solve than conventional LP models. We will see in Chapters 8, 9, and 10 that IP opens up the possibility of modelling a surprisingly wide range of practical problems.

Another type of model which we discuss briefly is known as a *stochastic programming model.* This arises when some of the data are uncertain but can be specified by a probability distribution. Although data in many linear programming models may be uncertain, their representation by *expected* values alone may not be sufficient. Situations in which a more explicit recognition of the probabilistic nature of data may be made, but the resultant model still converted to a linear program, are described. In Chapter 3 we mention *chance constrained* models and in Chapter 4 *multi-staged models with recourse*, both of which fall in the category of stochastic programming. An example of the use of this latter type of model is given in Sections 12.24, 13.24 and 14.24 where it is applied to determining the price of airline tickets over successive periods in the face of uncertain demand. A good reference to stochastic programming is Kall and Wallace (1994).

CHAPTER 5

Applications and Special Types of Mathematical Programming Model

5.1 Typical Applications

The purpose of this section is to create an awareness of the areas where linear programming (LP) is applicable. To categorize totally those industries and problems where LP can, or cannot, be used would be impossible. Some problems clearly lend themselves to an LP model. For other problems the use of an LP model may not provide a totally satisfactory solution but may be considered acceptable in the absence of other approaches. The decision of when to use, and when not to use, LP is often a subjective one depending on an individual's experience.

This section can do no more than try to give a 'feel' for those areas in which LP can be applied. In order to do this a list of industries and areas in which the technique has been applied is given. This list is by no means exhaustive but is intended to include most of the major users. A short discussion is given of the types of LP models which are of use in each area. References are given to some of the relevant published case studies. In view of the very wide use which has been made of LP it would be almost impossible to seek out every reference to published case studies. Nor would it be helpful to submerge the reader in a mass of often superfluous literature. The intention is to give sufficient references to allow the reader to follow up published case studies himself. From the references given here it should be possible to find other references if necessary. In many cases practical applications are illustrated by problems in Part 2.

Although the intention is mainly to consider *linear programming* applications in this chapter, the resultant models can very often naturally be extended by *integer programming* or *non-linear programming models*. In this way more complicated or realistic situations can often be modelled. These topics and further applications are considered more fully in Chapters 7, 8, 9, and 10.

The subject of linear programming does not have clearly defined boundaries. Other subjects impinge on, and merge with, linear programming. Two types of model which have, to some extent, been studied independently of LP are considered further in this chapter. Firstly *economic models* which are sometimes referred to as *input–output* or *Leontief models* are considered in

Section 5.2. Such models can often be regarded as a special type of LP model. Secondly, *network models* which arise frequently in Operational Research are considered in Section 5.3. Such models are, again, often usefully considered as special types of LP model. Another area which also has connections with LP is not considered in this book in view of its limited practical applicability to date. This is the *theory of games*. Game theory models can sometimes be converted into LP models (and vice versa).

The following list of applications should indicate the surprisingly wide applicability of linear programming and in consequence its economic importance. Many more references exist than are given. The purpose of the references is to give a 'window' into the literature.

The Petroleum Industry

This is by far the biggest user of LP. Very large models involving thousands, and occasionally tens of thousands, of constraints have been built. These models are used to help make a number of decisions starting with where and how to buy crude oil, how to ship it, and which products to produce out of it. Such 'corporate models' contain elements of *distribution*, *resource allocation*, *blending*, and possibly *marketing*. A typical example of the sort of model which arises in the industry is the REFINERY OPTIMIZATION problem of Part 2. Descriptions of the use of LP in the petroleum industry are given by Manne (1956), Catchpole (1962), and McColl (1969).

The Chemical Industry

The applications here are rather similar to those in the petroleum industry although the models are rarely as large. Applications usually involve *blending*, or *resource allocation*. An application is described by Royce (1970).

Manufacturing Industry

Linear programming is frequently used here for resource allocation. The '*product mix*' example described in Section 1.2 is an example of this type of application. Resources to be allocated are usually processing capacity, raw materials, and manpower. A multi-period problem of this type, considered in relation to the engineering industry, is the FACTORY PLANNING problem of Part 2. Other common applications of LP in manufacturing are *blending* and *blast furnace burdening* (the steel industry). Three references to the application of LP here are Lawrence and Flowerdew (1963), Fabian (1967), and Sutton and Coates (1981).

Transport and Distribution

Problems of distribution can often be formulated as LP models. Two classic examples are the *transportation* and *transhipment* problems which are considered in Section 5.3 as they involve *networks*. A simple DISTRIBUTION problem of this type is presented in Part 2. An extension of this problem, DISTRIBUTION 2, involves *depot location* as well and requires integer programming. *Scheduling* problems (e.g. lorries, aircraft, tankers, trains, buses, etc.) can often be tackled through integer programming. One such problem is the MILK COLLECTION problem given in Part 2. Problems of *assignment* (trains to mines and power stations) arising in transport have also been tackled through mathematical programming. Applications of mathematical programming in distribution are described by Eilon, Watson-Gandy and Christofides (1971) and Markland (1975).

Finance

A very early application of mathematical programming was in *portfolio selection*. This was due to Markowitz (1959). Given a sum of money to invest, the problem was how to spend it among a portfolio of shares and stocks. The objective was to maintain a certain expected rate of return from the investment but to minimize the variance of that return. The model which results is a quadratic programming model.

Agarwala and Goodson (1970) suggest how LP can be used by a government to design an *optimum tax package* to achieve some required aim (in particular an improvement in the balance of payments).

LP is increasingly being used in *accountancy*. The economic information which can be derived from the solution to an LP model can provide accountants with very useful costing information. This sort of information is described in detail in Section 6.2. A description of how LP can be used in accountancy is given by Salkin and Kornbluth (1973).

Spath, Gutgesell and Grun (1975) describe how an LP model is applied by a mail order firm in order to minimize the total interest cost on all credits.

An interesting extension of a finance model to allow the user to set *goals* interactively (rather than objectives) and operate within the degrees of freedom permitted by the constraints is described by Jack (1985).

A very important potential area of application is Yield (Revenue) Management which is concerned with setting prices for goods at different times in order to maximize revenue. It is particularly applicable to the hotel, catering, airline and train industries. An example of its use is the YIELD MANAGEMENT problem in Part 2.

Agriculture

LP has been used in agriculture for *farm management*. Such models can be used to decide what to grow where, how to rotate crops, how to expand production, and where to invest. An example of such a problem is the FARM PLANNING problem of Part 2. Swart, Smith and Holderby (1975) apply a multi-period LP model to planning the expansion of a large dairy farm. Other general references are Balm (1980) and Fokkens and Puylaert (1981).

Blending models are often applicable to agriculture problems. It is often desired to blend together livestock feeds or fertilizer at minimum cost. Glen (1980) describes a model for beef cattle ration formulation.

Distribution problems often arise in this area. The distribution of farm products, in particular milk, can be examined by the network type of LP model described in Section 5.3.

Models for planning the growth and harvesting of a number of agricultural products are described by Glen (1980, 1988, 1995, 1996, 1997).

Quadratic programming has been used for determining *optimal prices* for the sale of milk in the Netherlands. This is described by Louwes, Boot and Wage (1963). The AGRICULTURAL PRICING problem of Part 2 is based on this study. A mixed integer programming model for irrigation in a developing country is described by Rose (1973).

Health

The obvious application of mathematical programming in this area is in problems of *resource allocation*. How are scarce resources, e.g. doctors, nurses, hospitals, etc., to be used to best effect? In such problems there will obviously be considerable doubt concerning the validity of the data, e.g. how much of a nurse's time does a particular type of treatment really need? In spite of doubts concerning much of the data in such problems it has been possible to use mathematical programming models to suggest plausible policy options. McDonald, Cuddeford and Beale (1974) describe a non-linear programming model for allocating resources in the United Kingdom health service. Revelle, Feldmann and Lynn (1969) describe how a non-linear programming model can be used for controlling tuberculosis in an underdeveloped country.

Warner and Prawda (1972) describe a mathematical programming model for scheduling nurses. Redpath and Wright (1981) describe how linear programming is used to decide intensity and direction of beams for irradiating cancerous tumours.

Mining

A number of interesting applications of mathematical programming occur in mining. The straightforward applications are simply ones of *resource allocation*, i.e. how should manpower and machinery be deployed to best effect?

Blending problems also occur when it is necessary to mix together ores to achieve some required quality.

Two examples of mining problems are given in Part 2. The MINING problem concerns what combination of mines a company should operate in successive years. The OPENCAST MINING problem is to decide what the boundaries of an opencast mine should be. References to the application of mathematical programming in mining are Young, Fergusson and Corbishley (1963) and Meyer (1969).

Manpower Planning

The possible movement of people between different types of job and its control by recruitment, promotion, retraining, etc., can be examined by linear programming. An example of such a problem, MANPOWER PLANNING, is given in Part 2. Applications of mathematical programming to manpower planning are described by Price and Piskor (1972), Davies (1973), Vajda (1975), Charnes *et al.* (1975), and Lilien and Rao (1975).

Food

The food industry makes considerable use of linear programming. *Blending* (sausages, meat pies, margarines, ice cream, etc.) is an obvious application often giving rise to very small and easily solved models.

Problems of *distribution* also arise in this industry giving rise to the network type models described in Section 5.3.

