Mathematical Modeling
Various Models from Textbooks

Tony Hürlimann

Department of Informatics
University of Fribourg
CH – 1700 Fribourg (Switzerland)
tony.huerlimann@unifr.ch

March 13, 2025

(First Edition: Sep 03, 2015)

 Department of Informatics
University of Fribourg
CH-1700 Fribourg Switzerland
Email: info@matmod.ch
WWW: https://matmod.ch

Contents

List of Figures

List of Tables

PREFACE

This text is a collection of many (mostly very small) mathematical models formulated in the LPL modeling language and executable directly though the Internet. The models are from many sources and are of varying quality. A wide range of applications is covered, but there is no claim of a systematical order. The collection has grown over time. For more information about a particular model, the corresponding book must be consulted. The models are not for professional modeling in OR, most of them are introductory examples to do the first step in modeling. Furthermore, there is no explicit documentation for the models as I do normally when writing a model.

After a short introduction to LPL in Chapter 1, the next Chapter 2 contains small models from exercises in several known operations research textbooks. There are models from the textbook of Hillier [12], Murty [15], Shapiro [20], and many from Winston [24]. If possible, a small problem description was copied from the book. However, for full explanation the book must be consulted.

Chapter 3 implements a few models from the GAMS Library.

Chapter 4 implements a few examples from the French book of Billonnet [1].

The last Chapter implements miscellaneous models collected over the time.

All models are part of an automatic testing of LPL. When a new LPL version is deployed all these models must run correctly. Selected models can also be used for self study of modeling concepts or in classes.

The reader does not need to install any software to run and solve the models compiled in this text. Only an Internet browser and an Internet connection is

needed. A link appears in the title of each model. Click on it, and you are on the Internet site where the model is prompted in a text box. You can modify it interactively. Clicking the button "Send" on the web page, sends the model to the LPL-server, runs it and returns the result after it has been solved in a new browser page. If you know well the syntax of LPL, you can even write an entirely new model and present it to the LPL-server as well. Just click "Run and Solve" to solve it.

However, the solution time is limited to 60secs, and large and complex models cannot be solved this way.

The book was compiled using the typesetting system LaTeX. However, the documentation of all models was written in LPL's own "literate documentation system". LPL contains a documentation tool (similar to *javadoc* for Java programs) to automatically translate a documented model into a full LaTeX code. Additional information for this tool can be found in the Reference Manual of LPL [22].

Availability of the LPL-Modeling-System

The mathematical modeling and all case studies in this book are based on the software LPL. It is my own contribution to the field of computer-based mathematical modeling.

Currently, there is an implementation for Windows. A Linux console version is available – write me an email.

LPL – the software – together with documentation containing the Reference Manual, several papers and examples can be downloaded at:

MATMOD

CHAPTER 1

INTRODUCTION

"The essence of mathematics is not to make simple things complicated, but to make complicated things simple."

— S. Gudder

"Ich betrachte diese einleitenden theoretischen Betrachtungen als die schwierigsten, weil wir die ganze Zeit das Gehirn bemühen müssen. Nachher können wir an seiner Stelle die Mathematik verwenden!"

— A. Eddington

This chapter introduces briefly the concept of a model and related notions. Then it gives a rudimentary overview of the model building process in the way it is used for the examples in the following chapters. Finally, since all models are formulated in the executable modeling language LPL, the basic elements of this language are briefly presented.

1.1. Model and Related Concepts

Solving a small problem is like solving a mini research problem. Both require a certain insight to see the problem from the right angle, and then explore that idea until one finds a solution. Problem solving is fun! But even more exciting than solving a particular instance of a problem by any ad-hoc method or by *trial and error* is to find a general method and representation to solve it. In this book, we are not so much interested in finding a solution to a specific problem in a precise way. We want to write a general statement that describes it. If we can transform a problem this way, we can give the transformation to existing software (called a *solver* in this book) to solve it generally. **It is the process of building and composing a mathematical model**. However, there is not as much "operative" mathematics involved as the reader may suppose. No complicated derivation or algebraic transformation is used or has to be learnt. The reader only needs to have an idea of the basic formal mathematical and logical notation and how to read and use it.

So what is a model? The word *model* has many meanings: "I want to be a model father for my children", "She works as a photo model", "My son plays with the railway model", "This is an exact model of the Titanic", "The car was an old model", "She is an artist, she can model a head with clay", Rutherford came up with a model of the atom which had electrons orbiting around a nucleus, an orrery is a model of our solar system. In operations research (OR)[1] we use mathematical models to find the shortest round-trip of a journey or the "best" way to use resources in production.

We use models everywhere, even if we do not recognize them for what they are. Many of those we are using every day, are becoming routine instruments: The most elementary budget involves a simplified representation of the future – this is a *model*. Our daily way to the work office makes implicit estimations about the traffic. All our decisions are based on implicit or explicit, on more or less correct assumptions and models. In fact, we could say that modeling itself is a ubiquitous human activity; it is one of the fundamental ways in which human understand and modify the world.

While a *model* is a simplified representation of a problem or a situation, *modeling* is the process of building and refining it for greater insight and improved decision-making. Although the final goal of the modeling process is to obtain a model to work with, the modeling process itself is often as important, because its discovery path teaches us more about the problem than the resulting model. Mostly, we do not use the settled model as a static piece of knowledge. What makes it more interesting is the fact that it can be used as a vehicle to

[1] Operations research is a branch of applied mathematics and decision theory, which uses mathematical models, statistics, and algorithms to aid in decision-making. It is most often used to analyze complex real-world systems, typically with the goal of improving or optimizing performance.

further research and queries varying the initial questions and problems.

Models, and especially mathematical models, are capable of giving deeper *insight* to understand better how "things tick". The consequences are that we can make *better decisions* and save costs or get better returns. Better models are also a *vehicle to communication* and to accumulate and *store our knowledge* and to give us an analytic framework to *explore the world*, testing alternatives and variants.

Science is modeling – and mathematics as a research of patterns is at the very core of (better) modeling. Why mathematics? Mathematics is about building concepts, making new links, searching for new patterns – this is inherently linked to *abstraction*. Abstracting is a process of *determining the similarities and properties* of different objects that make it possible to address them as one. It is also a process of *finding analogous structures*, *leaving out the unessential* in different contexts – this is very much linked to *modeling*. Of course, it is often difficult to classify, to structure – but it is at the core of all scientific activities.

Modeling is a powerful means to solve problems, however, it is all but easy. So, how can modeling be learnt? Problems, in practice, do not come neatly packaged and expressed in mathematical notation; they turn up in messy, confusing ways, often expressed, if at all, in somebody else's terminology. Whereas problem solving can generally be approached by more or less well-defined techniques, there is seldom such order in the problem posing mode. Therefore, a modeler needs to learn a number of skills. He must have a good grasp of the system or the situation which he is trying to model; he has to choose the appropriate mathematical tools to represent the problem formally; he must use software tools to solve the models; and finally, he should be able to communicate the solutions and the results to an audience, who is not necessarily skilled in mathematics.

It is often said that modeling skills can only be acquired in a process of learning-by-doing; like learning to ride a bike can only be achieved by getting on the saddle. It is true that the "case study approach" is most helpful, and many university courses in (mathematical) modeling use this approach. But it is also true – once some basic skills have been acquired – that theoretical knowledge about the mechanics of bicycles can deepen our understanding and enlarge our faculty to ride it. This is even more important for modeling industrial processes. It is not enough to exercise these skills, one should also acquire methodologies and the theoretical background of modeling. In applied mathematics, more time than it is currently spent, should be given to the study of *discovery*, expression and *formulation of the problem*, initially in non-mathematical terms. So, the novice first needs to be an observer and then, very quickly, a doer. Modeling is not learnt only by watching others building models, but also by being actively and personally involved in the modeling process.

The use of computer modeling tools to simulate, visualize, and analyze mathematical structures has spread steadily. In the eighties, the "micro-computer" – a word which disappeared as quickly as it emerged – was the archetype of a whole generation of self-made quick and messy models. Everyone produced their own simulation tool, mathematical toolbox, etc. This phenomenon has now almost vanished. We have powerful packages such as Mathematica, Maple, Axiom, the NAG-library, Matlab, R, Python packages, Julia packages, and powerful MIP solvers to mention just a few, for solving complex problems. But the modeling process still needs more than easy-to-use solving tools: It also needs tools to integrate different tasks: such as data manipulation tools, representation tools for the model structure, viewing tools to represent the data and structure in different ways, reporting tools, etc. Some tasks can be done by different software packages: data are preferably manipulated in databases and spreadsheets, and are best viewed by different graphic tools.

This brings us to the heart of this book: Modeling is still an art with tools and methods built in an ad-hoc way for a specific context and for a particular type of model. While I believe that modeling will always be an art in the sense of a creative process, I also believe that we can build general computer-based modeling tools that can be used not only to help to find the solution to a mathematical model, but also to support the modeling process itself. The extraordinary advances in science, on one hand, and in computing performance, on the other hand, leave a cruel gap that can be bridged by general and efficient modeling management systems and mathematical modeling languages. As already mentioned, all models presented in this book are formulated with the modeling language LPL. After the problem is formulated in this language specification, our task is finished. The rest – solving a model instance – will be done by the software.

1.2. The Model Building Process

The immediate approach to solving problems is by *trial and error*, as mentioned. An alternative and much more general approach is by creating a model – in our cases a mathematical model. This has the advantage that the designed model can then be applied to a whole class of similar problems and is not only applicable to the specific problem example at hand.

Building a mathematical model means to conceptualize, to invent, to contrive a scheme to turn a "specification" into a operational set of (mathematical) formulas or other concepts. Finding a good model for a problem is not a trivial task. It is a sloppy process; false steps and dead-ends are common. The idea that there exists a rational, error-free path from the problem description to a useful model is quite unrealistic. Model building is also non-deterministic: In two situations, you can come up with very different models for the same problem. There does not exist a unique way to design a model and no tool is

right for everything. In short, modeling is a *heuristic* process.

This being said, there are nonetheless rules and guidelines for designing good models. The most important ideas are: (1) "identify the essential objects and their attributes" leaving away what is accidental. (2) "find a consistent abstraction". Abstraction is the ability to engage with the concept and ignore the irrelevant details, it is the faculty to extract common features from specific examples.

One of the first books that tried to describe this process of modeling was G. Polya's *How to Solve It* (1957) [9]. He decomposes the process into 4 different steps:

1. **Understanding the problem:** You need to understand the problem: What are the data? What is unknown? What is the condition? Is it sufficient, redundant, irrelevant, contradictory? Draw a figure! Introduce a suitable notation! Write it down! If you are stuck, start again!
2. **Devising a plan:** Do you know a related or a similar problem? Find the connection between the data and the unknowns! Maybe you need to design auxiliary problems and intermediary steps! Look at the unknowns! Go back to the definitions! Decompose the problem and solve the parts! Did you use all data? Did you use all conditions?
3. **Carrying out the plan:** Write it down step by step! Can you show that each step is correct?
4. **Looking back:** Examine the solution! Can you check the result or the argument? Can you derive the result differently?

Other methodologies exist. However, most of them decompose the modeling live cycle into simpler steps, as in:

- understand the problem
- describe the objective in words
- describe each constraint in words
- define the decision variables
- express the objective in terms of the decision variables
- express each constraint in terms of the decision variables
- formalize stepwise and check the result
- iterate and do not stop after the first draft.

The most important rule is not to be fixed on a single approach: if you are stuck, free your mind and try a different technique.

A small example follows to illustrate the main points. Suppose you have to solve the following problem:

"Knowing that the length plus the width of a rectangle is 27, while the area plus the difference of the length and the width is 183, find the length and width!"

One of the key steps in mathematical modeling is to find out what we are looking for. In this example, it can easily be stated: we are looking for the *length* and the *width* of the rectangle. So, we immediately introduce two symbols for them: L and W. Passing from the words to the symbols is a crucial and necessary step and should not be underestimated when building a mathematical model. One of the difficulties in modeling often stems from the fact, that we cannot easily identify the "objects" in the problem. If we have identified all essential objects, we have reached an important milestone. The rest often follows straight away.

Now, having L and W, we can easily derive from the first part of the statement that their sum is 27. Therefore, we write it down as: $\mathrm{L}+W=27$. *Write it down on a piece of paper! Do not do it mentally only! The paper is patient, you can throw it away and begin again from scratch!* The second part of the statement says, that the area plus the difference is 183. The area is $\mathrm{L}\cdot W$ and the difference is $\mathrm{L}-W$. Therefore we have $\mathrm{L}\cdot W+\mathrm{L}-W=183$. Did we use all information? Yes! Is something missing? No! Well, probably not. Can we now derive, what L and W are? Yes! So, let's see. We now have:

$$\begin{aligned}\mathrm{L}+W&=27\\ \mathrm{L}\cdot W+\mathrm{L}-W&=183\end{aligned}$$

That's the model! We are done. To find a solution, we do some algebraic transformations which is not our modeling business anymore. We leave it to a program to find the correct values for L and W. (The correct and unique solution is $\mathrm{L}=15$ and $W=12$, as you can easily verify.)

