AN INVESTIGATION OF COMPUTER BASED TOOLS FOR MATHEMATICAL PROGRAMMING MODELLING

A Thesis submitted for the degree of Doctor of Philosophy

by

Cormac Anthony Lucas

Department of Mathematics and Statistics, Brunel University,

December 1986

Dedicated to my dear parents

Alphonsus Anthony and Mary Antoinette Lucas

ACKNOWLEDGEMENTS

I wish to thank my supervisor, Dr. Gautam Mitra, for the excellent guidance he has given me over the past four years. His encouragement and enthusiasm have greatly helped to maintain my motivation and interest in mathematical programming.

The advisory staff of Brunel's computer unit, especially Mr R Pank, have been very helpful in overcoming computer problems. I wish to thank them for their expertise and willingness to help.

Mrs Mary Storey and Mrs Pam Denham typed part of this thesis, and Mrs Barbara Yates drafted some of the diagrams, to all of these I am extremely grateful.

Finally, I would like to express my appreciation to the Science and Engineering Research Council for awarding me a grant to pursue a Ph.D. course of advanced study and research at Brunel University.

CONTENTS

CHAPTER 1 INTRODUCTION 1

1.1 Historical development of linear programming and mathematical programming systems 1

1.2 Historical development of computer assisted LP modelling 3

1.3 Mathematical programming: the major issues 4

1.4 Mathematical programming modelling: the major issues 6

1.5 Research focus and the structure of the thesis 10

CHAPTER 2 STRATEGY AND TACTICS OF LP MODELLING 12

2.1 Introduction 12

2.2 A logical analysis of the problem 12

2.3 Derivation of a mathematical statement: an example 15

2.4 LP user formulation of the problem 20

2.5 Further examples 23

CHAPTER 3 CURRENT APPROACHES TO COMPUTER ASSISTED MATHEMATICAL PROGRAMMING MODELLING 34

3.1 Introduction 34

3.2 Introductory and teaching systems 36

3.3 Activity based modelling systems 37

3.4 Modelling systems employing row-wise specification 38

3.5 Generic modelling tools 41

3.6 Artificial intelligence aids 44

CHAPTER 4 COMPUTER ASSISTED MATHEMATICAL PROGRAMMING (MODELLING) SYSTEM: CAMPS 49

4.1 Introduction 49

4.2 Salient and novel features of CAMPS 49

4.3 An annotated example 54

4.4 Support for separable and integer programming reformulation 62

CHAPTER 5 DESIGN AND IMPLEMENTATION ISSUES 65

5.1 Introduction 65

5.2 System overview 66

5.3 Menu and screenform control 69

5.4 Strategy and tools for handling data tables 70

5.5 Sreen management tools 74

5.6 Analysis of model and creation of matrix generator program 75

5.7 Integration with ANALYZE and model documentation 77

CHAPTER 6 AN APPROACH TO COMPUTER ASSISTED REFORMULATION OF INTEGER, SEPARABLE AND FUZZY PROGRAMMING PROBLEMS 78

6.1 Introduction 78

6.2 Statement of the general LP problem and notation 79

6.3 Analysis of bounds for linear forms 80

6.4 Representation of logical restrictions 83

6.5 Strategies for separating variables in nonlinear programming problems 88

6.6 Reformulation of fuzzy decision problems as max-min LP problems 96

6.7 Automatic approach to reformulation: a summary of issues 99

CHAPTER 7 DISCUSSIONS, NEW DIRECTIONS AND CONCLUSIONS 101

7.1 Introduction 101

7.2 Artificial intelligence and mathematical programming modelling 103

7.3 Programmer's interface 106

7.4 Summary and conclusions 109

REFERENCES 112

APPENDIX 1 A COMPARISON OF CAMPS WITH OTHER SYSTEMS 125

CHAPTER 1

INTRODUCTION

1.1 Historical development of linear programming and mathematical programming systems

Mathematical programming is concerned with the efficient use or allocation of limited resources to meet a desired objective. When all the relationships between the variables are linear then the most extensive and popular method of optimisation is linear programming (LP). The earliest and also most established method to solve LP problems was developed in 1947 by G.B. Dantzig [DANTZI51]. This method, well known as the simplex, has been extended since then to solve large problems. In practice LP problems tend to be large and sparse and it is well known that the number of non zero elements is much smaller than the number of elements in the constraint matrix. Thus a vast amount of computer storage is required if a medium to large matrix is stored explicitly, and substantial computer effort is needed to update the full matrix at each iteration. This led to a revision of the computational aspects of the simplex algorithm in 1953. Dantzig [DANTZI53] and Orchard-Hays [ORCHAR56] created a more accurate and faster algorithm, which took full advantage of the mathematical properties of sparse simultaneous linear equations.

In addition to solving LP problems there are many algorithms for the solution of certain problem structures and known forms of nonlinearities. The very early mathematical programming systems used to represent upper bounds on variables as separate equations. Subsequently these were handled implicitly. Further, when sets of variables have common

upper bounds such that the constraint takes the following form

$$\sum_j x_{jk} = b_k, \quad k = 1,2,\ldots t$$

then the right hand sides, b_k, are general upper bounds (GUB) on the appropriate sets of variables. These are also handled implicitly by the mathematical programming system [BEALE70] and hence reduce the problem size.

If in the statement of the LP problem it is desired that some or all of the variables are constrained to take integer values (MIP), then this can be solved by a Branch and Bound algorithm. This method employs the approach of proposing mutually exclussive subproblems of new LPs and solves these by the dual simplex algorithm [MITRA70]. Recently considerable development has taken place to improve the performance of integer optimisers [BEALE85], [HOFPAD85] and [WOLSEY85].

For certain classes of nonlinearities, the function may be approximated by the representation

$$\begin{aligned} \lambda_1 + \ldots + \lambda_k &= 1 \\ a_1\lambda_1 + \ldots + a_k\lambda_k &= x_t \\ b_1\lambda_1 + \ldots + b_k\lambda_k &= f(x_t) \end{aligned}$$

where $f(x_t)$ is the (variable separable) nonlinear function and (b_i, a_i) are the coordinates of the k interpolation points. If any of the two adjacent λ's are constrained to be non negative then the set of variables is called an ordered set (of variables) of type two. If the λ's are constrained to be 0-1 then the corresponding set of variables is called an ordered set (of variables) of type one. Most mathematical programming systems have an extended Branch and Bound procedure where these two problem types can be solved more efficiently [TOMLIN70],

[SCICON78] than by conventional separable programming [BEALE68].

Other well established features include the solution of nonlinear programming problems by the method of approximation and recurrent call to the LP optimiser. This is a means whereby recursive solutions are obtained to a finer linear approximation in the neighbourhood of the first LP optimium solution. The process is continued until convergence or until a satisfactory solution is obtained. Another major feature of current mathematical programming systems is the use of the BASIC procedure. Large problems are broken down into smaller subproblems. These subproblems are solved and their solutions used to obtain a good basic starting point to the overall problem. This approach leads to a considerable saving in time in arriving at an optimal solution. A full review of LP optimisers can be found in [TAMIZ86], while that of special features is described in [ADBELM72].

1.2 Historical development of computer assisted LP modelling

While progress in the computational solution methods has led to the development of powerful and robust optimisers for large scale LP models and some restricted IP models, the ability to describe and communicate models to the optimiser has not progressed as rapidly. Since the product form of the inverse and the revised simplex method require that the data is processed in a column-wise fashion, the early (simplex) optimisers expected the problem matrix to be presented in this way. Due to sparsity in all realistic models, it is convenient to communicate the non zero values only. These can naturally be supplied by indicating the row and column positions and the element value for each non zero entry. For ease of interpreting the variables and restrictions, instead of using row and column numbers, unique names are introduced to indicate rows and

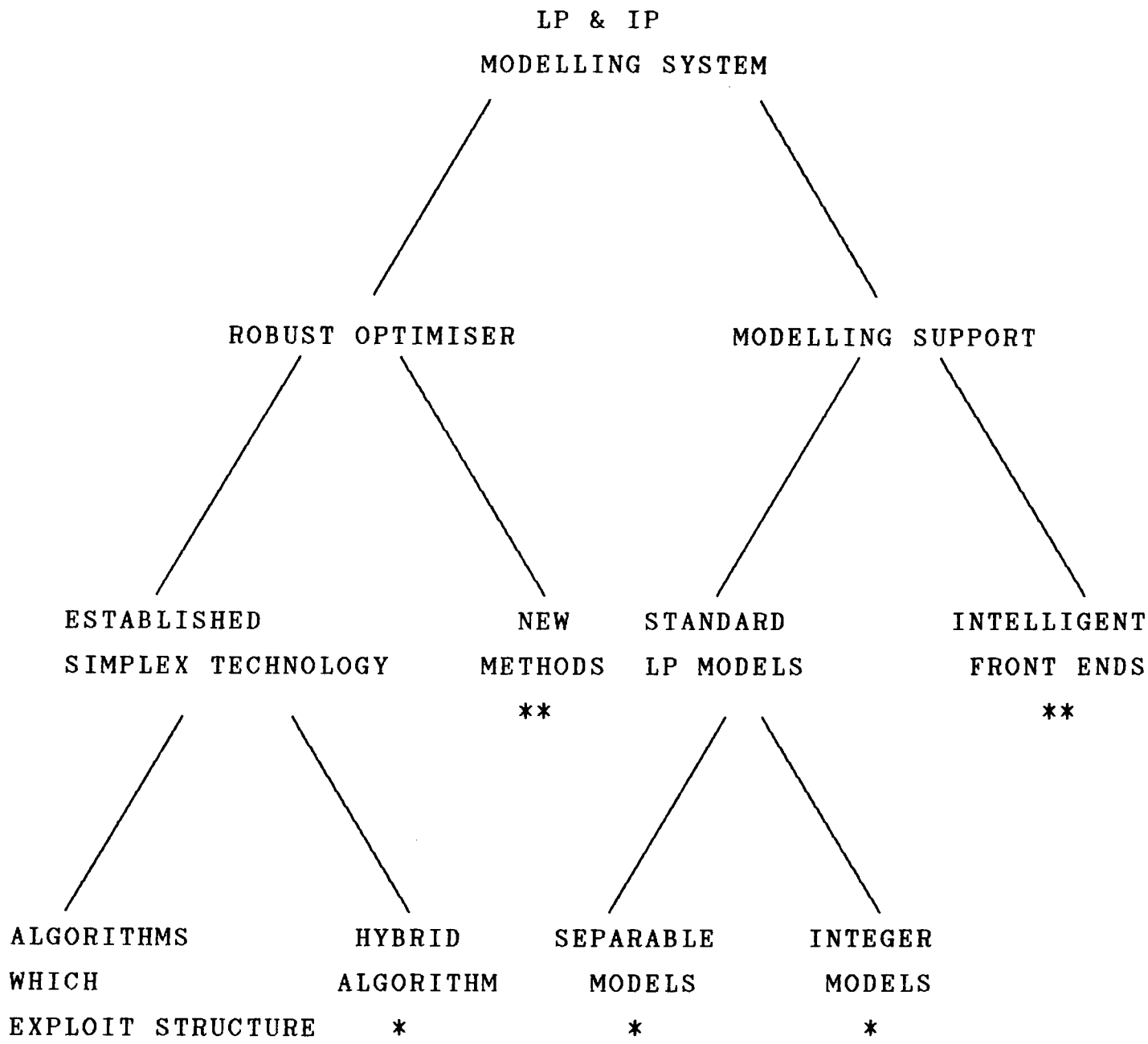
columns. This led to IBM's LP90 [BEALE68] input format which has now been superseded by the current defacto standard MPSX/370 [IBM76].

Creating these fixed format column order files is a tedious task. From around 1958 special purpose computer programs were created with a view to automating this step. In the early days, computer programs were written in either FORTRAN or assembler to generate the problem matrix. Then similar programs were written to read the results and create the desired reports. These programs were model specific and inflexible. Hence any new application required that a new program be written. In order to create and investigate applications more efficiently, more general purpose programs were developed. The next generation of tools were called matrix generators and report writers. These special purpose languages were mostly interpretive and were data driven generalised programs providing support for both the modeller and the optimiser. Thus the modeller was still involved in understanding certain conventions employed in the input specification of the optimiser. This approach gave greater flexibility in model formulation and solution analysis, but required careful and detailed matrix and report specification, using special languages, which possessed only a rudimentary syntax.

1.3 Mathematical programming: the major issues

Two major computational tasks need to be undertaken to investigate an application which involves mathematical programming. To start with there is the requirement of constructing the mathematical formulation and then specifying the problem data to represent the application. Subsequently, it is necessary to solve the proposed (optimisation) problem by suitable optimisation software. To develop software for these two tasks calls for two separate and distinct skills. Considerable research is directed to

each of these two fields in their own right. Display 1.1 gives an overview of the current state of the art.



Display 1.1

Over the last thirty years, there has been a steady development in optimisation methods to solve progressively larger problems efficiently and robustly. This progress is due, as much to the development of sparse matrix manipulation, as to improvements in computer hardware [DARMIT77], [GREENB78], [GIMUSW84], [TAMIZ86]. Techniques such as triangular factors of the basis matrix are used in preference to the product form of the inverse. A major contribution from the field of

computer science came from Kalan [KALAN71] with the introduction of the unique element pool storage strategy. This scheme takes advantage of the fact that the number of distinct non zeros is considerably less than the total number of non zero elements, thus leading to the concept of 'super sparsity'. In recent times some non simplex type methods, Karmarkar [KARMAR84], have proven to be faster in optimisation for some classes of large structured problems.

Although much effort continues to be invested in creating faster optimisers capable of solving larger problems, the biggest burden of mathematical programming is the amount of human time and resource it takes to describe, translate and investigate a model.

1.4 Mathematical programming modelling: the major issues

A modeller can possibly follow four alternative approaches to obtain a computer representation of his LP model. Each of these alternatives calls for varying skills and provides different scope in creating applications. The general skills and specific requirements about model structure and MPSX matrix formats are now described for these approaches.

(i) High level language approach

In this approach, programs are written in a high level language. These programs create the problem matrix in MPSX format.

Examples: FORTRAN, PL1.

General skills: Modelling, computer programming.

Model structure: Problem has to be conceived as a matrix comprising a sequence of columns.

MPSX Format: Knowledge essential.

Model documentation: This is created 'off line' as a pen and paper

exercise. It is usually hard to keep the documentation uptodate with model evolution.

(ii) Matrix Generator, Report Writer approach

In this approach a program is written in a traditional MGRW language to create the problem matrix.

Examples: OMNI [HAVERL76], DATAFORM [KETRON75].

General skills: Modelling, some computer programming.

Model structure: Problem has to be conceived as a matrix comprising a sequence of columns.

MPSX Format: Only naming convention needs to be considered.

Model documentation: It is possible to relate the mathematical model directly to the MG program. Still model documentation in a mathematical form is undertaken as a pen and paper exercise.

(iii) Modelling language approach

In this approach programs are written in a modelling language to create the problem.

Examples: GAMS [BISMEE82], ULP [WITMCC85], MGG/RGG [SCICON75].

General skills: Modelling, only superficial knowledge of computer programming.

Model structure: The models can be presented entirely in an equational form.

MPSX Format: It is inessential to know this format.

Model documentation: The source program reflects the mathematical model fairly closely. Thus model documentation is no longer 'off line' and stays uptodate with model evolution.

(iv) Interactive program generator approach

In this approach an executable program is created after an interactive session with the modeller. This generated program creates the problem matrix.

Examples: CAMPS [LUCMIT85], SIMP [CARMON86].

General skills: Modelling.

Model structure: The models can be presented entirely in an equational form.

MPSX Format: It is inessential to know this format.

Model documentation: Documentation is automatic and is supplied as a special feature including full textual annotation. Model documentation is not 'off line' and stays uptodate with model evolution.

In the present research, four major issues have been identified which are important in any approach towards computer supported modelling. These may be itemised as (a) data (base) storage and manipulation, (b) high level (natural) language documentation, (c) analysis of model and solution, and (d) computer support for reformulation.

(a) Data (base) storage and manipulation

It is now well accepted by analysts who are planning to create applications in business or industry, that the models proposed by them must communicate with existing management information systems. Palmer et al [PALMER84] and Mitra and Darby-Dowman [MITDAR85], show why it is important to have such an integrated approach.

(b) High level (natural) language documentation

A mathematical documentation of the model plays an important role in communication between two analysts but a limited role as a means of communication between a problem owner and a modeller. It is not difficult to interpret the mathematical documentation at a higher level as an English language description of the mathematical problem [EDS86]. Documentation at this level is of great value as a means of communication between the problem owner and the analyst.

(c) Analysis of model and solution

During the development phase of a model, one or more pertinent details are often omitted. While the analyst can infer that some data has been supplied incorrectly, or that some further detail concerning the model is required, it is very difficult to obtain advice on how to analyse the model. Similarly, when a solution is obtained for a large model it is often necessary to carry out a summary analysis of a few relevant decision variables or to investigate different scenarios. These aspects have been extensively investigated by O'Neill [ONEILL78] and Greenberg [GREENB83].

(d) Computer support for reformulation

In many applications reformulation methods have to be introduced to represent nonlinearities or logical restrictions as linear/integer programs. Usually this is achieved by following established but complex procedures. These techniques are, in short, 'tricks of the trade' that the modeller has to learn in order to develop his skills and expertise. It is observed that a natural way of progress in

this area is to support the modelling of such reformulation techniques with computer software facilities.

1.5 Research focus and the structure of the thesis

The purpose of this research has been to increase the speed as well as the productivity of the LP/IP modelling process by addressing some of the issues mentioned above. A computer based LP modelling system called CAMPS (this acronym is derived from Computer Assisted Mathematical Programming System) is implemented as part of the research. The system encourages the analyst to follow a certain modelling strategy which is set out in chapter two and involves a progressive definition of the problem. This forces the modeller to structure his applications in a systematic way while the system participates in trapping model inconsistencies and promoting logically correct definitions.

In common with many of the modelling systems, there are special features which help the modeller with problem formulation. Many of these features can also be found in other current generation modelling systems which are reviewed in chapter three.

The complete system (CAMPS) and some of its major features are described in chapter four. A small example is introduced to illustrate a typical session with the system.

Mathematical documentation of the model can be generated by the system. This documentation can be enhanced by introducing textual annotations at the model input stage. The full documentation can be presented as four components: the conceptual model, data tables, MPSX names and model solution. By integrating the system with a solution analyser (due to

Greenberg [GREENB83]) the use of annotated documentation is extended further. The solution analyser manipulates these textual annotations, held by CAMPS, and provides a discourse with the modeller. This discourse may take the form of advice giving which is very useful when investigating a model. The major aspects of integrating the two systems are discussed in chapter five.

In chapter six the techniques of reformulating nonlinear problems, fuzzy linear programming problems and logical restrictions are presented. As a result of investigating a number of nonlinear problems the system has been extended with special constructs. A blueprint for a system implementation to support reformulation of fuzzy linear programming problems and logical restrictions is also discussed in chapter six.

A summary of the major research results and conclusions is presented in chapter seven. Two main areas of further development are also considered which are seen to be natural ways of enhancing the power of present day LP modelling systems. These are a programmer's interface which helps in creating specialist models rapidly, and artificial intelligence techniques which use a rule base, a knowledge base and a natural language dialogue (as appropriate) to create applications.

CHAPTER 2

STRATEGY AND TACTICS OF LP MODELLING

2.1 Introduction

Formulating linear and integer programming models for industrial (optimisation) problems requires experience and specialist skill. The method of analysing a physical problem is discussed in section 2.2. The logical sequence of steps which lead to a mathematical statement of the model are set out in section 2.3; these concepts are illustrated by an example. Having obtained a mathematical statement it is necessary to prepare the data for suitable processing by a computer based LP system. This aspect is discussed in section 2.4. Further examples are considered in section 2.5 to explain these principles of modelling. The purpose of analysing the components which are used to construct LP models and of considering a range of models is to highlight the major features which need to be introduced into a general modelling support system.

2.2 A logical analysis of the problem

A modeller, when he comes across an industrial (optimisation) problem, does not necessarily find it well described in summary form. It is more than likely he is presented with a description of the problem containing details which may be irrelevant for modelling purposes; further it may also contain a number of gaps. Hence the first task of the modeller is to consider only the modelling requirements and extract the quantative relationships which are germane to that task. Having identified these items he produces a compact statement of the problem which contains only these pertinent details. The examples which are presented in section

3 of this chapter, and the planning model considered later are first described in this summary form.

Model Entities

After identifying the key components of the model his next task is to discover the underlying structure in the model. This amounts to finding a way of categorising the modelling information. The following is an illustrative list of typical categories (entities) that are found in practical problems.

- number of (decentralised) geographical locations
- number of planning periods
- number of different products
- number of grades of people
- number of age groups

This categorisation helps him to decide to what details the quantitative information relating to the problems should be requested and incorporated in the model. It also indicates to what details the answers are to be provided.

Model Variables

Once the categories are defined the model (decision) variables or the unknowns are broadly identified. An analysis of the decision variables may also suggest new categories at this stage. The point to note here is that the model variables are mostly detailed by categories. For the purpose of illustration a number of decision variables taken from different contexts are considered below.

- Production Planning: The quantity X_{pm} of a certain product p manufactured on a machine m.

- Distribution Planning: The quantity X_{prn} of a product p that is shipped from a source r to an outlet n.
- Inventory Scheduling: The quantity X_{pt} of a product p that is kept as closing stock at the end of a period t.
- Project Analysis: Whether one should invest in project p at the beginning of time period t, or not invest in this project Y_{pt} = 1 or 0 may be represented by this zero-one variable Y_{pt}.

Model Constraints

The constraints connect the decision variables and express the physical restrictions of the problem. By and large these are also detailed by categories. A few examples of these are set out below.

- Material Balance Equation

$$XO_t + XP_t - XC_t = D_t \quad , \; t = 1,2,\ldots T.$$

In this equation XO_t represents the opening inventory, XC_t represents the closing inventory, and XP_t the quantity to be produced. They are all decision variables pertaining to the time period t. D_t represents the customer demand for the product and is an input information.

- Capacity Restrictions

$$\sum_{p=1}^{P} X_{pm} \cdot t_{pm} \leqslant A_m \quad , \; m = 1,2,\ldots M.$$

Here p = 1,2,...P indicates the range of products which are manufactured on machines m = 1,2,...M. The rate of production is indicated by t_{pm}, that is, the time taken to produce one unit of product p on machine m. A_m indicates the number of hours that machine m is available. X_{pm} is the production variable and the constraints express the capacity of production for the machine m as limited by the number of hours of its availability.

- Blending Requirement

$$\sum_{c=1}^{C} X_{cp}\, b_{cr} \quad \left\{ \begin{array}{c} < \\ \text{or} \\ = \\ \text{or} \\ > \end{array} \right\} Q_{pr} \quad \begin{array}{l} p=1,\ldots P \\ r=1,\ldots R \end{array}$$

In this case c = 1,2,...C is the number of components used to be blended into p = 1,2,...P products. The components for instance could be different crudes and the products could be different types of gasoline. The index range r = 1,...R indicates the quality requirements. Typical requirements are maximum vapour pressure, minimum volatility index etc. Thus b_{cr}, Q_{pr} are input information pertaining to linear blending rates and quality requirements respectively. X_{cp} is the decision variable indicating fractions (by volume or weight) of the components c that are blended to derive the product p. Thus

$$\sum_{c=1}^{C} X_{cp} = 1 \ , \ p = 1,\ldots P.$$

Thus in the discussion of the model variables and model constraints the subscripts p,m,n,c,r,t etc which have been introduced indicate entities taken from the context of the model. Identifying these entities amounts to setting out the basic structure of the model.

2.3 Derivation of a mathematical statement: an example

It follows from the preliminary analysis presented in the last section that in order to derive a mathematical statement of the model one has to formally define the matrix elements of the constraint relations. In order to do this it is necessary to define the subscripts and their ranges. Note that the matrix elements themselves may be derived out of tabular input information relating to the problem. These matrix elements may be

considered to be model descriptors and are often referred to as "technology coefficients". The model (decision) variables in contrast are output information. Their values are obtained by solving the model. The sequence of steps leading to the derivation of a model thus naturally emerges and is set out below.

Step 1 Define the subscripts (entities) and their ranges (sets and dimensions).

Step 2 Define model variables, model constraints and the matrix coefficients in terms of these subscripts (step 1).

Step 3 Specify the linear relationships in a row-wise fashion which connect the items defined in step 2.