As in other manufacturing industries problems of *resource allocation* arise which can be tackled through linear programming.

The FOOD MANUFACTURE problem of Part 2 is an example of a multi-period blending problem in the food industry. A more complicated version of this problem is described by Williams and Redwood (1974). Jones and Rope (1964) describe another linear programming model in the food industry.

Energy

The electricity and gas supply industries both use mathematical programming to deal with problems of resource allocation. The TARIFF RATES (POWER GENERATION) problem of Part 2 involves scheduling electric generators to meet varying loads at different times of day. This problem is similar to that described by Garver (1963). This problem is extended to the HYDRO POWER model which illustrates another important application. Archibald, Buchanan, McKinnon and Thomas (1999) describe how a similar problem can be tackled by stochastic dynamic programming.

Linear programming can also be applied to *distribution* problems involving the design and use of supply networks.

Applications of mathematical programming in these areas are also described by Babayer (1975), Fanshel and Lynes (1964), Muckstadt and Koenig (1977), and Khodaverdian, Brameller and Dunnett (1986).

Pulp and Paper

Problems of *resource allocation* in the manufacture of paper give rise to linear programming models. Such models frequently involve an element of *blending*. In addition the possibility of *recycling waste paper* has also been examined by linear programming as described by Glassey and Gupta (1974).

A totally different type of problem arising in the paper industry (and the glass industry) is the *trimloss* problem. This is the problem of arranging orders for rolls of paper of different widths so as to minimize the wastage. This problem can be tackled by linear (or sometimes integer) programming. This problem is described by Eisemann (1957). It has also been considered by Gilmore and Gomory (1961, 1963).

Advertising

The problem of spreading one's advertising budget over possible advertising outlets (e.g. television commercials, newspaper advertisements, etc.) has been approached through mathematical programming. These problems are known as *media scheduling* problems.

Authors differ over the usefulness of mathematical programming in tackling this type of problem. Selected references are Engel and Warshaw (1964), Bass and Lonsdale (1966), and Charnes *et al.* (1968). The last-mentioned reference gives a very full list of references itself.

Defence

Problems of *resource allocation* give rise to military applications of linear programming. Such an application is described by Beard and McIndoe (1970).

The siting of missile sites is described by Miercort and Soland (1971).

Other Applications

There are numerous other applications of mathematical programming. A few of the less usual ones are given here since they might otherwise go unnoticed.

Heroux and Wallace (1973) describe a multi-period linear programming model for land development.

Souder (1973) discusses the effectiveness of a number of mathematical programming models for research and development. Feuerman and Weiss (1973) show how a knapsack integer programming model can be used to help design multiple choice-type examinations. Kalvaitis and Posgay (1974) apply integer programming to the problem of selecting the most promising kind of mailing list.

Wardle (1965) applies linear programming to forestry management.

Problems of pollution control have been tackled through mathematical programming. Applications are described by Loucks, Revelle and Lynn (1968).

Kraft and Hill (1973) describe a 0–1 integer programming model for selecting journals for a university library.

The use of linear programming for performance evaluation in a number of organizations (particularly in the public sector) through the use of a type of model known as *Data Envelopment Analysis* has become important. This is described through the EFFICIENCY ANALYSIS problem in Part 2. References are Charnes, Cooper and Rhodes (1978), Farrell (1957), Land (1991) and Thanassoulis, Dyson and Foster (1987).

A much fuller list of papers on mathematical programming applications has been compiled by Riley and Gass (1958). The special editions of the journal *Mathematical Programming Studies* Nos 9 (1975) and 20 (1982) are devoted to applications.

5.2 Economic Models

A widely used type of national economic model is the *input–output model* representing the interrelationships between the different sectors of a country's economy. Such models are often referred to as *Leontief models* after their originator, who built such a model of the American economy. This is described by Leontief (1951). Input–output models are often regarded usefully as a special type of linear programming model.

The Static Model

The *output* from a particular industry or a sector of an economy is often used for two purposes: (i) for immediate consumption, e.g. coal to be sold to domestic consumers; (ii) as an *input* to other industries or sectors of the economy, e.g. coal to provide power for the steel industry. Since the outputs from many industries will be able to be split in this way between (exogenous) consumption and as (endogenous) inputs into these same industries, a complex set of interrelationships will exist. The input–output type of model provides about the simplest way of representing these relationships. A number of strong (and usually over-simplified) assumptions are made regarding the inter-industry relationships. The two major assumptions are as follows:

(i) The output from each industry is directly proportional to its inputs, e.g. doubling all the inputs to an industry will double its outputs.

(ii) The inputs to a particular industry are all in fixed proportions, e.g. it is not possible to decrease one input and compensate for this by increasing another input. These fixed proportions are determined by the technology of the production process. In other words there is *non-substitutability* of inputs.

In order to demonstrate such a model we will consider a very simple example.

Example. A Three-industry Economy

We suppose that we have an economy made up of only three types of industry: coal, steel, and transport. Part of the outputs from these industries are needed as inputs to others, e.g. coal is needed to fire the blast furnaces that produce steel, steel is needed in the machinery for extracting coal, etc. The necessary inputs to produce one unit of output for each industry are given in the *input–output* matrix in Table 5.1.

It is usual in such tables to measure all units of production in monetary terms. We then see that, for example, to produce £1 in worth of coal requires £0·1 of coal (to provide the necessary power), £0·1 of steel (the steel 'used up' in the 'wear and tear' on the machinery) and £0·2 of transport (for moving the coal from the mine). In addition £0·6 of labour is required. Similarly the other columns of Table 5.1 give the inputs required (£s) for each £ of steel and each £ of transport (lorries, cars, trains, etc.).

Notice that the value of each unit of output is exactly matched by the sum of the values of its inputs.

This economy is assumed to be '*open*' in the sense that some of the output from the above three industries is used for exogenous consumption. We will assume that these 'external' requirements are (in £ millions)

Coal	20
Steel	5
Transport	25

Such a set of exogenous demands is known as a *bill of goods*.

Table 5.1 An input–output matrix

	Outputs		
Inputs	Coal	Steel	Transport
Coal	0·1	0·5	0·4
Steel	0·1	0·1	0·2
Transport	0·2	0·1	0·2
Labour	0·6	0·3	0·2

A number of questions naturally arise concerning our economy which a mathematical model might be used to answer:

(i) How much should each industry produce in total in order to satisfy a given bill of goods?
(ii) How much labour would this require?
(iii) What should the price of each product be?

If variables x_c, x_s, and x_t are used to represent the total quantities of coal, steel, and transport produced (in a year) we get the following relationships:

$$x_c = 20 + 0{\cdot}1x_c + 0{\cdot}5x_s + 0{\cdot}4x_t, \tag{1}$$

$$x_s = 5 + 0{\cdot}1x_c + 0{\cdot}1x_s + 0{\cdot}2x_t, \tag{2}$$

$$x_t = 25 + 0{\cdot}2x_c + 0{\cdot}1x_s + 0{\cdot}2x_t. \tag{3}$$

For example, equation (1) tells us that we must produce enough coal to satisfy external demand (£20m), input to the coal industry ($0{\cdot}1x_c$), input to the steel industry ($0{\cdot}5x_s$), and input to the transport industry ($0{\cdot}4x_t$).

Equations (1), (2), and (3) can conveniently be rewritten as

$$0{\cdot}9x_c - 0{\cdot}5x_s - 0{\cdot}4x_t = 20, \tag{4}$$

$$-0{\cdot}1x_c + 0{\cdot}9x_s - 0{\cdot}2x_t = 5, \tag{5}$$

$$-0{\cdot}2x_c - 0{\cdot}1x_s + 0{\cdot}8x_t = 25, \tag{6}$$

Such a set of equations in the same number of unknowns can generally be solved uniquely. In this case we would obtain the solution

$$x_c = 56{\cdot}1, \qquad x_s = 22{\cdot}4, \qquad x_t = 48{\cdot}1.$$

The total labour requirement can then easily be obtained as

$$0{\cdot}6 \times 5{\cdot}61 + 0{\cdot}3 \times 22{\cdot}4 + 0{\cdot}2 \times 48{\cdot}1 = 50.$$

Clearly (4), (5), and (6) could be regarded as the constraints of a linear programming model. An objective function could be constructed and we could maximize it or minimize it subject to the constraints. As the model stands, however, there would be little point in doing this since there is generally only one feasible solution. The objective function would, therefore, have no influence on the solution.

Once, however, we extend this very simple type of input–output model we frequently obtain a genuine linear programming model.

The model described above is unrealistic in a number of respects. Equations (4), (5), and (6) give no real limitation to the productive capacity of the economy. It can fairly easily be shown that so long as a particular, positive, bill of goods can be produced (the economy is a 'productive' one) then these relationships guarantee that any bill of goods, however large, can be produced. This is clearly unrealistic. Firstly, we would expect there to be some limitation on productive capacity preventing more than a certain amount of output from each industry in a given period of time. Secondly, we would expect the output from an industry only to be effective as the input to another industry after a certain time has elapsed. This second consideration leads to *dynamic input–output models* which are considered below. Before doing this, however, we will consider the problem of modelling limited productive capacity in the case of a *static* model.