The problems here and their well polished models in the next chapters give the impression that modeling is easy and straightforward. It is not! The main goal of this text is to give you the complete solution on the modeling process, indeed. But I also try to give you a possible step by step procedure on how one could model them. Of course, this itself is an elaborated recipe on how to proceed. In reality, it was a "painful" process that was anything but a straight path. This has to be kept in mind, when you are going through these steps of modeling. Nevertheless, it can be helpful to learn techniques and methods on how to build models, on how to attack a problem. To stimulate further your capability of modeling, each "case study" contains a number of questions that you can solve yourself. The proposed answer, also given in this text, is not necessarily the "right" approach, but gives you an indication how

I did it. Maybe this also has an educational effect. Anyway, the reader can only learn something from the examples, when he goes through *all* steps and questions himself. It is better to find a wrong answer (which can be corrected afterwards) than to look up the solution right away.

The approach, I propose for the case studies, is as follows: First, the problem is presented by a short description, this is marked by a title **Problem**. The information given in this part should be enough to get started with the modeling steps.

Next follows a polished procedure of the model building steps in **Modeling Steps**. Normally, we begin by identifying the data, that means, the *sets* and the numerical *data* used in the model. The critical step is the introduction of the *variables*. The reader should be convinced that he understood this important part in the model building process. Often the introduction of the variables determine the rest of the modeling process. In the next step, we develop the conditions (the *constraints*) that hold about the variables. Finally, the goal, the *objective function*, is put forward.

The presentation of the model building process is normally given by enumerating the steps to suggest to the reader a logical and sequential order he can follow. Again the reader, nevertheless, should be aware, that this sequential ordering is only an already polished outline of the real model building process. It is itself already an abstraction of the labyrinthine process of modeling. However, I did not find a better way of presenting this process. I also wanted to break it down to the main points. To describe the "real process" – of what is really going on in the brain when modeling – is also not what is actually needed to reconstruct and comprehend the model building process. It would be a lengthy explanation of all the intricate and nebulous thoughts that one goes through while modeling. Modeling can be learnt, but not by following the connotations and tortuous brainwork that one goes through actually. It can only be learnt using case studies, if the presentation itself is a clear and unequivocal synopsis of the main steps. If the classical outline proposed in this book (data – variables – constraints) is the right one, is an open question. Surely, other methods exist.

After the presentation of the modeling steps, a short paragraph follows to layout the **solution** of the problem. This solution is normally found by running the model on the computer. Sometimes hints are given to check the result and to verify its correctness.

Finally, each case study ends with some **questions**. The reader can test his comprehension and explore model variants. The questions should stimulate him to investigate various topics.

1.3. The Modeling Language LPL

Mathematical notation is a powerful language to specify formal knowledge. The LPL modeling language is nothing else than a computer executable form for a part of that great "language" – created over the last 3000 years. A model written in LPL can directly be parsed and transformed into various other representations by a program called the *LPL interpreter*. The interpreter can also communicate with various "solvers" – other programs that solve the problem numerically, it can retrieve the solution and report it in various forms. It can also read (write) large numbers of data from (to) databases. LPL is great in prototyping large and complicated models used in a professional context. It is not only used for small problems. The design principle behind LPL always was: "Make it as concise as possible, but not more concise."

The core of the modeling system LPL is the modeling language. Each model has the following basic syntactic structure:

```
model <modelname> ;
  <statement list>
end
```

`model` and `end` are keywords and `<modelname>` is a user-defined identifier. Basically, the *statements* can be classified into *declarations* and *instructions*. They can be in any order, but the instructions are executed from top to bottom. With this respect the LPL language is like a programming language. The main declarations are: *set, parameter, variable, constraint*. The main instructions are: *solve, if, while, for*, and the *assignment*. The instructions are similar to instructions of a classical programming language.

Each of the declarations begins with a keyword and ends with a semicolon:

```
set .... ;        --declare sets, lists
parameter .... ;  --declare tables of data
variable .... ;   --declare tables of unknowns
constraint .... ; --declare tables of formulas
```

A **set** declaration is similar to the definition of a mathematical set. For example, $I, J = \{1 \ldots 3\}$ (where J is just another name of I) is a set with three elements in the classical mathematical notation. In LPL, this is formulated as follows (we prefer lowercase letters to designate sets, since in LPL we use these names also as indices, if no confusion occurs):

```
set i, j := [1..3];
```

There is one important difference between mathematical sets and sets in LPL: Sets (also called *index-sets*) in LPL are always ordered. These sets can then be used to build vectors, matrices, and higher dimensional tables of **parameters** (the data), **variables**, or **constraints**. For example, the two-dimensional table

$a_{i,j}$ with $i, j \in I$ (that is, a 3×3 data matrix)

$$a_{i,j} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

can be declared in LPL as follows (the single entries are listed in lexicographical ordering, row by row from left to right, the commas are optional):

```
parameter a{i,j} := [1 2 3 , 4 5 6 , 7 8 9];
```

The main difference between the mathematical notation and LPL's notation is that in LPL we do not need to distinguish between the set names and the index names. Of course, we could also write:

```
set I; J;                    -- set name uppercase
parameter a{i in I,j in J}; -- index names lowercase
```

to make a difference between set name and index name, but this is not necessary in most contexts. A further difference, as already mentioned, between mathematical sets and sets defined in LPL is that mathematical sets are unordered, while in LPL they are always ordered. However, this difference only becomes important when we are referencing, for example, the elements by their relative or absolute position, as in expression $i - 1$ (an expression that returns the previous element of element i).

A declaration of a vector of unknown quantities (called **variables**) is written in mathematical notation as x_j with $j \in J$. In LPL we write it as follows:

```
variable x{j};
```

If the variables are integer or binary we write:

```
integer variable x{j} [0..100];
binary variable y{i};
```

The expression `[0..100]` means that the integer variables `x{j}` are bound to the range $[0 \ldots 100]$. Binary variable can only be zero or one. This means: $y_i \in \{0, 1\}$ with $i \in I$.

Constraints are defined in the same way. A constraint vector:[2]

[2]The formula uses a very common mathematical symbol ($\sum$), which is a shortcut for a summation. Hence,

$$\sum_{j \in J} x_j \quad \text{with } J = \{1 \ldots n\} \text{ is the same as } x_1 + x_2 + \ldots + x_n$$

where n is a positive integer. It is supposed that the reader is familiar with this notation. In the context of this book and if it is clear to which set the index j is

$$\sum_{j \in J} x_j + y_i \leq 10 \quad \text{for all } i \in I$$

The constraint is written in LPL as follows:

```
constraint D{i}: sum{j} x[j] + y[i] <= 10;
```

In LPL, each constraint has a name (here `D`) and can be indexed like a variable or a parameter. If the context allows it, the indexes within the expressions can even be dropped (but this is a matter of style). So the constraint can also compactly be written as follows:

```
constraint D{i}: sum{j} x + y <= 10;
```

The solve statement is identical to the constraint, except that it begins with the keyword `maximize` or `minimize`. If there is no objective function and we are only looking for a feasible solution then in LPL we can just use the keyword `solve`.

```
maximize obj: sum{j} x;
minimize obj: sum{j} x;
solve;
```

A Simple Example

These basic syntax elements allows us to write complete mathematical models in LPL. This is illustrated by a simple example:

Suppose your company produces laptops and printers. A laptop takes 6 hours of work and generates a revenue of 300€, while a printer takes 2 hours of work and generates a revenue of 200€. Furthermore, the space in our laboratory is limited to $350m^2$ and each product takes $5m^2$ of space. We want to find the "best" production mix, that means, how many of each can be manufactured in a week, if the total hours of work is limited to 300 hours per week?

---

referring, we also often use the shorter notation

$$\sum_{j} x_j \quad \text{instead of} \quad \sum_{j \in J} x_j \quad \text{with } J = \{1 \ldots n\}$$

In concordance with the LPL syntax, we often do not introduce the symbol for the set J, we just use a index name j. So we use the notation

$$j \in \{1 \ldots n\} \quad \text{instead of} \quad j \in J\,,\ J = \{1 \ldots n\}$$

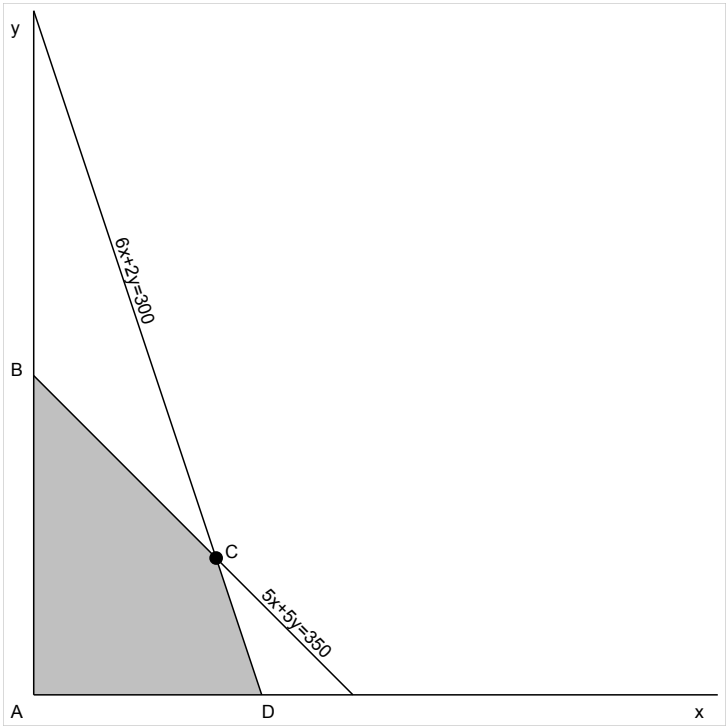

Figure 1.1: The Feasible Space

To find the optimal production mix, we introduce two symbols x and y for the (unknown) quantities for the two products. Then the production is limited to $6x + 2y \leq 300$ hours, and to $5x + 5y \leq 350$ m^2. This is best displayed by a diagram (see Figure 1.1). The points (x, y) within the quadrangle ABCD are the only points that do not violate the two requirements. The point C $= (40, 30)$ is particularly interesting, because it defines the production mix with the highest revenue, that means, $300x + 200y$ – the revenue of the mix – cannot be larger without violating either the time or the space constraint. Mathematically we say, that point C *maximizes* the expression $300x + 200y$. We can formulate the problem as a linear system in the following way:

$$\begin{array}{ll} \max & 300x + 200y \\ \text{subject to} & 6x + 2y \leq 300 \\ & 5x + 5y \leq 350 \\ & x, y \geq 0 \end{array}$$

The formulation says to maximize an expression such that the "subject to" conditions (called constraints) are fulfilled. In LPL the problem can be formulated as follows (see model tutor01):

```
model simple;
  variable x; y;
  maximize revenue: 300*x + 200*y;
  constraint A: 6*x + 2*y <= 300;
  constraint B: 5*x + 5*y <= 350;
end
```

As expected, solving the model gives $x = 40$ and $y = 30$.

The simple example is an instance of an important class of problems having a huge number of real applications. In the operations research (OR) community the class is called *linear programs* or LP. The general linear model containing n constraints and m variables can be compactly formulated as follows:

$$\begin{array}{lll} \max & \sum_{j \in J} c_j x_j & \\ \text{subject to} & \sum_{j \in J} A_{i,j} x_j \leq b_i & \text{for all } i \in I \\ & x_j \geq 0 & \text{for all } j \in J \\ \text{with} & J = \{1 \ldots m\}, \quad I = \{1 \ldots n\}, & m, n \geq 0 \end{array}$$

An even more compact formulation using matrix notation for the model is:

$$\begin{array}{ll} \max & cx \\ \text{subject to} & Ax \leq b \\ & x \geq 0 \end{array}$$

In LPL the general model can be formulated as follows:

```
model myLP;
  set i; j;
  parameter A{i,j}; c{j}; b{i};
  variable x{j};
  constraint C{i}: sum{j} A*x <= b;
  maximize obj: sum{j} c*x;
end
```

This is very close to the mathematical notation. The model first declares the two sets i and j. The data matrix and the two vectors are compactly declared as $A_{i,j}$, c_j and b_i.[3] There size will depend of the number of elements of the two sets. The variable vector x_j is also declared in a similar way. (The positivity of the variable is automatically assumed by LPL, if no range is given.) The constraints and the maximizing function are also very close to the mathematical notation. The model does not contain data. They can be added by

[3] Note – as already mentioned – in LPL syntax index names can be the same as set names if no ambiguity is created.

a separate submodel. For example, to formulate the small example above in this general form, the data can be added as :

```
model data;
  i := [1..2];   j := [1..2];
  A{i,j} := [6 2 , 5 5];
  c{j} := [300 200];
  b{i} := [300 350];
end
```

Data – we already gave an example – can be placed *within* the model code or it can be read from files using the `Read` function. The following data matrix:

$$a_{i,j} = \left\{ \begin{array}{rrrrr} 2 & 3 & -4 & 1 & 15 \\ -5 & 4 & 8 & 1 & 5 \\ 2 & 1 & 14 & 1 & 17 \\ 3 & 4 & 5 & 1 & 7 \\ 0 & 8 & -3 & 10 & 1 \end{array} \right\} \quad \text{with } i \in I, j \in J$$

could be written in LPL as:

```
set i, j := [1..5];
parameter a{i,j} := [2 3 -4 1 15,   -5 4  8  1 5,
      2 1 14 1 17,   3 4  5 1  7,    0 8 -3 10 1 ];
```

The matrix elements are listed in lexicographical order (other formats exist). Of course, for large data tables, it is better to read them from files or database tables. A comma-separated second identifier just introduces another name for the same set. Hence, `i` and `j` represent the same set, but they may be used as different index names.