In its simplest and most standard form an LP model can be stated in the following way:

- Subscripts, Ranges:

$i = 1,...m,\ j = 1,...n.$

- Variables, constraints, coefficients:

$x : x_j\ ,\ j = 1,...n\ ,\qquad r : r_i\ ,\ i = 1,...m\ ,$

$c : c_j\ ,\ j = 1,...n\ ,\qquad b : b_i\ ,\ i = 1,...m\ ,$

$A : a_{ij}\ ,\ i = 1,...m\ ,\ j = 1,...n.$

- Linear objective function and constraints:

$$\text{Max} \sum_{j=1}^{n} c_j x_j \ ,$$

$$\text{subject to } r_i: \quad \sum_{j=1}^{n} a_{ij} x_j = b_i \ , \ i = 1,\ldots m,$$

$$x_j \geqslant 0 \ , \ j = 1,\ldots n.$$

However, in real life applications the corresponding models possess more detailed structure than this standard form. As a result of such structure the A-matrix turns out to be highly sparse and b,c can also be sparse. In practice, therefore, formulating a model requires specifying only the non zero coefficients of the A-matrix as used in stating the linear constraint relations.

In deriving the mathematical statement of an LP model and especially the linear constraint relations it is often convenient to prepare a material flow diagram for the problem. This enables the modeller to visualise and set out the balance relations, the capacity restrictions etc. The principles of LP modelling discussed so far are illustrated in the derivation of a production cum distribution model considered here and further models described in section 2.5.

A Production cum Distribution Problem: An Example.

A clothing manufacturer has two factories, Southall (FT1) and Leeds (FT2). In the Southall factory he can manufacture Shirts (P1) and Denim Jeans (P3), whilst in Leeds he can manufacture Shirts (P1), Skirts (P2) and Denim Jeans (P3). The manufacturer ships these products directly to three main dealers in quantities of thousands. The dealers are Young Londoner (DL1), Beaute Paris (DL2) and Wiener Mode Anzug (DL3). The

manufacturer knows his production costs, the transport costs and the monthly production capacity of his factories. The dealers send their requirements for the next month on the first day of each month. All the numerical data relating to the problem are set out in table 2.1. The line diagram 2.1 illustrates the possible relationships between factories, products and dealers.

DEALERS REQUIREMENTS AND PRODUCTION CAPACITY IN UNITS OF THOUSANDS					
Product	Dealer Requirements			Factory Capacity	
	DL1	DL2	DL3	FT1	FT2
P1	50	10	30	35	54
P2	15	15	20	-	60
P3	20	60	30	85	45

PRODUCTION AND TRANSPORT COST IN POUND STERLING PER ITEM						
Factory	Production costs			Transport costs		
	P1	P2	P3	DL1	DL2	DL3
FT1	1.5	-	5.6	0.6	1.2	1.4
FT2	1.8	7.0	6.2	0.7	1.3	1.5

Table 2.1

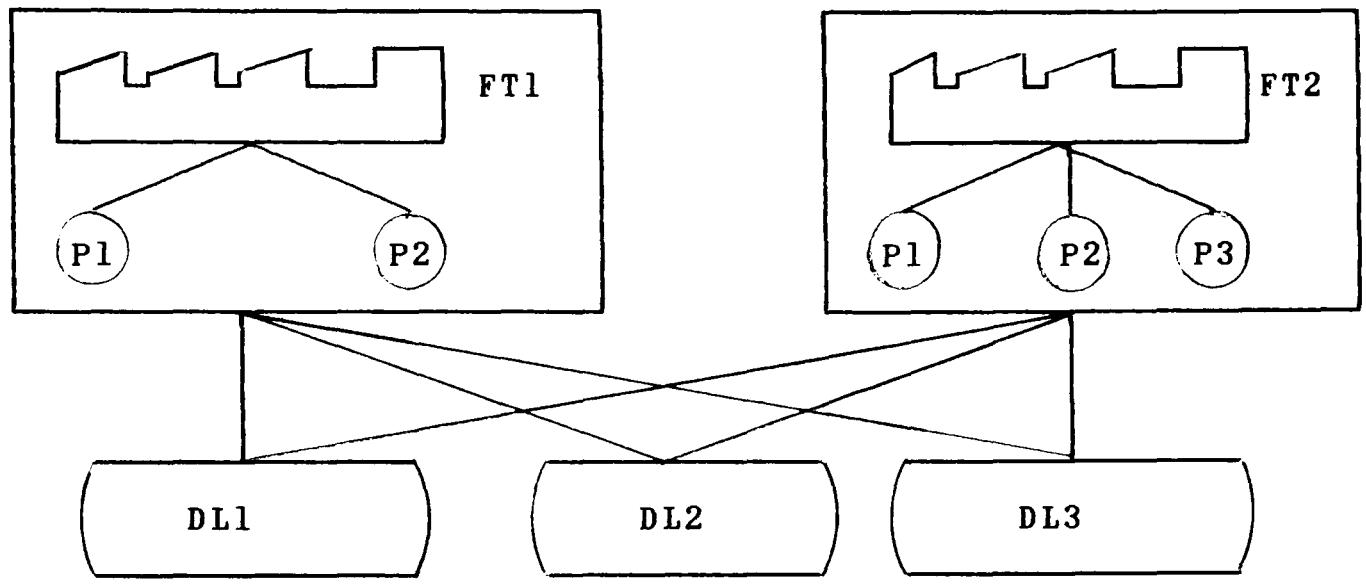


Diagram 2.1

The manufacturer at the beginning of each month, needs to formulate and solve a simple linear programming problem. A mathematical statement of this problem is set out below.

- Subscripts and Dimensions.

i = 1,2 denotes the factories
j = 1,2,3 denotes the products
k = 1,2,3 denotes the dealers.

- Model Variables

X_{ijk} the quantity of product j manufactured in factory i and shipped to dealer k. However, for i = 1 (Southall) the product j = 2 skirts and it's shipment are not defined.

$$\left.\begin{array}{l} \text{That is } i = 1, \quad j = 1,3 \\ \qquad\quad\; i = 2, \quad j = 1,2,3 \end{array}\right\} \; k = 1,2,3$$

- Model Coefficients (Descriptors)

P_{ij} the cost of producing one unit of product j at factory i,

t_{ik} the cost of transporting one unit of each product from factory i to dealer k,

c_{ijk} the derived cost of production as well as transport for given i,j,k which may be expressed as

$$c_{ijk} = p_{ij} + t_{ik}$$

a_{ij} the production capacity of the factory i for the product j,

r_{jk} the requirement of the dealer k for the product j.

- Linear Constraint Relations: A Mathematical Statement.

Minimise

$$\text{Cost} = \sum_{i=1}^{2} \sum_{k=1}^{3} \left[c_{i1k}\, x_{i1k} + c_{i3k}\, x_{i3k} \right] + \sum_{k=1}^{3} c_{22k}\, x_{22k}$$

Subject to the constraints:

capacity of production

$$\sum_{k=1}^{3} x_{ijk} \leqslant a_{ij} \quad , \quad \left.\begin{array}{l} i = 1,\ j = 1,3 \\ i = 2,\ j = 1,2,3 \end{array}\right\} \quad ,$$

and satisfying dealer requirements

$$\left.\begin{array}{l} \displaystyle\sum_{i=1}^{2} x_{ijk} = r_{jk} \quad , \ j = 1,3 \\ \qquad x_{22k} = r_{2k} \quad , \end{array}\right\} \ k = 1,2,3$$

and $x_{ijk} \geqslant 0.$

2.4 LP user formulation of the model

The mathematical statement of the model set out in the last section is concise and convenient for communication and discussion by mathematicians and analysts. However, for the purpose of processing the model by a computer based LP system and deriving numerical solutions, this form is abstract and unsuitable.

Model information is usually presented to an industrial LP system in a compact form and it is appropriate to highlight a few features of LP input at this point.

(i) All applicable LP models display a high degree of sparsity of the constraint matrix.

(ii) Only the non zero coefficients of the matrix are specified as input.

(iii) Instead of a row index and a column index, one uses a row name and a column name to specify a non zero coefficient of the matrix.

(iv) Feature (iii) requires that a suitable name is given for the rows and columns of the matrix.

IBM's MPSX input format is industry's de facto standard for model specification: this is described in [IBM76] and also in the CAMPS manual [LUCMIT85].

To obtain the LP user formulation the following model variable and constraint names are first defined.

- Model Variable name

FT1P1DL1 The amount of product P1 produced in the factory FT1 and shipped to the dealer DL1 etc.

- Model Constraint Names

COSTROW The objective row
FT1P1CAP The capacity constraint corresponding to the product P1 produced in factory FT1,
REQP1DL1 The requirement of the product P1 by the dealer DL1

etc.

The sparse but complete constraint matrix in terms of these row and column names is set out in tableau 2.1. The corresponding MPSX format input data file in line images is set out in display 2.1.

```
                                    ...  ..RMAT LINE IMAGE INPUT
NAME          EXPROD
ROWS
 N  COSTROW
 L  FT1P1CAP
 L  FT1P3CAP
 L  FT2P1CAP
 L  FT2P2CAP
 L  FT2P3CAP
 E  REQP1DL1
 E  REQP2DL1
 E  REQP3DL1
 E  REQP1DL2
 E  REQP2DL2
 E  REQP3DL2
 E  REQP1DL3
 E  REQP2DL3
 E  REQP3DL3
COLUMNS
    FT1P1DL1  FT1P1CAP          1.000000  REQP1DL1          1.000000
    FT1P1DL1  COSTROW           2.100000
    FT1P1DL2  FT1P1CAP          1.000000  REQP1DL2          1.000000
    FT1P1DL2  COSTROW           2.700000
    FT1P1DL3  FT1P1CAP          1.000000  REQP1DL3          1.000000
    FT1P1DL3  COSTROW           2.900000
    FT1P3DL1  FT1P3CAP          1.000000  REQP3DL1          1.000000
    FT1P3DL1  COSTROW           6.200000
    FT1P3DL2  FT1P3CAP          1.000000  REQP3DL2          1.000000
    FT1P3DL2  COSTROW           6.800000
    FT1P3DL3  FT1P3CAP          1.000000  REQP3DL3          1.000000
    FT1P3DL3  COSTROW           7.000000
    FT2P1DL1  FT2P1CAP          1.000000  REQP1DL1          1.000000
    FT2P1DL1  COSTROW           2.500000
    FT2P1DL2  FT2P1CAP          1.000000  REQP1DL2          1.000000
    FT2P1DL2  COSTROW           3.100000
    FT2P1DL3  FT2P1CAP          1.000000  REQP1DL3          1.000000
    FT2P1DL3  COSTROW           3.300000
    FT2P2DL1  FT2P2CAP          1.000000  REQP2DL1          1.000000
    FT2P2DL1  COSTROW           7.700000
    FT2P2DL2  FT2P2CAP          1.000000  REQP2DL2          1.000000
    FT2P2DL2  COSTROW           8.300000
    FT2P2DL3  FT2P2CAP          1.000000  REQP2DL3          1.000000
    FT2P2DL3  COSTROW           8.500000
    FT2P3DL1  FT2P3CAP          1.000000  REQP3DL1          1.000000
    FT2P3DL1  COSTROW           6.900000
    FT2P3DL2  FT2P3CAP          1.000000  REQP3DL2          1.000000
    FT2P3DL2  COSTROW           7.500000
    FT2P3DL3  FT2P3CAP          1.000000  REQP3DL3          1.000000
    FT2P3DL3  COSTROW           7.700000
RHS
    RHSVALUE  FT1P1CAP         36.000000  FT1P3CAP         85.000000
    RHSVALUE  FT2P1CAP         54.000000  FT2P2CAP         60.000000
    RHSVALUE  FT2P3CAP         45.000000  REQP1DL1         50.000000
    RHSVALUE  REQP1DL2         10.000000  REQP1DL3         30.000000
    RHSVALUE  REQP2DL1         15.000000  REQP2DL2         15.000000
    RHSVALUE  REQP2DL3         20.000000  REQP3DL1         20.000000
    RHSVALUE  REQP3DL2         60.000000  REQP3DL3         30.000000
ENDATA
```

Display 2.1

	F T 1	F T 1	F T 1	F T 1	F T 1	F T 1	F T 2	F T 2	F T 2	F T 2	F T 2	F T 2	F T 2	F T 2	F T 2	R E L	R H S
PRODUCTION	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	A	V
	1	1	1	3	3	3	1	1	1	2	2	2	3	3	3	T	A
VARIABLES	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	I	L
	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	O	U
	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	N	E
COST																	
COSTROW	2.1	2.7	2.9	6.2	6.8	7.0	2.5	3.1	3.3	7.7	8.3	8.5	6.9	7.5	7.7	FREE	
FACTCAP																	
FT1P1CAP	1	1	1													LE	35
FT1P3CAP				1	1	1										LE	85
FT2P1CAP							1	1	1							LE	54
FT2P2CAP										1	1	1				LE	60
FT2P3CAP													1	1	1	LE	45
DEALEREQ																	
REQP1DL1	1						1									EQ	50
REQP1DL2		1						1								EQ	10
REQP1DL3			1						1							EQ	30
REQP2DL1										1						EQ	15
REQP2DL2											1					EQ	15
REQP2DL3												1				EQ	20
REQP3DL1				1									1			EQ	20
REQP3DL2					1									1		EQ	60
REQP3DL3						1									1	EQ	30

Tableau 2.1

2.5 Further examples

Blending of Gasoline Products

An oil company in an off shore island maintains a reserve of five basic components, Butane, Light Naptha, Heavy Naptha, Catalytic Naptha and Catalytic Reformate which are blended and replenished on a weekly basis to meet the demands for two grades of gasoline called GAS1 and GAS2. The availability, the linear blending coefficients and the costs for these components are tabulated in table 2.2. The quality requirements and the volume demands for the two gasoline products are set out in table 2.3. The oil company wishes to derive an LP model that must be solved on a weekly basis to find the optimal blending of the components.

Blending Components

Component	Availability thousands of barrels	Research octane number	Vapour pressure	Volatility index	Code name	Cost, cents per gallon
Butane	3.5	120.0	60.0	105	BU	5.2
Light naptha	2.0	84.5	18.0	30	LN	6.4
Heavy naptha	4.0	73.0	4.0	12	HN	8.3
Catalytic naptha	10.5	96.0	6.4	15	CN	10.2
Catalytic reformate	8.0	99.0	2.5	3	CR	11.0

Table 2.2

Gasoline Requirements

Needed volume, thousands of barrels	Minimum research octane number	Maximum vapour pressure	Minimum volatility index	Code name
10.0	95.0	11.0	18	GAS1
6.0	98.0	12.0	20	GAS2

Table 2.3

Diagram 2.2 shows how the two products connect the five components.

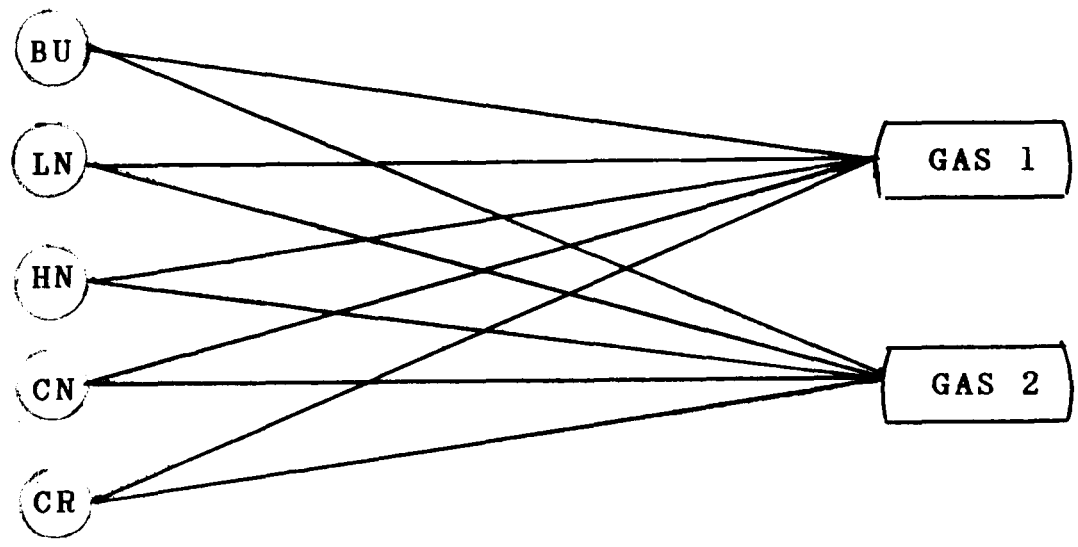


Diagram 2.2

- Subscripts and Dimensions

$i = 1,...5$ denotes the components,
$j = 1,2,3$ denotes the three quality indices: octane number, vapour pressure, volatility index,
$k = 1,2$ denotes the two gasoline products.

- Model Variables

x_{ik} The amount of component i that is blended into the product k.

- Model Coefficients

a_i The amount of component i that is available for blending
b_{ij} the linear blending coefficient for component i and quality index j,
c_i the cost of component i,
r_{kj} the blending quality requirement for the product k against quality index j,
d_k the demand for the gasoline product k.

- Linear Constraint Relations: A Mathematical Statement.

$$\text{Minimise} \quad \sum_{i=1}^{5} \sum_{k=1}^{2} c_i \, x_{ik}$$

subject to

Availability restriction $\sum_{k=1}^{2} x_{ik} \leqslant a_i \;, \; i = 1...5 \;,$

Demand balance

$$\sum_{i=1}^{5} x_{ik} = d_k \;, \; k = 1,2 \;,$$

and
Blending requirements

$$\left.\begin{array}{ll} \sum_{i=1}^{5} x_{ik}\, b_{i1} \geqslant d_k\, r_{k1} & j = 1 \text{ Octane specification} \\ \sum_{i=1}^{5} x_{ik}\, b_{i2} \leqslant d_k\, r_{k2} & j = 2 \text{ Vapour pressure} \\ \sum_{i=1}^{5} x_{ik}\, b_{i3} \geqslant d_k\, r_{k3} & j = 3 \text{ Volatility index} \end{array}\right\} \; k=1,2,$$

and $x_{ik} \geqslant 0 \qquad i = 1...5, \; k = 1,2.$

LP User Formulation

- Model Variable Name

BUGAS1, LNGAS1... CRGAS2	The amount of Butane used to produce GAS1... until amount of Catalytic Reformate used to produce GAS2.

- Model Constraint Name

AVAILBU,... AVAILCR	The restrictions on availability for the five components.
DEMGAS1, DEMGAS2	The demand balance equations for the two products
BLOCTGS1...BLVLTGS2	The six constraints for blending requirements.

The matrix of the constraint relations is now set out in tableau 2.2.

	BUGAS1	LNGAS1	HNGAS1	CNGAS1	CRGAS1	BUGAS2	LNGAS2	HNGAS2	CNGAS2	CRGAS2	TYPE	RHSVAL
COST	5.2	6.4	8.3	10.2	11.0	5.2	6.4	8.3	10.2	11.0	FR	
AVAILABILITY												
AVAILBU	1					1					LE	3.5
AVAILLN		1					1				LE	2.0
AVAILHN			1					1			LE	4.0
AVAILCN				1					1		LE	10.5
AVAILCR					1					1	LE	8.0
DEMANDS												
DEMGAS1	1	1	1	1	1						EQ	10.0
DEMGAS2						1	1	1	1	1	EQ	6.0
BLENDING												
REQUIREMENTS												
BLOCTGS1	120	84.5	73	96	99						GE	950.0
BLVAPGS1	60	18	4	6.4	2.5						LE	110.0
BLVLTGS1	105	30	12	15	3.0						GE	180.0
BLOCTGS2						120	84.5	73	96	99	GE	588.0
BLVAPGS2						60	18	4	6.4	2.5	LE	72.0
BLVLTGS2						105	30	12	15	3.0	GE	120.0

Tableau 2.2

A Multi Time Period Multi Mode Production Problem

A company manufactures three products P1, P2, and P3 (NUTS, BOLTS, and WASHERS) and has at its disposal three machines M1, M2, and M3. The company can undertake normal and overtime production and needs to plan for two time periods, say WINTER and SUMMER. Any product left after the second time period has very little resale value. The necessary information concerning the operation of the company is set out in tables 2.4, 2.5, 2.6.

It is necessary to find an LP formulation that maximises the profit of the company's operation over the two periods.

- Subscripts and Dimensions

Let the four indices i, j, k, l be defined as

i = 1,2	the index for the two time periods, Summer and Winter,
j = 1,2	the index for the two modes of production, Normal, Overtime,
k = 1,2,3	the index for the three product types, P1, P2, P3,
l = 1,2,3	the index for the three machines, M1, M2, M3.

- Model Variables

X_{ijkl}	the quantity that is produced in the category i, j, k, l,
Y_{ik}	the quantity of product k stored in period i,
Z_{ik}	the quantity of product k sold in period i.

- Model Coefficients

The following information relating to the problem are available in the table TABH.

t_{ijkl}	number of hours required to produce one unit of the product type k on the machine l, in the time period i, using Normal or Overtime production j,
a_{ijl}	machine availability in hours for the machine l in period i and mode j.

In the table TABD

P_{ik}	selling price	for the product type k in time period i,
d_{ik}	demand,	
s_k	storage cost for the product type k in one time period,	
h_k	the corresponding storage capacity,	

r_k the final resale value at the end.

In the table TABC,

c_{ijkl} the production cost in the category i, j, k, l.

- Linear Constraint Relation

The profit function of the problem may be expressed as

$$\text{Profit} = \sum_{i=1}^{2} \sum_{j=1}^{2} \sum_{k=1}^{3} \sum_{l=1}^{3} (p_{ik} - c_{ijkl}) x_{ijkl}$$

$$- \sum_{k=1}^{3} s_k \, y_{1k} + \sum_{k=1}^{3} (r_k - p_{2k}) y_{2k}$$

In an optimal plan Profit must be maximised subject to the constraints

(i) machine availability

$$\sum_{k=1}^{3} t_{ijkl} \cdot x_{ijkl} \leqslant a_{ijl} \text{ , for all i,j,l;}$$

(ii) stock balance in the two periods,

$$\sum_{j=1}^{2} \sum_{l=1}^{3} x_{1jkl} - y_{1k} - z_{1k} = 0 \text{ for period 1, and all k}$$

and

$$\sum_{j=1}^{2} \sum_{l=1}^{3} x_{2jkl} + y_{1k} - y_{2k} - z_{2k} = 0 \text{ for period 2, and all k}$$

(iii) minimum demand to be satisfied

$$z_{ik} \geqslant d_{ik} \text{ , for all i , and k ;}$$

(iv) upper bound on storage,

$$y_{1k} \leqslant h_k \text{ for all k ;}$$

(v) non negativity of the variables,

$y_{ik} \geqslant 0$ for all i,k, and $x_{ijkl} \geqslant 0$, for all i,j,k,l.

LP User Formulation

- Model Variable Name

Production:
T1NP1M1...T2OP3M3 The production variables $x_{1111} \cdots x_{2233}$,

Storage:
T1P1STR... The storage variables y_{11} etc.,

Amount meeting demand:

T1P1D... The quantities that are allocated to satisfy demand z_{11} etc.,

TABLE OF MACHINE HOURS (TABH)

	SUMMER PERIOD (H1)								WINTER PERIOD (H2)							
	Normal (N) Working hours			(O) Overtime			Total hours Available (AV)		Normal (N) Working hours			(O) Overtime			Total hours Available (A	
	P1	P2	P3	P1	P2	P3	Normal W-Hrs	Over time	P1	P2	P3	P1	P2	P3	Normal W-Hrs	Ove tim
MACHINE 1 (M1)	4	5	6	3	4	5	100	80	5	6	7	4	5	5	110	90
MACHINE 2 (M2)	7	6	6	6	5	5	100	90	8	7	7	7	6	6	110	100
MACHINE 3 (M3)	3	-	-	2	-	-	40	30	4	-	-	3	-	-	50	40

P1 = NUTS P2 = BOLTS P3 = WASHERS

TABLE 2.4

TABLE OF PRODUCTION COSTS (TABC)

	SUMMER PERIOD						WINTER PERIOD					
	Normal Working hours			Overtime			Normal Working hours			Overtime		
	P1	P2	P3	P1	P2	P3	P1	P2	P3	P1	P2	P3
MACHINE 1	2	3	4	3	4	5	3	4	5	4	5	6
MACHINE 2	4	3	2	5	4	3	5	4	3	6	5	4
MACHINE 3	1	–	–	2	–	–	2	–	–	3	–	–

P1 = NUTS P2 = BOLTS P3 = WASHERS

TABLE 2.5

TABLE OF ADDITIONAL DATA (TABD)

		SUMMER PERIOD			WINTER PERIOD		
		NUTS	BOLTS	WASHERS	NUTS	BOLTS	WASHERS
SALE PRICE		10	10	9	11	11	10
MINIMUM DEMAND		25	30	30	30	25	25
STORAGE DATA	CAPACITY	20	20				
	COST	1	1	1			
	RESALE VALUE				2	1	1

TABLE 2.6

- Model Constraint Name

PROFIT	Objective row.
T1M1AN	Availability of machine 1, time period 1 and normal production,
T1P1ST	Stock balance equation time period 1 product 1.

The other three constraints are satisfied by upper bound and lower bound restrictions. The right hand side column is called RHS and the bound is called LIM and the full model is set out in tableau 2.3.