In our small example we assumed that once we had decided how much each industry should produce in order to meet a specified bill of goods we could provide the labour required. If labour were in short supply it might limit our productive capacity. There would then be interest in seeing what bills of goods are or are not producible in a particular period of time. Returning to our example, if we were to limit labour to 40 (£m per year) we could not produce our previous bill of goods. But what bill of goods could we produce? Answers to this question can be explored through linear programming. Variables will now represent our bill of goods:

Coal	y_c
Steel	y_s
Transport	y_t

Equations (4), (5), and (6) will give the constraints

$$0{\cdot}9x_c - 0{\cdot}5x_s - 0{\cdot}4x_t - y_c = 0, \tag{7}$$

$$-0{\cdot}1x_c + 0{\cdot}9x_s - 0{\cdot}2x_t - y_s = 0, \tag{8}$$

$$-0{\cdot}2x_c - 0{\cdot}1x_s + 0{\cdot}8x_t - y_t = 0. \tag{9}$$

The labour limitation gives the constraint

$$0{\cdot}6x_c + 0{\cdot}3x_s + 0{\cdot}2x_t \leqslant 40. \tag{10}$$

Achievable bills of goods will be represented by the values of y_c, y_r, and y_t in feasible solutions to (7), (8), (9), and (10). Specific solutions can be found by introducing an objective function. For example, we might wish to maximize the total output:

$$x_c + x_s + x_t. \tag{11}$$

Alternatively we might weight some outputs more heavily than others by giving x_c, x_s, and x_t different objective coefficients. We might wish simply to maximize production in one particular sector of the economy, such as steel, and simply maximize x_s. This is clearly a situation of the type referred to in Section 3.2 in which it is of interest to experiment with a number of different objectives rather than simply concentrate on one.

We have considered only labour as a limiting factor in productive capacity. In practice there could well be other resource limitations such as processing capacity, raw material, etc. Such limitations could, of course, easily be incorporated in a model by extra constraints. Limited resources of this sort are sometimes known as *primary goods*. Primary goods only provide inputs to the economy. They are not produced as outputs as well. A major advantage of treating such models as linear programming models is that a lot of subsidiary economic information is also obtained from solving such a model. Such information is described very fully in Section 6.2. In particular, valuations are obtained for the constraints of a model. These valuations are known as *shadow prices*. For the type of model considered here we would obtain meaningful valuations for the primary goods. In this way a pricing system could be introduced into our model. This would give suitable prices for the outputs from all the industries.

Although any number of primary goods can be considered in a linear programming formulation of an input–output model it is quite common only to consider labour. In practice, particularly in the relatively simple economies of developing countries, it may well not be unreasonable to consider labour as the overall limitation. If this can be done there is another less obvious advantage to be gained in the applicability of such a model. It has already been pointed out that an input–output model assumes *non-substitutability* of the inputs, i.e. it is not possible to vary the relative proportions in which all the inputs are used to produce the output of a particular industry. In practice this might well be a far from realistic assumption. For example, we might well be able to produce each unit of coal by using more power (more coal) and less machinery (less steel). To model this possibility would require a variation in the coefficients of the input–output matrix. It has been shown that if there is only one primary good (usually labour), then it will only be worthwhile to concentrate on one production process for each industry. This is the result of the *Samuelson substitution theorem* which we will not prove. Such theoretical results and a fuller description of input–output models are given in Dorfman, Samuelson, and Solow (1958). Another good reference is Shapiro (1979). The importance of this result is that we need not worry about the apparent non-substitutability limitation so long as we only have one primary good. There will be one, and only one best set of inputs (production processes) for each industry. This best production process will remain the best no matter what bill of goods we are producing. There is, of course, the problem of finding, for each industry, that production process which should be used. Once, however, this has been done, no matter what the bill of goods, we need only incorporate this one production process (column of the input–output matrix) into all future models. In fact the finding of the best production process for each industry can be done by linear programming. To illustrate how this may be done as well as illuminating the import of the Samuelson substitution theorem we will extend our small example. Table 5.2 gives two possible sets of inputs (production processes) to produce one unit of the three industries.

In practice there might be many more (possibly an infinite number) than two processes for each industry.

On the face of it we might think it advantageous to use some combination of the two processes for producing coal. The first process is more economical on

Table 5.2 An input–output matrix with alternative production processes

	Outputs					
Inputs	Coal		Steel		Transport	
Coal	0·1	0·2	0·5	0·6	0·4	0·6
Steel	0·1	—	0·1	0·1	0·2	0·2
Transport	0·2	0·1	0·1	—	0·2	0·05
Labour	0·6	0·7	0·3	0·3	0·2	0·15

coal but uses some steel as well, which the second process does not. Similarly some mixture of the two processes for producing steel and the two processes for producing transport might seem appropriate. Moreover, which processes are used might seem likely to depend upon the particular bill of goods.

We repeat, however, that our intuitive idea would be false. There will be exactly one best process for each industry and this will be used whatever bill of goods we have. Instead of the variables x_c, x_s, and x_t in our original model we can introduce variables x_{c1}, x_{c2}, x_{s1}, x_{s2}, x_{t1}, and x_{t2} to represent the total quantities of coal, steel, and transport produced by each process. Using the same bill of goods as before, instead of constraints (1), (2), and (3), we obtain

$$x_{c1}+x_{c2}=20+0{\cdot}1x_{c1}+0{\cdot}2x_{c2}+0{\cdot}5x_{s1}+0{\cdot}6x_{s2}+0{\cdot}4x_{t1}+0{\cdot}6\ x_{t2}, \tag{12}$$

$$x_{s1}+x_{s2}=\ 5+0{\cdot}1x_{c1}+0{\cdot}0x_{c2}+0{\cdot}1x_{s1}+0{\cdot}1x_{s2}+0{\cdot}2x_{t1}+0{\cdot}2\ x_{t2}, \tag{13}$$

$$x_{t1}+x_{t2}=25+0{\cdot}2x_{c1}+0{\cdot}1x_{c2}+0{\cdot}1x_{s1}+0{\cdot}0x_{s2}+0{\cdot}2x_{t1}+0{\cdot}05x_{t2}. \tag{14}$$

These equations can be written as

$$0{\cdot}9x_{c1}+0{\cdot}8x_{c2}-0{\cdot}5x_{s1}-0{\cdot}6x_{s2}-0{\cdot}4x_{t1}-0{\cdot}6x_{t2}\ =20, \tag{15}$$

$$-0{\cdot}1x_{c1}-0{\cdot}0x_{c2}+0{\cdot}9x_{s1}+0{\cdot}9x_{s2}-0{\cdot}2x_{t1}-0{\cdot}2x_{t2}\ =\ 5. \tag{16}$$

$$-0{\cdot}2x_{c1}-0{\cdot}1x_{c2}-0{\cdot}1x_{s1}-0{\cdot}0x_{s2}+0{\cdot}8x_{t1}+0{\cdot}95x_{t2}=25. \tag{17}$$

We are considering labour as the only primary good and will limit ourselves to 60 (£m). This gives the constraint

$$0{\cdot}6x_{c1}+0{\cdot}7x_{c2}+0{\cdot}3x_{s1}+0{\cdot}3x_{s2}+0{\cdot}2x_{t1}+0{\cdot}15x_{t2}\leqslant 60. \tag{18}$$

There will generally be more than one solution to a system such as this. In order to find the 'best' solution we will define an objective function. One possible objective function would, of course, be to ignore constraint (18) and minimize the expression on the left-hand side representing labour usage. Alternatively we might specify another objective function. For this example we will do this and simply maximize total output:

$$x_{c1}+x_{c2}+x_{s1}+x_{s2}+x_{t1}+x_{t2}. \tag{19}$$

Our resultant optimal solution gives

$$x_{c1} = 64{\cdot}6,$$

$$x_{c2} = 0,$$

$$x_{s1} = 22{\cdot}6,$$

$$x_{s2} = 0,$$

$$x_{t1} = 0,$$

$$x_{t2} = 44{\cdot}6.$$

Notice that the Samuelson substitution theorem has worked in this case. One process in each industry is the best to the total exclusion of all the others. Moreover, it could be shown that these processes will be the best no matter what the bill of goods is. The optimal solution will be made up of only the variables x_{c1}, x_{s1}, and x_{t2} no matter what the right-hand side coefficients in constraints (15), (16), (17) and (18) are. We could, therefore, confine all our attention to these processes and ignore the others. It should be noted, however, that these 'best' processes are only the best because of the objective function (19) which we have chosen. If instead of maximizing output we were to choose another objective, it might be preferable to switch to another process in some cases. It will never, however, be worth 'mixing' processes. Once we consider more than a single primary good such mixing may well, however, become desirable.

The Dynamic Model

We have pointed out that our static model assumed that we could ignore time lags between an output being produced and used as an input to another (or the same) industry. This unrealistic assumption can be avoided by introducing a *dynamic model.* It has already been shown in Section 4.1 that linear programming models can often be extended to multi-period models. We can do much the same thing with the static type of input–output model. In practice some of the output from an economy will immediately be consumed (e.g. cars for private motoring) while some will go to increase productive capacity (e.g. factory machinery). Such alternative uses for the output will result in different possible growth patterns for the economy, i.e. we can live well now but neglect to invest for the future or we can sacrifice present day consumption in the interests of future wealth producing capacity. A simple example of such a problem, ECONOMIC PLANNING, leading to a dynamic input–output model is given in Part 2. Rather than discuss dynamic input–output models further here, the discussion is postponed to the specific discussion of the formulation of this problem in Part 3. A description of dynamic input–output models of this type is given by Wagner (1957).