Writing the results can be done using the `Writep` and `Write` functions. The simplest way to output the values of the variable vector `x{j}` is as follows;

```
Writep(x);
```

However, sometimes we need to format the output. To write each value of x_j together with the set element name on a separate line, we may write in LPL:

```
Write{j}('Element: %s Value= %6.2f \n', j,x);
```

The string in single quotes contains the literal text and formatting instructions that must be written. `%s` is a place holder for strings and `%6.2f` is a place holder for floating point data (with width 6 and 2 decimals). The values of the following two parameters `,j,x` replace the placeholders in the string. This expression means to write (for each element in the set `j`) on a separate line the following two elements, that is `(j,x)`. More information on this powerful "reporting instruction" can be found in the reference manual of LPL.

Data can also be assigned by the assignment instruction. The table assignment

$$a_{i,j} = c_j + b_i \quad \text{for all } i \in I, j \in J$$

would be formulated in LPL as follows:

```
a{i,j} :=  c[j] + b[i];
```

Another often used instruction in the presented case studies is the **for** loop. Suppose, one would like to optimize a model 10 times (with different parameters), then one could write:

```
set loop := [1..10];
for{loop} do
  maximize obj: ... ;
  ....
end
```

The loop can contain a sequence of any instructions.

The reader should now be able to read and recognize the most basic statements of LPL in order to work with the case studies in the following chapters. There is no room here to explain each detail of the language. The interested user can get a free LPL version together with the definite language reference guide and tutor examples. It can be downloaded from the LPL-site.

1.4. Mathematical Notation

The mathematical notation in this book deviates in somewhat from the common notations in three ways.

1.4.1 Set-Names versus Index-Names

The LPL syntax uses some shortcuts in mathematical notation which is reflected also in the notation of mathematical expressions used in this book. As already mentioned, one can use an set name also as an index name and vice versa. So we use the following shorter notation in LPL :

```
set i := 1..10;
set j := 1..5;
parameter b{i,j} := i*j;
parameter a{i} := sum{j} b[i,j];
```

instead of the longer notation (although this second notation is also perfectly correct in LPL) :

```
set I := 1..10;
set J := 1..5;
parameter b{i in I,j in J} := i*j;
parameter a{i in I} := sum{j in J} b[i,j];
```

This is also reflected in the mathematical notation often used in this book as :

$$\begin{aligned} b_{i,j} &= i \cdot j && \text{for all } i, j \\ a_i &= \sum_j b_{i,j} && \text{for all } i \\ i &\in \{1, \ldots, 10\} \\ j &\in \{1, \ldots, 5\} \end{aligned}$$

instead of the usual notation :

$$\begin{aligned} b_{i,j} &= i \cdot j && \text{for all } i \in I,\ j \in J \\ a_i &= \sum_{j \in J} b_{i,j} && \text{for all } i \in I \\ i &\in I && I = \{1, \ldots, 10\} \\ j &\in J && J = \{1, \ldots, 5\} \end{aligned}$$

The difference is that we do not use the set names I or J, if the context is clear, only index names i or j are written.

1.4.2 Logic-mathematical Notation

LPL does not distinguish between Boolean and mathematical operators. All Boolean operator return 0 or 1, which might be interpreted as *false* and *true*. Hence, one can mix these operators in an expression. This is also true in constraints. We may write, for example:

```
a=0 or b=1
(a=1) + (b=10)
```

The first expression is common in all kind of programming languages and is a Boolean expression that is *true* if a is zero or if b is one. The second expression seems to be a little bit strange. But on the light that all expressions are numerical, we have: $a = 1$ is zero or one, and also $b = 10$ is zero or one, so the result of the second expression is zero, one or two. This is also used in the mathematical notation and in constraints. For example, in the model will07, the constraint (with variables $d_{m,t} \in \{0, 1\}$ and $x_{m,t} \geq 0$) is used:

$$d_{m,t} \leftarrow x_{m,t} > 0$$

The expression means: "the mine is working if ore is extracted from it", or "if a certain positive quantity of ore is delivered from a mine then the variable d is defined to be true".

1.4.3 Logical Operators

LPL uses various logical and Boolean operators that can also be used in constraints, see Table 1.1. Many problems can be formulated using Boolean or logical constraints.

Meaning	Math. notation	LPL syntax
and	$x \wedge y$	`x and y`
or	$x \vee y$	`x or y`
not	$\neg x$	`~x`
not and	x nand y $(\neg(x \wedge y)$	`x nand y`
not or	x nor y $(\neg(x \vee y)$	`x nor y`
implication	$x \rightarrow y$	`x -> y`
rev. impl.	$x \leftarrow y$	`x <- y`
exclusive or	$x \dot{\vee} y$	`x xor y`
equivalence	$x \iff y$	`x <-> y`
indexed and	$\bigwedge_i x_i$	`and{i} x[i]`
indexed or	$\bigvee_i x_i$	`or{i} x[i]`
indexed xor	$\dot{\bigvee}_i x_i$	`xor{i} x[i]`
forall	$\forall_i x_i$	`forall{i} x[i]`
exists	$\exists_i x_i$	`exist{i} x[i]`
indexed nand	$\mathbf{nand}_i\, x_i$	`nand{i} x[i]`
indexed nor	$\mathbf{nor}_i\, x_i$	`nor{i} x[i]`
at least 2	$\text{atleast}(2)_i\, x_i$	`atleast(2){i} x[i]`
at most 2	$\text{atmost}(2)_i\, x_i$	`atmost(2){i} x[i]`
exactly 2	$\text{exactly}(2)_i\, x_i$	`exactly(2){i} x[i]`

Table 1.1: Logical Operators

CHAPTER 2

VARIOUS TEXTBOOK MODELS

"Approach your problem from the right end and begin with the answer. Then one day, perhaps you will find the final question."

— R. van Gulik

"It isn't that they can't see the solution. It is that they can't see the problem."

— G. K. Chesterton

"No mathematician can be a complete mathematician unless he is also something of a poet."

— Weierstrass

This book presents models from various textbooks in operations research. All models are implemented in LPL and can be executed straight away.

CHAPTER 3

BILLIONNET

3.1. A MIQP problem (bill018)

—- Run LPL Code , HTML Document –

Problem: Implement and solve the following iQP-problem (problem with a quadratic objective function and integer variables) in LPL.

$$\begin{array}{lll} \min & \sum_{i\in I}(x_i - a_i)^2 & \\ \text{subject to} & \sum_{i\in I} b_i x_i \leq K & \\ & x_i \in Z & \text{forall } i \in I = \{1\ldots5\} \end{array}$$

with $K = 50$, $a = \{30, 20, 4, 8, 16\}$, and $b = \{3, 2, 5, 7, 1\}$ (see [1] p.18).

Modeling Steps:

Listing 3.1: The Complete Model implemented in LPL [22]

```
model p9 "A MIQP problem";
  set i := [1..5];
  parameter a{i} := [30 20 4 8 16];
            b{i} := [ 3  2 5 7  1];
  integer variable x{i} [-50..50];
  constraint C1: sum{i} b*x <= 50;
  minimize obj: sum{i} (x-a)^2;
  Writep(obj,x);
end
```

3.2. Localisation Optimal (bill023)

—- Run LPL Code , HTML Document –

Problem: A company wants to build m new factories in a regions located at $m <= n$ of n potential sites to serve its new p costumers with the new product. The demand q_j of each customer are given as well as the capacity N_k in each factory, with $j \in J = \{1 \dots p\}$ and $k \in K = \{1 \dots m\}$. Since we will not generate stock the following equation must hold: $\sum_{j \in J} q_j = \sum_{k \in K} N_k$, that is, the capacity is chosen in a way to just cover all the demand.

Furthermore, the cost are given as $c_{k,i}$ for building the factory k at site i, with $k \in K = \{1 \dots m\}$ and $i \in I = \{1 \dots n\}$, as well as the distances (in km) between the site i and the costumer j, which is $d_{i,j}$. The unit cost for one km is $C = 6$ Euros.

At which sites the factories should be built in order to minimize overall costs (building costs and transportation costs)? (see [1] p.22).

Modeling Steps: We introduce a binary variable $x_{k,i}$ which is 1 if the factory k is built at site i and 0 otherwise. Furthermore, we need a variable $y_{i,j} \geq 0$ for the quantity transported from site i to costumer j in order to be able to formulate the transportation costs.

1. Now we can express the total costs: (1) the buildung costs are:

$$\sum_{k \in K, i \in I} c_{k,i} x_{k,i}$$

   (note that $x_{k,i}$ is one if a factory k is built at site i). (2) the transportation costs are:

$$\sum_{i \in I, j \in J} C d_{i,j} y_{i,j}$$

   The total should be minimal.

2. A factory k can only be built at exactly one site i, hence we have constraint

$$\sum_{i} x_{k,i} = 1 \quad \text{forall } k \in K$$

3. At each site i we only build at most one factory k, hence constraint

$$\sum_{k} x_{k,i} \leq 1 \quad \text{forall } i \in I$$

4. The demand of each customer must be exactly fulfilled:

$$\sum_{i} y_{i,j} = q_j \quad \text{forall } j \in J$$

5. At each site i the quantity produced must be the quantity transported to the different customers j (no stock):

$$\sum_{j} y_{i,j} = \sum_{k} N_k x_{k,i} \quad \text{forall } i \in I$$

Listing 3.2: The Complete Model implemented in LPL [22]

```
model p8 "Localisation Optimal";
  set i := [1..5] "locations";
      j := [1..9] "clients";
      k := [1..3] "factories";
  parameter c{k,i} "costs" := [
     2000 2500 3000 2000 2800
     1500 2000 1800 2100 1800
     2300 3000 2500 1800 2600 ];
    d{i,j} "distances" := [
      250 250 200 450 650 700 500 600 600
      300 200 550 750 750 600 350 350 200
      500 300 300 300 350 350 350 450 550
      650 400 350 200 200 350 450 550 700
      700 400 700 700 450 200 150 150 400 ];
    N{k} "capacity" := [1800 1200 2200];
    q{j} "demand" := [550 370 540 660 500 480 750 650
        700];
    C "unit costs" := 6;
  binary variable x{k,i} "factory k is built at i?";
  variable y{i,j} "quantity of product";
  constraint
    C1{k}: sum{i} x = 1;
    C2{i}: sum{k} x <= 1;
    C3{j}: sum{i} y = q;
    C4{i}: sum{j} y = sum{k} N*x;
  minimize cost: sum{k,i} 1000*c*x + sum{i,j} C*d*y;
  Writep(cost,y);
end
```

3.3. Optimal Portefolio (bill026)

—- Run LPL Code , HTML Document –

Problem: We want to invest a certain amount of money into a subset of a given number n of different titles. The set of titles is $I = \{1 \dots n\}$. Let r_i be the expected earnings in percent and v_i the variance of these earnings. (The variance is a measure of risk for a title: the larger the variance the larger the risk that the realized earning will deviate substantially from the expected earnings). We decide to invest in exactly $K < n$ titles. We suppose that the earnings of the different titles are independent from each others. Furthermore, we desire to have a yield of at least ρ. In addition, there is a minimal lo_i and maximal part up_i of the whole investment that can be placed into one single title.

What is the part of each title in the portfolio such that the total risk is minimal? (see [1] p.25ff).

Modeling Steps: We introduce a variable x_i which is the part (percentage of the whole amount) that is to be placed on title i. We also introduce a binary variable y_i saying whether the title i is used/not used in the final portfolio.