VARIABLES→	T1 NP1	T1 NP1	T1 NP1	T1 NP2	T1 NP2	T1 NP3	T1 NP3	T1 OP1	T1 OP1	T1 OP1	T1 OP2	T1 OP2	T1 OP3	T1 OP3	T1 1PS	T1 2PS	T1 3PS	T2 NP1	T2 NP1	T2 NP1	T2 NP2	T2 NP2	T2 NP2	T2 NP3	T2 NP3	T2 OP1	T2 OP1	T2 OP2	T2 OP2	T2 OP3	T2 OP3	T2 1PS	T2 2PS	T2 3PS	T1 1PD	T1 1PD	T1 1PD	T1 2PD	T1 2PD	T2 2PD		RHS
CONSTRAINTS ↓	M1	M2	M3	M1	M2	M1	M2	M1	M2	M3	M1	M2	M1	M2	TR	TR	TR	M1	M2	M3	M1	M2	M3	M1	M2	M1	M2	M3	M1	M2	M3	TR	TR	TR	D	D	D	D	D	D		
PROFIT	8	6	9	7	7	5	7	7	5	8	6	6	4	6	-1	-1	-1	7	5	8	6	6	4	6	6	4	7	5	5	3	5	-9	-9	-9							N	
T1M1AN	4			5		6																																			LE	100
T1M2AN		7			6		6																																		LE	100
T1M3AN			3																																						LE	40
T1M1AO								3			4		5																												LE	80
T1M2AO									6			5		5																												LE	90
T1M3AO										2																															LE	30
T2M1AN																		5			6		7																		LE	110
T2M2AN																			8			7		7																	LE	110
T2M3AN																				4																					LE	50
T2M1AO																							4		5		5														LE	90
T2M2AO																								7		6		6													LE	100
T2M3AO																									3																LE	50
T1P1ST	1	1	1					1	1	1					-1																								-1			EQ	
T1P2ST				1	1						1	1				-1																							-1			EQ	
T1P3ST						1	1						1	1			-1																							-1			EQ	
T2P1ST															1			1	1	1				1	1	1			-1													EQ	
T2P2ST																1					1	1					1	1			-1											EQ	
T2P3ST																	1						1	1					1	1		-1										EQ	
BOUND																																											
LIM																																											
UP															20	20																												
LO																																							25	30	30	25	30	25		

Tableau 2.3

CHAPTER 3

CURRENT APPROACHES TO COMPUTER ASSISTED MATHEMATICAL PROGRAMMING

3.1 Introduction

Linear and integer programming have a diverse range of applications, and since the late nineteen sixties a number of alternate computer based systems have been created to formulate models and to analyse their solutions. In as much as analysts still like to write high level application programs, the method of generating LP matrices using high level languages such as FORTRAN, PL1, etc., remains a popular technique. The scope of these systems is limited and these systems are not considered any further.

Fourer [FOURER83] in his widely quoted review paper has attempted to classify modelling systems as matrix generators and general purpose modelling languages. A careful analysis of these systems, their implementation and run time characteristics shows that the boundary between these two approaches is rather blurred. For all practical purposes in both types of systems the matrix layout and specification provides the common theme. Thus all the systems provide suitable language constructs to specify the main body of the constraint matrix, right hand side values, bounds or variables and the relationships for each constraint. The table manipulation, data manipulation and conditional transfer of control and other language features are available in varying degrees, depending upon when and how these systems were implemented.

By and large systems which are general enough to create a broad range of linear programming applications are categorised into five main classes. The first class of such sytems are called teaching or introductory systems. Their purpose is to introduce undergraduate or postgraduate students or new industry recruits to the methods of LP formulation. These are discussed in section 3.2. A number of earlier systems which are mentioned in chapter one, have survived the test of time and are still in use in many key industries such as the chemical and energy industries. These systems tend to model the problem using a column-wise specification and are called activity based methods. This column-wise specification of the model sometimes makes it easier to conceive the model, and hence some new systems also employ this strategy for model description. Many of these earlier systems are described by Fourer [FOURER83] as matrix generators and these systems are reviewed in section 3.3. In more recent times, there have been developments in modelling languages. Many of these offer the ability to describe a model in the equation form. From a mathematical point of view, and for many modellers, this seems a more natural way of describing the model. Systems which support this type of model creation employing row-wise generators are described in section 3.4. Substantial system development effort has gone into creating LP based corporate modelling systems. Two well known and perhaps most successful examples of these, PLANETS [EDS86] as used by General Motors, and PLATOFORM [PALMER84] as used by Exxon, are discussed in section 3.5. Some recent developments in block structured systems and generic modelling tools are also included in this section. Finally, modelling systems which are influenced by artificial intelligence ideas are briefly described in section 3.6.

3.2 Introductory and teaching systems

These systems are aimed at introducing linear programming and encouraging newcomers to learn the art of modelling decision problems. In most cases the users are expected to possess a limited knowledge of the computer and how to program it. These teaching systems are simple to use, and help the beginner to describe, and investigate, elementary problems such as food mix, transhipment and so on. The software is usually supplemented by good quality courseware such as text books with illustrative examples. In addition to teaching modelling, these are also used to teach advanced algorithmic methods such as parametric simplex steps and how to interpret results. Whereas the systems used are excellent in presenting and editting small problems, they cannot be extended to larger and more realistic industrial problems which have, in general, a hundred or more rows and columns.

The one common theme throughout these systems is that they are easy to use, although they do adopt alternative ways of presenting the model. For instance in the LINDO [SCHRAG81A], [SCHRAG81B] system the model is presented one equation at a time, as if it is directly transcribed from the presentation seen in the text book. MICROSOLVE [JENSEN86], uses menus and screenforms in order to present the model. What's Best [HOLDAY86], is a typical spreadsheet based method which uses LOTUS123 [LOTUS84], to create the linear programming interface. Thus a person who knows how to complete the spreadsheet cells does not need to learn anything new to represent the LP matrix other than the linear equation form.

The optimisers which go with these systems often solve much larger models than can be realistically specified using these approaches. In

such cases the user is advised to write special programs to generate the matrix. This calls for programming skill and limits the scope of applying these systems to larger models.

3.3 Activity based modelling systems

A column-wise description of the linear programming model is naturally suited for input to the revised simplex algorithm. Thus the early systems were developed along these lines. These include DATAMAT [MIT75], GAMMA3 [SPERRY78], MaGen [HAVERL77], OMNI [HAVERL76], IBM MGRW [IBM77] and APEX-II MRG [CONTRO74]. Over the years these systems have been improved by incorporating industrial experience. Their implementations have been invariably extended to deal with large models and most of the obvious bugs have been removed. These systems are hence reliable and very attractive from that point of view to serious industrial users.

For these systems, clausal forms to specify columns are difficult to comprehend leading to poor model documentation. Thus it is not easy to communicate the model in the source form. These two points can easily be seen in the example set out in Appendix one, showing input specification of a model using the OMNI system. These column-wise systems also lead to unnecessary amounts of code; for instance if there are three sets of variables in a model, where a particular row is undefined, then this requires the 'if clause' to be repeated three times to define this exception. For multi time period problems the modeller is required to understand a further set of constructs to represent the matrix.

3.4 Modelling systems employing row-wise specification

Modelling systems which employ equation forms or row-wise specification of the LP problem are distinguished in the following way. These systems were designed later than the column-wise systems described in the last section. Thus they profit from the later developments in special purpose application languages and incorporate many powerful language constructs. Some of these systems were developed as compilers with associated executors and a run time support library, and have the advantage of efficiency in execution with alternative data sets. Thus the same executable program representing a model can be run with different sets of table data for different model sizes.

Another important design consideration for these systems is that a modeller finds it easier to conceive an LP problem in the equation form. The designers of these systems also claimed that the source programs (which specify the model in the equation form) serve as an adequate documentation which may be used to communicate between analysts. A number of these systems such as UIMP [MITELL82] [UNICOM77], DATAFORM [KETRON75] and MGRW [IBM77] additionally incorporate column-wise generation capability. This is because some models, or often parts of models, are best presented in an activity basis. For example it is always clear to present the right hand side vector in a column form. The various points discussed so far are best illustrated by the full example set out in Appendix one and also by considering a few language features of the systems which are discussed in this section.

The logical operator '$' introduced by GAMS [BISMEE82] represents 'such that' and is used to manipulate tables. the power of this operator is illustrated by the following typical statement which sums over the set D

```
YR(R) = SUM ( D $ RD(R,D), YD(R,D));
```

all values YD(R,D) when RD(R,D) is defined. Consider a manufacturing problem where the sets P, I and M denote processess, plants and machines respectively and let the parameter K(M,I) denote the number of units of available capacity of machine M in plant I. Also let the parameter B(M,P) describe the required number of units of capacity of machine M per unit level of process P. Consider the table PPOS(P,I) with parameters having zero one values and defined by the statement

```
PPOSS(P,I) = SUM ( M $ ( K(M,I)EQ 0), B(M,P)NE 0)EQ 0
```

In this statement the expression B(M,P)NE 0 takes the value one if B(M,P) > 0, otherwise it takes the value zero (ie the machine M is dependant upon process P). This is then summed over all machines such that K(M,I) = 0, that is, all machines not at plant I. If the resulting sum equals zero then PPOSS(P,I) takes the value one and thus the process is independant of unavailable machines and is taken into consideration. Otherwise the process is dependant upon at least one unavailable machine and is not considered. The purpose of creating such a table is that in one row generator statement all the corresponding constraints may be specified/controlled by the zero one entries in this table.

LMC [MEFEAV77] is another row-wise modelling system which also has interactive capability. In LMC as in LINDO [SCHRAG81A] it is possible to specify input in an equation (textual) form. Additionally it is also possible to create large scale matrices, matrix pictures or display an equation. It uses an English-like discourse language to manipulate data

tables, but this is not a practical proposition to deal with any reasonably sized data table.

Sets of entities and constructs to manipulate these sets play an important role in all these systems. GAMS [BISMEE82] and LPMODEL [KARIRO80] allow three dimensional sets but in practice they are combined and mapped into one extended set by short hand notation. An entity in a set can be referenced either by a numbered element or as an alphanumeric entity name. GAMS goes one step further, whereby sets can be extended at the time of table data entry if this proves to be convenient.

ULP [WITMCC85] contains an extensive collection of reserved words which can be profitably used to state compactly a range of constraints. This is illustrated below with the language constructs of LPMODEL and ULP. Consider, for instance, the material balance relations taken over three time periods as specified in LPMODEL.

```
MATERIALS.PERIOD_1?≤INITIAL_STOCK.MATERIALS

MATERIALS.PERIOD_2?=MATERIALS.PERIOD_1? -
     SUM[PRODUCTS:COMPOSITION.MATERIALS.PRODUCTS ×
                  PRODUCTS.PERIOD_1?]

MATERIALS.PERIOD_3?=MATERIALS.PERIOD_2? -
     SUM[PRODUCTS:COMPOSITION.MATERIALS.PRODUCTS ×
                  PRODUCTS.PERIOD_2?]
```

The corresponding formulation in ULP, using the reserved word NEXT reduces to

```
CONSTRAIN(MATERIALS,NEXT(PERIODS):
        S(MATERIALS,NEXT(PERIODS)) - S(MATERIALS,PERIODS)
        + COMPOSITION(MATERIALS,PRODUCTS) * X(PRODUCTS,PERIODS) =0)

BOUND (S(MATERIAL,'PERIOD 1') ⩽ INITIAL STOCK(MATERIALS))
```

The reserved word (NEXT in this example) reduces the source statement and also enhances model clarity. Complex constraints can be represented using words such as NETWORK, as these take advantage of well known model structures.

Other features of modelling systems include looping and transfer of control. More recently, MAGIC [DAYWIL86] has introduced FORTRAN like constructs which also include FOR and END loop statements. Currently there are many new row-wise systems under development such as EXPRESS LP [DASH86] and [FOURER86] whose modelling concepts follow the ideas set out in this section.

More recently there has been a move towards producing smart interactive editors for existing modelling systems. These usually generate a modelling language. PLATOFORM [PALMER84] and PAM [WELCH86] are two good examples of systems generating statements in an existing modelling system, DATAFORM. A discussion of PLATOFORM, which is a corporate system, is postponed to the next section. PAM, however, is a more general tool and adds to the productivity of creating applications using DATAFORM.

3.5 Generic modelling tools

Many large corporations are the most committed users of management science based planning and decision making tools. Energy industries

such as oil and gas companies and large multi national corporations such as General Motors and General Electric are typical examples of these. The planning problems of these organisations generally fall into broad classes of long range planning (5 year time horizon) and operational planning on a weekly or monthly time frame. Many of these organisations have developed their own generic (mathematical programming) model generation tools to deal with a range of business problems. These tools are not only used for model generation but also to carry out scenario analysis and management reports or financial requirements, resource utilisation and so on.

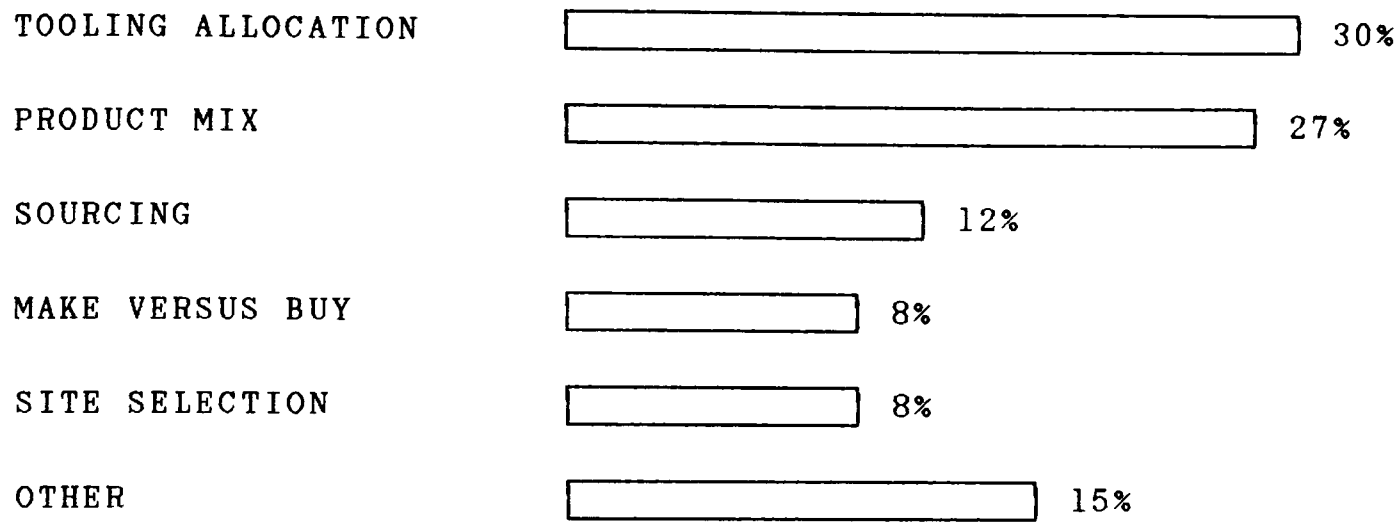
Geoffrion in his structured modelling, [GEOFF85] and [GEOFF86] has tried to develop the framework of a unified system which is designed to aid: (i)management communication, (ii)mathematical representation and (iii)computer execution. Within this framework management science models such as mathematical programming, decision trees, graph problems, markov chains, and queuing problems can be all represented. The main aim of his work is to improve the present state whereby modelling is a low productivity process with poor managerial acceptance. He reports three implementations of his work which are LEXICON, IIS, and UCLA. However, the use of any of these in a real corporate environment is not reported by him.

PLATOFORM, as reported by Palmer et al, is perhaps the earliest example of the use of mathematical programming modelling as a model generation tool for corporate planning models. Within EXXON, PLATOFORM is used extensively to generate a range of planning problems (long range, strategic and operational). Often submodels germane to a particular country's operation are extracted and solved to investigate a specific decision problem. The PLATOFORM system actually generates DATAFORM

modelling statements which create these models and their corresponding reports. The system, as currently implemented, uses a friendly menu driven front end as well as making use of corporate information held within the DATAFORM database.

The management science group of General Motors have developed PLANETS [EDS86], which is an acronym for Production Location Analysis NETwork System. PLANETS was originally implemented in 1974 and has evolved into a flexible framework for scenario description and analysis. The system is designed by individuals, with no prior computer or mathematical programming background, to evaluate complex business problems. It is a tool for generating mathematical models, facilitated by the conversational definition of a variety of business problems in a structured manner, using standard business terminology with a comprehensive network of computer programs. This mathematical representation of the problem is then automatically solved by using commercially available optimisation tools. PLANETS interprets the resulting mathematical programming output and then provides both financial and operational results in an easily understandable business format. A range of business problems such as marketing, manufacturing, capital costs, purchasing and distribution can be modelled separately and combined as appropriate. The system also allows specification and investigation of multiple objectives. Since actual problem formulation and data input are facilitated by PLANETS through the use of standard 'building block' terminology, PLANETS has been referred to as an open-ended scenario and model building 'language' for business planners. It is worth reporting the statistics of different planning models which have been studied using PLANETS. The histogram of these figures is set out below.

BREAKDOWN OF PLANETS STUDIES BY TYPE



Although PLANETS is a generic tool by which business planning problems can be specified and investigated, it is not sufficiently general whereby other decision problems such as crew scheduling, paper trim loss, etc can be modelled using the system. This contrasts with modelling systems such as DATAFORM and UIMP which are more of an analyst's tool as opposed to a corporate planner's tool and allow such problems to be investigated.

3.6 Artificial intelligence aids

Artificial intelligence and prototyping aids are used increasingly to create complex application models. Currently many researchers are working towards the creation of 'intelligent mathematical programming systems'. Reasoning mechanisms may be introduced into these to enable them to learn to build well formulated models from incomplete specifications with a discourse that is 'natural' for the analyst. This goal can be partitioned into four subgoals that reflect the central strategy of building an intelligent system. These four sub goals are set out below: (1) development of the structural specification of a model, (2) development of tools for assessing model validity and quality, (3) incorporating learning

mechanisms, (4) development of tools for the interactive analysis of the model solution. It is also necessary to undertake analysis and integration of submodels, automatic generation of queries to an external database and infeasibility and unboundedness analysis for general LP models.

The use of artificial intelligence in mathematical programming modelling systems dates back to a system created by Shen and Krulee [SHEKRU73] in 1973. Simple English statements are supplied by the user, and from these statements a mathematical model is created. The system processes the sentences and produces property lists for each set (set names are recognised via the dictionary -lexicon- look up). Then, by analysing the property lists and basic sentences, variables of the model can be identified resulting in a variable requirement table. Finally the problem is fully constructed in a compact linear algebraic form. The following example illustrates the process. Consider the following dialogue with the system.

P1 COSTS MY COMPANY $1.5 AT F1 AND $1.8 AT F2.*

P2 COSTS $7.0 AT SOUTHALL F2.*

THE COSTS OF PRODUCING P3 IS $5.6 AT F1 AND $6.2 AT F2.*

THE TRANSPORT COSTS FROM F1 TO D1, D2, D3 ARE $0.6, $1.2, AND $1.4 RESPECTIVELY.*

THE TRANSPORT COSTS FOR D1, D2, D3 ARE $0.7, $1.3, AND $1.5 FROM F2.*

THE DEALER D1 REQUIRES 50 UNITS OF P1, 15 UNITS OF P2, AND 20 UNITS OF P3.*

WHILE DEALER D2 REQUIRES 10 UNITS OF P1, 15 UNITS OF P2, AND 60 UNITS OF P3.*

THE CAPACITIES OF F1 ARE 36 UNITS FOR P1, 0 UNITS FOR P2, AND 85

UNITS FOR P3.*

THE AVAILABLE CAPACITIES AT F2 FOR P1, P2, P3, ARE 54 UNITS, 60 UNITS, AND 40 UNITS.*

DEALER D3 REQUIRES 30 UNITS OF P1, 20 UNITS OF P2, AND 30 UNITS OF P3.*

DETERMINE THE QUANTITIES OF P1, P2, P3 TO BE PRODUCED AT F1 AND F2.*

The resulting model is stated in the algebraic form as set out below.

THE PROBLEM IN FORMULA FORM .....

MINIMISE + 2.2 V1 + 2.7 V2 + 2.9 V3 + 2.5 V4 + 3.1 V5 +3.3 V6 +
7.7 V7 + 8.3 V8 + 8.5 V9 + 6.2 V10 + 6.8 V11 + 7.0 V12
+ 6.9 V13 + 7.5 V14 + 7.7 V15

SUBJECT TO 1 V1 + 1 V2 + 1 V3 < = 36
1 V10 + 1 V11 + 1 V12 < = 85
1 V4 + 1 V5 + 1 V6 < = 54
1 V7 + 1 V8 + 1 V9 < = 60
1 V13 + 1 V14 + 1 V15 < = 40
1 V1 + 1 V4 = 50
1 V7 = 15
1 V10 + 1 V13 = 20
1 V2 + 1 V5 = 10
1 V8 = 15
1 V11 + 1 V14 = 60
1 V3 + 1 V6 = 30
1 V9 = 20
1 V12 + 1 V15 = 30

V1, V2,V3, V4, V5,V6, V7, V8,V9,V10,V11,V12, V13,V14, V15 > = 0

LPFORM [MURSTO85] is a currently proposed system purporting to employ artificial intelligence techniques. It uses an LP generator [STOHR85], IBM's MPSX system for solving linear and integer mathematical programs [IBM76], IBM's SQL database management system (DBMS) [ASTCHA75] and ANALYZE [GREENB83], a solution analyser. LPFORM is probably the first mathematical programming system implemented in PROLOG. The knowledge in LPFORM consists of a number of rules relevant to the formulation of LP problems. This knowledge is not specific to any given application. Specific application knowledge and data values for the coefficients of the LP tableau are stored in the DBMS. The system is at an experimental stage and as yet no user interface has been designed. An illustration of a transportation example provides an insight into some of the rules contained in the knowledge base. Consider

Minimise $$\sum_i \sum_j c_{ij} x_{ij}$$

Subject to $$\sum_j x_{ij} \leqslant s_i$$

$$\sum_i x_{ij} \geqslant d_j$$

then the following gives the internal representation of the data schema and problem definition after interaction with the user.

Data schema

a. TRANS-COSTS (Vendor, Warehouse, C, $ per unit)

b. SUPPLY (Vendor, S, units)

c. DEMAND (Warehouse, D, units)

Problem Definition Statements

a. CREATE-BLOCKS (Trans-problem, [Vendors, Warehouses])

b. LINK-BLOCKS (ALL, [Vendors, Warehouses], X)

c. CREATE-BLOCKS (Vendors, Vendor = [V1..V3])

d. CREATE-BLOCKS (Warehouses, Warehouses = [W1,W2])

e. MINIMISE (Trans-costs)

Firstly, a, b, c define the tables c_{ij}, s_i, d_j, and the units field is used to check that the data for the problem is expressed in compatible units. In the problem definition, the first statement, a, defines the problem name and major blocks, Vendors, Warehouses. The next statement defines the variable x_{ij}. Statements c and d result in constraint definitions. Since statement c is by vendors, the system can infer that the right hand side value is SUPPLY and similarly the demand constraint is created.

CHAPTER 4

COMPUTER ASSISTED MATHEMATICAL PROGRAMMING (MODELLING) SYSTEM: CAMPS

4.1 Introduction

In this chapter a new mathematical programming modelling system called CAMPS is described. It is an interactive system and comprises a set of integrated 'program generation' and data management tools which are controlled by a series of menus and screenforms. The design objectives are broad: thus the system encourages non expert LP users to come to grips with the task of conceptualising and describing LP models whereas the expert LP user is also supported in his requirements to construct large and complex models. The contents of this chapter are organised as follows. Section 2 describes the salient and novel features of CAMPS and an example of model construction using CAMPS is illustrated in Section 3. The method of automated reformulation of separable and 0-1 integer programming is considered in Section 4. For illustrative purposes the problem of section 3 is reformulated using ULP [WITMCC85] and OMNI [HAVERL76] in the appendix and contrasts the CAMPS approach with these well known systems.

4.2 Salient and novel features of CAMPS

Computer Assisted Mathematical Programming (Modelling) System (CAMPS) is an interactive system designed to aid model formulation, matrix generation and model management. The main menu set out in display 4.1 and the information flow diagram display 4.2 together provide an outline of the structure and the major functions of the system. A full user

specification of the system is given in [LUCMIT85].

```
                        . . . C A M P S . . .

  USER:                                                    DATE:
  MODEL:                                                   TIME:
                                SEC:

 ..........................................................

                              1.INPUT
                              2.GENERATE
                              3.OPTIMISE
                              4.REPORT
                              5.UTILITIES
                              6.LOGOUT

                                        TYPE NUMBER<<    >>:
```

Display 4.1.