Aggregation

To sum up the characteristics of a whole industry or sector of an economy in one column of an input–output matrix obviously requires a large amount of simplification of the real situation. It is necessary to group together many different industries into one. This *aggregation* is necessary in order to obtain a reasonable size of problem. Most input–output models are aggregated into less than 1000 industries. The problem of aggregation is obviously of paramount concern to the model builder. Unfortunately very little theoretical work has been done to indicate *mathematically* when aggregation is and is not justified. Three criteria which common sense would suggest to be good grounds for aggregating particular industries are: (i) substitutability, (ii) complementarity, and (iii) similarity of production processes. Problems of this sort are discussed more fully by Stone (1960).

In view of their sophistication and efficiency commercial mathematical programming packages provide a useful way of solving input–output models. Even if the model is of the simplest kind described above and only requires the solution of a set of simultaneous equations such packages are of use. Almost all packages contain an inversion routine which is very useful for inverting large matrices (sets of simultaneous equations). It should, however, be pointed out that input–output models are often quite dense. In this respect they are untypical of general linear programming models. As already mentioned in Section 2.1, in a thousand-constraint model one would expect only about 1% of the coefficients to be non-zero. For an input–output model this figure could well be as high as 50%. As a result input–output models can take a long time to solve on a computer and run into numerical difficulties. It is sometimes worth exploiting the special structure of an input–output linear programming model and using a special purpose algorithm. Dantzig (1955) describes how the simplex algorithm can be adapted to this purpose.

5.3 Network Models

The use of models involving networks is very widespread in operational research. Problems involving distribution, assignment, and planning (critical path analysis and PERT) frequently give rise to the analysis of networks. Many of the resultant problems can be regarded as special types of linear programming problem. It is often more efficient to use special purpose algorithms rather than the revised simplex algorithm. Nevertheless it is important for the model builder to be aware when he is dealing with a special kind of linear programming model. In order to solve his model it may be useful to adapt the simplex algorithm to suit the special structure. It may even, sometimes, be worth ignoring the special structure and using a general purpose package program. Since such programs are often highly efficient and well designed their speeds outweigh the algorithmic efficiency of less well designed but more specialized programs.

It is not intended that the coverage of this topic be comprehensive. The main aim is simply to show the connection between network models and linear programming models. References are given to much fuller treatments of the subject.

The Transportation Problem

This famous type of problem first described by Hitchcock (1941) is usefully regarded as one of obtaining the minimum cost flow through a special type of network.

Suppose that a number of suppliers ($S_1, S_2, \ldots, S_m$) are to provide a number of customers ($T_1, T_2, \ldots, T_n$) with a commodity. The transportation problem is how to meet each customer's requirement, while not exceeding the capacity of any supplier, at minimum cost. Costs are known for supplying one unit of the commodity from each S_i to each T_j. In some cases it may not be possible to supply a particular customer T_j from a particular supplier S_i. It is sometimes useful to regard these costs as infinite in such cases. In distribution problems these costs will often be related to the distances between S_i and T_j. It is assumed that the capacity of each supplier (over some period such as a year) is known and the requirement of each customer T_j is also known. In order to describe the problem further we will consider a small numerical example.

Example 1. A Transportation Problem

Three suppliers (S_1, S_2, S_3) are used to provide four customers (T_1, T_2, T_3, T_4) with their requirements for a particular commodity over a year. The yearly capacities of the suppliers and requirements of the customers are given below (in suitable units)

Suppliers	S_1	S_2	S_3	
Capacities (per year)	135	56	93	
Customers	T_1	T_2	T_3	T_4
Requirements (per year)	62	83	39	91

The unit costs for supplying each customer from each supplier are given in Table 5.3 (in £/unit).

Table 5.3

	Customer			
Supplier	T_1	T_2	T_3	T_4
S_1	132	—[a]	97	103
S_2	85	91	—	—
S_3	106	89	100	98

[a] A dash indicates the impossibility of certain suppliers for certain depots or customers.

We can easily formulate this problem as a conventional linear programming model by introducing variables x_{ij} to represent the quantity of the commodity sent from S_i to T_j in a year. The resultant model is:

Minimize

$$132x_{11} + Mx_{12} + 97x_{13} + 103x_{14} + 85x_{21} + 91x_{22} + Mx_{23} + Mx_{24} + 106x_{31} + 89x_{32} + 100x_{33} + 98x_{34} \quad (1)$$

subject to

$$\begin{array}{llllllllllllll}
x_{11} + x_{12} + x_{13} + x_{14} & & & & & & & & & & & \leqslant & 135, & (2)\\
& & & & x_{21} + x_{22} + x_{23} + x_{24} & & & & & & & \leqslant & 56, & (3)\\
& & & & & & & & x_{31} + x_{32} + x_{33} + x_{34} & & & \leqslant & 93, & (4)\\
x_{11} & & & & +x_{21} & & & & +x_{31} & & & = & 62, & (5)\\
& x_{12} & & & & +x_{22} & & & & +x_{32} & & = & 83, & (6)\\
& & x_{13} & & & & +x_{23} & & & & +x_{33} & = & 39, & (7)\\
& & & x_{14} & & & & +x_{24} & & & +x_{34} & = & 91, & (8)
\end{array}$$

$$x_{ij} \geqslant 0, \quad \text{all } i, j.$$

This model obviously has a very special structure to which we will refer later. Notice that we have included variables for non-allowed routes in the model with objective coefficients M (some very large number). This has been done simply to preserve the pattern of the model. In practice, if we were to solve the model as a linear programming problem of this form we would simply leave these variables out.

Constraints (2), (3), and (4) are known as *availability constraints*. There is one such constraint for each of the three suppliers. These constraints ensure that the total quantity out of a supplier (in a year) does not exceed his capacity. Constraints (5), (6), (7), and (8) are known as *requirement constraints*. These constraints ensure that each customer obtains his requirement. In some formulations of the transportation problem constraints (2), (3), and (4) are treated as '=' instead of '$\leqslant$'. If the sum total of the availabilities exactly matches the sum total of the requirements then this is acceptable since all capacities must obviously be completely exhausted. In a case such as our numerical examples, however, this is not so. Total capacity (284) exceeds total demand (275). This can be coped with by introducing a dummy customer T_5 with a requirement for the excess of 9. If the cost of meeting this requirement of T_5 from each supplier S_i, is made zero we have equated total capacity to total demand with no inaccuracy in our modified model. The three constraints (2), (3), and (4) could then be made '='. When special algorithms are used to solve the transportation problem the employment of devices such as this is sometimes necessary. For a conventional linear programming formulation of the problem this is not necessary. For a general transportation problem with m suppliers $(S_1, S_2, \ldots, S_m)$ and n customers $(T_1, T_2, \ldots, T_n)$ there will be m availability constraints and n requirement constraints giving a total of $m + n$ constraints. If each supplier can be potentially used for each customer there will be mn variables in the linear programming model. Clearly for practical problems involving large numbers of suppliers and customers the linear programming model could be very large. This is one motive for using special algorithms.

The above problem can be looked at graphically as illustrated in Figure 5.1.

In the network of Figure 5.1 we have the suppliers S_1, S_2, and S_3 and the five customers T_1, T_2, T_3, T_4, and T_5 (including the dummy customer). S_i and T_j provide the *nodes* of the network to which we have attached the (positive) capacities or (negative) requirements. The possible supply patterns S_i to T_j provide the *arcs* of the network to which we have attached the unit supply costs. Our problem can now be regarded more abstractly as one where we wish to obtain the *minimum cost flow* through the network. The S_i nodes are 'sources' for the flow entering the system and the T_j nodes are 'sinks' where flow leaves the system. We must ensure that there is continuity of flow at each node (total flow in equals total flow out). These conditions give rise to material balance constraints of the type discussed in Section 3.3.

If x_{ij} represents the quantity of flow in the arc i to j we obtain the following constraints:

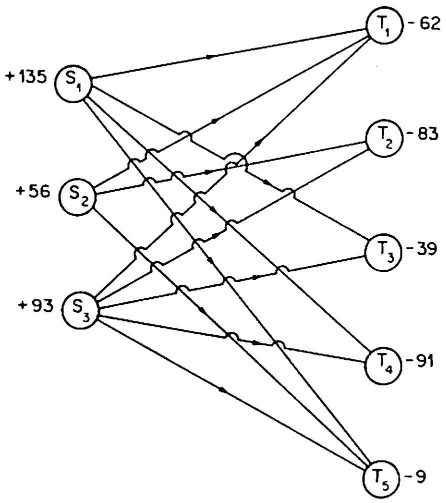

Figure 5.1

$$-x_{11}-x_{13}-x_{14}-x_{15} = -135, \tag{9}$$
$$-x_{21}-x_{22}-x_{25} = -56, \tag{10}$$
$$-x_{31}-x_{32}-x_{33}-x_{34}-x_{35} = -93, \tag{11}$$
$$x_{11}+x_{21}+x_{31} = 62, \tag{12}$$
$$x_{22}+x_{32} = 83, \tag{13}$$
$$x_{13}+x_{33} = 39, \tag{14}$$
$$x_{14}+x_{34} = 91, \tag{15}$$
$$x_{15}+x_{25}+x_{35} = 9. \tag{16}$$

These constraints are clearly equivalent to the constraints (2) to (8). We have, however, added the dummy customer T_5. This has resulted in the additional variables x_{15}, x_{25}, and x_{35}, and an added constraint (16), but allowed us to deal entirely with '=' constraints. We have also revised the signs on both sides of the availability constraints. For more general minimum cost flow problems, which we consider later, it is convenient to give negative coefficients to flows *out* of a node and positive coefficients to flows *in*. Therefore this convention has been applied here.