1. The sum of all invested parts must be one, this is formulated by the constraint `C1`.
2. The yield must be at least ρ, The earning on a title is the expected rate times the invested part of this title. The sum must be ρ. This is constraint `C2`.
3. The number of titles in the final portfolio is K. This gives constraint `C3`.
4. The parts must be between the minimal and maximal requirements. $lo_i y_i$ is the minimal requested amount to be placed in a title i: if $y_i = 0$ (the part of i is zero), then $lo_i y_i = 0$, if $y_i = 1$ (there *is* a positive part of i in the portfolio), then $lo_i y_i$ is lo_i. $up_i y_i$ is the maximal requested amount to be placed in a title i: if $y_i = 0$, then $up_i y_i = 0$, if $y_i = 1$ then $up_i y_i$ is up_i. Hence x_i should be between $lo_i y_i$ and $up_i y_i$: if $y_i = 0$ then $x_i = 0$ anyway, if $y_i = 1$ then x_i is between lo_i and up_i. That is exactly what is requested. The constraint is `C4`.
5. Finally, we want to minimize the risk. The total risk is measured by the sum of the (positive and negative) deviations from the expected eanrings on each title. The deviation of a single title i can be measured by $v_i x_i^2$. The quadratic term makes sure that the risk increased with an increasing (positive *or* negative deviation. Hence the sum of these risks is minimized.

Listing 3.3: The Complete Model implemented in LPL [22]

```
model p11 "Optimal Portefolio";
  set i := [1..10] "titles";
  parameter r{i} "Expected earning" := [
     5 9 3 12 5 4 8 2 4 9 ];
    v{i} "Variance" := [
      30 145 7 180 34 10 100 6 10 120 ];
    rho := 3  "Requested interest yield" ;
    K   := 4  "Number of different titles in the
       portfolio";
    lo{i}:= 0.01 "minimal proportion of a title";
    up{i}:= 1    "maximal proportion of a title";
  variable x{i} "Part attributed in the portfolio";
  binary variable y{i} "Title used/not used in the
     portfolio";
  constraint
    C1: sum{i} x = 1;
    C2: sum{i} r*x >= rho;
    C3: sum{i} y = K;
    C4{i}: lo*y <= x <= up*y;
  minimize risk: sum{i} v*x^2;
  Writep(risk,y,x);
end
```

3.4. Optimal Positioning on the Market (bill028)

— Run LPL Code , HTML Document –

Problem: Consider a number m of products and a number of potential customers n. Each product has a given number of attributes out of all p attributes. Let the set of products, customers, and attributes be $J = \{1 \dots m\}$, $I = \{1 \dots n\}$, and $K = \{1 \dots p\}$. We measure the "value" of an attribute k of product j by a positive number $d_{j,k}$. Each customer has a given concept of an "ideal" product in mind, which is expressed by the values of its attributes. These data are stored in the table $z_{i,k}$ and the costumer also assigns a given weight $w_{i,k}$ to each attribute. We suppose that each customers chooses the product that corresponds the best to his ideal. The distance between a product j and the "ideal" product for customer i is given by $\sqrt{\sum_{k \in K} w_{i,k}(d_{j,k} - z_{jk})^2}$. We denote the minimum over all products for a customer i as R_i. Hence

$$R_i = \sqrt{\sum_{k \in K} w_{i,k}(d_{j,k} - z_{jk})^2}$$

A company now wants to design a new product with the attributes such that it maximises the profit by attracting as many customer as possible supposing that the company gets a revenu c_i if the customer is attracted by the new product.

Furthermore, we suppose that the introduced product with characteristic values of x_k with $k \in K$ costs $f(x_1, x_2, \dots, x_p)$. Let this function be given by the quadratic expression as follows ($p = 5$):

$$f(x_1, x_2, \dots, x_p) = 0.6x_1^2 - 0.9x_2 - 0.5x_3 + 0.1x_4^2 + x_5$$

In addition the characteristcs of the product are only possible under the following linear constraints Φ:

$$\begin{aligned}
&x_1 - x_2 + x_3 + x_4 + x_5 \leq 10\\
&0.6x_1 - 0.9x_2 - 0.5x_3 + 0.1x_4 + x_5 \leq -0.64\\
&x_1 - x_2 + x_3 - x_4 + x_5 \geq 0.69\\
&0.157x_1 + 0.05x_2 \leq 1.5\\
&0.25x_2 + 1.05x_4 - 0.3x_5 \geq 4.5\\
&2 \leq x_1 \leq 4\\
&0 \leq x_2 \leq 8\\
&3 \leq x_3 \leq 9\\
&0 \leq x_4 \leq 5\\
&4 \leq x_5 \leq 10
\end{aligned}$$

What characteristics should this product contain and which customers will be attracted by it? (see [1] p.27ff)

Modeling Steps: We introduce a variable x_k with the characteristic "value" for attribute k and a binary variable y_i saying that the customer i is/is not attracted by the product.

1. A customer i is attracted by the product, if the product fulfills the following condition:
$$\sum_{k \in K} w_{i,k}(x_k - z_{i,k})^2 \leq R_i$$
We can formulated this condition for each customer i by the following constraints (where h is a large number, such that $\sum_{k \in K} w_{i,k}(d_{j,k} - z_{jk})^2 - h \leq R_i$):
$$\sum_{k \in K} w_{i,k}(x_k - z_{i,k})^2 - h(1 - y_i) \leq R_i$$

2. The other constraints are given by the possible charactreistics Φ.

3. We need to maximize the profit, which is the revenue ($= \sum_i c_i y_i$) and the costs given by $f(x_1, x_2, \ldots, x_p)$.

Listing 3.4: The Complete Model implemented in LPL [22]

```
model p11 "Optimal Positioning on the Market";
  set i := [1..25] "customers";
  set j := [1..10] "products";
  set k := [1..5] "attributes";
  parameter d{j,k} := [
      0.62 5.06 7.82 0.22 4.42
      5.21 2.66 9.54 5.03 8.01
      5.27 7.72 7.97 3.31 6.56
      1.02 8.89 8.77 3.10 6.66
      1.26 6.80 2.30 1.75 6.65
      3.74 9.06 9.80 3.01 9.52
      4.64 7.99 6.69 5.88 8.23
      8.35 3.79 1.19 1.96 5.88
      6.44 0.17 9.93 6.80 9.75
      6.49 1.92 0.05 4.89 6.43 ];
    z{i,k} := [
      2.26 5.15 4.03 1.74 4.74
      5.51 9.01 3.84 1.47 9.92
      4.06 1.80 0.71 9.09 8.13
      6.30 0.11 4.08 7.29 4.24
      2.81 1.65 8.08 3.99 3.51
      4.29 9.49 2.24 9.78 1.52
```

```
    9.76 3.64 6.62 3.66 9.08
    1.37 6.99 7.19 3.03 3.39
    8.89 8.29 6.05 7.48 4.09
    7.42 4.60 0.30 0.97 8.77
    1.54 7.06 0.01 1.23 3.11
    7.74 4.40 7.93 5.95 4.88
    9.94 5.21 8.58 0.13 4.57
    9.54 1.57 9.66 5.24 7.90
    7.46 8.81 1.67 6.47 1.81
    0.56 8.10 0.19 6.11 6.40
    3.86 6.68 6.42 7.29 4.66
    2.98 2.98 3.03 0.02 0.67
    3.61 7.62 1.79 7.80 9.81
    5.68 4.24 4.17 6.75 1.08
    5.48 3.74 3.34 6.22 7.94
    8.13 8.72 3.93 8.80 8.56
    1.37 0.54 1.55 5.56 5.85
    8.79 5.04 4.83 6.94 0.38
    2.66 4.19 6.49 8.04 1.66 ];
  w{i,k} := [
    9.57 2.74 9.75 3.96 8.67
    8.38 3.93 5.18 5.20 7.82
    9.81 0.04 4.21 7.38 4.11
    7.41 6.08 5.46 4.86 1.48
    9.96 9.13 2.95 8.25 3.58
    9.39 4.27 5.09 1.81 7.58
    1.88 7.20 6.65 1.74 2.86
    4.01 2.67 4.86 2.55 6.91
    4.18 1.92 2.60 7.15 2.86
    7.81 2.14 9.63 7.61 9.17
    8.96 3.47 5.49 4.73 9.43
    9.94 1.63 1.23 4.33 7.08
    0.31 5.00 0.16 2.52 3.08
    6.02 0.92 7.47 9.74 1.76
    5.06 4.52 1.89 1.22 9.05
    5.92 2.56 7.74 6.96 5.18
    6.45 1.52 0.06 5.34 8.47
    1.04 1.36 5.99 8.10 5.22
    1.40 1.35 0.59 8.58 1.21
    6.68 9.48 1.60 6.74 8.92
    1.95 0.46 2.90 1.79 0.99
    5.18 5.10 8.81 3.27 9.63
    1.47 5.71 6.95 1.42 3.49
    5.40 3.12 5.37 6.10 3.71
    6.32 0.81 6.12 6.73 7.93 ];
  c{i} := [1 .2 1 .2 .9 .9 .1 .8 1 .4 1 .3
           .1 .3 .5 .9 .8 .1 .9 1 1 1 .2 .7 .7];
  R{i} := min{j} (sum{k} w*(d-z)^2);
  h := 2000;
  lo{k} := [2 0 3 0  4];
```

```
    up{k} := [4 8 9 5 10];
  variable x{k} [lo..up] "Characteristic value of
      attribute k";
  binary variable y{i} "Customer attracted or not";
  constraint
    C1{i}: sum{k} w*(x-z)^2 - h*(1-y) <= R;
    C2: x[1] - x[2] + x[3] + x[4] + x[5] <= 10;
    C3: 0.6*x[1] -0.9*x[2] - 0.5*x[3] + 0.1*x[4] + x[5]
        <= -0.64;
    C4: x[1] - x[2] + x[3] - x[4] + x[5] >= 0.69;
    C5: 0.157*x[1] + 0.05*x[2] <= 1.5;
    C6: 0.25*x[2] + 1.05*x[4] - 0.3*x[5] >= 4.5;
  expression Fx: 0.6*x[1]^2 - 0.9*x[2] - 0.5*x[3] + 0.1*x
      [4]^2 + x[5];
  maximize profit: sum{i} c*y - Fx;
  Writep(profit,y,x);
end
```

3.5. Optimal Portefolio (bill053)

—- Run LPL Code , HTML Document –

Problem: We want to invest a certain amount of money into a subset of a given numberr n of different titles. The set of titles is $I = \{1 \dots n\}$. Let r_i be the expected earnings in percent and $v_{i,j}$ the covariance of these earnings. (The covariance is a measure of risk for a title: the larger the covariance the larger the risk that the realized earning will deviate substantially from the expected earnings). Furthermore, we desire to have a yield of at least ρ. What is the part of each title in the portfolio such that the total risk is minimal? (see [1] p.54ff). Compare this model with the model Bill26. What is the difference?

Modeling Steps: We introduce a variable x_i which is the part (percentage of the whole amount) that is t be placed on title i. We also introduce a binary variable y_i saying whether the title i is used/not used in the final portfolio.

1. The sum of all invested parts must be one, this is formulated by the constraint `C1`.
2. The yield must be at least ρ, The earning on a title is the expected rate times the invested part of this title. The sum must be ρ. This is constraint `C2`.
3. Finally, we want to minimize the risk. The total risk is measured by the sum of the (positive and negative) deviations from the expected eanrings on each title. The deviation of a single title i can be measured by $v_i x_i^2$. The quadratic term makes sure that the risk increased with an increasing (positive *or* negative deviation. Hence the sum of these risks is minimized.

Listing 3.5: The Complete Model implemented in LPL [22]

```
model p11 "Optimal Portefolio";
  set i,j := [1..5] "titles";
  parameter r{i} "Expected earning" := [
     4 2 3 3 4 ];
    v{i,j} "Variance" := [
      21.5  -17.25 -2.75  2.75 -14.75
     -17.25  56.5  -0.25  0.25  -3.5
      -2.75  -0.25  0.5  -0.5    3.25
       2.75   0.25  -.5   0.5   -3.25
     -14.75  -3.5   3.25 -3.25  49.5 ];
    rho := 3.75  "Requested interest yield" ;
  variable x{i} [0..10000] "Part attributed in the
      portfolio";
  constraint
```

```
    C1: sum{i} x = 1;
    C2: sum{i} r*x >= rho;
  minimize risk: sum{i,j} v[i,j]*x[i]*x[j];
  Writep(risk,rho,x);
end
```

3.6. Piecewise Linear Function (bill227)

—- Run LPL Code , HTML Document –

Problem: This is a small model to show the syntax of *piecewise linear function* in LPL. Let the following small problem be given:

$$\begin{array}{ll} \min & f(y) + z_1 - 2z_2 + z_3 - 2y \\ \text{subject to} & 4.5y + z_1 \leq 27.5 \\ & y + 2f(y) + 2z_1 + 2z_2 \leq 24 \\ & 1 \leq y \leq 7\ ,\ z_1 \leq 6\ ,\ z_2 \leq 4 \\ & z_1\ ,\ z_2\ ,\ z_3 \geq 0 \end{array}$$

where $f(y)$ is a non-linear function that can be approximated by a piecewise linear function defined in the interval $[1,7]$ and given by the 4 points $[y, f(y)]$ as follows: $[1,3]$, $[3,5]$, $[5,3]$, $[7,5]$. Solve the problem by transforming it into a integer linear model. The model is from [1] p.227).