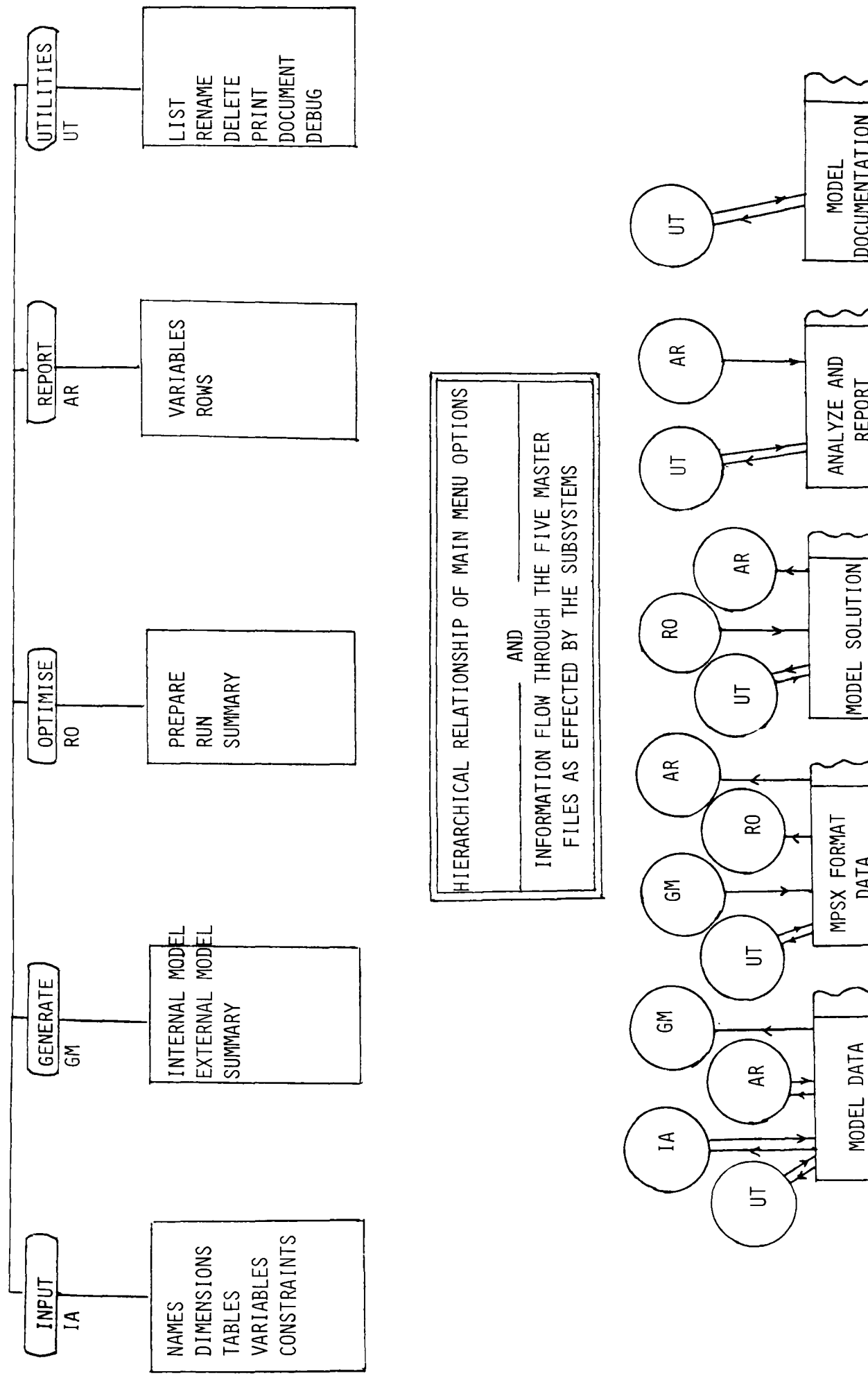
The INPUT (and AMEND) option is used to construct and/or update all aspects of a model created entirely within CAMPS. Display 4.3 illustrates the options under this subsystem and reflects the modelling

```
                        . . . C A M P S . . .

  USER:                                                    DATE:
  MODEL:                                                   TIME:
                                SEC:

 ..........................................................

                              1.NAMES
                              2.DIMENSIONS
                              3.TABLES
                              4.VARIABLES
                              5.CONSTRAINTS
                              6.RETURN

                                        TYPE NUMBER<<    >>:
```

Display 4.3.

methodolgy set out in section 3 of chapter 2.

The subscripts of the model correspond to 'basic entities' which are elements of 'sets' and in actual models these 'sets' could represent geographical regions, materials or time periods. This progressive approach


INPUT
IA
NAMES
DIMENSIONS
TABLES
VARIABLES
CONSTRAINTS
GENERATE
GM
INTERNAL MODEL
EXTERNAL MODEL
SUMMARY
OPTIMISE
RO
PREPARE
RUN
SUMMARY
REPORT
AR
VARIABLES
ROWS
UTILITIES
UT
LIST
RENAME
DELETE
PRINT
DOCUMENT
DEBUG
HIERARCHICAL RELATIONSHIP OF MAIN MENU OPTIONS
AND
INFORMATION FLOW THROUGH THE FIVE MASTER FILES AS EFFECTED BY THE SUBSYSTEMS
IA
GM
UT
AR
MODEL DATA
UT
AR
RO
MPSX FORMAT DATA
UT
AR
ANALYZE AND REPORT
UT
MODEL DOCUMENTATION
UT
AR
RO
MODEL SOLUTION

Display 4.2

to model definition allows avoidance of a procedural language by replacing it with an option driven program generator approach. The syntax of commands are captured in their context and thus mistakes introduced by erroneous keystrokes are kept to a minimum. This is because predefined indices, sets and names are prompted at the appropriate fields of the screenforms. For instance, at the time of defining variables and tables, currently defined sets are displayed. At the time of entering the linear forms, the operators (+,-,*) are prompted and a linear term is forced to comply with the dimensions of the summation indices and the row indices. This is discussed further in the example given in section 3.

The first four options of the main menu are designed to facilitate construction and investigation of a model, whereas the fifth, the UTILITIES option, provides model management support. In CAMPS the usual model management functions such as DELETE, RENAME, LIST and PRINT are augmented by a further option called DOCUMENT. Tabular displays of the input data, variable(MPSX) and row(MPSX) names, and tabulated results are essential aspects of documentation as supplied by all known systems. In addition to these a mathematical formulation of the model is also provided by CAMPS. This mathematical statement can be enhanced by textual annotations specific for a given application. These explanatory texts are introduced at the input stage.

The REPORT subsystem allows information relating to the rows, columns and reduced costs to be examined. The analysis module within REPORT is now designed to interface with the interactive model and solution analysis system ANALYZE by Greenberg [GREENB83]. For each 'basic entity' a textual annotation may be supplied and a unique two character stub is extracted out of this text. This stub is used to create the

'syntax file' of ANALYZE. Thus the facilities of the ANALYZE which provide tools for identifying structural infeasibility and the discourse model to support explanatory dialogue can be reached in this way [GRLUMI86]. The OPTIMISE option uses the FORTLP system [TAMIYA85]. For all practical purposes this is treated as a black box, although a few algorithm control parameters can be set under this option.

LP/IP models are created in MPSX format under the GENERATE subsystem. Within the GENERATE subsystem externally created models are also accepted but REPORT and DOCUMENT options cannot be used in this case. Whereas CAMPS itself is designed for high level interaction in the modeller's form, at the GENERATE subsytem level a programmer's interface for model generation is also available. Thus it is possible to create MPSX models using data tables and model descriptions not held within CAMPS. In this approach the system held subroutine library for model generation is used. This approach is somewhat similar to the ideas put forward by Forrest [FORRES86]. CAMPS has also been used in this way to create set covering models in MPSX format [ELDMIT86]. These models were supplied in a non standard format.

In order to deal with well established structured models or restrictive modelling situations, a compendium of reserved words has been introduced into the TABLES and ROWS section of the system. A reserved table, RESTRICT, with appropriate dimensions is created by default as an internal table of 0-1 entries. It is used subsequently to deal with undefined entries in the primary tables. NETWORK, CONVEX and REFER are reserved row names. NETWORK is used to create a compact network model with balanced flows. CONVEX and REFER are used to achieve separable programming (set type one and set type two) model reformulation within the system [LUMIYA86].

4.3 An annotated example

In this section a problem taken from the book by Jensen and Barnes [JENBAR80] is considered. This example is specially chosen as it displays the typical structure of an integrated production and distribution model. The example is also adopted by Geoffrion [GEOFF85] and Bradley [BRACLE85], [CLEMEN84] to illustrate their systems.

The Tanglewood Manufacturing Co. has four plants located around the country. The fabrication and assembly cost per chair and the minimum and maximum monthly production for each plant are shown in table 4.1.

PLANT	Cost	Max Production	Min Production
	$		
Washington	5.00	500	0
Philadelphia	7.00	750	400
Denver	3.00	1000	500
Buffalo	4.00	250	250

FABRICATION COST AND PRODUCTION RESTRICTIONS BY PLANT

Table 4.1

The company obtains the twenty pounds of wood required to make each chair from two suppliers who have agreed to supply any amount ordered. In return, the company guarantees the purchase of at least 8 tons of wood per month from each supplier. The cost of wood is $0.10/lb from supplier 1 and $0.075/lb from supplier 2. The shipping cost in $/lb from each supplier to each plant is shown in table 4.2.

$/lb of wood	Washington	Philadelphia	Denver	Buffalo
ONTARIO	0.01	0.02	0.04	0.04
QUEBEC	0.04	0.03	0.02	0.02

SHIPPING COST FROM SOURCE TO PLANT

Table 4.2

The chairs are sold in New York, Houston, San Francisco and Chicago.

Transportation costs in $/chair between the cities and plants are listed in table 4.3. Finally table 4.4 shows the minimum demand that must be satisfied, the maximum demand that can be satisfied and the selling price for chairs in each city.

$/Chair	New York	Houston	San Francisco	Chicago
Washington	1.00	1.00	2.00	0.00
Philadelphia	3.00	6.00	7.00	3.00
Denver	3.00	1.00	5.00	3.00
Buffalo	8.00	2.00	1.00	4.00

TRANSPORTATION COST BETWEEN PLANTS AND CITIES

Table 4.3

City	Selling Price Per Chair	Max Demand	Min Demand
New York	$20.00	2000	500
Houston	15.00	400	100
San Francisco	20.00	1500	500
Chicago	18.00	1500	500

SELLING PRICE AND DEMAND RESTRICTIONS BY CITY

Table 4.4

It is desired to find the optimal production and shipment so as to maximise profit. A mathematical statement of this problem is set out below.

-Subscripts and Dimensions

i=1,2	denotes the timber merchants,
j=1,2,3,4	denotes the wood plants,
k=1,2,3,4	denotes the chair retailers.

-Model Coefficients (Descriptors)

c_j	the cost of producing one chair at wood plant j,
n_j	the minimum production of chairs at wood plant j,
q_j	the maximum production of chairs at wood plant j,
p_k	the selling price of chairs at chair retailer k,
l_k	the minimum amount of chairs sent to chair retailer k,
h_k	the maximum amount of chairs sent to chair retailer k,
t_{jk}	the shipment cost between wood plant j and chair retailer k,
m_{ij}	the shipment cost between timber merchant i and wood plant j,
s_i	the cost of wood at timber merchant i,
d_i	the minimum order amount at timber merchant i.

-Model Variables

z_{ij} The quantity of wood bought from timber merchant i and processed in wood plant j,

y_{jk} the number of chairs bought by chair retailer k from wood plant j.

-Linear Constraint Relations: A Mathematical Statement

Maximise

$$\text{Profit} = \sum_{j=1}^{4}\sum_{k=1}^{4} (p_k y_{jk} - c_j y_{jk} - t_{jk} y_{jk})$$

$$- \sum_{i=1}^{2}\sum_{j=1}^{4} (m_{ij} z_{ij} + s_i z_{ij})$$

Subject to the constraints:

minimum demand of the timber merchant i,

$$\sum_{j=1}^{4} z_{ij} \geqslant d_i \qquad i=1,2$$

production at plant j within allowable range,

$$\left.\begin{array}{l} \displaystyle\sum_{k=1}^{4} y_{jk} \geqslant n_j \\ \displaystyle\sum_{k=1}^{4} y_{jk} \leqslant q_j \end{array}\right\} \quad j=1,2,3,4$$

meeting customer demand at k within allowable range,

$$\left.\begin{array}{l} \displaystyle\sum_{j=1}^{4} y_{jk} \geqslant l_k \\ \displaystyle\sum_{j=1}^{4} y_{jk} \leqslant h_k \end{array}\right\} \quad k=1,2,3,4$$

stock balance at plant j,

$$\sum_{i=1}^{2} z_{ij} - \sum_{k=1}^{4} 20 y_{jk} = 0 \qquad j=1,2,3,4.$$

This problem was created using CAMPS, and descriptive names for tables and variables were used instead of one character algebraic symbols. For example c_j is replaced by PLNTCOST(j). Displays 4.4 to 4.18 comprise the major sequence of screenforms and illustrate how the main components are defined. The method of defining names is illustrated by the table names screenform set out in display 4.4. The sets, the reference indices and the corresponding textual annotations are shown in display 4.5. The text for each individual elment of a set is entered using the screen shown in display 4.6. Table dimension and annotations are shown in display 4.7. A compact method for entering data which can accomodate multidimensional (up to six) tables is illustrated in display 4.8. The system also supports a faster spreadsheet type method of entering one and two dimensional tables. The model variables are defined in display 4.9 and similarly, display 4.10 shows how the model rows are defined. Display 4.11 presents the right hand side definitions while the method of entering linear form relations is set out in display 4.12 to display 4.18.

In order to illustrate the method of specifying linear relations and the restrictions introduced to ensure consistency of dimensions, consider the linear form shown in display 4.15. This group of constraints is defined for the index k and is summed over index j. Hence tables and variables which are dimensioned by indices j and k are only displayed in this screenform.

A mathematical statement of the problem is obtained using the documentation facility of the UTILITY subsystem and is illustrated in displays 4.19a and 4.19b. This formulation is sufficiently detailed for communication between analysts. In the linear expressions for the objective row and the constraint rows each term is annotated: a feature also found in GAMS [BISMEE82].

```
SEC:    NAMES SECTION                                    MODEL: TANGWOOD

                TABLE NAME                          TEXT
.....................................................................

                  ..........          ..................
                  .PLNTCOST.          .PLANT-COST------.
                  ..........          ..................

PLNTCOST                              PLANT COST
PLNTMIN                               MIN PRODUCTION
PLNTMAX                               MAX PRODUCTION
CUSTPRCE                              CUSTOMER PRICE
CUSTLDMD                              MIN CUST DMND
CUSTHDMD                              MAX CUST DMND
TCSTPTC                               TRAN COST TO CST
TCSTPTC                               TRAN COST FR SRC
```

Display 4.4

```
SEC:    INDICES SECTION                                  MODEL: TANGWOOD

 SET NAME       TEXT                  INDICES          LLIM    ULIM     STEP
1. I-       TIMBER MERCHANTS     i----------------     ---1    ---2      -1
2. J-       WOOD-PLANTS-----     j----------------     ---1    ---4      -1
3. K-       CHAIR RETAILERS-     k----------------     ---1    ---4      -1
4. --       ----------------     ----------------     ----    ----      --
5. --       ----------------     ----------------     ----    ----      --
6. --       ----------------     ----------------     ----    ----      --
7. --       ----------------     ----------------     ----    ----      --
8. --       ----------------     ----------------     ----    ----      --
```

Display 4.5

```
SEC:    INDICES SECTION                                  MODEL: TANGWOOD

          SET NAME I-                  TEXT TIMBER-MERCHANTS
.....................................................................

                              ..................
                              .ONTARIO---------.
                              ..................

ONTARIO
QUEBEC
```

Display 4.6

SEC: TABLES SECTION MODEL: TANGWOOD

	TABLE NAME	TEXT	TYPE	INDICES
1.	PLNTCOST	PLANT-COST------	-REAL--	j----------
2.	PLNTMIN-	MIN-PRODUCTION--	-REAL--	j----------
3.	PLNTMAX-	MAX-PRODUCTION--	-REAL--	j----------
4.	CUSTPRCE	CUSTOMER-PRICE--	-REAL--	k----------
5.	CUSTLDMD	MIN-CUST-DMND---	-REAL--	k----------
6.	CUSTHDMD	MAX-CUST-DMND---	-REAL--	k----------
7.	TCSTPTC-	TRAN-COST-TO-CST	-REAL--	j-,k-------
8.	TCSTSTP-	TRAN-COST-FR-SRC	-REAL--	i-,j-------

Display 4.7

SEC: TABLES SECTION MODEL: TANGWOOD

TABLE NAME PLNTCOST TYPE -REAL--

j = 1 :FOR WASHINGTON

.....
.------5.----.
.....

Display 4.8

SEC: VARIABLES SECTION MODEL: TANGWOOD

	VARIABLE NAME	TEXT	TYPE	INDICES
1.	WOFSTP--	TIMBER-SHIPPED--	-REAL--	i-,j-------
2.	CHFPTC--	CHAIRS-SOLD-----	-REAL--	j-,k-------
3.	--------	----------	-------	----------
4.	--------	----------	-------	----------
5.	--------	----------	-------	----------
6.	--------	----------	-------	----------
7.	--------	----------	-------	----------
8.	--------	----------	-------	----------

Display 4.9

SEC: ROWS SECTION MODEL: TANGWOOD

	ROW NAME	TEXT	RTYPE	INDICES
1.	WMINSRC-	MIN-AMT-SHIPPED-	GE	i----------
2.	MPROD---	MIN-AMT-PRODUCED	GE	j----------
3.	XPROD---	MAX-AMT-PRODUCED	LE	j----------
4.	CLOW----	MIN-CUST-DEMAND-	GE	k----------
5.	THIGH---	MAX-CUST-DEMAND-	LE	k----------
6.	BSTOCK--	STOCK-BALANCE---	EQ	j----------
7.	PROFIT--	MAXIMISE-PROFIT-	FR	----------
8.	--------	----------	--	----------

Display 4.10

SEC: ROWS SECTION MODEL: TANGWOOD

	ROW NAME	TEXT	R.H.S.
1.	WMINSRC-	MIN-AMT-SHIPPED-	SCRLDMD-
2.	MPROD---	MIN-AMT-PRODUCED	PLNTMIN-
3.	XPROD---	MAX-AMT-PRODUCED	PLNTMAX-
4.	CLOW----	MIN-CUST-DEMAND-	CUSTLDMD
5.	THIGH---	MAX-CUST-DEMAND-	CUSTHDMD
6.	--------	----------	--------
7.	--------	----------	--------
8.	--------	----------	--------

Display 4.11

SEC: ROWS SECTION MODEL: TANGWOOD

ROW NAME WMINSRC(i)

```
SUM OVER j                1.00000*WOFSTP (i ,j )
FOR ALL i
```

Display 4.12

SEC: ROWS SECTION MODEL: TANGWOOD

ROW NAME MPROD (j)

```
SUM OVER k                1.00000*CHFPTC  (j ,k )
FOR ALL j
```

Display 4.13

```
     SEC:   ROWS SECTION                                    MODEL: TANGWOOD

                                ROW NAME XPROD    (j )

.....................................................................

SUM OVER k                  1.00000*CHFPTC   (j ,k )
FOR ALL j
```

Display 4.14

```
     SEC:   ROWS SECTION                                    MODEL: TANGWOOD

                                ROW NAME CLOW     (k )

.....................................................................

SUM OVER j                  1.00000*CHFPTC   (j ,k )
FOR ALL k
```

Display 4.15

```
     SEC:   ROWS SECTION                                    MODEL: TANGWOOD

                                ROW NAME THIGH    (k )

.....................................................................

SUM OVER j                  1.00000*CHFPTC   (j ,k )
FOR ALL k
```

Display 4.16

```
     SEC:   ROWS SECTION                                    MODEL: TANGWOOD

                                ROW NAME PROFIT

.....................................................................

SUM OVER j ,k       -PLNTCOST(j )*CHFPTC   (j ,k )
SUM OVER j ,k       CUSTPRCE(k )*CHFPTC   (j ,k )
SUM OVER j ,k       -TCSTPTC (j ,k )*CHFPTC   (j ,k )
SUM OVER i ,j       -TCSTSTP (i ,j )*WOFSTP   (i ,j )
SUM OVER i ,j       -SCRPRCE (i )*WOFSTP   (i ,j )
```

Display 4.17

SEC: ROWS SECTION MODEL: TANGWOOD

ROW NAME BSTOCK (j)

```
.....................................................................

SUM OVER i              1.00000*WOFSTP  (i ,j )
SUM OVER k        -     20.00000*CHFPTC  (j ,k )
FOR ALL j
```

Display 4.18

4.4 Support for separable and integer programming reformulation

In CAMPS, support for reformulating separable and integer programming problems has been provided. A description of this approach is given in [LUMIYA86]. For instance special table types, variable types (to define SOS type 1 and type 2 variables) and row names (CONVEX*, REFER*) are used to construct separable programming problems. These facilities have been used to reformulate ten representative nonlinear optimisation problems taken from Hock and Schittkowski [HOCSCH81]. In reformulation support bound analysis of the linear form [BRMIWI75], [WILLIA83] plays a fundamental role. CAMPS does not necessarily achieve the most compact or tightest reformulation, but it carries out a range of burdensome algebraic manipulation.

```
*****************************************
*                                       *
*                                       *
*   Model Documentation                 *
*                                       *
*   Prepared by    ...CLucas            *
*                                       *
*   Problem name   ...TANGWOOD          *
*                                       *
*   Date           ...07/01/86          *
*                                       *
*   Time           ...11:45             *
*                                       *
*                                       *
*****************************************
```

INDICES

```
i  =1,   2      # .. TIMBER MERCHANTS .. #
j  =1,   4      # .. WOOD PLANTS      .. #
k  =1,   4      # .. CHAIR RETAILERS  .. #
```

TABLES

```
PLNTCOST(j)     # PLANT COST          ..by.. WOOD PLANTS       .. #
PLNTMIN(j)      # MIN PRODUCTION      ..by.. WOOD PLANTS       .. #
PLNTMAX(j)      # MAX PRODUCTION      ..by.. WOOD PLANTS       .. #
CUSTPRCE(k)     # CUSTOMER PRICE      ..by.. CHAIR RETAILERS   .. #
CUSTLDMD(k)     # MIN CUST DMND       ..by.. CHAIR RETAILERS   .. #
CUSTHDMD(k)     # MAX CUST DMND       ..by.. CHAIR RETAILERS   .. #
TCSTPTC(j,k)    # TRAN COST TO CST ..by.. WOOD PLANTS       ..and.. CHAIR RETAILERS   .. #
TCSTSTP(i,j)    # TRAN COST FR SRC ..by.. TIMBER MERCHANTS ..and.. WOOD PLANTS       .. #
SCRPRCE(i)      # SOURCE PRICES       ..by.. TIMBER MERCHANTS  .. #
SCRLDMD(i)      # SOURCE DEMANDS      ..by.. TIMBER MERCHANTS  .. #
```

VARIABLES

```
WOFSTP(i,j)     # TIMBER SHIPPED      ..by.. TIMBER MERCHANTS ..and.. WOOD PLANTS       .. #
CHFPTC(j,k)     # CHAIRS SOLD         ..by.. WOOD PLANTS      ..and.. CHAIR RETAILERS   .. #
```

ROWS

```
WMINSRC(i)      # MIN AMT SHIPPED   ..by.. TIMBER MERCHANTS .. #
MPROD(j)        # MIN AMT PRODUCED  ..by.. WOOD PLANTS      .. #
XPROD(j)        # MAX AMT PRODUCED  ..by.. WOOD PLANTS      .. #
CLOW(k)         # MIN CUST DEMAND   ..by.. CHAIR RETAILERS  .. #
THIGH(k)        # MAX CUST DEMAND   ..by.. CHAIR RETAILERS  .. #
BSTOCK(j)       # STOCK BALANCE     ..by.. WOOD PLANTS      .. #
PROFIT          # MAXIMISE PROFIT   #
```

CONSTRAINTS

```
Row name WMINSRC(i)                     # MIN AMT SHIPPED ..restriction.. #

Sum over j  [ +1.000000*WOFSTP(i,j) ]
                                        #  ..for.. TIMBER SHIPPED   #

..ge..SCRLDMD(i)                        # .. SOURCE DEMANDS   .. #

                                        For all i


Row name MPROD(j)                       # MIN AMT PRODUCED..restriction.. #

Sum over k  [ +1.000000*CHFPTC(j,k) ]
                                        #  ..for.. CHAIRS SOLD      #

..ge..PLNTMIN(j)                        # .. MIN PRODUCTION   .. #

                                        For all j
```