The transportation problem also arises in less obvious contexts than distribution. We give a numerical example below of a production planning problem.

Example 2. Production Planning

A company produces a commodity in two shifts (regular working and overtime) to meet known demands for the present and future. Over the next four months the production capacities and demands (in thousands of units producible) are

	January	February	March	April
Regular working	100	150	140	160
Overtime	50	75	70	80
Demand	80	200	300	200

The cost of production of each unit is £1 if done in regular working or £1·50 if done in overtime. Units produced can be stored before delivery at a cost of £0·30 per month per unit.

The problem is how much to produce each month to satisfy present and future demand.

It is convenient to summarize the costs in Table 5.4 (in £).

Clearly it is impossible to produce for demand of an earlier month. This is represented by a dash in the positions indicated (an infinite unit cost). The other

Table 5.4

Production		Demand January	February	March	April
January	Regular	1	1·3	1·6	1·9
	Overtime	1·5	1·8	2·1	2·4
February	Regular	—	1	1·3	1·6
	Overtime	—	1·5	1·8	2·1
March	Regular	—	—	1	1·3
	Overtime	—	—	1·5	1·8
April	Regular	—	—	—	1
	Overtime	—	—	—	1·5

unit costs arise from a combination of production and storage costs, e.g. production in January by overtime working for delivery in March gives a unit cost of £1·50 (production) + £0·60 (storage) = £2·10. This cost matrix is of the same form as that given for the transportation problem in Table 5.3. Although the problem here is not one of distribution it can still therefore be regarded as a transportation problem. In this case there are eight sources and five sinks including the 'surplus' demand of 45 units.

Transportation problems are obviously expressed much more compactly in a square array such as Tables 5.3 and 5.4 rather than as a linear programming matrix. This is one virtue of using a special purpose algorithm. Dantzig (1951) uses the simplex algorithm but works within this compact format. The special structure results in the algorithm taking a particularly simple form. An alternative algorithm for the transportation problem is due to Ford and Fulkerson (1956). This algorithm is usefully thought of as a special case of a general algorithm for finding the minimum cost flow through a network. Such problems are considered below and described in Ford and Fulkerson (1962).

As a result of their special structure transportation problems are particularly easy to solve in comparison with other linear programming problems of comparable size. They also have (together with some other network flow problems) the very important property that so long as the availabilities and requirements at the sources and sinks are integral the values of the variables in the optimal solution will also be so. For example, so long as the right-hand side coefficients in the constraints (2) to (8) of the linear programming problem of Example 1 are integers, the variable values in the optimal solution will be as well. This rather surprising property of the transportation problem is computationally very important in many circumstances since it avoids the necessity of using *integer* *programming* to ensure that variables take integer values. As will be discussed in Chapters 8, 9, and 10 integer programming models are generally much more difficult to solve than linear programming models.

Sufficient conditions for a model to be expressible as a network flow problem are discussed in Section 10.1. The recognition of such conditions is important since it allows the use of specialized efficient algorithms and avoids the use of computationally expensive integer programming.

A further constraint that sometimes applies to transportation problems is that there are limits to the possible flow from a source to a sink. This gives rise to the *capacitated transportation problem*. There may be both lower and upper limits for the flow in each arc. For the linear programming formulation of the transportation problem (such as exemplified in Example 1 above) such limits can be accommodated by simple bounds on the variables:

$$0 \leqslant l_{ij} \leqslant x_{ij} \leqslant u_{ij}.$$

Frequently l_{ij} will be 0. Capacitated transportation problems can, like the ordinary transportation problem, be solved by straightforward extensions to the special purpose algorithms mentioned above.

Another non-distribution example of the transportation problem is described by Stanley, Honig and Gainen (1954), who describe how the problem arises in deciding how a government should award contracts.

The Assignment Problem

This is the problem of assigning n people to n jobs so as to maximize some overall level of competence. For example person i might take an average time t_{ij} to do job j. In order to assign each person to a job and to fill each job so as to minimize total time for all tasks our problem would be:

$$\text{Minimize} \quad \sum_{i,j} t_{ij}x_{ij}$$

$$\text{subject to} \quad \sum_{i} x_{ij} = 1 \text{ for all } j, \tag{17}$$

$$\sum_{j} x_{ij} = 1 \text{ for all } i, \tag{18}$$

where

$$x_{ij} = \begin{cases} 1 & \text{if person } i \text{ is assigned to job } j, \\ 0 & \text{otherwise.} \end{cases}$$

This can obviously be regarded as a special case of the transportation problem. We can regard it as a problem with n sources and n sinks. Each source has an availability of 1 unit and each sink has a demand of 1 unit. Constraints (17) impose the condition that each job be filled. Constraints (18) impose the condition that every person be assigned a job.

It might appear that this problem demands integer programming in order to ensure that x_{ij} can only take the values 0 or 1. Fortunately, however, because this problem is a special case of the transportation problem the integrality property mentioned above holds. If we solve an assignment problem as a conventional linear programming model we can be certain that the optimal solution will give integer values to the x_{ij} (0 or 1). If marriage is regarded as an assignment problem of this kind Dantzig has suggested that the integrality property shows that monogamy leads to greatest overall happiness!

Obviously assignment problems could be solved as linear programming models, although the resultant models could be very large. For example, the assigning of 100 people to 100 jobs would lead to a model with 10 000 variables. It is much more efficient to use a specialized algorithm. One of the specialized algorithms for the transportation problem could obviously be applied. The most efficient method known is one allied to the Ford and Fulkerson algorithm but more specialized. This is known as the Hungarian method and is described by Kuhn (1955).

The Transhipment Problem

This is an extension of the transportation problem, due to Orden (1956). In this problem it is possible to distribute the commodity through intermediate sources and through intermediate sinks as well as from sources to sinks. In Example 1 we could allow flow (at a certain cost) between suppliers S_1, S_2, and S_3 as well as between customers T_1, T_2, T_3, and T_4. It might be advantageous sometimes to send a commodity from one supplier to another before dispatching it to the customer. Similarly it might be advantageous to send a commodity to a customer via another customer first. The transhipment problem allows for these possibilities.

If we extend Example 1 to allow the use of certain intermediate sources and sinks our graphical representation would be of the form of Figure 5.2.

Costs have now been attached to the arcs between sources and the arcs between sinks. Notice that it is sometimes possible to go either way between sources (or sinks), at not necessarily the same cost.

It is possible to convert a transhipment problem into a transportation problem. To do this the sources and sinks are considered firstly as being all sources and then as all sinks. When considered as sinks, sources have no availabilities, and when considered as sources, sinks have no requirements. Flow from a sink to a source is not allowed. For the transhipment extension of Example 1 illustrated in Figure 5.2 we can draw up the unit cost array of Table 5.5. 'Sources' T_1, T_2, T_3, and T_5 will have zero availabilities and 'sinks' S_1, S_2, and S_3 zero requirements.

Transhipment problems can obviously be formulated as linear programming models just as the transportation problem can. Again it is often desirable to use

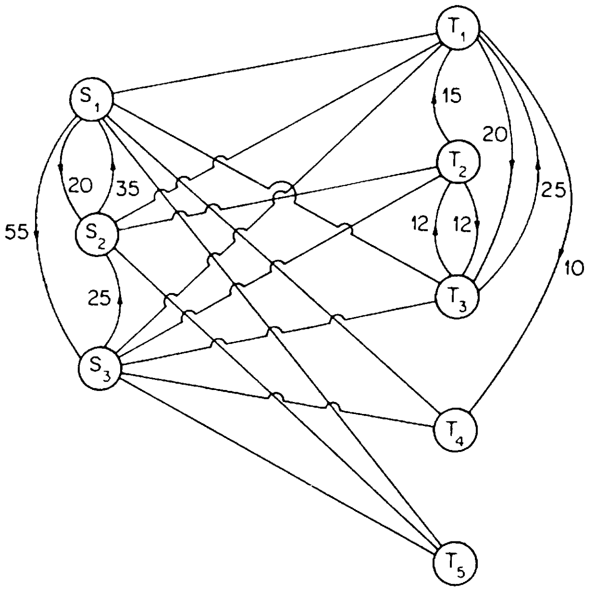

Figure 5.2

a specialized algorithm such as that described by Dantzig (1951) or by Ford and Fulkerson (1962).

As with transportation problems, transhipment problems can be extended to capacitated transhipment problems where the arcs have upper and lower capacity limitations. These can also be solved by specialized algorithms.

An application of the transhipment problem outside the field of distribution is described by Srinivasan (1974).

Table 5.5

	Sinks							
Sources	S_1	S_2	S_3	T_1	T_2	T_3	T_4	(T_5)
S_1	—	20	55	132	—	97	103	0
S_2	35	—	—	85	91	—	—	0
S_3	—	25	—	106	89	100	98	0
T_1	—	—	—	—	—	20	10	—
T_2	—	—	—	15	—	12	—	—
T_3	—	—	—	25	12	—	—	—
T_4	—	—	—	—	—	—	—	—
(T_5)	—	—	—	—	—	—	—	—

The Minimum Cost Flow Problem

The transportation, transhipment, and assignment problems are all special cases of the general problem of finding a minimum cost flow through a network. Such problems may have upper and lower capacities attached to the arcs in the capacitated case. The uncapacitated case will be considered here.