Modeling Steps: Let $f(y)$ be a non-linear function approximated by $k = \{1 \ldots K\}$ points as follows:

$$[a_k, f(a_k)] \quad \text{with } 0 \leq a_1 \leq \ldots \leq a_K$$

To translate the piecewise linear function into a linear model the following steps have to be processed:

1. Add a variable $e \geq 0$ and replace each occurrence of $f(y)$ by e.
2. Add a variable vector:
$$\lambda_k \geq 0,\ k = \{1 \ldots K\}$$
3. Add the following four constraints to the model:
$$\begin{array}{l} e = \sum_k \lambda_k f(a_k) \\ y = \sum_k \lambda_k a_k \\ \sum_k \lambda_k = 1 \\ \text{Sos2}(\lambda_k) \quad \text{forall } k \in K \end{array}$$

$\text{Sos2}(< variable - list >)$ is a special constraint saying that exactly 2 consecutive variables in the list are different from zero, all others must be zero. (For more theoretical background see [13], Chapter 9.3.)

A $\text{Sos2}(\lambda_k,\ \forall k = \{1 \ldots K\})$ can be transformed into linear constraints as follows:

1. Add a binary variable vector:

$$x_k \geq 0,\ k = \{1 \ldots K-1\} \text{ with } x_k \in \{0,1\}$$

2. Add the following constraints:

$$\lambda_k \leq x_k + x_{k+1} \quad , \text{ forall } k = \{1 \ldots K-1\}$$
$$\sum_k x_k = 1$$

In LPL, the piecewise linear function is added using the << ... >> notation. LPL then translates the model autoamtically into an integer linear model:

Listing 3.6: The Complete Model implemented in LPL [22]

```
model Bill4_18 "Piecewise Linear Function";
  set k:=[1..4] "pieces";
  parameter a{k}:=[1,3,5,7];
            b{k}:=[3,5,3,5];
  variable y [1..7]; z1 [0..6]; z2 [0..4]; z3; e;
  constraint
    A: 4.5*y + z1 - z3 <= 27.5;
    B: y + 2*e + 2*z1 + 2*z2 <= 24;
    C: e = <<y, {k}(a,b)>>;
  minimize obj: e + z1 - 2*z2 + z3 -2*y;
  Writep(obj);
end
```

The constraint defining the piecewise linear function in LPL is:

```
C: e = <<y, {k}(a,b)>>;
```

This notation could also be explicitly added to the model by replacing the constraint with the following piece of code in LPL:

```
-- the following is equivalent to the <<piecewise-
    linear-code>>
variable la{k};
constraint
  C1: e = sum{k} la*b;
  C2: y = sum{k} la*a;
  C3: sum{k} la = 1;
  C4: Sos2({k} la);
```

The `Sos()` constraint (`C4`) can also be replaced by the linear constraints as follows:

```
-- the following three instructions are equivalent to
    the Sos2() constraint
binary variable x{k|k<#k};
constraint C41{k}: la <= x[k] + x[k+1];
constraint C42   : sum{k} x = 1;
```

A more involving model using piecewise linear functions is given by the model Bill320.

CHAPTER 4

GAMS

4.1. Linear Quadratic Control Problem (abel)

—- Run LPL Code , HTML Document –

Problem: Linear Quadratic Riccati Equations are solved as a General Nonlinear Programming Problem instead of the usual Matrix Recursion.

This model is from the GAMS model library (GAMS SEQ=64). See also [14]

Modeling Steps:

Listing 4.1: The Complete Model implemented in LPL [22]

```
model ABEL "Linear Quadratic Control Problem";
  set n,np "states"  :=[consumpt, invest];
      m,mp "controls":=['gov-expend', money];
      k    "horizon" :=['1964-i' '1964-ii' '1964-iii'
   '1964-iv' '1965-i' '1965-ii' '1965-iii' '1965-iv'];
  parameter
    a{n,np} "state vector matrix" := [0.914  -0.016,
        0.097  0.424];
    b{n,m}  "control vector matrix" := [0.305  0.424,
        -0.101  1.459];
    wk{n,np}"penalty matrix for states - input" :=
        [0.0625 ., . 1];
    lambda{m,mp} "penalty matrix for comtrols" := [1 .,
        . 0.444];
    c{n}  "constant term" := [-59.4,  -184.7 ];
    xinit{n} "initial value" := [387.9,  85.3 ];
    uinit{m} "initial controls" := [110.5, 147.1];
    xtilde{n,k} "desired path for x" := xinit*1.0075^(k
        -1);
    utilde{m,k} "desired path for u" := uinit*1.0075^(k
        -1);
    w{n,np,k}   "penalty matrix on states" := if(k<#k,wk[
        n,np],100*wk[n,np]);
```

```
  variable  x{n,k}  "state variable"  := xinit;
            u{m,k}  "control variable"  := uinit;
  constraint
    stateq{n,k|k<#k} "state constraint": x[n,k+1] = sum{
        np} a*x[np,k] + sum{m} b*u + c;
    bound{n}: x[n,1] = xinit;
  minimize criterion: 0.5*(sum{k,n,np}(x-xtilde)*w[n,np,k
      ]*(x[np,k]-xtilde[np,k]))
     + 0.5*(sum{k,m,mp|k<#k}(u-utilde)*lambda*(u[mp,k]-
         utilde[mp,k]));
  Writep(criterion,x,u);
end
```

4.2. Agreste Farm Level Model of NE Brazil (agreste)

—- Run LPL Code , HTML Document –

Problem: This is a farm level model of the north east region of brazil. There is only one farm type - medium sized. There are 3 types of land and risk on revenue is considered.

This model is from teh GAMS model library (GAMS SEQ=88). See also [17]

Modeling Steps:

Listing 4.2: The Complete Model implemented in LPL [22]

```
model AGRESTE "Agreste Farm Level Model of NE Brazil";
 set c       "crops" := [
             cotton_h   banana   sugar_cane  beans_arr
             beans_cor  oranges   manioc  corn sisal];
     tm  "months" := [ jan, feb, mar, apr, may, jun, jul,
          aug, sep, oct, nov, dec ];
     p   "cropping activities" := [
             crop_02       -- cotton_herbaceo
             crop_05       -- banana
             crop_10       -- sugar cane
             crop_15       -- oranges
             crop_16       -- manioc
             crop_17       -- corn
             crop_19       -- sisal
             crop_25       -- beans-de-corda   corn
             crop_29       -- cotton_herbaceo  beans-de-
                arranca corn
             crop_30       -- cotton_herbaceo  beans-de-
                corda   corn
             crop_33       -- beans-de-arranca corn
             crop_36       -- beans-de-arranca manioc corn
         ];
     s,sp  "land types" := [
       good  --flat and near water or humid and low lying
       medium  -- hilly or more arid than good
       pasture -- cultivated pasture or tree crops
                           ];
     sc{s}       "crop lands" := [ good , medium ];
     r           "livestock feeding alternatives" := [
         rec_1 rec_2 rec_3 ];
     ty           "year" := [ 1960..1969 ];
     dr           "family consumption bundle alternatives
         " := [ one , two , three ];
```

```
     km          "technology characteristics" := [
                 equipment, fertilizer, seeds, sprouts,
                     itech ];
  parameter
    landc{s}    "land data (ha)" := / good  8.775, medium
         11.64, pasture  21.92/;
    rations{r}  "lvstk rations feeding alternative (cr
        per head)":= / rec_1  5.141 , rec_2  21.646 ,
        rec_3  49.845/;
    xcropl{p,s} "limits on cropping activities" := /
        crop_19 good 0.036, crop_19 medium 0.298/;
    ldp{s,s} "land downgrading possibilities" := /
                    :good     medium:
     good             1          .
     medium          -1          1
     pasture            .       -1  /;
/* land req for lvstk seems high but ok - pg 65 para 3 */
    lio{s,r} "land requirements for lvstk feed
       alternatives (ha per head)" := /
                   :rec_1   rec_2    rec_3:
     medium         1.407   0.611    0.631
     pasture        0.209   2.03     0.9  /;
    labor{p,tm}  "labor requirements for cropping (mandays
        per ha)" := /
                :jan   feb   mar   apr   may   jun   jul
                   aug   sep   oct nov  dec:
      crop_02  4.79 10.89 18.70  9.83  8.03 20.95 15.67
          15.48 11.35  8.38 8.23 4.54
      crop_05  8.24  0.89  5.43  3.31 11.13 10.33  3.27
          2.65  5.47  3.67 3.18 1.44
      crop_10  5.78  3.37  4.03  7.09  4.64  5.91  7.67
          9.23  5.93  9.05 7.59 6.37
      crop_15  2.00  2.36  4.13  2.13  7.18  5.88  3.21
          9.20  6.89  3.30 9.70 8.87
      crop_16  5.71  7.60  7.28  8.15  8.48 10.71  9.34
          12.76  8.93 10.22 6.18 8.34
      crop_17  9.19 14.42 10.59 12.01  3.47 10.75  5.88
          7.30  4.14  4.25 3.49 2.11
      crop_19  1.22  1.11  0.56  0.78  1.19  0.44  1.89
          3.04  2.93  4.99 7.73 9.44
      crop_25 12.46 11.35 24.38 22.03 16.91  9.47 10.97
          9.70  4.93  6.41 0.53 0.89
      crop_29  5.54 10.30  9.24  8.97 15.82 13.00 14.74
          10.64  5.64  4.90 4.73 4.92
      crop_30  9.18  3.94 16.15 23.69 28.12 20.58 17.73
          15.06  4.58  9.76 7.39 2.67
      crop_33  6.87 10.19 10.61  9.14 18.38 11.50  9.22
          13.36  3.85  3.87 2.65 1.09
      crop_36  7.44 10.21  9.63 16.18 20.11 16.86 14.86
          14.00  8.04  6.69 5.49 5.26 /;
```

```
 llab{tm,r} "labor requirement for lvstk feed (mandays
    per head)" := /
                    :rec_1       rec_2       rec_3:
   jan          4.261       1.873       1.933
   feb          2.854       1.262       1.302
   mar          0.040       0.040       0.040
   apr          0.040       0.040       0.040
   may          0.040       0.040       0.040
   jun          0.040       0.040       0.040
   jul          0.040       0.040       0.040
   aug          0.040       0.040       0.040
   sep          0.040       0.040       0.040
   oct          0.040       0.040       0.040
   nov          7.075       3.095       3.195
   dec          7.075       3.095       3.195/;
cbndl{c,dr}  "consumption bundles (tons per bundle)" :=
    /
                    :one        two       three:
   beans_arr      0.225      0.152      0.15
   manioc         0.965      2.64       0.935
   corn           0.235      0.232      0.581 /;
crev{c,ty}  "crop revenue time series (cr per ha)" := /
            :1960 1961 1962 1963 1964 1965 1966 1967
               1968 1969:
  banana     6399 8193 9215 8581 7988 7228 5923 6738
     7221 7842
  sugar_cane 1667 1899 1898 2230 2265 2067 1499 1903
     1901 1929
  beans_cor   546  585 1091  651  470  621  605  518
     519  772
  manioc      986 1543 2658 1639 1146 1380 1357 1702
     1560 1580
  corn        267  378  544  329  320  351  310  322
     294  349 /;
price{c} "crop reference prices (cr per kg)" := /
  cotton_h  1.9 banana    4   sugar_cane 35
  beans_arr 2.5 beans_cor 1.5 oranges    10
  manioc   0.17 corn     0.15 sisal        1 /;
 /* wage data: pg 159, working capital: is this correct
    : =6.4*1113, family labor available: correct? */
fwage "family reservation wage rate (cr/man-month)"
       :=  75;
twage  "temporary labor wage rate    (cr/man-month)" :=
    250;
pwage  "permanent labor wage rate     (cr/man-year)"
   :=2054;
vsc    "value of self-consumption       (cr/bundle)" :=
    934;
wcbar  "working capital                        (cr)"
   :=7123.2;
```

```
famlab "family labor           man equivalent wrks)" :=
     70.5;
lprice "livestock price              (cr per head)" :=
    211;
vetpr  "cost of vetinanry servic          per head)" :=
     1;
dpm    "man days per man month"                      :=
    25;
/* some of these yields look odd: medium land has
   higher land than good */
yield{p,c,s}  "crop yields (kg per ha)" := /
                         :good    medium:
   crop_02 cotton_h       848     569
   crop_05 banana         221     174
   crop_10 sugar_cane      45      30
   crop_15 oranges         92       .
   crop_16 manioc        4456    3964
   crop_17 corn           725     563
   crop_19 sisal         2244    1666
   crop_25 beans_cor      251     211
   crop_25 corn           373     264
   crop_29 cotton_h       269     149
   crop_29 beans_arr      285     221
   crop_29 corn           536     544
   crop_30 cotton_h       403     133
   crop_30 beans_cor      115     352
   crop_30 corn           361     212
   crop_33 beans_arr      274     260
   crop_33 corn           594     442
   crop_36 beans_arr      288     287
   crop_36 manioc        3408    1031
   crop_36 corn           503     328   /;
set ps{p,s} "process-land possibilities" := (sum{c}
   yield > 0);
parameter
  ravg{c} "average crop revenues (cr per ha)" := (sum{
     ty} crev)/#ty;
  prdev{c,ty|ravg} "price deviations for crops (cr per
     ton)" := 1000*price*(crev/ravg - 1);
  phi    "risk factor" := 1;
  techc{p,km} "cropping technology rquirements (cr per
     ha)" := /
            :equipment  fertilizer  seeds  sprouts:
   crop_02         .          31       91        .
   crop_05         .           .        .       45
   crop_10         .          36        .        .
   crop_15         .         106        1      184
   crop_16         .          20        .        .
   crop_17         .          42       55        .
   crop_29         4          22        .       19
```

```
    crop_30        .            .        27        .
    crop_33       11           42         .        .
    crop_36        .           98         6        1  /;
 pcost{p}   "cropping technology costs (cr per ha)" :=
     sum{km} techc;
 a{p}       "cropping constants" := if (sum{s} yield[p
     ,'cotton_h',s]=0,1);
variable
  xcrop{p,s}   "cropping activities (ha)";
  xliver{r}    "livestock activity defined on feed
      techniques (head)";
  lswitch{s}   "land downgrading (ha)";
  sales{c}     "crop sales (ton)";
  cons{dr}     "on-farm consumption (ton)";
  flab{tm} [0..famlab]"family labor (man-days)";
  tlab{tm}     "temporary labor (man-days)";
  plab         "permanent labor (workers)";
  pdev{ty}     "positive price deviations (cr)";
  ndev{ty}     "negative price deviations (cr)";
  xlive        "livestock production (head)";
  xprod{c}     "crop production (ton)";
  rationr      "livestock ration requirements (cr)";
  revenue      "from crop and livestock sales (cr)";
  cropcost     "accounting: cropping activities cost (
      cr)";
  labcost      "accounting: labor costs - including
      family (cr)";
  vetcost      "accounting: vetinary services cost (cr)
      ";
constraint
  landb{s}: sum{p|ps and sc} a*xcrop + sum{sp} ldp[sp]*
      lswitch[sp] + sum{r} lio*xliver <= landc "land
      balance";
  lbal    : xlive = sum{r} xliver  "livestock balance";
  rliv    : rationr = sum{r} rations*xliver  "livestock
       ration requirements definition";
  mbalc{c}: sum{s,p} yield*xcrop/1000 >= sales + sum{dr
      } cbndl*cons  "material balance: crops";
  dprod{c}: xprod = sum{p,s} yield*xcrop/1000  "crop
      production definition";
  labc{tm}: sum{p,s|ps} labor*xcrop + sum{r} llab*
      xliver <= flab + tlab + dpm*plab  "labor supply-
      demand relation";
  cond    : sum{dr} cons = 1  "on farm consumption
      definition";
  ddev{ty}: sum{c} prdev*sales = pdev-ndev  "crop price
       deviation definition";
  arev    : revenue = lprice*xlive + 1000*(sum{c} price
      *sales) "accounting: revenue definition";
  acrop   : cropcost = sum{p,s|ps} pcost*xcrop  "
```

```
        accounting: cropping cost definition";
    alab    : labcost = (fwage*(sum{tm} flab) + twage*(
        sum{tm} tlab))/dpm + pwage*plab "accounting:
        labor cost definition";
    awcc    : cropcost + rationr + vetcost + twage/dpm*(
        sum{tm} tlab) + pwage*plab <= wcbar "accounting:
        working capital requirements";
    avet    : vetcost = vetpr*xlive   "accounting:
        vetinary costs";
    bound{p,s|xcropl}: xcrop <= xcropl;
    income frozen: revenue + vsc*(sum{dr} cons)-labcost-
        rationr-vetcost
            -cropcost-phi*(sum{ty} (pdev+ndev)/#ty) "farm
                income definition";
  maximize obj1: income;
  Writep(income);
  phi := 0;
  maximize obj2: income;
  Writep(income);
end
```