Display 4.19A

```
Row name XPROD(j)                          ¤ MAX AMT PRODUCED..restriction.. ¤

Sum over k  [ +1.000000*CHFPTC(j,k) ]
                                           ¤  ..for.. CHAIRS SOLD       ¤
..le..PLNTMAX(j)                           ¤ .. MAX PRODUCTION    .. ¤

                                           For all j


Row name CLOW(k)                           ¤ MIN CUST DEMAND ..restriction.. ¤

Sum over j  [ +1.000000*CHFPTC(j,k) ]
                                           ¤  ..for.. CHAIRS SOLD       ¤
..ge..CUSTLDMD(k)                          ¤ .. MIN CUST DMND     .. ¤

                                           For all k


Row name THIGH(k)                          ¤ MAX CUST DEMAND ..restriction.. ¤

Sum over j  [ +1.000000*CHFPTC(j,k) ]
                                           ¤  ..for.. CHAIRS SOLD       ¤
..le..CUSTHDMD(k)                          ¤ .. MAX CUST DMND     .. ¤

                                           For all k


Row name PROFIT                            ¤ MAXIMISE PROFIT ..no restriction.. ¤

Sum over j ,k  [ -PLNTCOST(j)*CHFPTC(j,k) ]
                                           ¤ PLANT COST        ..for.. CHAIRS SOLD       ¤
Sum over j ,k  [ +CUSTPRCE(k)*CHFPTC(j,k) ]
                                           ¤ CUSTOMER PRICE    ..for.. CHAIRS SOLD       ¤
Sum over j ,k  [ -TCSTPTC(j,k)*CHFPTC(j,k) ]
                                           ¤ TRAN COST TO CST ..for.. CHAIRS SOLD       ¤
Sum over i ,j  [ -TCSTSTP(i,j)*WOFSTP(i,j) ]
                                           ¤ TRAN COST FR SRC ..for.. TIMBER SHIPPED    ¤
Sum over i ,j  [ -SCRPRCE(i)*WOFSTP(i,j) ]
                                           ¤ SOURCE PRICES     ..for.. TIMBER SHIPPED    ¤

..fr..0


Row name BSTOCK(j)                         ¤ STOCK BALANCE    ..restriction.. ¤

Sum over i  [ +1.000000*WOFSTP(i,j) ]
                                           ¤  ..for.. TIMBER SHIPPED    ¤
Sum over k  [ -20.000000*CHFPTC(j,k) ]
                                           ¤  ..for.. CHAIRS SOLD       ¤

..eq..0

                                           For all j
```

Display 4.19B

CHAPTER 5

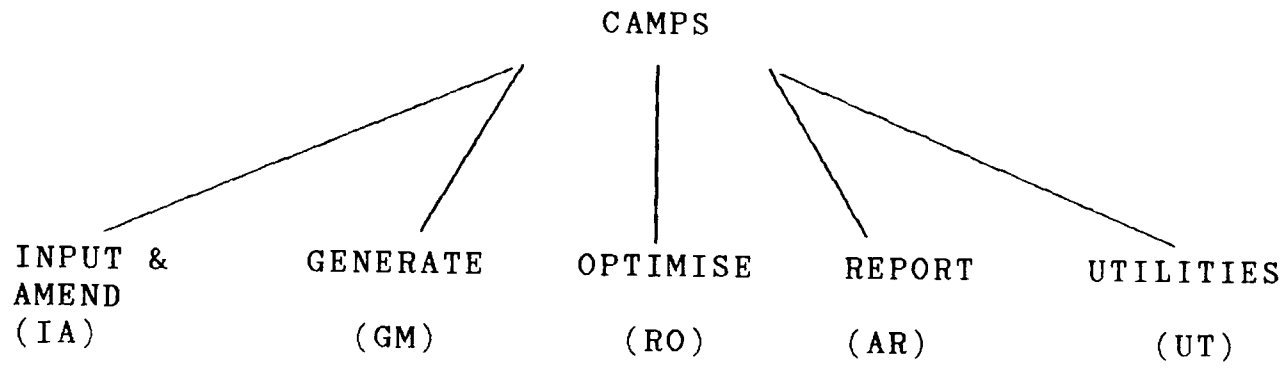
DESIGN AND IMPLEMENTATION ISSUES

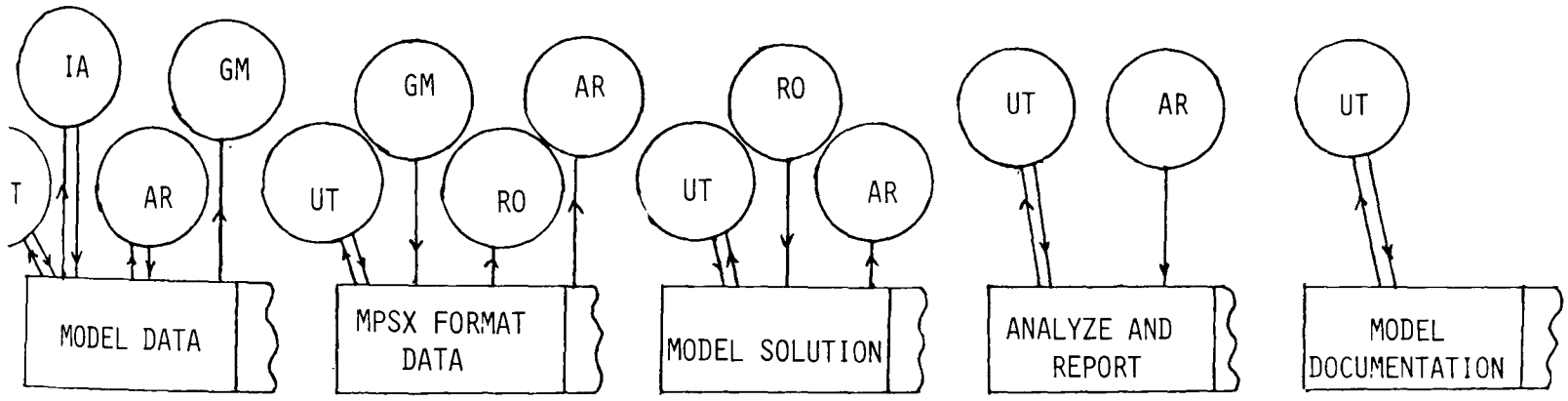
5.1 Introduction

The internal design and strategy followed to implement CAMPS is presented in summary form in this chapter. Section 5.2 contains the system overview covering the main functions of CAMPS and the supporting file structure. These external files are referred to as master files and the actions of the main programs on these master files are also presented in this section. The main logic of controlling menus and screenforms is provided in section 5.3. The implications of changing the external design of a screenform or menu and its effect on the screen data structure is also discussed in this section. The method of managing the internal data structures of CAMPS is briefly considered in section 5.4. CAMPS is conceived to serve as a work station. Thus all the major controls are supplied using the visual display unit. Section 5.5 details the various screen tools which have been adopted for communication using the visual display unit. In common with many application systems, INPUT & AMEND constitutes the main function of CAMPS. The other major task in CAMPS is code generation; the target language in CAMPS is FORTRAN. The implementation language of the system is also FORTRAN. The main issues of code generation and the compile, link and load sequence is described in section 5.6. Finally, the method of constructing external model documentation and that of integrating CAMPS with ANALYZE, are presented in section 5.7.

5.2 System overview

At the top-most level CAMPS comprises a suite of five main programs. A short FORTRAN driving program makes calls to the operating system in order to run these main (subsystems) programs. Display 5.1 illustrates the hierarchy of the system options and the information flow through the five master files as effected by the subsystem.





Display 5.1

The subsystems are mostly FORTRAN based and are integrated with calls to the screen tools which are written in PL1. The INPUT & AMEND subsystem is implemented in FORTRAN. The other subsystems are also written in FORTRAN but contain many calls to the operating system commands and functions. FORTLP [TAMIYA85] is the optimiser for the system and is looked upon as a black box. This program is called with

a limited number of modifiable control variable settings.

There are five master files in the system. These are the Model Master File, the MPSX Input Master File, the Solution Master File, the Report Master File and the Documentation Master File. Any item of data in these five master files is referred to as a Data Module and is identified by the model name which may occupy up to eight characters. For these five master files, the name of any module is synonymous with the Model Name. The contents of each of the files and how and why they are processed by different subsystems is described below. The short names given in the information flow, display 5.1, are used for all the subsystems.

(I) Model Data Master File

For each model a Data Module is created and altered by the INPUT & AMEND (IA) subsystem. Each Data Module comprises dimension, table, model variable, model constraint, and linear relationship information.

UT subsystem accesses it for List, Delete, Rename and similar functions.
IA subsystem uses it for INPUT & AMEND functions.
GM subsystem uses it for generation.
AR subsystem uses it to gather information for analysis and report.

(II) MPSX Input Master File

For each model the GM program creates MPSX input data which is then held in this Master File. MPSX input data from some external source can also be similarly held.

GM subsytem generates each of these data modules in MPSX format.
RO subsystem processes each of these modules to provide a solution.
AR subsystem may use an MPSX format Data Module to prepare a report.
UT subsystem uses it for List, Delete, Rename and such-like functions.

(III)Solution Master File

Each model in MPSX input format, when successfully solved by the optimiser, leads to a solution which can be held in this Master File.

RO subsystem generates the solution data module.
AR subsystem loads the solution information for the purpose of analysis and report.
UT subsystem accesses it for List, Delete, Rename and similar functions.

(IV) Report Master File

ANALYSIS & REPORT subsystem using the Model Data, the MPSX Data and the Model Solution can produce user reports. These user reports are normally printed as text modules of report.

AR subsystem produces text modules of report which are held in this file.
UT subsystem accesses it for List, Delete, Rename and other such-like functions.

(V) Documentation Master File

For the purpose of users own reference, and for communicating with others, a mathematical statement of a model is produced by the system

and is called Model Documentation. The UT subsystem uses a Data Module to produce a Documentation Module which may be printed or stored in this Master File. The UT subsystem also accesses this Master File for the purposes of List, Delete, Rename and other actions.

5.3 Menu and screenform control

The display text used for screens and menus is held in a screenform control file. If this display text needs to be changed, this is achieved by running a separate program which creates the new screen data structure for the reformatted screens. The screenform control file also contains a number of accompanying data tables which describe positions of fields, types of screens (ie whether it is a menu or screenform), menu level and other related information. These data tables are then consulted by the menu and screenform control program to obtain screen layouts. The programs for menu and screenform have essentially the same structure. A skeleton outline of the menu control algorithm is set out in display 5.2.

When control passes to the screen handler, logical checks are introduced to test for completeness of the model. Deletion of a data table name leads to the deletion of a table of numeric data and, by implication, bounds may become equal to zero or right hand sides may be removed or possibily a linear relationship may become undefined. Suitable advice is supplied at screen level to inform the modeller of these consequences. When a specific item of data is either supplied or amended, the screen control program calls a field control tool. To maintain 'model completeness' lists of admissible data values are created at screen level. Subsequently, these are used at field level to check for validity of data input.

```
    Menu counter = 0
1   Display menu of current menu counter
2   Get user response
    If invalid user response then
         Sound action
         Goto 2
    Endif
    If menu level is top and user response equals no of menu options
    then
         return
    Endif
    If menu level is top then
         Menu counter = user response
    Elseif user response equals no of menu options then
         Menu counter =0
    Else
         Screen counter = 0
         Do for i equals 1 to menu counter less 1
              Screen counter = screen counter+no of menu options(i)-1
         Continue
         Screen counter = screen counter + user response
         Display screen of current screen counter
         Call screen handler
    Endif
    Goto 1
```

Display 5.2

5.4 Strategy and tools for handling data tables

The internal storage structure of CAMPS is made up of four main classes of data arrays held in blank common. The first three arrays store real, integer and character data while the fourth array (again integer) consists of the parameters and pointers used in CAMPS. Two main subroutines are used to manipulate these data arrays. These subroutines access and return the global parameters and the local data tables to the main store. The following illustration of local data update provides the data maintenance philosophy of CAMPS. Consider display 5.3 representing the information supplied by a user in the dimension definition screenform.

```
1. I- PLANTS———————— i———————————— 1—— 3—— 1-
2. J- FACTORIES————— j-,k————————— 1—— 2—— 1-
```

Display 5.3

The local tables and global parameters that can be updated in this screenform are listed and described in display 5.4. There are also three local text arrays which are not affected by amending the fields of this screenform.

Table Name	Description
STNMCH	Address in a text array of the name of the chosen set.
STTEXA	Address in a text array of any associated annotation of the set name. A zero indicates there is no text.
STIDCH	Address in a text array of the names of chosen indices.
STNID	The number of indices associated with a given set.
STLLIM	The number of the first element in the set.
STULIM	The number of the last element in the set.
STSTEP	The value of the increment of a given set.
SETSIZ	The number of elements in a given set.
INDSIZ	For each index, the number of elements in it's set.
SETPNT	For each index, this is the address in a text array of the set name it references.
TEXPNT	For each index this is the address in a text array of any associated annotation of the set name as in SETPNT.
TSET	The total number of sets defined.
TSETID	The total number of indices defined.
TSETSZ	This represents the total number of elements for all sets.

Display 5.4

The data table values and parameters to represent display 5.3 together with the text arrays are shown in display 5.5.

STNMCH

No	Element
1	1
2	2

SETNAM

No	Element
1	I
2	J

SETPNT

No	Element
1	1
2	2
3	2

STTEXA

No	Element
1	1
2	2

TEXNAM

No	Element
1	PLANTS
2	FACTORIES

TEXPNT

No	Element
1	1
2	2
3	2

STIDCH

No	Element
1	1
2	2
3	3

INDNAM

No	Element
1	i
2	j
3	k

SETSIZ

No	Element
1	3
2	2

STLLIM

No	Element
1	1
2	1

STULIM

No	Element
1	3
2	2

STSTEP

No	Element
1	1
1	1

PARAMETER POOL

	No	Element
TSET	1	2
TSETID	2	3
TSETSZ	3	5

STNID

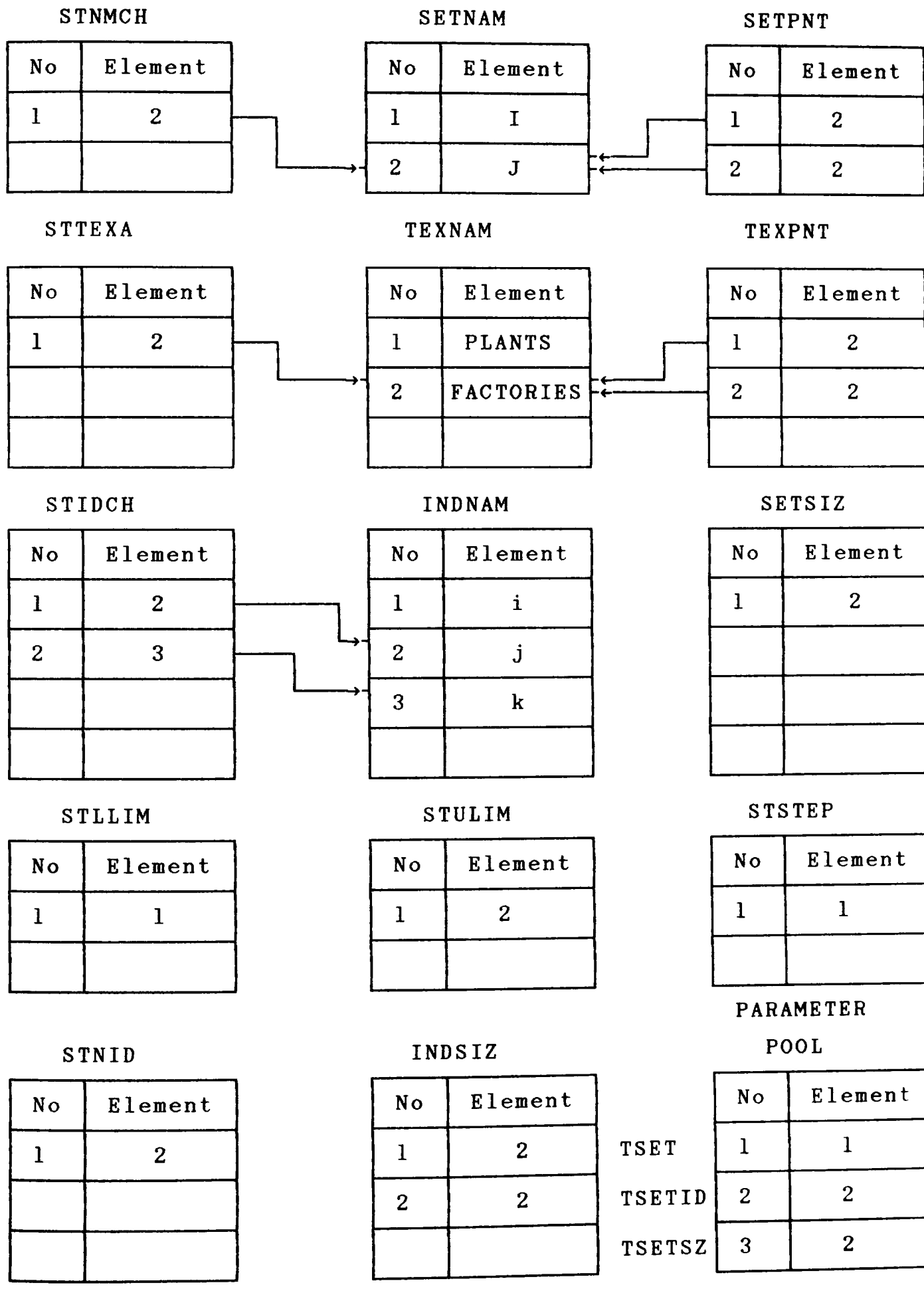
No	Element
1	1
2	2

INDSIZ

No	Element
1	3
2	2
3	2

Display 5.5

If at this stage the set I is removed then the updated local data tables are shown in display 5.6



Display 5.6

5.5 Screen mangement tools

The screen management tools are all written in PL1. In all these procedures there is one or more calls to the operating system to carry out the desired screen functions. The screen itself is defined as a matrix of twenty four rows and eighty columns and is addressed by row and column numbers. A brief description of the PL1 procedures is set out below.

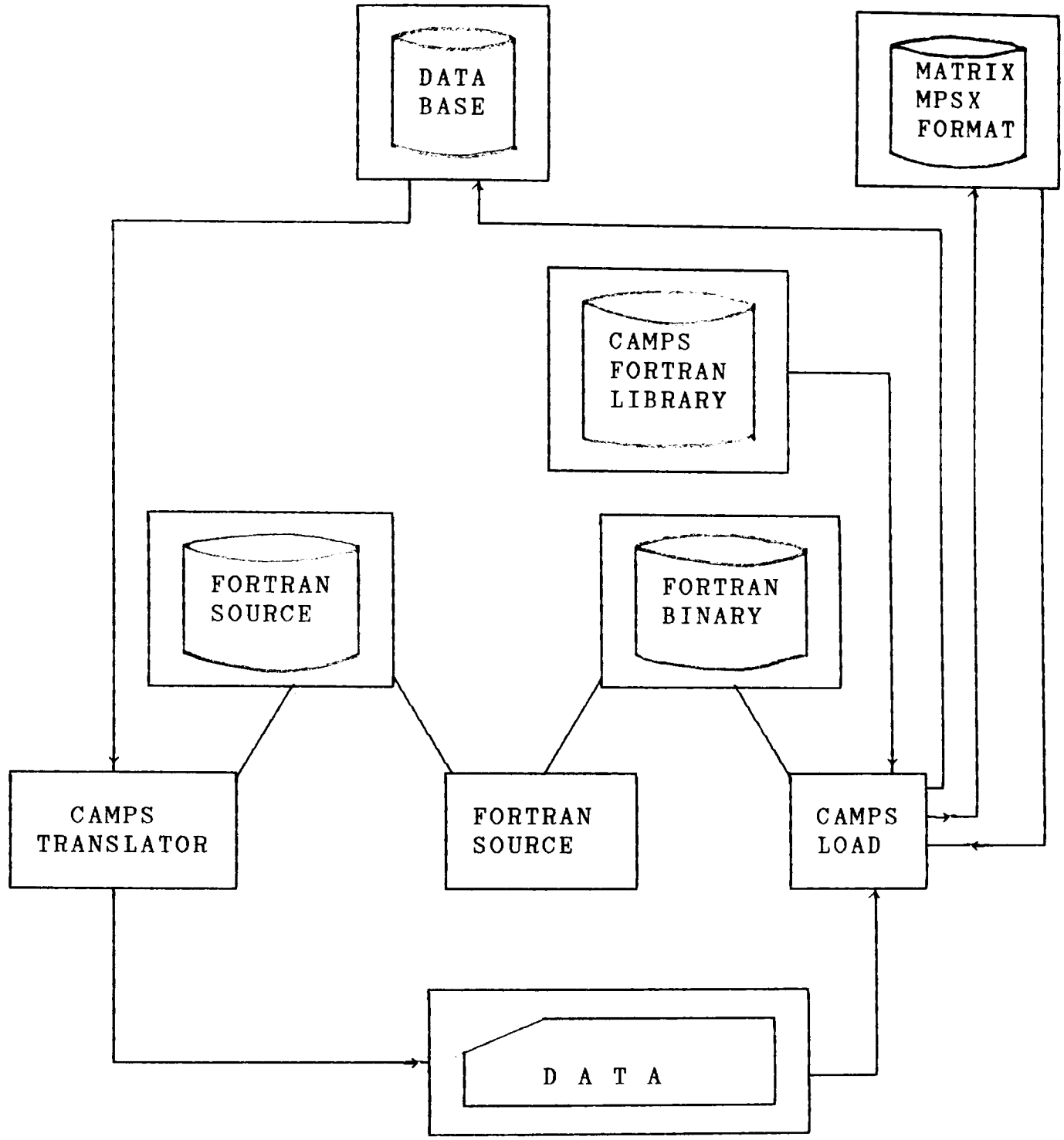
clear_screen	the screen is cleared and the cursor is positioned in the top left hand corner.
clear_to_eol	text to the right hand side of the cursor is cleared to the end of the row.
clear_to_eos	all text below the cursor and to the right in the cursor row, is cleared.
position_cursor	the cursor is positioned at the row and column coordinates specified provided they are within the screen dimensions.
position_curs_rel	the cursor is positioned relative to it's current position. Therefore negative arguments are allowed. Screen dimensions must not be violated.
read_text	unechoed characters are read from the screen and if defined they are echoed back. The DEL function key has the effect of erasing the previous typed character and the cursor is repositioned one space to the left.
write_text	the text supplied in the arguments of this procedure is echoed onto the screen starting from the current cursor position.
read_char	one unechoed character is read from the screen.

ring_bell	an audio bell is sounded.
erase_character	the character to the left of the cursor is erased and the cursor is relocated one place to the left.
get_curs_position	the coordinates of the current cursor position are returned as row and column numbers.
term_screen	the last procedure called in order to disconnect the visual display unit.
init_screen	the first procedure called in order to invoke and initialise the visual display unit.

5.6 Analysis of model and creation of matrix generator program

The model generation subsystem involves three stages; display 5.7 illustrates the information flow of this subsystem through these three stages.

The first stage of the model generation subsystem is the translation phase. A FORTRAN program analyses the model in order to create FORTRAN code. This is used to generate the matrix of the model. Due to the logical analysis and progressive definition of the model components as used by the INPUT & AMEND subsystem, the task of creating code is made considerably easier. There are three main tasks of the CAMPS translator. The first generates the declaration statements and also creates an internal numeric ordering of the rows and columns. Next, a data file is created together with a matching set of subroutine calls which enable the matrix generator program to access this data. Finally, lines of code are generated to internally represent the linear relationships, bounds, right hand side values and the type of linear relationships that exist in the model. The ordering of the matrix is always found by matching rows and column names as given in the model



Display 5.7

definition of INPUT & AMEND. Any exceptions or restrictions on linear relationships result in FORTRAN IF statements, while reserved words create calls to appropriate subroutines. For each line of code generated, there is a character count in order to control when a continuation line is needed.

The main task in the second stage is to compile the generated code. This generated code is then linked to a library of FORTRAN run time subroutines. Some of these subroutines apply simple analysis to the

matrix to check for inconsistencies within the model. When the program is run, if any incompleteness in the model is detected, these subroutines inform the CAMPS load system and the program stops.

The final stage of the generate subsystem is to run the program. The compilation messages are first interrogated and if the compilation is successful, then the program is loaded and run. There are two possible outcomes which the CAMPS load function copes with. If a successful matrix is generated then this is passed back to the database. The other outcome is that the CAMPS generated program detects an unsuccessful situation. Some information is given and the program halted.

5.7 Integration with ANALYZE and model documentation

The integration with ANALYZE closely follows the philosophy for creating external model documentation. CAMPS creates a syntax file together with a separate MPSX input file with MPSX names constructed in accordance with the requirements of ANALYZE. The syntax file provides descriptions of the different name classes. In the result of an unsuccessful exit from the optimiser (ie unbounded solution or no feasible solution) this MPSX input file and syntax file are passed to ANALYZE. ANALYZE uses these in its discourse model [GREENB86], [GRLUMI86] and attempts to provide some rational explanation of the model failure. When creating external documentation, a new file is created giving an annotated mathematical description of the problem. CAMPS maintains data tables which indicate whether a name used in the definition of a model has some text associated with it. If such texts exists, then these are displayed in a predefined documentation format.