Example 3. Minimum Cost Flow

The network in Figure 5.3 has two sources 0 and 1 with availabilities of 10 and 15. There are three sinks 5, 6, and 7 with requirements 9,10, and 6 respectively. Each arc has a unit cost of flow associated with it.

The arcs are 'directed' in the sense that only flow in the direction marked by the arrow is allowed. If flow is allowable in the opposite direction as well, this is indicated by another arc in the reverse direction. This happens in the case of the two arcs between node 2 and node 4.

The problem is simply to satisfy the requirements at the sinks by flow through the network from the sources at total minimum cost. In this case the total availability exactly equals the total requirement. This can always be made possible by the use of a dummy sink if necessary as described for the transportation problem in Example 1.

The linear programming formulation of Example 3 is:

Minimize

$$5x_{02} + 4x_{13} + 2x_{23} + 6x_{24} + 5x_{25} + x_{34} + 2x_{37} + 4x_{42} + 6x_{45} + 3x_{46} + 4x_{76}$$

subject to

$$\begin{array}{lr}
-x_{02} = -10, & (19) \\
-x_{13} = -15, & (20) \\
x_{02} - x_{23} - x_{24} - x_{25} + x_{42} = 0, & (21) \\
x_{13} + x_{23} - x_{34} - x_{37} = 0, & (22) \\
x_{24} + x_{34} - x_{42} - x_{45} - x_{46} = 0, & (23) \\
x_{25} + x_{45} = 9, & (24) \\
x_{46} + x_{76} = 10, & (25) \\
x_{37} - x_{76} = 6. & (26)
\end{array}$$

In order to be systematic about this formulation it is convenient to regard each constraint as arising from the *material balance requirement* at each node. For example, at node 2 it is necessary to ensure that the total flow in ($x_{02} + x_{42}$) is the same as the total flow out ($x_{23} + x_{24} + x_{25}$). This is achieved by constraint (21). At node 7 which is a sink the total flow in (x_{37}) must again be the same as the total flow out ($x_{76} + 6$). This gives constraint (26).

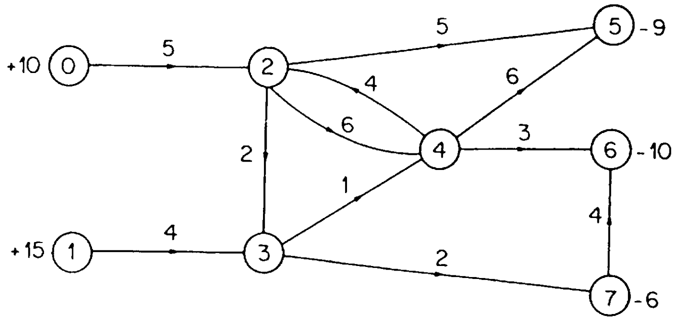

Figure 5.3

The matrix of coefficients in constraints (19) to (26) of the model above is known as the *incidence matrix* of the network in Figure 5.3. It clearly has a very special structure. This structure is further discussed in Section 10.1 since, like the transportation problem, the minimum cost flow problem (whether capacitated or not) can be guaranteed to yield an optimal *integer* solution so long as the availabilities, requirements, and arc capacities are integer.

As with the other types of model so far discussed in this section it is generally more efficient to use specialized algorithms. Those due to Dantzig (1951) and Ford and Fulkerson (1962) are also applicable here.

A comprehensive survey of applications of the minimum cost network flow problem is given by Bradley (1975). Other useful references are Glover and Klingman (1977) and Jensen and Barnes (1980).

It is sometimes possible to convert a linear programming model into a form which is immediately convertible into a network flow model. A procedure for doing this, or showing such a conversion to be impossible, is given in Section 5.4.

If arcs have lower or upper bounds (or both) on their capacities, then (as with the special case of the transportation problem) it is possible to adapt the special algorithms to cope with this. It is, however, worth pointing out that such models can be converted to the uncapacitated case. This might be necessary if a program was being used which could not deal with such bounds.

Suppose the flow from node i to node j had a lower bound of l (and cost c_{ij}). An extra node i' can be added with a new arc from i to i'. If there is an external flow l *out* of i and *into* i', as shown in Figure 5.4, this provides the necessary restriction. Similarly Figure 5.5 demonstrates how an upper bound of u can be imposed on the flow in arc ij.

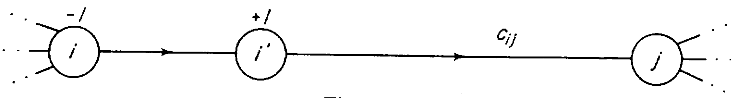

Figure 5.4

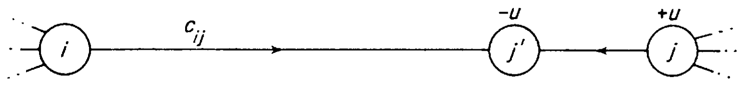

Figure 5.5

It is important to ensure that a minimum cost network flow problem is well defined. For example the unit flow costs are generally non-negative. If negative costs are allowed it is important to ensure that cost cannot be minimized indefinitely (giving an *unbounded* problem). This could happen, for example if arc 2–4 were given in a unit cost of -6 instead of $+6$. Going round the loop indefinitely would continuously reduce the cost.

A minimum cost network flow problem, DISTRIBUTION, is given in Part 2.

An extension of the problem of finding the minimum cost flow of a single commodity through a network is the problem of minimizing the cost of the flows of several commodities through a network. This is the *minimum* cost *multi-commodity network flow problem*. There will be capacity limitations on the flows of individual commodities through certain arcs as well as capacity limitations on the total flow of all commodities through individual arcs. For example, in the network of Figure 5.3 one commodity might flow between source 0 and sink 5 and a second commodity flow between source 1 and sinks 6 and 7. This type of problem can again be formulated as a linear programming model. The resultant model has a block angular structure of the type discussed in Section 4.1. The block angular structure makes the decomposition procedure of Dantzig and Wolfe, which is discussed in Section 4.2, applicable. In fact this leads to another linear programming formulation of the problem. This aspect of the minimum cost multi-commodity network flow problem is discussed by Tomlin (1966).

Apart from decomposition there are no special algorithms applicable to the general minimum cost multi-commodity network flow problem. The (often large) linear programming model resulting from such a problem is best solved by the standard revised simplex algorithm using a package programme.

Charnes and Cooper (1961a) formulate a traffic flow problem as a minimum cost multi-commodity network flow model.

This extension of the single-commodity network flow linear programming model to more than one commodity destroys the property that guarantees an integral optimal solution. Fractional values for the flows may result from the optimum solution to the linear programming model even if all capacities, availabilities, and requirements are integral. If the nature of the problem requires an optimal integer solution it is necessary to resort to *integer programming*.

Another important extension of the minimum cost network flow model is the *generalized network flow model*. This is sometimes known as the *network flow with gains model*. In this extension the flow in an arc may alter between the two nodes. A multiplier is then associated with each arc which gives the factor by which flow is altered. Situations which require this modification result from, for example, evaporation, wastage or application of interest rates. Glover and Klingman (1977) give applications. If it is necessary that the flows be *integer* then this can no longer be guaranteed from a linear programming solution. It is necessary to use integer programming methods. Nevertheless it is possible to exploit this simple structure to good effect in the algorithms used. Glover *et al.* (1978) describe such a method. In fact any 0–1 integer programming problem can be converted into such a generalized network model where flows must be integer. This is shown by Glover and Mulvey (1980).

The Shortest Path Problem

This is the problem of finding a shortest path between two nodes through a network. Rather surprisingly this problem can be regarded as a special case of the minimum cost flow problem.

Example 4. Finding the Shortest Path Through a Network

In the network in Figure 5.6 we wish to find the shortest path between node 0 and node 8. The lengths of each arc are marked.

We can reduce this problem to one of finding a minimum cost flow through the network by giving node 0 an availability of 1 unit (a source) and giving node 8 a requirement of 1 unit (a sink). Because of the property that minimum cost flow (as with transportation, transhipment and assignment) problems have of guaranteeing integral optimal flows, when solved as linear programming models, we can be sure that this minimal cost flow through each arc in Figure 5.6 will be 0 or 1. Exactly one of the arcs out of node 0 will therefore have a flow of 1 and exactly one of the flows into node 8 will have a flow of 1. Similarly intermediate nodes on the flow path will have exactly one arc with flow in and one with flow out. The 'cost' of the optimal flow path will give the shortest route between 0 and 8.

Although it is possible to use conventional linear programming to solve shortest path problems it would be more efficient to use a specialized algorithm. One of the most efficient such algorithms is due to Dijkstra (1959).