4.3. Ajax Paper Company Production Schedule (ajax)

—- Run LPL Code , HTML Document –

Problem: This sample model is taken from the cybernet pds/apex sample library of models. A paper manufacturer can produce four different types of paper on three different machines. Given a fixed demand schedule the objective is to find a production plan that maximizes monthly profits.

This model is from the GAMS model library (GAMS SEQ=60). See also CDC, PDS/APEX Sample Model Library, 1977. Control Data Corporation.

Modeling Steps:

Listing 4.3: The Complete Model implemented in LPL [22]

```
model AJAX "Ajax Paper Company Production Schedule";
 set    m      "machines at mills" := [ machine_1,
     machine_2, machine_3 ];
        g      "paper grades" := [ '20-bond-wt', '25-bond-
            wt', 'c-bond-ext', 'tissue-wrp' ];
 parameter prate{g,m}  "production rate (tons/hour)" := [
    53  52  49 , 51  49  44 , 52  45  47 , 42  44  40];
  pcost{g,m}  "production cost ($ per ton)" := [
    76  75  73 , 82  80  78 , 96  95  92 , 72  71  70];
  demand{g} := [30000 20000 12000  8000];
  price{g} := [77  81  99  105];
  avail{m} "available machine time (hours/month)" := [672
       600 480];
 variable   outp{g,m}    "production  (tons/month)";
 constraint
   Cap{m}:  sum{g} outp/prate <= avail;
   Dem{g}:  sum{m} outp = demand;
 maximize profit: sum{g} demand*price - sum{g,m} pcost*
     outp;
  Writep(profit);
  Write('machine time report\n            avail-h  used-h
       unused-h  marginal\n');
  Write{m}('%10s  %7.1f  %6.1f  %8.2f  %8.2f\n', m,avail,
      Cap,avail-Cap,- GetValue(Cap,3));
  Write('production allocation report\n
      demand   sold    unsold    marginal\n');
  Write{g}('%10s  %7.1f  %6.1f  %8.2f  %8.2f\n', g,demand
      ,Dem,demand-Dem, GetValue(Dem,3));
end
```

4.4. Sample Problem (AMPL) (ampl)

— Run LPL Code , HTML Document –

Problem: A sample problem to demonstrate the power of modeling systems (compares GAMS with AMPL).

This model is from teh GAMS model library (GAMS SEQ=74). See also [8].

Modeling Steps:

Listing 4.4: The Complete Model implemented in LPL [22]

```
model AMPL "Sample Problem (AMPL)";
  set  p  "products"  := [ nuts, bolts, washers ];
       r  "raw materials" := [ iron, nickel ];
       t  "periods" := [ 1..5 ];
  parameter
    b{r} "initial stock" := /iron 35.8 , nickel 7.32/;
    d{r} "storage cost" := /iron   0.03, nickel 0.025/;
    f{r} "residual value" :=/iron  0.02, nickel -0.01/;
    m  "maximum production" :=  123;
    a{r,p} "raw material inputs to produce a unit of
        product" := [
       0.79   0.83   0.92
       0.21   0.17   0.08];
    c{p,t} "profit" := [
         1.73  1.8  1.6  2.2  .
         1.82  1.9  1.7  0.95 .
         1.05  1.1  0.95 1.33 .];
  variable
    x{p,t|t<#t} "production level";
    s{r,t} "storage at beginning of period";
  constraint
    limit{t} "capacity constraint":  sum{p} x <= m;
    balance{r,t|t>1} "raw material balance":  s = s[t-1]
        - sum{p} a*x[p,t-1];
    sUpper{r}: s[r,1] <= b;
  maximize profit: sum{p,t}c*x + sum{r,t}if(t=#t,f,-d)*s;
  Writep(profit);
end
```

4.5. Bid Evaluation (bid)

—- Run LPL Code , HTML Document –

Problem: A company obtains a number of bids from vendors for the supply of a specified number of units of an item. Most of the submitted bids have prices that depend on the volume of business.

This model is from the GAMS model library (GAMS SEQ=19). See also [3], pp. 28–36.

Modeling Steps:

Listing 4.5: The Complete Model implemented in LPL [22]

```
model BID "Bid Evaluation";
  set v       "vendors"  := [ a, b, c, d, e ];
      s       "segments" := [ 1..5 ];
      vs{v,s} "vendor bit possibilities";
      cl  "column labels" :=[setup, price, q_min, q_max];
  parameter req "requirements" :=  239600.48;
    bid{v,s,cl}  "bid data";
  Read{v,s,cl}('bid.txt', v,s,cl,bid);
  /*  get minimum domains and ripple total cost up the
      segments */
  {v,s}(vs:=bid[v,s,'q_max']),
      for{vs[v,s]|s>1} (bid[v,s,'setup'] := bid[v,s-1,'
          setup']
        + bid[v,s-1,'q_max']*(bid[v,s-1,'price']
        -bid[v,s,'price']));
  variable pl{v,s}    "purchase level";
  binary   plb{v,s}   "purchase decision";
  constraint
    demand: req = sum{vs[v,s]} pl   "Demand constraint";
    mpl{vs[v,s]}:bid[v,s,'q_min']*plb <= pl <= bid[v,s,'
        q_max']*plb;
    oneonly{v} : sum{s|vs} plb <= 1 "At most one deal";
  minimize cost:
    sum{vs[v,s]} (bid[v,s,'price']*pl + bid[v,s,'setup']*
        plb);
  Writep(cost,pl,plb);
end
```

4.6. Bid Evaluation (bid1)

—- Run LPL Code , HTML Document –

Problem: A company obtains a number of bids from vendors for the supply of a specified number of units of an item. Most of the submitted bids have prices that depend on the volume of business.

This model is from the GAMS model library (GAMS SEQ=19). See also [3], pp.28–36.

Modeling Steps:

Listing 4.6: The Complete Model implemented in LPL [22]

```
model BID "Bid Evaluation";
  set v       "vendors" := [ a, b, c, d, e ];
      s       "segments" := [ 1..5 ];
      vs{v,s} "vendor bit possibilities";
      cl "column labels" :=[setup, price, q_min, q_max];
  parameter req "requirements" :=  239600.48;
    bid{v,s,cl}  "bid data" := /
                      :setup      price     q_min     q_max:
      a 1           3855.84     61.150         .     33000
      b 1         125804.84     68.099     22000     70000
      b 2                .      66.049     70000    100000
      b 3                .      64.099    100000    150000
      b 4                .      62.119    150000    160000
      c 1          13456.00     62.190         .    165600
      d 1           6583.98     72.488         .     12000
      e 1                .      70.150         .     42000
      e 2                .      68.150     42000
          77000/;
  /*  get minimum domains and ripple total cost up the
      segments */
  {v,s}(vs:=bid[v,s,'q_max']),
      for{vs[v,s]|s>1} (bid[v,s,'setup'] := bid[v,s-1,'
          setup']
        + bid[v,s-1,'q_max']*(bid[v,s-1,'price']
        -bid[v,s,'price']));
  variable
    pl{v,s}  "purchase level";
    binary plb{v,s} "purchase decision";
  constraint
    demand "demand constraint": req = sum{vs[v,s]} pl;
    minpl{vs[v,s]} "min purchase": pl >= bid[vs,'q_min']*
        plb;
    maxpl{vs[v,s]} "max purchase": pl <= bid[vs,'q_max']*
        plb;
```

```
    oneonly{v} "at most one deal":  sum{s|vs} plb <= 1;
  minimize cost:
    sum{vs[v,s]} (bid[vs,'price']*pl + bid[vs,'setup']*
        plb);
  Writep(cost,pl,plb);
end
```

4.7. Blending Problem I (blend)

— Run LPL Code , HTML Document –

Problem: The problem is to blend a new alloy from other purchased alloys.

The model is from The GAMS model library (GAMS SEQ=2). See also [10], chap 3.4.

Modeling Steps:

Listing 4.7: The Complete Model implemented in LPL [22]

```
model BLEND "Blending Problem I";
  set  alloy :=[a b c d e f g h i]   "products on the
      market";
       elem  :=[lead zinc tin]   "required elements";
  parameter  compdat{elem,alloy}  := [
          10   10   40   60   30   30   30   50   20
          10   30   50   30   30   40   20   40   30
          80   60   10   10   40   30   50   10   50];
    price{alloy}:=[4.1 4.3 5.8 6.0 7.6 7.5 7.3 6.9 7.3];
    rb{elem}:=/lead 30 zinc 30 tin 40/ "required blend";
    ce{alloy}:=sum{elem} compdat-100 "composition error (
        pct-100)";
  variable  v{alloy} "purchase of alloy (pounds)";
  constraint
    pc{elem}:  sum{alloy} compdat*v = rb "purchase
        constraint";
    mb frozen: sum{alloy} v = 1  "material balance";
    ac frozen: sum{alloy} price*v "accounting: total cost
        ";
  minimize cost1: ac;
  Writep(ac,v);
  Unfreeze(mb);
  minimize cost2: ac;
  Writep(ac,v);
end
```

4.8. Hanging Chain COPS 2.0 3 (chain1)

—- Run LPL Code , HTML Document –

Problem: Find the chain (of uniform density) of length L suspended between two points with minimal potential energy. This model is from the COPS benchmarking suite. See http://www-unix.mcs.anl.gov/ more/cops/. COPS performance tests have been reported for nh = 50, 100, 200, 400.