CHAPTER 6

AN APPROACH TO COMPUTER ASSISTED REFORMULATION OF INTEGER, SEPARABLE AND FUZZY PROGRAMMING PROBLEMS

6.1 Introduction

It is well known that reformulations of integer, and variable separable programming problems also require considerable insight and modelling skill. The experience with use of modelling support systems has shown that there is a great scope for providing automatic support for reformulating such nonlinear programming problems. The purpose of this chapter is to present a unified approach towards a range of such problems. The methods described here can fit naturally into most LP modelling support systems.

The contents of this chapter are organised as follows. In section 6.2 the LP is defined in a general form in order to introduce notation which is used in the rest of the chapter. Analysis of bounds for linear forms is well known in the context of model reduction [BRMIWI75], [WILLIA83]. Some of the bound analysis results which are pertinent to model reformulation as well are presented in section 6.3. The principles and methods underlying the reformulation technique are described in section 6.4. The main emphasis of this section is to show how logical statements (clausal forms) can always be restated as equivalent integer forms involving 0-1 integer variables. Strategies for separating variables to represent a wide range of nonlinear programming problems are presented and discussed in section 6.5. Reformulation of the fuzzy progamming problem as a max-min LP problem and the relationship of this approach to IP reformulation methods are presented in section 6.6.

The general scope and applicability of these reformulation methods are discussed in section 6.7.

6.2 Statement of the general LP problem and notation

The general LP problem can be stated in the following form:

- Subscripts and their ranges

$$i = 1,\ldots m,\ j = 1,\ldots n.$$

- Variables, constraints, and matrix coefficients:

$$x_j,\ j = 1\ldots n, \quad r_i,\ i = 1\ldots m, \quad d_j,\ j = 1,\ldots n,$$
$$c_j,\ j = 1\ldots n, \quad b_i,\ i = 1\ldots m,$$
$$a_{ij}, i = 1\ldots m,\ j = 1\ldots n.$$

- Linear objective function and constraints:

$$\text{Max} \sum_{j=1}^{n} c_j x_j \quad , \tag{1}$$

$$\text{subject to } r_i: \quad \sum_{j=1}^{n} a_{ij} x_j \varrho_i b_i \quad , \quad i = 1,\ldots,m$$

where ϱ_i is an (in)equality relation of the form "$\leqslant$", "$\geqslant$" or "=",

$d_j : \ell_j \leqslant x_j \leqslant u_j \ , \ j = 1,\ldots,n,$

and ℓ_j may be $-\infty$ or finite and u_j may be $+\infty$ or finite.

6.3 Analysis of bounds for linear forms

- Use of Analysis in Model Reduction

Consider the restrictions r_i and d_j of the linear programming problem set out in (1) expressed as two sets R and D of Linear Form constraints and Structural constraints respectively.

$$R = \{(x_1, \ldots x_n) \mid \sum_{j=1}^{n} a_{ij} x_j \varrho_i b_i, \; i = 1, \ldots, m\} \qquad (2)$$

$$D = \{(x_1, \ldots x_n) \mid \ell_j \leqslant x_j \leqslant u_j, \quad j = 1, \ldots, n\} \qquad (3)$$

It is well known [BRMIWI75], [WILLIA83], that by considering the constraint sets R and D logically and iteratively, in many real life problems one may deduce the following:

(i) whether a constraint in set R is redundant,

(ii) whether a constraint from set R may be removed and replaced by a tighter bound in the set D,

(iii) whether a bound in the set D is redundant.

All these results follow from the analysis of the bounds on the linear forms.

- An Analysis of the Linear Form

Let

$$F_i = \sum_{j=1}^{n} a_{ij}x_j, \quad i = 1,\ldots,m \tag{4}$$

denote the ith linear form.

Introduce two index sets P_i, and N_i, column indices of the positive and negative coefficients of the row i respectively:

$$P_i = \{j \mid a_{ij} > 0\}, \quad N_i = \{j \mid a_{ij} < 0\}, \quad i = 1,\ldots,m. \tag{5}$$

Let

$$L_i \leqslant F_i \leqslant U_i \ , \quad i = 1,\ldots,m \tag{6}$$

denote the bounds on the linear form F_i. From the definition of the structural bounds $(\ell_j \leqslant x_j \leqslant u_j)$ the following is easily deduced:

$$U_i = \sum_{j\in P_i} a_{ij}u_j + \sum_{j\in N_i} a_{ij}\ell_j \ , \tag{7}$$

$$L_i = \sum_{j\in P_i} a_{ij}\ell_j + \sum_{j\in N_i} a_{ij}u_j \ . \tag{8}$$

In any of the following cases:

(a) ρ_i is "$\leqslant$" and $U_i \leqslant b_i$,

(b) ρ_i is "$\geqslant$" and $L_i \geqslant b_i$.

the ith Linear Form constraint is redundant and may be removed from

the problem. Further, it is relevant in the present context to make the following observations concerning this analysis.

(i) L_i, may be $-\infty$ or finite and U_i may be $+\infty$ or finite. However, for finite values of ℓ_j, u_j, j=1,...n it follows from (7)and (8) that L_i, U_i are finite.

(ii) If the linear form constraints are connected by logical restrictions then L_i, U_i values as necessary may be employed to (re)formulate these as 0-1 mixed integer programs.

(iii) The derived bounds may be used in the improved reformulation and partial solution of integer programs.

(iv) It is not well known and rarely discussed in the literature that this analysis constitutes an essential part of any procedure for the reformulation of nonlinear, not variable separable functions into variable separable functions with arguements defined between upper and lower bounds. These can be obtained for the appropriate variable using (7) and (8).

The following examples illustrate some of the principles stated here and serve as an understanding for computing the various bounds that are later used to linearise the functions.

Let the constraint sets R and D be as defined below.

$$R = \{ (x_1,x_2,x_3) \mid x_1 + 2x_2 - x_3 \leqslant 11 \}$$

$$D = \{ (x_1,x_2,x_3) \mid 0 \leqslant x_1 \leqslant 1,\ 0 \leqslant x_2 \leqslant 2,\ 0 \leqslant x_3 \leqslant 4 \}$$

The bounds on the Linear Form F_1 may be deduced as

$$L_1 = -4, \quad U_1 = 5.$$

Thus $U_1 < b_1$, hence the constraint is redundant.

Further, consider R and D as defined as below.

$$R = \{ (x_1,x_2,x_3) \mid x_1 + x_2 - 2x_3 = 2 \}$$

$$D = \{ (x_1,x_2,x_3) \mid 0 \leqslant x_1 \leqslant 1, \; 0 \leqslant x_2 \leqslant 3, \; 0 \leqslant x_3 \leqslant 4 \}$$

Since $a_{13} < 0$ and ρ_1 is "=" an improved bound on x_3 is given by

$$x_3 \leqslant (b_1 - U_1)/a_{13}$$

Now $U_1 = 4$, $b_1 = 2$, $a_{13} = -2$ and hence $x_3 \leqslant 1$ is the new bound which may be introduced in the set D. The bounds for x_1 and x_2 using the new lower bound of x_3 are

$$x_1 \leqslant (b_1 - L_1)/a_{11}, \qquad x_2 \leqslant (b_1 - L_1)/a_{12}$$

giving $x_1 \leqslant 4$ and $x_2 \leqslant 4$ thus u_1 and u_2 are valid bounds. These new bounds are computed from the way the bounds on the Linear Forms are constructed using the two sets P_i and N_i.

6.4 Representation of logical restrictions

Preliminary Considerations and Notation

It is well known that a large range of logical relationships connecting variables and constraint sets may be represented as integer or mixed

integer programs [CONTRO79], [WILLIA78], [SIMONN66], [DANTZI63].

Recently Jeroslow et al [BLJELO85] have set out an exposition and also present experimental results which connect integer programming with propositional logic and theorem proving. They, for instance, consider three well known clausal forms, conjunctive normal form, disjunctive normal form and Horn sentence. They then show how the equivalent integer forms may be constructed. The interest in this chapter is to interpret such theory and to automate reformulation methods which use mixed integer programming. The reformulation methods set out in this section do not necessarily lead to the tightest formulation.

Let

Δ_i i = 1,2,...	denote logical variables which may take values .TRUE. or .FALSE.,
ε_i	take the value 1, if and only if Δ_i is .TRUE., and 0, if and only if Δ_i is .FALSE.,
$\vee$	denote inclusive .OR.,
$\dot{\vee}$	denote exclusive .OR.,
$\wedge$	denote .AND.,
$\equiv$	denote equivalence.

Representing .OR.

If the condition $\Delta_1 \vee \Delta_2 \vee \Delta_3 \vee \ldots \vee \Delta_m$ is required to hold then this can be represented by the constraint

$$\varepsilon_1 + \varepsilon_2 + \ldots + \varepsilon_m \geqslant 1. \qquad (9)$$

Similarly exclusive .OR. relations as in the requirement $\Delta_1 \dot{\vee} \Delta_2 \ldots \dot{\vee} \Delta_m$ can be represented by the constraint

$$\delta_1 + \delta_2 + \ldots + \delta_m = 1. \qquad (10)$$

Furthermore, the relations

$$\delta_1 + \delta_2 + \ldots + \delta_m \geqslant k \qquad (11)$$

and

$$\delta_1 + \delta_2 + \ldots + \delta_m = k \qquad (12)$$

where k is an integer and $1 \leqslant k \leqslant m$, represent the two statements "k or more alternatives hold at any time" and "exactly k alternatives hold at any time".

Representing OR and AND equivalence relations

Let Y denote a logical variable and y the corresponding 0-1 variable. Then the condition : Y is .TRUE. if and only if $\Delta_1 V \Delta_2 V \Delta_3 \ldots \Delta_m$ is .TRUE. (which is expressed as $Y \equiv \Delta_1 V \Delta_2 V \ldots \Delta_m$), can be represented by the constraint

$$-(m-1) \leqslant \delta_1 + \delta_2 + \ldots + \delta_m - my \leqslant 0 \qquad (13)$$

Similarly the logical condition $Y \equiv \Delta_1 \wedge \Delta_2 \wedge \ldots \Delta_m$ can be represented by the constraint

$$0 \leqslant \delta_1 + \delta_2 + \ldots + \delta_m - my \leqslant m-1 \qquad (14)$$

Logically Relating the Linear Form Constraints

A linear form constraint involving n variables represents a point set in E^n. If a number of these are stated and need to be satisfied then these invoke the logical .AND. operation.

For example consider the relations

$$\begin{array}{l} R_1 = \{(x_1 \ldots x_n) \mid \sum_{j=1}^{n} a_{1j}x_j \leqslant b_1\} \\ \vdots \qquad\qquad\qquad\qquad \vdots \\ R_m = \{(x_1 \ldots x_n) \mid \sum_{j=1}^{n} a_{mj}x_j \leqslant b_m\} \end{array} \qquad (15)$$

and let P denote the proposition that $x \in R$, where

$$R = \{(x_1 \ldots x_n) \mid \sum_{j=1}^{n} a_{ij}x_j \leqslant b_i,\ i = 1 \ldots m\}. \qquad (16)$$

If P_i denotes the proposition that $x \in R_i$, $i = 1 \ldots m$ then P is given by the logical form $P = P_1 \wedge P_2 \wedge \ldots P_m$.

To represent the logical .OR. relation of these propositions $P_1, P_2, \ldots P_m$ it is necessary to consider the structural constraint set D as in (3) where some or all ℓ_j, u_j $j = 1,\ldots n$ are finite such that the bounds $U_i, i = 1 \ldots m$ are finite. Also from the redundancy consideration it is required that $b_i < U_i$, $i = 1,..,m$.

Thus the inclusive .OR. relation $P_1 \vee P_2 \vee \ldots P_m$ is given by the integer and mixed integer forms (9) and (17).

$$\sum_{j=1}^{n} a_{ij}x_j - B_i(1 - \varepsilon_i) \leqslant b_i, \quad i = 1 \ldots, m. \qquad (17)$$

In (17) B_i is a finite value such that for $\varepsilon_i = 0$, $B_i + b_i$ is greater than or equal to the upper bound U_i of F_i defined in (4). Thus any finite value for B_i such that

$$B_i + b_i \geqslant U_i \quad , \quad i = 1, \ldots, m, \qquad (18)$$

leads to a valid formulation. The exclusive .OR. and the two forms of k-fold alternatives for these propositions, are similarly obtained by introducing (17) together with (10), (11) or (12) as appropriate.

An Example

This is taken from [WILLIA78] and modified.

Let $R_1 = \{ (x_1,x_2) \mid x_1 + x_2 \leqslant 4 \}$

$R_2 = \{ (x_1,x_2) \mid -x_1 + x_2 \leqslant 0 \}$

$R_3 = \{ (x_1,x_2) \mid 3x_1 - x_2 \leqslant 8 \}$

and Let $D = \{ (x_1,x_2) \mid 0 \leqslant x_1 \leqslant 5, 0 \leqslant x_2 \leqslant 5 \}$

Then

$S = R \wedge D = R_1 \wedge R_2 \wedge R_3 \wedge D$ is as shown in Diagram 6.1.

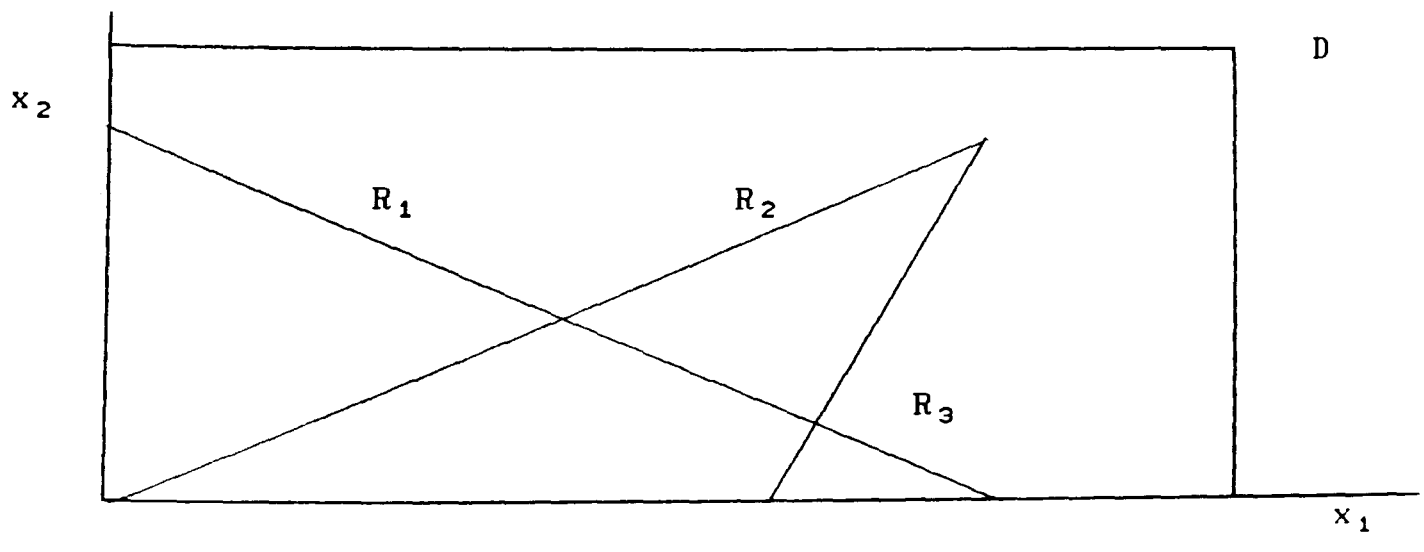


Diagram 6.1

The three bounds on the linear forms may be computed as

$$U_1 = 10,\ U_2 = 5,\quad U_3 = 15.$$

A formulation which uses the logical .OR. as well as .AND. relation is $T \equiv R_1 \vee (R_2 \wedge R_3)$ which may be stated as

$$\begin{aligned} x_1 + x_2 - 6(1 - \varepsilon_1) &\leqslant 4, \\ -x_1 + x_2 - 5(1 - \varepsilon_2) &\leqslant 0, \\ 3x_1 - x_2 - 7(1 - \varepsilon_2) &\leqslant 8, \\ \varepsilon_1 + \varepsilon_2 &\geqslant 1 \quad \text{and} \quad \varepsilon_1, \varepsilon_2 = 0,1. \end{aligned}$$

The constraint region T in this case is as shown in Diagram 6.2.

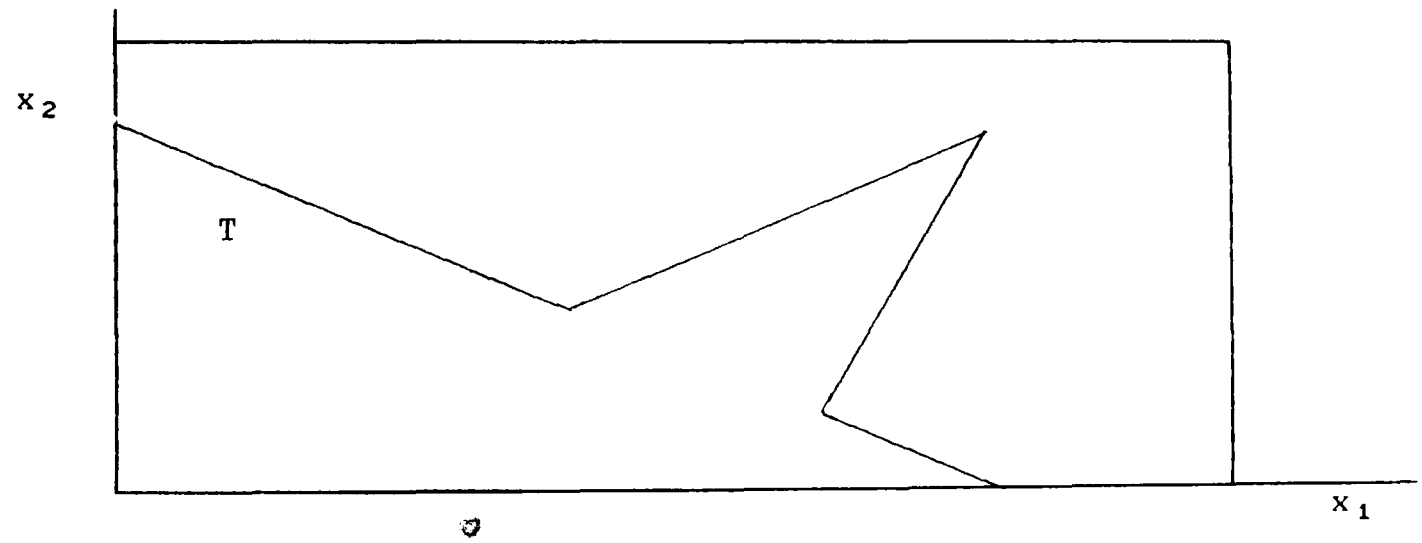


Diagram 6.2

6.5 Strategies for separating variables in nonlinear programming problems

Linearisation of Variable Separable Programming Problems

The problem

$$\text{Max} \quad \sum_{j=1}^{n} f_j(x_j)$$

$$\text{subject to} \quad \sum_{j=1}^{n} g_{ij}(x_j) \leqslant b_i \ , \quad i = 1,\ldots,m,$$

is a general statement of the variable separable programming problem. In order to carry out piecewise linear approximations to the objective and the constraint functions, it is necessary to make two further assumptions concerning this problem.

(i) The functions $f_j(x_j)$, $j = 1,..,n$
are all single valued.

(ii) The arguments x_j, $j = 1,...n$ of these functions have finite ranges $(\ell_j \leqslant x_j \leqslant u_j \ , \quad j = 1,...,n)$.

The construction of piecewise linear approximations using weighting variables, convexity row, reference row, function row and the methods of solution are well discussed in [BRHAMA77] and [MITRA76].

An Analysis of Nonlinear Programming Test Problems

It has been claimed by proponents of the separable programming method of solving nonlinear programming problems that a large class of nonlinear (not variable separable) programming problems can be transformed into variable separable programming problems. In order to investigate the reality of this claim a comprehensive collection of nonlinear programming test problems which have been put together in [HOCSCH81], have been analysed and a selection of these formulated and solved.

Consider the test problems in the format

$$
\begin{aligned}
&\text{Maximise} && f(x_1,\ldots,x_n) \\
&\text{subject to} && g_i(x_1,\ldots,x_n) \leqslant b_i \;,\; i = 1,\ldots,m_1 \\
& && g_i(x_1,\ldots,x_n) = b_i \;,\; i = m_1+1,\ldots,m \\
&\text{and} && \ell_j \leqslant x_j \leqslant u_j \;,\; j = 1,\ldots,n.
\end{aligned}
$$

The frequency distribution of the 115 test problems is set out in Table 6.1. In [HOCSCH81] the problems are numbered from 1 to 119, however, there are no problems numbered 58, 82, 94, 115!

The following types of objective and constraint functions are found in the set of test problems.

Objective function types

(i) Constant objective function ...function code C.

(ii) Linear objective function ...function code L.

(iii) Quadratic objective function ...function code Q.

(iv) Sum of squares objective function ...function code S.

(v) Generalised polynomial objective function...function code P.

This is of the form

$$
f(x) = a_o + \sum_{i=1}^{n} a_i x_i + \sum_{i,j=1}^{n} a_{ij} x_i x_j + \sum_{i,j,k=1}^{n} a_{ijk} x_i x_j x_k + \ldots \qquad (19)
$$

It may be observed that in the geometric programming problem [DEMBO76] a more general form is introduced which is called the signomial function and is expressed as

$$
f(x) = \sum_{j \in J} c_j \prod_i x_i^{d_{ij}} \qquad (20)
$$

where J is used to label the terms appearing in the signomial function. In (19) a_o, a_i, a_{ij} etc. and in (20) c_j, d_{ij} are given real values.

(vi) General function ...function code G.

Constraint types

(i) Only upper and lower bounds on the variables ...code B

(ii) Linear constraint functions ...code L

(iii) Quadratic constraint functions ...code Q

(iv) Generalised Polynomial constraint functions ...code P

This is of the same form as (19) or (20).

(v) Generalised constraint functions ...code G.

		Objective Function Codes						
		C	L	Q	S	P	G	Row sum
Constraint Function Codes	B			1	1	5	2	9
	L			10		8	6	24
	Q	1	7	18	2	9	1	38
	P		2	2		14	3	21
	G		3	6		7	7	23
Column Sum		1	12	37	3	43	19	115

Table 6.1

Experimental Investigations

Some of the methods described in this section together with the bound analysis discussed earlier, were applied to reformulate 10 out of 115 test problems discussed earlier in this section. CAMPS was used to aid these reformulations and generate these models. These problems are discussed and the investigations are reported in [LUCMIT86].

Manipulation of Nonlinear Functions to Variable Separable Form.

The principal motivation of deriving variable separable formulations of nonlinear functions is to approximate these functions by piecewise linear forms. Consequently a standard mathematical programming system (e.g. MPSX) can be used to solve these classes of nonlinear programming problems. In order to apply a piecewise linear approximation it is required that the variables of the separable formulation, which are derived from the original nonlinear functions, be bounded. It is therefore necessary to apply a bound analysis to determine these bounds. In practical applications it is possible to impose realistic bounds on any unconstrained variable which may appear in the problem.

McCormick and Jackson [JACMCC84] have done considerable work on the (reformulation) factorisation of highly complex nonlinear programming problems. They analytically derive the hessian and gradient of the 'factored' forms and are interested in the sensitivity properties of the resulting nonlinear models.

A few frequently occuring instances of nonlinearities (nonlinear terms as well as nonlinear forms) are now considered and the methods of reformulating these are briefly discussed.

Product Term

A product term, x_1x_2, may be replaced by $(y_1^2 - y_2^2)$ with the additional constraints $y_1 = \frac{1}{2}(x_1 + x_2)$ and $y_2 = \frac{1}{2}(x_1 - x_2)$. If $(\ell_j \leq x_j \leq u_j)$ then, given finite ℓ_j and u_j, finite bounds L_i and U_i may easily be derived such that $(L_i \leq y_i \leq U_i)$, $i = 1,2$.

By repeated application of this technique a variable separable formulation

of a higher order product term may be obtained.

Quadratic Function

For a general quadratic function, $\Phi(x_1,\ldots x_n)$ a more compact variable separable formulation may be obtained.

$$\text{Let } \Phi(x_1,\ldots x_n) = \tfrac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij}x_ix_j$$

$$\text{replace } \Phi(x_1,\ldots x_n) \text{ by } \Psi(y_1,\ldots y_r) = \sum_{k=1}^{r} d_k y_k^2$$

with the constraints

$$y_k = \sum_{j=k}^{n} \acute{q}_{kj}x_j \qquad k = 1,\ldots r \tag{21}$$

where r is the rank of the symmetric matrix $Q = ||q_{ij}||$.