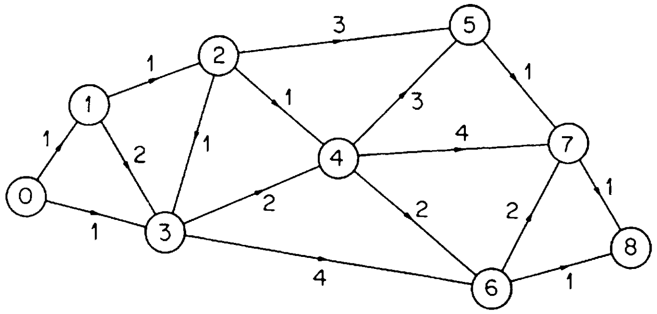

Figure 5.6

Maximum Flow Through Network

When a network has capacity limitations on the flow through arcs there is often interest in finding the maximum flow of some commodity between sources and sinks. We will again consider the network of Example 3 but our objective will now be to maximize the flows into the sources and out of the sinks rather than these quantities being given. Each arc has now been given an upper capacity which is the figure attached to it in Figure 5.7.

Example 5. Maximizing the Flow Through a Network

This problem can again be formulated as a linear programming model. The variables and constraints will be the same as those in Example 3 apart from the introduction of five new variables x_{S0}, x_{S1}, x_{5T}, x_{6T}, and x_{7T} representing the flows into sources 0 and 1 and out of sinks 5, 6, and 7. The resultant model is

$$\text{Maximize} \qquad x_{S0} + x_{S1}$$

subject to

$$
\begin{aligned}
x_{S0} - x_{02} &= 0, && (27)\\
x_{S1} - x_{13} &= 0, && (28)\\
x_{02} - x_{23} - x_{24} - x_{25} + x_{42} &= 0, && (29)\\
x_{13} + x_{23} - x_{34} - x_{37} &= 0, && (30)\\
x_{24} + x_{34} - x_{42} - x_{45} - x_{46} &= 0, && (31)\\
x_{25} + x_{45} - x_{5T} &= 0, && (32)\\
x_{46} + x_{76} - x_{6T} &= 0, && (33)\\
x_{37} - x_{76} - x_{7T} &= 0, && (34)
\end{aligned}
$$

$$x_{02} \leqslant 12,\ x_{13} \leqslant 20,\ x_{23} \leqslant 6,\ x_{24} \leqslant 3,\ x_{25} \leqslant 6,\ x_{34} \leqslant 7,\ x_{37} \leqslant 9,\ x_{42} \leqslant 2,\ x_{45} \leqslant 5,\ x_{46} \leqslant 8,\ x_{76} \leqslant 4.$$

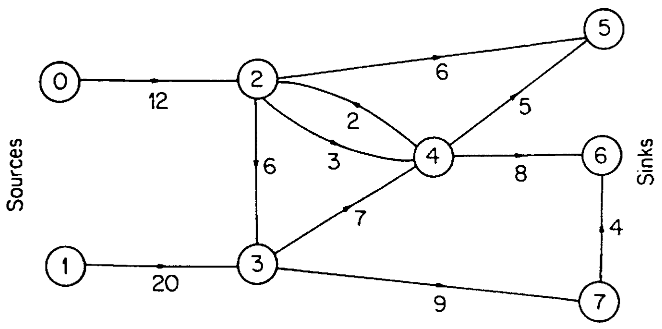

Figure 5.7

Again this type of model has the property that the optimal solution will give integer flows so long as the capacities are integer.

It is again more efficient to use a specialized algorithm for this type of problem. Such an algorithm is described by Ford and Fulkerson (1962).

Critical Path Analysis

This is a method of planning projects (often in the construction industry) which can be represented by a network. The arcs of the network represent *activities* occupying a duration of time, e.g. building the walls of a house, and the nodes are used to indicate the termination and beginning of activities. Once a project has been represented by such a network model the network can be analysed to answer a number of questions such as

(i) How long will it take to complete the project?
(ii) Which activities can be delayed if necessary and by how long without delaying the overall project?

Such a mathematical analysis of this kind of network is known as *critical path analysis*. The arcs for those activities in the network which cannot be delayed without affecting the overall completion time of the project can be shown to lie on a path. This *critical path* is in fact the *longest path* through the network. The problem of finding the critical path is a special kind of linear programming problem although the special structure of the problem makes a specialized algorithm appropriate.

Example 6. Finding the Critical Path in a Network

The network in Figure 5.8 represents a project of building a house. Each arc represents some activity forming part of the project. The durations (days) of the activities are attached to the corresponding arcs. The arc 4–2 marked with a broken line is a dummy activity having no duration. Its only purpose is to prevent activity 2–5 starting before activity 3–4 has finished.

In order to formulate this problem as a linear programming model we can introduce the following variables:

t_0	start time for activities 0–1, 0–3, and 0–2
t_1	start time for activity 1–3
t_2	start time for activity 2–5
t_3	start time for activity 3–4
t_4	start time for activities 4–2 and 4–5
t_5	start time for activity 5–6
z	finish time for the project.

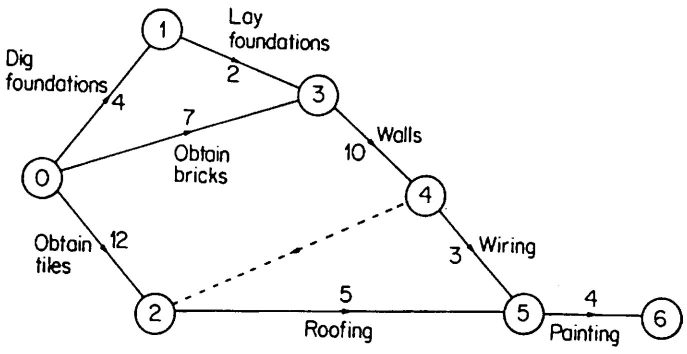

Figure 5.8

Our model is then:

Minimize z

subject to

$$\begin{array}{rrrrrrrl} -t_0 & +t_1 & & & & & & \geqslant 4, \quad (35)\\ -t_0 & & +t_2 & & & & & \geqslant 12, \quad (36)\\ -t_0 & & & +t_3 & & & & \geqslant 7, \quad (37)\\ & -t_1 & & +t_3 & & & & \geqslant 2, \quad (38)\\ & & & -t_3 & +t_4 & & & \geqslant 10, \quad (39)\\ & & t_2 & & -t_4 & & & \geqslant 0, \quad (40)\\ & & -t_2 & & & +t_5 & & \geqslant 5, \quad (41)\\ & & & & -t_4 & +t_5 & & \geqslant 3, \quad (42)\\ & & & & & -t_5 & +z & \geqslant 4. \quad (43) \end{array}$$

Each constraint represents a *sequencing relation* between certain activities. For example activity 3–4 cannot start before activity 1–3 has finished. This gives $t_3 \geqslant t_1 + 2$, which leads to constraint (38). Finally, since the project cannot be completed before activity 5–6 is finished we get constraint (43).

On solving this model we obtain the following results:

$$\text{Project completion time } (z) = 26 \text{ days}$$

$$\begin{aligned} t_0 &= 0,\\ t_1 &= 4,\\ t_2 &= 17,\\ t_3 &= 7,\\ t_4 &= 17,\\ t_5 &= 22. \end{aligned}$$

The critical path is clearly 0–3–4–2–5–6.

Building and conventionally solving the above linear programming model would be an inefficient method of finding the critical path. Special algorithms exist and there are widely used package programs for doing critical path analysis. Many extensions of the problem of scheduling a project in this way can be considered but are beyond the scope of this book. A full discussion of this subject is contained in Lockyer (1967).

One very practical extension of the problem is that of *allocating resources to the activities* in a project network. For example, in the network of Figure 5.8 both activity 4–5 (wiring) and activity 2–5 (roofing) may require people (although in this contrived example they would probably have different skills). If activity 4–5 requires three people and activity 2–5 requires six and there are only eight available, the optimal schedule given above is unattainable and one of those activities will have to be delayed or extended. The problem is then how to reschedule to achieve some objective. For example, the objective might well be to delay the overall completion time as little as possible. Alternatively there might be a desire to 'smooth' the usage of this and other resources over time. This extension to the problem is mentioned again in Section 9.5 since it gives rise to an integer programming extension to the linear programming problem of the type above. Nevertheless integer programming would be generally far too costly in computer time to justify solving this type of problem in this way. A problem which gives rise to a very simple network is job-shop scheduling. This problem of scheduling jobs on machines can be regarded as a problem of allocating resources to activities in a network. The operations in the job shop (machining, etc.) give the activities. Sequencing relations between those operations give a (simple) network structure. The resources to be allocated are the limited machines.

All the problems described in this section (apart from the minimum cost multi-commodity network flow problem) are best tackled through specialized algorithms rather than the revised simplex algorithm available on commercial package programs. The reason for describing these problems and showing how they can, if necessary, be modelled as linear programs is that many practical problems are made up, in part, of network problems. Such problems often, however, contain additional complications which make it impossible to use a pure network model. This is where the conventional linear programming formulation becomes important. Many practical linear programming models have a very large network component. Such a feature usually makes very large models easy to solve using package programs. Some package programs have special features to take advantage of some of the network structure within a model. An example of this is the *generalized upper bound* (GUB) type of constraint which was mentioned in Section 3.3. In the linear programming formulation of the transportation problem in Example 1 constraints (2), (3), and (4) could be regarded as GUB constraints and not explicitly represented as constraints if a package with this facility was used. Alternatively (and preferably because there are more of them) constraints (5), (6), (7), and (8) could be represented as GUB constraints. The use of the GUB facility makes the solution of many network flow problems (or problems with a network flow component) particularly easy by conventional linear programming.

Another virtue in recognizing a network flow component in a linear programming model is that many variables are likely to come out at integer values in the optimal solution. It has been pointed out that this happens for all the variables in most of the network flow problems described in this section so long as the right-hand side coefficients are integers. If a model is 'not quite' of a network flow kind it is probable that the great majority of variables will still take integer values in the optimal linear programming solution. The computational difficulties of forcing all these variables to be integer by integer programming will be much reduced. This topic is discussed much more fully in Chapter 10.

There is also great virtue to be gained from remodelling a problem in order to get it into the form of a network flow model. This then opens up the possibility of using a special purpose algorithm. An example of how this can sometimes be done for a practical problem is given by Veinott and Wagner (1962). Dantzig (1969) shows how a hospital admissions scheduling program can be remodelled to give a linear programming model with a large network flow component. He then exploits this structure by use of the GUB facility. Other examples of reformulations into network flow models are Daniel (1973), Wilson and Willis (1983), and Cheshire, McKinnon and Williams (1984).

An automatic way of either converting a linear programming model to a network flow model or showing such a conversion to be impossible is given in Section 5.4. The recognition or conversion of a model as a network structure relieves the need to use the computationally much more costly methods of integer programming.

In Section 6.2 the concept of the dual of a linear programming model is described. Every linear programming model has a corresponding model known as the *dual model*. The optimal solution to the dual model is very closely related to the optimal solution of the original model. In fact the optimal solution to either one can be derived very easily from the optimal solution to the other. It turns out that many practical problems give rise to a linear programming model which is the dual of a network flow model. In such circumstances it could well be worth using a specialized algorithm on the corresponding network flow model. Moreover, the dual of any of the types of network flow mode mentioned here (apart from the minimum cost multi-commodity network flow model) also has the property of guaranteeing optimal integer solutions (so long as the objective coefficients of the original model are integers). The recognition of this type of model can, again, be of great practical importance for this reason. This topic is further discussed in Sections 10.1 and 10.2.

In Part 2 the OPENCAST MINING problem can be formulated as the dual of a network flow problem. The formulation is discussed in Part 3. The MINING problem of Part 2 can be formulated as an integer programming model, a large proportion of which is the dual of a network flow model. Some examples of such models are given in Williams (1982).

One famous network problem which has not been discussed in this section is the *travelling salesman problem*. This is the problem of finding a minimum distance (cost) route round a given set of cities. This problem cannot generally be solved by a linear programming model, in spite of its apparent similarity to the assignment problem. It can, however, be modelled as an integer programming extension to the assignment problem and is fully discussed in Section 9.5.

5.4 Converting Linear Programs to Networks

The advantages of converting linear programs to minimum cost network flow models, if possible, have already been discussed in Section 5.3. They are further discussed in Section 10.1 since minimum cost network flow models have *integer* optimal solutions (so long as external flows in and out are integer). This relieves the need to use the much more costly procedures of integer programming.

We outline a method described by Baston, Rahmouni and Williams (1991) for converting linear programs to network flow models. Another procedure, but expressed in the more abstract language of matroid theory, is given by Bixby and Cunningham (1980).

In order to illustrate the method we will take a numerical example.

$$\begin{array}{lrrrrrrrrl}
\text{Minimize} & c_1x_1 & + c_2x_2 & + c_3x_3 & + c_4x_4 & & & & & \\
\text{subject to} & 2x_1 & & & & + x_5 & & & & = b_1, \\
& 6x_1 & & + 9x_3 & & & + x_6 & & & = b_2, \\
& -8x_1 + & 4x_2 & & - 8x_4 & & & + x_7 & & = b_3, \\
& & x_2 + & 3x_2 & - 2x_4 & & & & + x_8 & = b_4, \\
& & x_2 + & 3x_3 & & & & & + x_9 & = b_5, \\
& & \multicolumn{8}{r}{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9 \geqslant 0.}
\end{array}$$

Since the conversion does not depend on the objective or right-hand side coefficients we give these in a general form. In this example we assume all the original constraints were of the ' $\leqslant$ ' form and that *slack* variables have been added to make them equations. For ' $\geqslant$ ' constraints *surplus* variables would be subtracted. If any of the original constraints were equations then we would add *artificial* variables (variables constrained to take the value zero). These *logical* (slack, surplus or artificial) variables will represent arcs in the network created. For the case of artificial variables these arcs will finally be deleted.

We carry out the following transformations:

(i) Scale the rows and columns in order to make the constraint coefficients 0 or ± 1 if possible.

This may not be possible in which case the conversion to a network is not possible. In most practical problems (such as those referenced in Section 5.3) for which it is worth attempting a conversion these coefficients will already be 0 or ± 1.

In this example the resultant scaled coefficients are given below.

6	7	8	9	1	2	3	4	5	
$\frac{1}{2}c_1$	c_2	$\frac{1}{3}c_3$	$\frac{1}{2}c_4$						
1				1					$= b_1$
1		1			1				$= \frac{1}{3}b_2$
−1	1		−1			1			$= \frac{1}{4}b_3$
	1	1	−1				1		$= b_4$
	1	1						1	$= b_5$

It is also convenient to number the variables. The logical variables are numbered 1 to 5 and the original variables 6 to 9.

(ii) In this step the signs of the non-zero coefficients (±1) are ignored. The arcs corresponding to the logical variables are arranged in the form of a *spanning tree* of the network. This spanning tree must be *compatible* with the original variables of the model in the following sense: the original variables each form a *polygon* with some of the arcs of the spanning tree. Figure 5.9 illustrates how this is possible with the example.

The arc corresponding to variable 6 forms a polygon with the arcs of the tree corresponding to variables 1, 2 and 3 since variable 6 has non-zero entries in rows 1, 2 and 3. Similarly since variable 7 has non-zero entries in rows 3, 4 and 5, arc 7 forms a polygon with arcs 3, 4 and 5. Arcs 8 and 9 are similarly compatible with the tree.

It will not always be possible to find an arrangement of the logical arcs in the form of a spanning tree which is compatible with the other arcs in the manner demonstrated above. In such a case the network conversion is not possible. A systematic way of investigating whether it is possible to construct such a spanning tree or showing it to be impossible is described by Baston, Rahmouni and Williams (1991).

Having created such an *undirected* network the arcs are orientated in the following manner:

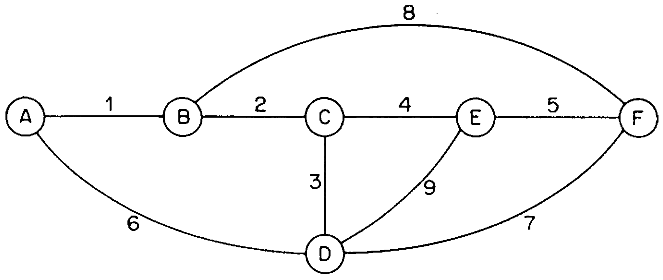

Figure 5.9

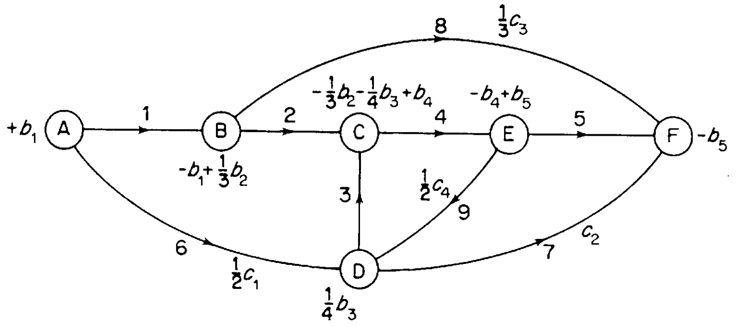

Figure 5.10

(iii) For each polygon the arcs of the tree in the polygon are given a direction *opposite* to that of the non-tree arc, when going round the polygon, if the entry in the column corresponding to this arc is $+1$. If the entry is -1 the arcs are given the *same* direction.

For this example the resulting orientations are shown in Figure 5.10. Arc 6, for example, has an opposite orientation to arcs 1 and 2 (variable 6 has $+1$ coefficients in rows 1 and 2) and the same orientation to arc 3 (variable 6 has a -1 coefficient in rows 3) when going round the polygon formed by arcs 1, 2, 3 and 6. Other arcs are orientated similarly according to this rule. It may not be possible to orientate the arcs in any manner compatible with the signs of the coefficients. In such a case a (directed) network is not constructible.

(iv) The (non-tree) arcs in the network are given unit costs equal to the scaled objective coefficients of corresponding variables.

(v) Each node is given an external flow *in* equal to the sum of the scaled right-hand side coefficients of the rows corresponding to those tree arcs leaving the node less the sum of the scaled right-hand side coefficients of the rows corresponding to those tree arcs entering the node.

In the example, in Figure 5.10, node C has tree-arc 4 leaving it (row 4 has scaled right-hand side b_4) and tree-arcs 2 and 3 entering it (rows 2 and 3 have scaled right-hand sides of $\frac{1}{3}b_2$ and $\frac{1}{4}b_3$ respectively). Hence the external flow into node C is $b_4 - \frac{1}{3}b_2 - \frac{1}{4}b_3$. (A negative external flow would, of course, be regarded as a positive flow out.) The other nodes have external flows calculated in a similar manner.

The resulting directed network with its external flows gives the required minimum cost network flow model equivalent to the original linear program. When solved the values of the flows in the arcs may have to be unscaled according to the scaling factors applied.