The coefficients $\acute{q}_{kj}$ and d_k can be determined by applying a standard method such as Gaussian reduction [STIEFE63].

Given finite bounds ℓ_j and u_j on x_j, $j = 1,\ldots n$, finite bounds L_k and U_k on y_k, $k = 1,\ldots r$, may be simply derived by considering the linear forms (21). Thus a piecewise linear approximation can be used.

Ratio of Linear Forms

$$\text{Let } \acute{H} = \sum_{j=1}^{n} \acute{h}_j x_j \text{ and } \tilde{H} = \sum_{j=1}^{n} \tilde{h}_j x_j \ .$$

The expression $(\acute{H}/\tilde{H})$ may be manipulated in the following way.

Replace $(\acute{H}/\tilde{H})$ by y_1 and introduce the constraint

$$\sum_{j=1}^{n} \acute{h}_j x_j = \sum_{j=1}^{n} \acute{\acute{h}}_j x_j y_1 .$$

As discussed earlier a variable separable formulation may be obtained for the product terms of the constraint. The finite bounds on $x_j, j = 1,...n$, provide bounds on $\acute{H}$ and $\acute{\acute{H}}$ such that $\acute{L} \leqslant \acute{H} \leqslant \acute{U}$ and $\acute{\acute{L}} \leqslant \acute{\acute{H}} \leqslant \acute{\acute{U}}$ from which bounds on y_1 may be obtained. If $\acute{\acute{L}} > 0$ or $\acute{\acute{U}} < 0$, the bounds on y_1 are finite and a piecewise linear formulation can be applied.

Power Forms - Constant Base

Consider the term $a^{x_1+x_2^2}$ where $a > 0$.

A variable separable formulation may be obtained by replacing $a^{x_1+x_2^2}$ by y_1 and introducing the constraint $\log y_1 = (\log a)(x_1 + x_2^2)$. The bounds L_1 and U_1 on y_1 can be derived from the bounds on x_1 and x_2.

Power Forms - Variable Base

Consider the term $x_1^{x_2}$. This term can be handled using the substitution $y_1 = x_1^{x_2}$ and introducing the constraints

$$x_1 = 10^{y_2} \tag{22}$$

$$y_1 = 10^{y_2 x_2} \tag{23}$$

The constraint (23) can be handled using the techniques for product terms and constant base power forms discussed earlier. For constraint (22) it is necessary that $0 < \ell_1 \leqslant x_1 \leqslant u_1$ from which the bounds on y_2 are easily derived.

To illustrate these methods, consider the following problem.

Maximise $x_1 + 2x_2 + x_3$

subject to
$$x_1x_2 + x_2e^{x_3}/(1 + x_1) + x_3 \leqslant 20 \quad (24)$$
$$x_1 + x_2 + x_3 \leqslant 4 \quad (25)$$
and
$$x_1, x_2, x_3 \geqslant 0 \quad (26)$$

From restrictions (25) and (26) it follows that

$$4 \geqslant x_1, x_2, x_3 \geqslant 0 \quad (27)$$

Rewrite
$$x_2/(1 + x_1) = y_1 \quad (28)$$

Using (27) and (28) $\ell_4 \leqslant y_1 \leqslant u_4$ where $\ell_4 = 0$, $u_4 = 4$.

Thus constraint (24) can be expressed as

$$x_1x_2 + y_1e^{x_3} + x_3 \leqslant 20 \quad (29)$$

The product terms of (27) are expressed as

$$y_2^2 - y_3^2 = y_1x_1, \quad y_4^2 - y_5^2 = y_1e^{x_3} \quad \text{and} \quad y_6^2 - y_7^2 = x_1x_2$$

Finally the complete formulation is given as follows

Minimise $x_1 + 2x_2 + x_3$

subject to

$$\begin{aligned}
y_6^2 - y_7^2 + y_4^2 - y_5^2 + x_3 &\leqslant 20 \\
y_1 - x_2 - y_2^2 + y_3^2 &= 0 \\
y_2 - \tfrac{1}{2}y_1 - \tfrac{1}{2}x_1 &= 0 \\
y_3 - \tfrac{1}{2}y_1 + \tfrac{1}{2}x_1 &= 0 \\
y_4 - \tfrac{1}{2}y_1 - \tfrac{1}{2}e^{x_3} &= 0 \\
y_5 - \tfrac{1}{2}y_1 + \tfrac{1}{2}e^{x_3} &= 0 \\
y_6 - \tfrac{1}{2}x_1 - \tfrac{1}{2}x_2 &= 0 \\
y_7 - \tfrac{1}{2}x_1 + \tfrac{1}{2}x_2 &= 0 \\
x_1 + x_2 + x_3 &\leqslant 4
\end{aligned}$$

Lower Bounds

$$y_3 \geq -2$$

$$y_5 \geq -\tfrac{1}{2}\, e^4$$

$$y_7 \geq -2$$

For the functions y_6^2, y_7^2, y_4^2, y_5^2, y_2^2, y_3^2, and e^{x_3} variables are introduced to linearise the functions over their respective domains.

6.6 Reformulation of fuzzy decision problems as max-min LP problems

Background to the Model

Fuzzy set theory was first introduced by Zadeh [ZADEH65] and subsequently Bellman and Zadeh [BELZAD70] discussed its application to decision problems. Later developments and applications of this approach are well discussed in the text book by Dubois and Prade [DUBPRA80]. In Fuzzy set theory an element x is defined to have a degree of membership of a given set say S. The degree of membership is denoted by a membership function $x\mu$ which is defined over the range [o,u] where u is a positive real number. For u=1 it is the normal fuzzy set, $\mu(x) \in [0,1]$. In the usual set theoretic terms x belongs to S is equivalent to $\mu(x) = 1$ and $\mu(x) = 0$ otherwise.

The major contribution of the seminal paper by Bellman and Zadeh [BELZAD70] was to establish the relationship between goals and constraints of a decision problem. In their words:

> "goals and the constraints constitute classes of alternatives whose boundaries are not sharply defined." They then proceed to explain that their modelling framework ..." erases the differences

between goals and constraints and makes it possible to relate in a relatively simple way the concept of a decision to those of the goals and constraints of a decision process..." In short, a broad definition of the concept of decision may be stated as:

Decision = Confluence of Goals and Constraints".

Fuzzy Programming as a decision model was mainly promoted by Zimmermann [ZIMMER78]. Its applications to media selection [ZIMWIE78], and power systems planning [SATSER82] are two of many applications which have been reported. Dyson [DYSON80] considers the multicriteria decision problems, analyses it following the Max-Min approach based on utility function and shows how the latter has the identical form to that of crisp equivalent formulation of the fuzzy LP.

Statement and Reformulation of Fuzzy Linear Programs

Consider the linear programming problem with 1,...k objective (goal) functions and m inexact (soft) restrictions defined as

$$\text{Max } Z \approx Cx$$

$$\text{subject to} \quad Qx \lesssim d$$

$$x \geq 0$$

where x is an n vector
d is an m vector
C is a k x n matrix
Q is an m x n matrix

Let $z = \begin{bmatrix} z_1 \\ \vdots \\ z_k \end{bmatrix}$ denote the 'aspiration levels' (that is the maximum these are expected to achieve) of these k objectives.
Define $A = \left|\left|\frac{C}{Q}\right|\right|$ a (k+m) x (n) matrix, $b = \left[\frac{z}{d}\right]$ a (k+m) vector

Let $$\mu_i(x) = f_i\left(\sum_{j=1}^{n} a_{ij}x_j\right)$$

denote the membership function of the ith goal or restriction, i=1,...k+m. A typical membership function is illustrated in diagram 6.3.

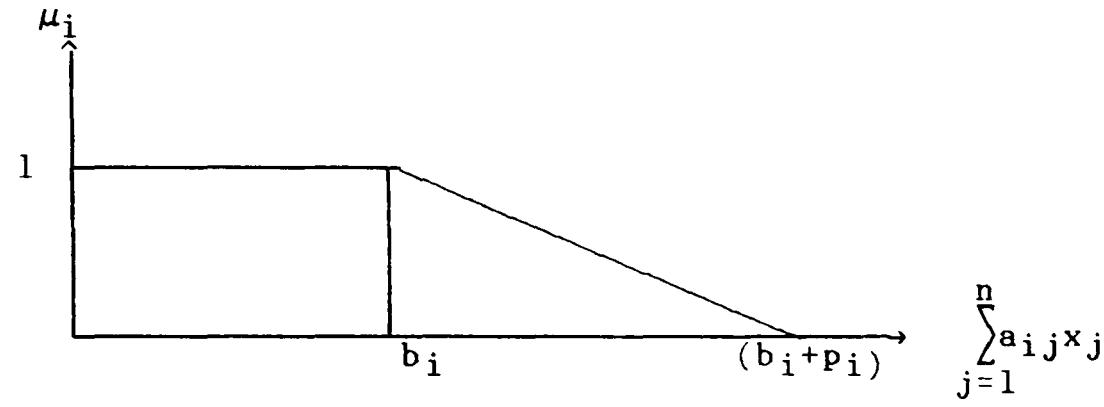


Diagram 6.3

Thus define

$$\mu_i(x) = \begin{cases} 1 & \text{if } \sum_{j=1}^{n} a_{ij}x_j \leqslant b_i \\ 1 - \dfrac{\left[\sum_{j=1}^{n} a_{ij}x_j - b_i\right]}{P_i} & \text{if } b_i < \sum_{j=1}^{n} a_{ij}x_j \leqslant (b_i+p_i) \\ 0 & \text{if } \sum_{j=1}^{n} a_{ij}x_j > b_i+p_i \end{cases}$$

If $\mu_D(x)$ denotes the membership function of the (optimal) decision set then following the usual (but much debated) approach of applying 'Min' as the intersection operator leads to the following

$$\mu_D(x) = \min_i \ \mu_i(x) \quad .$$

Thus maximum satisfaction of constraints and targets are achieved by solving the equivalent Max-Min linear program,

$$\text{Max} \quad \lambda$$

$$\text{subject to} \quad \lambda p_i + \sum_{j=1}^{n} a_{ij}x_j \leqslant b_i+p_i, \quad i = 1,\ldots k+m,$$

$$x_j \geqslant 0, \quad j = 1,\ldots n \ .$$

The following observations can be made for this model.

(a) The multiple objective (or goal) model illustrates Zadeh and Bellman's principle rather well. In the case of a single objective function, $k = 1$.

(b) The fuzzy goals and constraints are alternative ways of introducing soft constraints in the model.

(c) If the variables x_j are bounded, that is $\ell_j \leqslant x_j \leqslant u_j$ as in section 6.3, then U_i as introduced in the section may be used to check the consistency of the fuzzy membership function. Clearly $b_i + p_i \leq U_i$.

(d) If it is desired to construct models which involve crisp as well as fuzzy relations then reformulation methods of section 6.4 and section 6.6 can be naturally put together.

6.7 Automatic approach to reformulation: a summary of issues

The bound analysis plays a key role in automating the steps which are used in reformulating mixed integer, separable and fuzzy programming problems. For instance the algebraic relations which are used to separate variables are also applied to derive bounds on new variables introduced in the reformulation. These bounds are essential for piecewise linear approximation. The bounds on linear forms are also used in transforming propositions (which take logical forms) to equivalent mixed integer linear forms.

The methods described in this chapter do not necessarily achieve the most computationally efficient model after reformulation. Jeroslow [JEROSL86] has given examples of how tighter reformulations can be found. In this work the main aim has been to reduce the chore for an experienced analyst, and also to provide support for a problem owner who is capable of describing his problem but may not be experienced in reformulation techniques. Computer support in these areas offers increased scope and applicability of mathematical programming.

CHAPTER 7

DISCUSSIONS, NEW DIRECTIONS AND CONCLUSIONS

7.1 Introduction

The field of mathematical programming with its increasing acceptance as a proven, tested and robust tool stimulates much research towards the creation of computer based modelling systems. In conceiving and designing such a system, the first task is to identify the audience for whom the system is being built. Chapter three highlights the broad range of constituents who have different requirements from a computer based modelling system. Firstly there is the novice user who knows very little computer programming and thinks of a model as a set of equations. Typically, his requirements can be met by one of the available spreadsheet packages [CARMON86]. Then at the other end of the spectrum, there are dedicated corporate users who run different planning scenarios using large company databases. These systems have to offer flexible and secure access to a large database and cope with multiple users. Reports have to be quickly obtained and response time to crisis modelling has to be good. There is also a need to provide productivity tools for the analysts who create special purpose applications. The main thrust of the present research has been to investigate the type of tools which support these diverse range of modellers in creating their applications quickly and efficiently.

In building models there are many structures that are common to different models. The strategy for modelling these specific structures remains the same no matter what the application. It therefore seems natural that the knowledge of building these structures should be

embedded in the mathematical programming modelling system. Similarly, when a problem owner communicates model data, the system should validate this data within a specific range, and also establish that the units are consistent with the modellers description of the data. This level of support calls for the introduction of artificial intelligence techniques in model building. The scope of integrating artificial intelligence with mathematical programming is discussed in section two of this chapter. This also includes looking at ways in which a natural language could be employed in the modelling support. This ranges from a conversation with the problem owner, to the ability to provide advice concerning the model structure.

In order to harness the proven success of mathematical programming optimisers, it is often necessary to create very special models. No matter how powerful the systems constructs are and how flexible and general the modelling language is, there are often many situations where a certain part of the problem is modelled much more easily by allowing the model builder direct access to the system components which are used for generating the LP matrix. There are also many applications where external programs need to be created to implement special heuristics which have been tested and proven in a different environment. A conscious design of a programmer's (analysts) interface to support these specialist modelling tasks is therefore required. In section three of this chapter the broad criteria of such an interface are discussed. A summary of the major contributions of the research reported in this thesis is presented in the final section of this chapter.

7.2 Artificial intelligence and mathematical programming modelling

In this section, the background and possible use of artificial intelligence in mathematical programming modelling systems is discussed. From time to time considerable attention has been given by the specialists in the field of mathematical programming to the methods of artificial intelligence and vice versa. It is well known that the travelling salesman search methods and heuristics [LINKER71], [GLOVER85] are of interest to both mathematical programming and artificial intelligence specialists. In recent times the links between logic programming theorem proving and integer programming have been investigated by Jerslow [JERSLOW85], Williams [WILLIA86] and others. From the viewpoint of applied problem solving, it is well accepted that mathematical programming and artificial intelligence methods (especially expert systems) are perhaps the most successfully applied methodologies in industrial contexts. In his inspiring lecture in 1959, nobel laureate, Simon [SIMON60], developed and made a strong case for artificial intelligence. In a recent plenary presentation at the TIMS/ORSA meeting, Simon [SIMON86], recounted the success of these two methodologies and argued why they should coevolve. There are, however, not many reported developments which combine these two fields. A few papers that have tackled this integration of the two fields are Slagle and Hamburger [SLABAT85], Murphy and Stohr [MURSTO86] and the work due to Greenberg [GREENB85]. In this research three areas have been identified where the methods of artificial intelligence could be introduced to improve the mathematical programming modelling methodology. These are the scope of applying natural languages, introducing rule bases to modelling, and enhancing modelling support with a knowledge base.

- Natural Languages

Natural language communication to control a particular application is gaining popularity and acceptance. This approach is attractive for interactive communication and it is suitable for use by non experts, although for a skilled modeller it is cumbersome. Gaines [GAISHA84] illustrates how an expert system imitating a doctor, ELIZA [WEIZEN66], mimics the patient in order to create an illusion of intelligence. There is, however, scope for applying natural languages in three areas within the modelling support system. Firstly, it is possible to accept a definition of the model in a natural language from the problem owner and create a compact definition [SHEKRU73] so as to extract the exact information germane to the problem. The second use of natural languages is to create a textually annotated documentation of the model described in English rather than in mathematics. This allows the problem owner to have a simple understanding of the model and to be able to communicate it. Another important use of natural languages is in advice giving. The discourse models currently supplied by ANALYZE [GREENB83] provide narratives in the English language about the model structure and the possible causes for an unsatisfactory termination in the course of optimisation.

- Rule Base

The concept of using rule bases in particular problems can be applied in the context of CAMPS whereby data presentation, model generation, and model solution analysis can be overseen by introducing rule bases. These rule bases could include data validation. This could be used to specify a range of values for which a specific data item is defined, and it could also be used to both check and map data items to consistent

units. A second rule base could be used to verify that the indices of coefficients in linear relationships correspond to those of the summation indices and constraints. In the generation stage, a rule base could investigate that the model contains an objective function and constraints. It could also examine network structures to check that inputs and outputs balance.

- Knowledge Base

A knowledge base can be introduced to support the modeller's task in the following way. An expert modeller when faced with the task of modelling a new problem, consults a series of well known case studies of similar situations and draws upon such 'knowledge'. Thus it is natural to compile a collection of well known and established LP/IP modelling structures. These could be constraints such as material balances, upper and lower bounds, generalised upper bounds and quality constraints. In addition to the components of known variable types and constraints, the knowledge base may also contain complete submodels such as the common form of networks, product mix and blending problems. These aspects of the knowledge base are stored as 'templates' [MURSTO86]. One such template could be a production process (ie the classical transportation problem). This model requires inputs to be shipped to a location where they become outputs. This automatically implies a shipping cost. Further, the structure of the model is such that constraints exist for each input and output. The activities can also be created by an inference mechanism so that for every combination of supply and demand rows, where flows are allowed, it leads to an individual activity. Other information that can be stored concerning this problem are that each activity intersects one supply row and one demand row, supply rows are less than or equal to constraints, demand rows are greater than or equal

to constraints and the non zero coefficients are all ones. All these 'items of knowledge' can be obtained from the knowledge base and can be used by the inference making module to guide the modeller in conceiving and constructing his model.

7.3 Programmer's interface

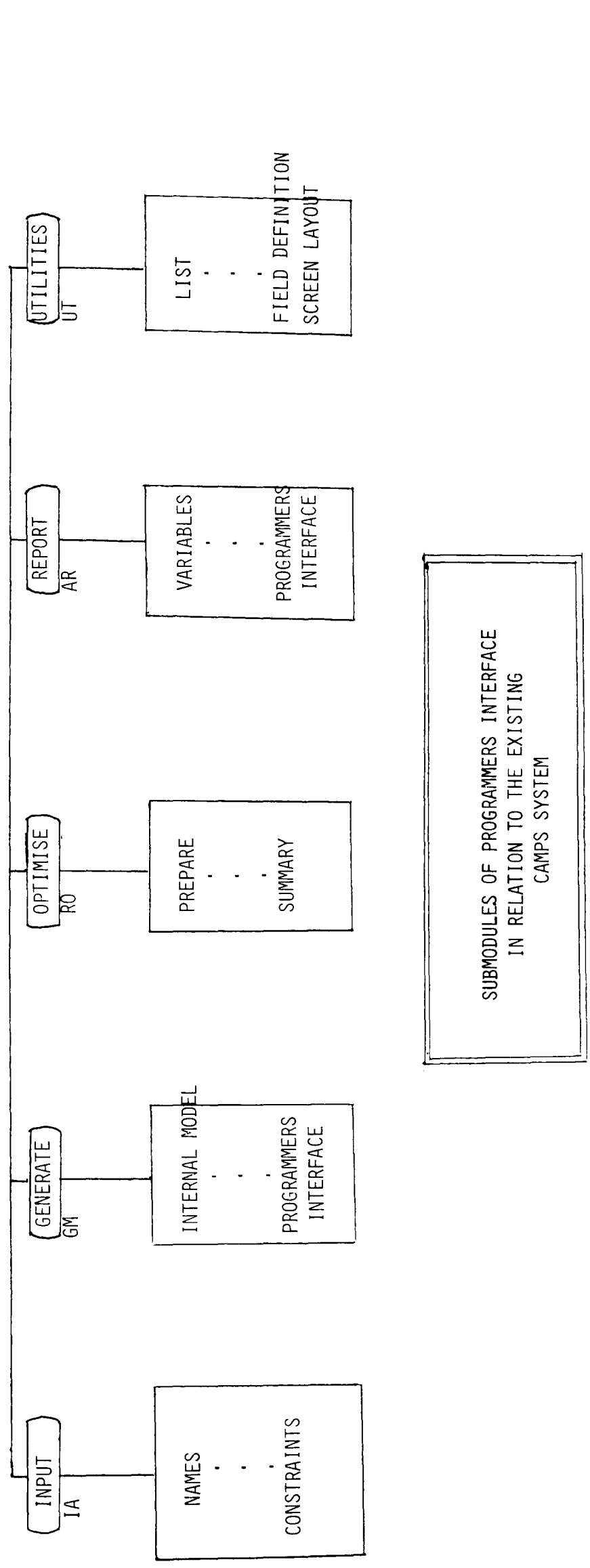
When creating an application, it is often desired to customise the interface between the problem owner and the computer to meet the requirements specific to that situation. Further, sometimes it is easier to describe a model using a programming language rather than be tied to the modelling methodology of a system such as CAMPS. In this way it is possible to introduce heuristic descriptions which are peculiar to the model. These ideas naturally lead to the extension of CAMPS whereby a programmer's interface is introduced. The programmer's interface comprises three new modules. One module of this interface is addressed towards creating data entry screenforms and is a new option under the UTILITIES subsystem. An analyst may use this module to create new screenforms and describe the relationships between these and the data tables which are defined within CAMPS. The screen support tool also incorporates a submodule with field definition and data validation. In this way the analyst can also introduce checks on data entered via these special purpose screenforms. The second interface module is an addition to the GENERATE subsystem. Within this module the analyst can directly program in the implementation language of CAMPS in order to create the linear/integer programming matrix. This offers greater flexibility in building models and many of the subroutines in the CAMPS library may be used which reduces the programming burden of the analyst. Similarly a programmer's interface is added to the REPORT & ANALYSIS subsystem. Through this module the analyst may specify any connections between

submodels and create driver programs for scenario analysis. The new modules are shown in the appropriate subsystems in display 7.1.

The programmer's interface and the scope of its use is now illustrated. In a typical interactive modelling system, such as CAMPS, the end user requirements invariably dictate that the screenform design for data entry goes through a development and update phase. This plays a vital role in the tailoring of an otherwise mathematically sound computer implemented model whereby the non expert user can communicate with the model. For instance, in an application such as combined heat and power (optimisation) scheduling or portfolio selection it is easy to specify screenforms for data entry that are much more appropriate for the corresponding problem than the basic screenforms used in CAMPS to enter data. This task is achieved through the screen support tool within the programmer's interface.

The experience with CAMPS has highlighted that some specialised applications are more efficiently constructed using a programming language. Bus crew scheduling [DARMIT85] and trim loss minimisation [DALUMI86], are examples of two such situations. An analysis of these two models shows that the constraint matrix for the underlying linear programs do not display any special structure. Thus it is necessary to devise special heuristics to generate the columns of the problem matrix. Therefore the analyst can create the generator program using the programmer's interface which has been added to the GENERATE subsystem. The link between the data entry and the model coefficients as defined by the analyst's program would be created through the screen support tool of the UTILITIES subsystem.

PLANETS is a versatile model generator system and is briefly reviewed


SUBMODULES OF PROGRAMMERS INTERFACE
IN RELATION TO THE EXISTING
CAMPS SYSTEM
OPTIMISE
RO
PREPARE
.
.
.
SUMMARY
REPORT
AR
VARIABLES
.
.
.
PROGRAMMERS
INTERFACE
UTILITIES
UT
LIST
.
.
.
FIELD DEFINITION
SCREEN LAYOUT
GENERATE
GM
INTERNAL MODEL
.
.
.
PROGRAMMERS
INTERFACE
INPUT
IA
NAMES
.
.
.
CONSTRAINTS

Display 7.1

in chapter three. It also has extensive facilities for scenario analysis. In order to create a comparable application, it is necessary to define the effects of changes in data values and the size and structure of submodels on the main model. The main thrust of such an application is built around running scenarios. This requires the use of all three programmer's interface modules. The module of the REPORT & ANALYSIS subsystem, is used to define the connections between the submodels in order to cope with 'What if' statements. The other two submodules are used as discussed before.

7.4 Summary and conclusions

This thesis is concerned with the analysis and design of methods leading to software techniques which can be used to support mathematical programming modelling. The sequence of logical steps which lead to the construction of an LP model may be stated as a progressive definition of dimensions, data tables, model variables, model constraints and the matrix coefficients which connect the last two entities. As a result of this research a computer based system is implemented which supports this approach to model description.

The experimental LP modelling system, CAMPS, is described in chapter four. A number of other modelling systems have command and syntax structure whereby the model description follows closely the mathematical statement of the LP. The motivation behind such an approach is to force the modeller to communicate his model in a form that serves also as full documentation. Whereas model documentation is essential, it is not necessary to tie the method by which the modeller communicates his model to the documentation requirements. In CAMPS the model is communicated and updated using menus and screenforms. Documentation

is divorced from this step and is obtained under a separate option. The experience with the system has shown that the menus and screenforms capture a model in far fewer keystrokes than by using a modelling language. Errors introduced due to mistyping are virtually removed: this is due to the progressive and automatic syntax checking and prompting mechanism of the system.

Earlier generation systems which involve high level languages, matrix generators and modelling languages are introduced and discussed in chapter one and are also considered in chapter three. It is shown that set against this, the program generator method, developed in this research, leads to a superior man-machine communication facility to describe LP models. The data structure and data management tools and overall system specification comprise the main design activity for system implementation. This itself is a research area in its own right. In chapter five some of these implementation aspects are considered and presented. The integration of model generation with the ANALYZE subsystem is also described. This integration of CAMPS with model and solution analysis capability extends the scope of annotated documentation. As a result it is possible to provide an advice giving discourse to the problem owner (modeller) when more information regarding the model needs to be supplied: for instance if the model is not solvable.

In chapter six a blueprint for integrating and automating a number of reformulation methods of mathematical programming is presented. The keyrole played by bound analysis in these models is also illustrated. Currently most modelling support systems only allow the user to create the underlying LP model. It is shown that the basic modelling tool can be naturally extended to incorporate reformulation support. By introducing the facility of algebraic manipulation it is possible to reduce

the chore of manual reformulation of models. This aspect may prove to be particularly valuable for problem owners who are capable of describing their problems precisely but may not be experienced in reformulation techniques. Computer support in these areas offers increased scope and applicability of mathematical programming.

References

[ADBELM72] Adams, N., Beglari, F., Laughton, M., and Mitra, G., Mathematical Programming Systems in Electrical Power Generation, Transmission and Distribution Planning, P.S.C.C., Grenoble, 1972.

[ASTCHA75] Astrahan, M.M., and Chamberlin, D.D., Implementation of a Structured English Query Language, Communication of the ACM, Vol. 18, No. 10, 580-588, 1975.

[BEALE68] Beale, E.M.L., Mathematical programming in practice, Pitman, 1968.

[BEALE70] Beale, E.M.L., Advanced Algorithmic features for General Mathematical Programming Systems, in Integer and Nonlinear Programming, Ed J. Abadie, North Holland, 1970.

[BEALE85] Beale, E.M.L., Integer Programming, in Computational Mathematical Programming, Ed. K. Schittkowski, Springer-verlag, 1985.

[BELZAD70] Bellman, R., and Zadeh, L.A., Decision Making in a Fuzzy Environment, Man. Sci, Vol 17, pp 141-164, 1970.

[BISMEE82] Bisschop, J., and Meeraus, A., On the Development of a General Algebraic Modelling System in Strategic Planning Environment, Mathematical Programming Study 20, 1982, North Holland.

[BLJELO85] Blair, C.E., Jeroslow, R.G., and Lowe, J.K., Some Results and Experiments in Programming Techniques for Propositional Logic, Report of Georgia Institute of Technology, January 1985.

[BRACLE85] Bradley, G.H. and Clemence, R., Implementation of a Structured Modeling Language for Optimization, paper presented at the 12th International Symposium on Mathematical Programming, M.I.T., 1985.

[BRHAMA77] Bradley, S.P., Hax, A.C., and Magnanti, T.L., Applied Mathematical Programming, Addison-Wesley, 1977.

[BRMIWI75] Brearly, A.L., Mitra, G., and Williams, H.P., Analysis of Mathematical Programming Models Prior to Applying the Simplex Algorithm, Mathematical Programming, Vol. 8, 54-83, 1975.

[CARMON86] Carmona, J., and Jones, C., S.I.M.P., Spreadsheet Interface to Mathematical Programming User's Guide, Department of Management Systems and Sciences, the University of Hull, 1986.

[CLEMEN84] Clemence Jr., R.D., A Structured Modeling System for Optimization, Master's Thesis, Naval Postgraduate School, Monterey, CA, 1984.

[CONTRO74] Control Data Corporation, APEX-II Reference Manual, No. 59158100, revision C,1974.

[CONTRO79] Control Data Corporation, APEX III Reference Manual, Version 1.2, No. 7607000, Revisions G, 1979.

[DALUMI86] El-Darzi, E., Lucas, C., and Mitra, G., An Investigation of Computer Based Methods for Minimising the Trim loss of Gasket Production, internal report, Coopers Payen Ltd, 1986.

[DANTZI51] Dantzig, G.B., Maximisation of a Linear Functions of Variables Subject to Linear Inequalities, in Activity Analysis of Production and Allocation, ed. T.C. Koopmans, Wiley, New York, 1951.

[DANTZI53] Dantzig, G.B., Notes on Linear Programming (pt3): Computational Algorithm of the Revised Simplex Method, The RAND Corporation Tech. Rept. RM-1266, 1953.

[DANTZI63] Dantzig, G.B., Linear Programming and Extensions, Princeton University Press, Princeton, N.J., 1963.

[DARMIT77] Darby-Dowman, K., and Mitra, G., Matrix Storage Schemes in Linear Programming, Internal report No. STR/15, Department of Statistics and O.R., Brunel University, 1977.

[DARMIT85] Darby—Dowman, K., and Mitra, G., An Extension of Set Partitioning with Application to Scheduling Problems, European Journal of O. R., 1985.

[DASH86] Dash Associates, XPRESS-LP, 1986.

[DAYWIL86] Day, R.E., and Williams, H.P., MAGIC: The Design and Use of an Interactive Modelling Language for Mathematical Programming, IMA 1986.

[DEMBO76] Dembo, R.S., A set of Geometric Programming Test Problems and their Solutions, Mathematical Programming, Vol.10, pp 192-213, 1976.

[DUBPRA80] Dubois, D., and Prade, H., Fuzzy Sets and Systems, Academic Press New York, 1980.

[DYSON80] Dyson, R.G., Maximin Programming, Fuzzy Linear Programming and Multi-Criteria Decision Making, JORS, Vol 31, pp 263-267, 1980.

[EDS86] Lucas, J., Expert System/Mathematical Programming applied to strategic decisions, presented to TIMS XXVII, Gold Coast, Australia, 1986 and winner of Franz Edelman award for Management Science achievement, TIMS, 1986

[ELDMIT86] El-Darzi, E., and Mitra, G., A collection of set covering and set partitioning test problems, internal report, Brunel University, 1986.

[FORRES86] Forrest, J.J.H., A Minimalist Approach to a Modelling Language, presented to TIMS/ORSA Joint National Meeting,Los Angeles,1986.

[FOURER83] Fourer, R., Modeling Languages versus Matrix Generators for Linear Programming, ACM Transactions on Mathematical Software, Vol. 9, No. 2, 143-183, 1983.

[FOURER86] Fourer, R., A New Algebraic Modeling Language for Linear Programming, presented at ORSA/TIMS Joint National Meeting Miami Beach, 1986.

[GAISHA84] Gaines, B.R., and Shaw, M.L.G., The Art of Computer Conservation, Prentice Hall, 1984.

[GEOFF85] Geoffrion, A.M., Structured Modeling, Western Management Science Institute, Graduate School of Management, University of California, Los Angeles, CA 90024, 1985.

[GEOFF86] Geoffrion, A.M., An Introduction to Structured Modeling. Working Paper No. 338, Western Management Science Institute, UCLA, 1986.

[GIMUSW84] Gill, P.E., Murray, W., Saunders, M.A., and Wright, M.H., Sparse Matrix Methods in Optimization, SIAM J. SCI STAT. COMPUT., Vol 5, No 3,1984.

[GLOVER85] Glover, F., Mathematical Programming/Artificial Intelligence links, presented at the 12th International Symposium on Mathematical Programming, M.I.T., Boston,1985.

[GREENB78] Greenberg, H.J., A Tutorial on Matricial Packing, Design and Implementation of Optimization Software, Ed Greenberg, H.J., Sijthoff & Noordhoff, The Netherlands, 185-224, 1978.

[GREENB83] Greenberg, H.J., A Functional Description of ANALYZE: A Computer-Assisted Analysis System for Linear Programming Models, ACM Transactions on Mathematical Software, Vol. 9, No. 1, March 1983, pages 18-56.

[GREENB85] Greenberg, H.J. Towards an Intelligent Mathematical Programming System, presented at the TIMS Workshop, Colorado School of Mines, Golden, CO, 1985 (notes available from author).

[GREENB86] Greenberg, H.J., A Natural Language Discourse Model to Explain Linear Programming Models, Technical Report, University of Colorado at Denver, Denver, 1986.

[GRLUMI86] Greenberg, H.J., Lucas, C., and Mitra, G., Computer-Assisted Modelling and Analysis of Linear Programming Problems: Towards a Unified Framework, presented at ORSA/TIMS, MIAMI 1986.

[HAVERL76] Haverly Systems, Omni Linear Programming System:User and Operating Manual, first edition, 1976.

[HAVERL77] Haverly Systems, MAGEN Reference Manual, 1977.

[HOCSCH81] Hock, W., and Schittkowski, K., Text Examples for Nonlinear Programming Codes, Springer Verlag, 1981.

[HOFPAD85] Hoffman, K., and Padberg, M., LP-Based Combinatorial Problem Solving, in Computational Mathematical Programming, Ed. K. Schittkowski, Springer-verlag, 1985.

[HOLDAY86] Holden-Day Inc, What's Best!, 4432 Telegraph Avenue, Oakland,CA 94609, 1986.

[IBM76] IBM Corporation, Mathematical Programming System Extended, MPSX/370, Reference Manual, SM19-1095-1, 1976.

[IBM77] IBM Coproration, MGRW Program Reference Manual, Program SH19-5014, 1977.

[JACMCC84] Jackson, R.H.F., and McCormick, G.P., The Polyadic Structure of Factorable Function Tensors with Application to High Order Minimisation Techniques, Dept. of Operations Research, George Washington University, 1984.

[JENBAR80] Jensen, P.A., and Barnes, J.W., Network Flow Programming, Wiley, New York, 1980.

[JENSEN86] Jensen, P.A., MICROSOLVE/OPERATIONS RESEARCH, Department of Mechanical Engineering, University of Texas at Austin, U.S.A., 1986.

[JEROSL85] Jeroslow, R.G., Computation-oriented reduction of predicate to propositional logic, internal report, Atlanta, Georgia, 1985.

[JEROSL86] Jeroslow, R., private communication, June 30, 1986.

[KALAN71] Kalan, J.E., Aspects of large-scale in-core linear programming, in proceedings of the 1971 annual conference of the A.C.M., Chicago, 1971.

[KARIRO80] Katz, S., Risman, L.J., and Rodeh, M., A System for Constructing Linear Programming Models, IBM Systems Journal 19, 505-520, 1980.

[KARMAR84] Karmarkar, N., A New Polynomial-Time Algorithm for Linear Programming, Combinatorica, Vol 4, No 4, 1984.

[KETRON75] Ketron Inc., MPSIII DATAFORM User Manual, 1975.

[LINKER71] Lin, S., and Kernighan, B.W., A heuristic technique for solving a class of Combinatorial optimisation problems, Princeton Conference on System Science, 1971.

[LOTUS84] Lotus Development Corporation, Symphony Reference Manual, 161 First Street, Cambridge, MA 02142, 1984.

[LUCMIT85] Lucas, C., and Mitra, G., Computer Assisted Mathematical Programming (Modelling) System: CAMPS, User Reference Manual, Brunel University, 1985.

[LUCMIT86] Lucas, C., and Mitra, G., Reformulation of nonlinear programming test problems to separable programming problems, Brunel University, 1986.

[LUMIYA86] Lucas, C., Mitra, G., Yadegar, J., and Darby-Dowman, K., Linear, Integer, Separable and Fuzzy Programming Problems: A Unified Approach Towards Reformulation, to be published in the JORS, UK.

[MEFEAV77] Mills, R.E., Fetter, R.B., and Averill, R.F., A Computer Language for Mathematical Program Formulation, Decision Sciences 8, 427-444, 1977.

[MIT75] M.I.T. Center for Computational Research in Economics and Management Science, DATAMAT Reference Manual, third edition, No. D0078, 1975.

[MITDAR85] Mitra, G., and Darby-Dowman, K., CRU-SCHED - A Computer Based Bus Crew Scheduling System, in Computer Scheduling of Public Transport 2, Ed. J.M. Rousseau, North Holland, 1985.

[MITELL82] Mitra, G., and Ellison, E.F.D., User Interface to Mathematical Programming:UIMP, ACM Transactions on Mathematical Software, Vol 8, No. 3, p229-255, 1982.

[MITRA70] Mitra, G., Designing Branch and Bound Algorithms for Mathematical Programming, Paper presented at 7th International Symposium on Mathematical Programming, the Hague, 1970.

[MITRA76] Mitra, G., Theory and application of Mathematical Programming, Academic Press, 1976.

[MURSTO85] Murphy, H.M., and Stohr, E.A., An Intelligent System for Formulating Linear Programs, Center for Research on Information Systems, Computer Applications and Information Systems Area, Graduate School of Business Administration, New York University, 1985.

[MURSTO86] Murphy, F.H., and Stohr, E.A. Incorporating Rules for Model Building in an Artificial Intelligence System for Formulating Linear Programs, Working Paper, New York University, 1986.

[ONEILL78] O'Neill, R.P., An interactive Query System for MPS Solution Information, in Design and Implementation of Optimization Software, Ed Greenberg, H.J., Sijthoff & Noordhoff, The Netherlands, 175-183, 1978.

[ORCHAR56] Orchard-Hays, W., The Revised Simplex Method, The RAND Corporation Tech. Rept. P-911,1956.

[PALMER84] Palmer, K.H., Boudwin, N.K., Patton, H.A., Rowland, A.J., Sammes, J.D., and Smith, D.M., A Model Management Framework for Mathematical Programming, An Exxon Monograph, John Wiley, New York, 1984.

[SATSER82] Satoh, H., and Serizawa, Y., An Application of Fuzzy Linear Programming to Expansion Planning of Electric Power Generation, IEE, Japan Technical Meeting, PE-82-3, 1982.

[SCHRAG81A] Schrage, L., Linear Programming Models with LINDO, The Scientific Press, 1981.

[SCHRAG81B] Schrage, L., User's Manual for LINDO, The Scientific Press, Palo Alto, CA, 1981.

[SCICON75] Scicon Computer Services, MGG User Guide, RWG User Guide, 1975.

[SCICON78] SCICON, Computer Services, SCICONIC User Manual, 1978.

[SHEKRU73] Shen, S.N.T., and Krulee, G.K., Solving Linear Programming Problems Stated in English by Computer, Proceedings ACM 73, 299-303, 1973.

[SIMON60] Simon, H.A., The Shape of Automation, reprinted in Perspectives on the Computer Revolution, Ed Z.W. Pylyshyn, Prentice Hall, 1970.

[SIMON86] Simon, H.A., Plenary talk at ORSA/TIMS Joint National Meeting, Miami, 1986.

[SIMONN66] Simonnard, M., Linear Programming, Prentice Hall International, 1966.

[SLABAT85] Slagle, J.R., and Hamburger, H., An Expert System for a Resource Allocation Problem, Communications of the ACM, Vol 28, No 9, 1985.

[SPERRY78] Sperry Univac, GAMMA3, User Manual for UNIVAC 1108 Computers, 1978.

[STIEFE63] Stiefel, E.L., An Introduction to Numerical Mathematics, Academic Press 1963.

[STOHR85] Stohr, E.A., A Mathematical Programming Generator System in APL, Working Paper 96, Center of Research in Information Systems, Graduate School of Business Administration, New York University, 1985.

[TAMIYA85] Tamiz, M., Mitra, G., and Yadegar, J., FORTLP: A Linear, Integer and Nonlinear Programming System, User Manual, Brunel University, 1985.

[TAMIZ86] Tamiz, M., Design Implementation and Testing of a General Linear Programming System Exploiting Sparsity, Phd thesis, Brunel University, 1986.

[TOMLIN70] Tomlin, J.A., Branch and Bound Methods for Integer and Non-convex Programming, in Integer and Nonlinear Programming, Ed J. Abadie, North Holland, 1970.

[UNICOM77] UIMP, User Interface to Mathematical Programming, UNICOM Consultants Limited, 1977.

[WEIZEN66] Weizenbaum, J., ELIZA- A Computer Program for the Study of Natural Language Communications Between Man and Machine, Communications of the ACM, 36-45, 1966.

[WELSH86] Welsh Jr., J.S., PAM - A Practitioner's Approach to Modelling, presented at ORSA/TIMS Joint National Meeting Miami Beach, 1986.

[WILLIA78] Williams, H.P. Model Building in Mathematical Programming, John Wiley & Sons, 1978.

[WILLIA83] Williams, H.P., A Reduction Procedure for Linear and Integer Programming Models, in Redundancy in Mathematical Programming, Edited by S. Zionts et al, 87-109, Springer Verlag, 1983.

[WILLIA86] Williams, H.P., Linear and Integer Programming Applied to Artificial Intelligence, internal report, University of Southampton, 1986.

[WITMCC85] Witzgall, C., and McClain, M., Problem and Data Specification for Linear Programs, U.S. Department of Commerce, National Bureau of Standards, Report NBSIR 85-3125, 1985.

[WOLSEY85] Wolsey, J., Mixed Integer Programming Model Formulations and Algorithms, presented at the 12th International Symposium on Mathematical Programming, M.I.T., Boston, 1985.

[ZADEH65] Zadeh, L.A., Fuzzy Sets, Information and Control, Vol. 8, p338-353, 1965.

[ZIMMER78] Zimmermann, H.J., Fuzzy Programming and Linear Programming with Several Objective Functions, Fuzzy Sets and Systems, Vol 1, pp 45-46, 1978.

[ZIMWIE78] Zimmermann, H.J., and Wiedey, G., Media Selection and Fuzzy Linear Programming, JORS, Vol 29, pp 1071-1084, 1978.

APPENDIX 1

A COMPARISON OF CAMPS WITH OTHER SYSTEMS

Using the sample problem of chapter four, a comparison of CAMPS problem specification method with those of ULP and OMNI is presented here. OMNI is a well established matrix generator system in which the linear programming matrix is specified a column at a time. ULP is a recently developed modelling language and incorporates many ideas also found in CAMPS. Thus the data entry which is separate from model definition follows the logical sequence whereby the sets are first defined and then the data tables. The model is then conceived in the equation form and generated using row statements. The problem formulations in ULP and OMNI have not been tested but were developed by reading user manuals, however the CAMPS formulation has been tested and the resulting model optimised.

```
TANGLEWOOD - ULP

*RANGES

MERCHANTS:ONTARIO,QUEBEC;
PLANTS:WASHINGTON,PHILADELPHIA,DENVER,BUFFALO;
RETAILERS:NEW YORK,HOUSTON,SAN FRANCISCO,CHICAGO;

*TABLES

PLANT COSTS(PLANTS): 5 7 3 4;
MIN PROD(PLANTS): 0 400 500 250;
MAX PROD(PLANTS): 500 750 1000 250;
```

```
SELL PRCE(RETAILERS):20 15 20 18;
MIN CUST DMND(RETAILERS): 500 100 500 500;
MAX CUST DMND(RETAILERS): 2000 400 1500 1500;
TRAN COST CUST(PLANTS,RETAILERS): 1.0 1.0 2.0 0.0
                                  3.0 6.0 7.0 3.0
                                  3.0 1.0 5.0 3.0
                                  8.0 2.0 1.0 4.0;
TRAN COST DLR(MERCHANTS,PLANTS): 0.01 0.02 0.04 0.04
                                 0.04 0.03 0.02 0.02;
SCR PRCE(MERCHANTS): 0.1 0.075;
SCR DMND(MERCHANTS): 8 8

UNKNOWN (X(MERCHANTS,PLANTS),Y(PLANTS,RETAILERS))
COMMENT (X(MERCHANTS,PLANTS)=AMOUNT TIMBER FROM MERCHANT TO
              PLANT)
COMMENT (Y(PLANTS,RETAILERS)=AMOUNT CHAIRS FROM PLANT TO
              RETAILER)

LPMAX  (SELL PRCE(RETAILERS)*Y(PLANTS,RETAILERS)
        -PLANT COSTS(PLANTS)*Y(PLANTS,RETAILERS)
        -TRAN COST CUST(PLANTS,RETAILERS)*Y(PLANTS,RETAILERS)
        -TRAN COST DLR(MERCHANTS,PLANTS)*X(MERCHANTS,PLANTS)
        -SCR PRCE(MERCHANTS)*X(MERCHANTS,PLANTS))

CONSTRAIN  (PLANTS:X(MERCHANTS,PLANTS)>SCR DMND(MERCHANTS))
CONSTRAIN  (RETAILERS:Y(PLANTS,RETAILERS)>MIN PROD(PLANTS))
CONSTRAIN  (RETAILERS:Y(PLANTS,RETAILERS)<MAX PROD(PLANTS))
CONSTRAIN  (PLANTS:Y(PLANTS,RETAILERS)>MIN CUST DMND(RETAILERS))
CONSTRAIN  (PLANTS:Y(PLANTS,RETAILERS)<MAX CUST DMND(RETAILERS))
```

```
CONSTRAIN  (MERCHANTS,RETAILERS:Y(PLANTS,RETAILERS)
                -20*X(MERCHANTS,PLANTS)=0)
```

TANGLEWOOD - OMNI

DICTIONARY

CLASS MER	Set of timber merchants:
ONT	Ontario
QUE	Quebec
CLASS PLA	Set of plants:
WAS	Washington
PHI	Philadelphia
DEN	Denver
BUF	Buffalo
CLASS RET	Set of retailers:
NEW	New York
HOU	Houston
SAN	San Francisco
CHI	Chicago

DATA

TABLE A Plant costs for production of

* CHAIRS

	COSTS
WAS	5
PHI	7
DEN	3
BUF	4

TABLE B Minimum production level at each plant

	MIN
WAS	0
PHI	400
DEN	500
BUF	250

TABLE C Maximum production level at each plant

	MAX
WAS	500
PHI	750
DEN	1000
BUF	250

TABLE D Selling prices to retailers

	PRC
NEW	20
HOU	15
SAN	20
CHI	18

TABLE E Minimum retailer demands

	MIN
NEW	500
HOU	100
SAN	500
CHI	500

TABLE F Maximum retailer demands

	MAX
NEW	2000
HOU	400
SAN	1500
CHI	1500

TABLE G Cost of transport from each plant to each retailer

*

	NEW	HOU	SAN	CHI
WAS	1.0	1.0	2.0	0.0
PHI	3.0	6.0	7.0	3.0
DEN	3.0	1.0	5.0	3.0
BUF	8.0	2.0	1.0	4.0

TABLE H Costs of transport from each merchant to each plant

*

	WAS	PHI	DEN	BUF
ONT	0.01	0.02	0.04	0.04
QUE	0.04	0.03	0.02	0.02

```
    TABLE I                          Costs of timber at each timber
*                                                merchant
          PCE
    ONT   0.1
    QUE   0.075

    TABLE J                          Minimum demand at each timber
*                                                merchant
          MIN
    ONT     8
    QUE     8

FORM ROW ID
*Maximise operating profit
        OBJ=OBJ
*Satisfy minimum production at plants limit
        PLN(PLA)=MIN
*Satisfy maximum production at plants limit
        PLX(PLA)=MAX
*Satisfy minimum order quantity
        MEN(PLA)=MIN
*Satisfy minimum customer demand limit
        CUN(RET)=MIN
*Satisfy maximum customer demand limit
        CUX(RET)=MAX
*Satisfy balance of wood stock at each plant
        WOB(PLA)=FIX
```

COLUMNS

*Shipping activity for wood from merchants

FORM VECTOR X(MER)(PLA)

*The amount of timber bought from merchant

MEN(PLA)=1

*The amount of wood consumed in making chairs

WOB(PLA)=-20

*The cost of buying and shipping timber

OBJ=-TABLE H ((PLA),(MER)) - TABLE I (PCE,(MER))

*Shipping activity for chairs from plants to retailers

FORM VECTOR Y(PLA)(RET)

*The amount of chairs produced at the plant

PLN(PLA)=1

*The amount of chairs produced at plant

PLX(PLA)=1

*The amount of chairs retailer buys

CUN(RET)=1

*The amount of chairs retailer buys

CUX(RET)=1

*Amount of chairs produced at plant

WOB(PLA)=1

*The effective profit of selling chairs

OBJ=TABLE D (PRC,(RET)) - TABLE A (COSTS,(PLA))

-TABLE G ((RET),(PLA))

RHS

FORM VECTOR RHSIDE

*Minimum plant production

PLN(PLA)=TABLE B (MIN,(PLA))

*Maximum plant production

PLX(PLA)=TABLE C (MAX,(PLA))

```
*Minimum order amount
        MEN(PLA)=TABLE J (MIN,(MER))
*Minimum customer demand
        CUN(RET)=TABLE E (MIN,(RET))
*Maximim customer demand
        CUX(RET)=TABLE F (MAX,(RET))
*Note the right hand sides for the balance rows and
*objective are zero
ENDATA
```