Taken from the GAMS model library (GAMS SEQ=231). See also [6] and [18].

Modeling Steps:

Listing 4.8: The Complete Model implemented in LPL [22]

```
model CHAIN "Hanging Chain COPS 2.0 3";
//this model is not correctly formulated
  set nh ,i "number of intervals for the discretization"
      := 1..50;
  parameter
    L  "length of the suspended chain" :=  4;
    a  "height of the chain at t=0 (left)" :=  1;
    b  "height of the chain at t=1 (left)" :=  3;
    tf "odes defined in [0 tf]"        :=  1;
    n  "number of subintervals" := #nh-1;
    h  "uniform interval length" := tf/n;
    tmin :=if(b>a,0.25,0.75);
  variable
    x{i}   "height of the chain" :=
      4*Abs(b-a)*((i-1)/n)*(0.5*((i-1)/n) - tmin) + a;
    u{i}   "derivative of x" := 4*Abs(b-a)*(((i-1)/n) -
       tmin);
  constraint
    x_eqn{i|i+1}: x = x[i-1] + 0.5*h*(u[i-1]+u);
    length_eqn:
      0.5*h*(sum{i|i>1} (Sqrt(1+u[i-1]*u[i-1]) + Sqrt(1+u
         *u))) = L;
    xfx1: x[1] = a;
    xfx2: x[#nh] = b;
  minimize energy "potential energy" :
    0.5*h*(sum{i|i>1} (x[i-1]*Sqrt(1+u[i-1]*u[i-1]) + x*
       Sqrt(1+u*u)));
  Writep(energy,x,u);
end
```

4.9. Chance Constraint Feed Mix Problem (chance)

— Run LPL Code , HTML Document –

Problem: formulated, a deterministic model and the chance constraint version deterministic equivalent.

This model is from the GAMS model library (GAMS SEQ=26). See also [3] Chap 9, pp.94–100.

Modeling Steps:

Listing 4.9: The Complete Model implemented in LPL [22]

```
model CHANCE "Chance Constraint Feed Mix Problem";
  set  f  "feeds" := [ barley, oats, sesame, grnd_meal ];
       n  "nutrients" := [ protein, fats ];
  parameter   price{f}  "feed prices (fgld per ton)" :=
      /barley 24.55 oats 26.75 sesame 39.00 grnd_meal
          40.50/;
    req{n} "requirements (pct)" :=/protein 21 fats 5/;
    mean{n,f} "feed characteristics (pct)" := [
                12.0      11.9      41.8       52.1
                 2.3       5.6      11.1        1.3 ];
    variance{f} "for protein" :=[0.28 0.19 20.5 0.62];
  variable x{f} "feed mix (pct)";
  expression cost: sum{f} price*x "total cost per ton";
  constraint
    mc "mix constraint": sum{f} x = 1;
    nbal{n} "nutrient balance": sum{f} mean*x >= req;
    cc{n} "chance constraint":
      sum{f} mean*x - 1.645*Sqrt(sum{f} variance*x^2) >=
          req;
  minimize obj2: cost subject_to mc,cc;
  Writep(cost,x);
  minimize obj1: cost subject_to mc,nbal;
  Writep(cost,x);
end
```

4.10. Organic Fertilizer Use in Intensive Farming (china)

—- Run LPL Code , HTML Document –

Problem: Models the use of organic and chemical fertilizers for triple-cropping in the southern Jiangsu province. Detailed attention is paid to removal and replenishment of plant nutrients and humus.

This model is from the GAMS model library (GAMS SEQ=56). See also [23]

Modeling Steps:

Listing 4.10: The Complete Model implemented in LPL [22]

```
model CHINA "Organic Fertilizer Use in Intensive Farming"
    ;
  set ca "all commodities" := [
             barley
             wheat
             e_rice     -- early rice
             m_rice     -- middle rice
             l_rice     -- late rice
             l_sc_rice -- late single crop rice
             g_manure   -- green manure
             gm_seeds   -- green manure seeds
             rapeseed
             azolla
             azolla_e   -- early azolla
             fodder
             g_feed     -- grain feed
             straw
             hyacinth
             silt
             nightsoil
             amm_water -- ammonia water
             amm_bi     -- ammonium bicarbonate
             ssp        -- single superphosphate
             pigs
             shoats
             c_straw    -- composted straw
             c_gm       -- composted green manure
             c_hyacinth -- composted hyacinth
             pig_m      -- pig-manure
             straw_b    -- straw bedding
             rapes_c    -- composted rapeseed
             vegetables                    ];
      c{ca}  "crops" := [ barley, wheat, e_rice, m_rice,
          l_rice, l_sc_rice, g_manure
```

```
        gm_seeds, rapeseed, azolla, azolla_e, fodder ];
      g{ca}  "grains"  := [ barley, wheat, e_rice, m_rice
          , l_rice, l_sc_rice ];
      cu{ca} "upland crops"    := [ fodder, vegetables ];
      cp{ca} "commodities purchased" := [ hyacinth, pig_m
          , silt, nightsoil, amm_water, amm_bi, ssp ];
      cs{ca} "commodities sold" := [ barley, wheat,
          e_rice, m_rice, l_rice, l_sc_rice
           rapeseed, pigs, shoats, straw, vegetables ];
      cf{ca} "fertilizers" := [ c_straw, c_gm, c_hyacinth
          ,  pig_m, straw_b, nightsoil
         azolla, rapes_c, amm_water, amm_bi, ssp ];
      s      "crop sequence" := [ bar_r_r, bar_sr, bar_r,
           wh_sr, wh_r, gm_r_r, gm_sr, gm_s_sr, gm_r,
          rape_sr, fallow_sr ];
      sh{s}  "sequences with higher fertilizer
          application" := [ bar_r_r, gm_r_r, wh_r, bar_r
          ];
      nh     "nutrients and humus" := [ n, p2o5, k2o,
          humus ];
      n{nh}  "nutrients"           := [ n, p2o5, k2o ];
      en     "effective nutrients" := [ n_imm, n_tot,
          p2o5_eff, k2o_exch, humus ];
      f      "fertilization intensity" := [ normal, high
          ];
      p      "pig raising activities"  := [ r_shoats,
          r_pigs  ];
      ss{s,f} "sequence possibilties"  := if(f='normal'
          or sh,1);
      t      "time periods" := [ nov_1, nov_mar, mar_2,
          apr_1, apr_2, may_1, may_jun, jun_2, jul_1,
          jul_2
            aug_1, aug_2, sep, oct_1, oct_2 ];
      s1; s2; s3; s4;
parameter
  paddy  "paddy land available  (mu)" :=  268.6;
  upland  "upland avaialable (mu)" :=  7.8;
  muperha   "conversion of mu to hectars (mu per hectar)
      " :=  15;
  jinperkg  "conversion of jin to kg (jin per kg)" :=
      2.0;
  yperd     "exchange rate (yuan per dollar)" :=  1.8;
  mcp{s,c}  "multi cropping patterns for paddyland (mu)"
       := /
                 :barley wheat e_rice m_rice l_rice
                     l_sc_rice g_manure gm_seeds rapeseed
                       azolla  azolla_e  fodder:
       bar_r_r    0.704 0.193  0.807     .      1.0
           .        .        .         .        .
           .       .
```

```
      gm_r_r          .  0.193   .807      .       1.0
          .      0.704        .          .         .
          .        .
      bar_sr      0.859      .      .       .       .
          1.0      0.141        .          .       1.0
          0.859         .
      wh_sr           .  0.859      .       .       .
          1.0      0.141        .          .       1.0
                  .         .
      gm_sr           .       .      .       .       .
          1.0      1.0          .          .       1.0
          0.859         .
      rape_sr         .       .      .       .       .
          1.0      0.141        .   0.859          1.0
                  .         .
      fallow_sr       .       .      .       .       .
          1.0      0.141        .          .         .
                  .    0.859
      gm_s_sr         .       .      .       .       .
          1.0      0.1       0.9           .         .
                  .         .
      bar_r       0.9         .      .      1.0      .
          .      0.1           .          .       1.0
          .        .
      wh_r            .   0.9       .      1.0      .
          .      0.1           .          .         .
          .        .
      gm_r            .       .      .      1.0      .
          .      1.0           .          .       1.0
          .        .  /;
  cdata{ca,s1}  "crop data" := /
   /* yield     : paddy land yield in jin per mu
      cash_cost : yuan per mu
      straw_y   : kg per kg output
      proc_price: yuan per ton
      quota_sale: tons */
                :yield cash_cost straw_y proc_price
                   quota_sale:
      barley     427.6  11.0       1.0      226.0
          .
      wheat      530.3  11.0       1.1      316.0
          .
      e_rice     690.3  14.9       0.8      232.0
          .
      m_rice     962.7  23.6       1.0      238.0
          .
      l_rice     558.2  18.9       0.8      272.0
          .
      l_sc_rice  857.7  25.2       1.0      233.0
          .
```

```
    g_manure  4968.1   2.3        .          .
        .
    rapeseed   221.1   9.8       1.5      720.0
               4
    azolla    1149.0    .         .          .
        .
    azolla_e  1149.0    .         .          .
        .
    fodder    1200      .         .          .
        .
    straw        .      .         .         44.0
        1.2
    pigs         .      .         .       1200.0
        4
    shoats       .      .         .       1800.0
               .
    vegetables   .      .         .        375.0
                 .  /;
yield{ca}    "paddy land crop yields (ton per mu)" :=
    cdata[ca,'yield']/jinperkg/1000;
yieldu{ca}   "upland yields (ton per mu)" := / fodder
     0.3, vegetables  0.24 /;
cxcrop{s}    "cash cost by sequence (yuan per mu)"  :=
     sum{c} cdata[c,'cash_cost']*mcp;
aqsprice{ca} "above quota sales price (yuan per ton)"
    := if(ca within c,1.5,1.1)*cdata[ca,'proc_price'];
grainquota   "grain sales quota (tons)" :=  100;
purdata{ca,s2}  "purchase prices and limits" := /
                  :price       quantity:
   /*            (yuan per ton)   (tons) */
    amm_water       30              .
    amm_bi         123              .
    nightsoil       20              7
    ssp            100            500
    pig_m           12          10000
    hyacinth             .         15    /;
pigio{ca,p} "pig raising input output relations" := /
                :r_shoats  r_pigs:
    shoats           1       -0.12
    pigs             .        1
    straw_b          8        8
    g_feed        -3.3     -2.6
    straw         -8.8     -8.8
    fodder        -0.55    -0.50 /;
cxpig{p}  "pig raising cashcost (yuan per ton)" := /
    r_shoats 90,  r_pigs 60 /;
gio{ca,ca1 in  ca|g[ca1]} "grain feed mixing recipes (
    kg per kg)" := if(ca=ca1,-1, ca='g_feed',1, 0);
/* labor use and supply */
lu{t,c}  "labor use for crops (gongs per mu per period
```

```
      )" := /
            :barley wheat e_rice m_rice l_rice l_sc_rice
                 g_manure gm_seeds rapeseed:
      nov_1     8.15  8.15     .      .     3.17     3.00
                 .        .   14.90
      nov_mar   7.45  7.45   7.75   5.57   6.33    13.07
                1.00    1.00  15.15
      mar_2     1.10  1.10   2.03   0.74     .       .74
                 .        .    1.85
      apr_1     0.4   0.4    3.25   0.70     .       .70
                 .        .       .
      apr_2     0.9   0.9    3.36   0.70     .       .70
                 .        .    1.0
      may_1     0.2   0.2    8.20   2.92     .       4.42
                3.00      .    0.9
      may_jun  10.80  8.80  13.58  12.12   1.50     5.27
                 .      6.70   8.40
      jun_2      .    2.00  2.35    4.40   3.00     8.99
                 .        .       .
      jul_1      .       .    2.48   5.41   3.00     5.00
                 .        .       .
      jul_2      .       .    0.70   5.59   3.00     4.50
                 .        .       .
      aug_1      .       .    8.42   1.00   9.35     1.42
                 .        .       .
      aug_2      .       .    1.88   3.65   4.92     4.54
                 .        .       .
      sep       1.30  1.30     .     3.20   5.09     4.15
                 .        .    1.00
      oct_1     1.30  1.30     .     6.13   1.52      .
                 1.25    1.25   0.50
      oct_2     1.25  1.25     .     4.37   0.12      .
                 1.0     1.00   0.30 /;
labj{t}    "labor adjustment coefficient (proportion)"
     := / aug_1 0.12 /;
days{t}    "number of days in periods (days)" := /
     nov_1 22, nov_mar 120, mar_2 16, apr_1 15 ,apr_2
         15, may_1 14, may_jun 32, jun_2 15
     jul_1 15, jul_2 12, aug_1 14, aug_2  21, sep 30,
         oct_1 14, oct_2 10 /;
tday  "total number of days in all periods  (days)" :=
     sum{t} days;
lsup  "labor supply  (gongs per day)" :=  137;
dlab{t,c}  "daily labor requiremnts for single crop (
    gongs per day per mu)" := lu/days;
lab{t,s}  "labor requirements for crop sequences (
    gongs per day per mu)" := sum{c} dlab*mcp;
/* fertilizer requiremets and supply */
nc{cf,nh} "nutrient composition (pct weight)" := /
                  :n      p2o5     k2o   humus:
```

```
    c_straw      0.63      0.11   0.85    21.4
    c_gm         0.40      0.11   0.35     2.34
    c_hyacinth   0.24      0.07   0.11     2.64
    pig_m        0.35      0.32   0.75      .
    straw_b      0.60      0.20   0.80    15.9
    azolla       0.34      0.02   0.12     2.58
    nightsoil    0.65      0.30   0.25      .
    rapes_c      4.60      2.50   1.40      .
    amm_water    3.75         .      .      .
    amm_bi      17.00         .      .      .
    ssp             .     15.00      .      . /;
nu{cf,nh,en}  "fertilizer utilization rates (pct)" :=
    /
    [c_straw,*,*]    n n_imm 10  n n_tot 76  p2o5
       p2o5_eff 20  k2o k2o_exch 14  humus humus 100
    [c_gm,*,*]       n n_imm 10  n n_tot 76  p2o5
       p2o5_eff 20  k2o k2o_exch 14  humus humus 100
    [c_hyacinth,*,*] n n_imm 10  n n_tot 76  p2o5
       p2o5_eff 20  k2o k2o_exch 14  humus humus 100
    [pig_m,*,*]      n n_imm  9  n n_tot 76  p2o5
       p2o5_eff  8  k2o k2o_exch 16  humus humus 100
    [straw_b,*,*]    n n_imm  9  n n_tot 76  p2o5
       p2o5_eff  8  k2o k2o_exch 16  humus humus 100
    [nightsoil,*,*]  n n_imm  9  n n_tot 76  p2o5
       p2o5_eff  8  k2o k2o_exch 16  humus humus 100
    [azolla,*,*]     n n_imm 20  n n_tot 76  p2o5
       p2o5_eff 20  k2o k2o_exch 14  humus humus 100
    [rapes_c,*,*]    n n_imm 10  n n_tot 76  p2o5
       p2o5_eff 20  k2o k2o_exch 14
    [amm_water,*,*]  n n_imm 20  n n_tot 40
    [amm_bi,*,*]     n n_imm 22  n n_tot 45
    [ssp,*,*]        p2o5 p2o5_eff 25 /;
 pno{c,n}  "plant nutrient offtake (pct weight of
     output)" := /
                 :n   p2o5  k2o:
    barley       3.0   1.0  2.0
    wheat        2.75  1.2  2.4
    e_rice       1.9   0.8  2.7
    m_rice       2.1   1.0  2.9
    l_rice       2.05  0.9  2.75
    l_sc_rice    1.9   0.9  3.0
    g_manure       .   0.11 0.35
    rapeseed     7.0   3.6  7.1/;
nup{cu,en}  "upland cropping nutrient requirement  (kg
     per mu)" := /
              :n_imm  n_tot p2o5_eff k2o_exch humus:
  fodder        2.0  11.0     1.0      2.5      .
  vegetables  4.58 21.35     2.57     3.93     . /;
soil{c,s3}  "soil data - supplies of p2o5 k2o nitrogen
     requirement and irrigation water" := /
```

```
                :p2o5_eff    k2o_exch    inr    iwr:
   /*  p r o p o r t i o n s) (tons per ha) */
    barley        0.6          0.3       0.47    4125
    wheat         0.6          0.3       0.45    4125
    e_rice       -0.1          0.25      0.45    4875
    m_rice        0.2          0.35      0.45    6000
    l_rice        0.6          0.75      0.3     4875
    l_sc_rice     0.05         0.25      0.24    8250
    g_manure      0.9          1              .     .
    rapeseed      0.5          0.4       0.58      ./;
cno{c,n}  "crop nutrient offtake (ton per mu)" := pno*
    yield/100;
enc{cf,en}   "effective nutrient content (pct weight)"
     := sum{nh} nu*nc/100;
kc  "k2o content of irrigation water (ppm)"   :=  3.5;
hdr "humus decomposition rate (kg per kg soil n)":= 9;
cxfert{cf}   "fertilization cash cost (yuan per ton)"
    := / azolla 1.3, rapes_c 200 /;
freq{c,en}   "effective nutrient requirements (ton per
     mu)" := if(
  en='n_imm',    cno[c,'n']*soil[c,'inr'],
  en='n_tot',    cno[c,'n'],
  en='p2o5_eff', cno[c,'p2o5']*(1-soil[c,'p2o5_eff']),
  en='k2o_exch', cno[c,'k2o']*(1-soil[c,'k2o_exch']) -
       soil[c,'iwr']*kc/muperha/1e6,
  en='humus',    hdr*cno[c,'n']*(1-soil[c,'inr']) );
mult{en} "nutrient req factor" := / n_imm 1.1, n_tot
    1.05, p2o5_eff 1.25, k2o_exch 1.1 , humus 1.0 /;
hyield{c}   "yield for high fertilizer applications (
    tons per mu)" := yield*1.07;
hreq{c,en}  "nutrient requirements for high
    application rates  (tons per mu)" := freq*mult;
sreq{s,en,f} "crop sequence nutrient requiremets (ton
    per mu)";
syield{ca,s,f} "yield of crop sequence (ton per mu)";
sys{s,f} "straw yield of crop sequences (ton per mu)";
DoAssign "a dummy name" :=
 {s,en} (sreq[s,en,'normal']  := sum{c} freq*mcp,
         sreq[s,en,'high']    := sum{c} hreq*mcp),
 {c,s}  (syield[c,s,'normal'] := yield*mcp,
         syield[c,s,'high']   := hyield*mcp),
 {s,f}  (sys    := sum{c} cdata[c,'straw_y']*syield,
    ---- e_rice straw is plowed under immediately,
        adjust straw yields and fertilizer requirement
         */
         sys    := sys - syield['e_rice',s,f]*cdata['
             e_rice','straw_y']),
 {s,en,f}(sreq := sreq - cdata['e_rice','straw_y']*
     syield['e_rice',s,f]*enc['c_straw',en]/100*mcp[s
     ,'e_rice']),
```

```
    {s,f} (syield['straw',s,f]   := sys,
           syield['rapes_c',s,f] := 0.8*syield['rapeseed
               ',s,f]);
  crec{ca,cf} "composting and fertilizing recipes" := /
                  :c_straw  c_gm  c_hyacinth  azolla:
      straw         0.12       .         .         .
      g_manure         .    0.12         .         .
      hyacinth         .       .      0.12         .
      pig_m         0.05    0.05      0.05         .
      silt          0.83    0.83      0.83         .
      azolla           .       .         .      1.0
      azolla_e         .       .         .      0.2 /;
  DoAssign1 := {cf[ca1]|~(sum{ca} crec[ca,ca1])} (crec[
      ca,ca1]:= 1);
  chemnall{ca,s4}   "chemical nitrogen (ammonium
      bicarbonate) allocations" := /
   /* crops: kg ammonium bicarbonate per mu of crop
       planted
      qsa  : quota sales allocation for pigs (kg per kg)
      aqsa : above quota sales allocation (kg per kg) */
                               :crops    qsa     aqsa:
      barley                     49          .   0.17
      wheat                      49          .   0.17
      e_rice                     53          .   0.17
      m_rice                     45          .   0.17
      l_rice                     45          .   0.17
      l_sc_rice                  45          .   0.17
      rapeseed                   45          .   0.45
      g_manure                    4          .     .
      pigs                          .   0.225    0.225/;
  schem{s}    "chemical fertilizer allocation for crop
      sequences (tons amm_bi per mu)" := 0.001*(sum{c}
      chemnall[c,'crops']*mcp);
/* model definition */
variable
  xcrop{s,f}   "paddy land cropping activities (mu)";
  xupland{ca}  "upland cropping activities    (mu)";
  xpig{p}      "pig raising activities        (ton)";
  xfeed{g}     "grain feed mixing             (ton)";
  xfert{ca}    "fertilization activities      (kg)";
  purchase{ca} "purchasing activities         (ton)";
  sales{ca}    "quota sales                   (ton)";
  aqsales{ca}  "above quota sales             (ton)";
  ccost        "cash cost                     (yuan)";
constraint
  mb{ca} "material balance (ton)": sum{s,f|ss} syield*
      xcrop + sum{p} pigio*xpig + if(cp,purchase)
    +if(cu,yieldu*xupland) + sum{g[ca1]} gio[ca,ca1]*
        xfeed[ca1] >=  if(cs,sales+aqsales) + sum{cf[ca1
        ]} crec[ca,ca1]*xfert[ca1];
```

```
   labor{t} "labor balance (gong per day)":  sum{s,f|ss}
       lab*xcrop <= (1+labj)*lsup;
   fert{en} "nutrient and humus balance (ton)": sum{s,f|
       ss} sreq*xcrop + 0.001*(sum{cu} nup*xupland)
     <= 0.01*(sum{cf} enc*xfert);
   chemn "chemical nitrogen allocation (ton)": sum{s,f|ss
       } schem*xcrop + sum{cs} (chemnall[cs,'qsa']*sales
     + chemnall[cs,'aqsa']*aqsales) >= purchase['amm_bi
         '];
   landp "paddy land constraint (mu)": sum{s,f|ss} xcrop
       <= paddy;
   landu "upland land constraint (mu)": sum{cu} xupland
       <= upland;
   gmseed "green manure seed requirements (mu)": sum{s,f|
       ss} ((1-0.16)*mcp[s,'gm_seeds'] - 0.16*mcp[s,'
       g_manure'])*xcrop >= 0;
   grainq "grain quota sales (ton)":  sum{g} sales >=
       grainquota;
   cdef "cash cost definition (yuan)": ccost = sum{s,f|ss
       } cxcrop*xcrop + sum{p} cxpig*xpig
     + sum{cf} cxfert*xfert + sum{cp} purdata[cp,'price
         ']*purchase;
   saleslower{cs}: sales >= cdata[cs,'quota_sale'];
   purchaseupper{cp|purdata[cp,'quantity']}: purchase <=
       purdata[cp,'quantity'];
 maximize income "brigade income (yuan)": sum{cs} (cdata[
     cs,'proc_price']*sales + aqsprice*aqsales) - ccost;
 -- solution report
 set su := ['<demand>', '<supply>', '<s_price>'] "
     fertilization demand and supply summary";
 parameter frep{en,su}:=if(su=1, 0.01*(sum{cf} enc*xfert)
      + GetValue(fert,5),
                           su=2, 0.01*(sum{cf} enc*xfert)
                               ,
                           su=3, -GetValue(fert,3));
 Writep(income,frep);
end
```

4.11. Financial Optimization: Financial Engineering (cmo)

— Run LPL Code , HTML Document –

Problem: Collateralized Mortgage Obligations (CMO) are used to restructure the cashflows from underlying mortgage collateral into a set of high quality bonds with different maturities. In the following mixed-integer programming is used to restructure a mortgage pool into six normal tranches and on zero tranch. The model is written in a general form to allow experimentation with different periodicity. The following version is annual.

This model is from the GAMS model library (GAMS SEQ=114). See also [5]

Modeling Steps:

Listing 4.11: The Complete Model implemented in LPL [22]

```
model CMO "Financial Optimization: Financial Engineering"
    ;
  set
    i,j    "tranches"       := [ n1 n2 n3 n4 n5 n6 , m ];
    n{i}   "normal tranches" := [ n1 n2 n3 n4 n5 n6 ];
    m{i}   "m tranches"      := [m];
    tp     "time periods"  := [ 0 1 2 3 4 5 6 7 8 9 10 ];
    ts{tp} "settlement date" := [ 0 ];
    t{tp}  "payment periods" := [ 1 2 3 4 5 6 7 8 9 10 ];
    tl{tp} "last payment"    := [ 10 ];
  /* note: all rates are monthly rates x 12. For
      aggregation we need to compute equivalent rates.
      (1+r/12)**12 == (1+rd)**d */
  parameter
    cg       "collateral annual gross coupon"  := 0.10;
    nom      "nominal value of collateral"     :=  100;
    s        "servicing rate"                  := 0.005;
    psa      "pricing speed"                   :=  115;
    d        "periodicity"                     :=    1;
    tmax     "term to maturity of collateral"  := #t;
    age      "age of the collateral in years"  := (360 -
        tmax*12/d)/12;
    minn     "minimum number of normal tranches" :=  3;
    minm     "minimum number of m tranches"    :=    1;
    --tranches data
    trcoupon{i}:= [0.085 0.085  0.085  0.0875 0.0875
        0.0875 /*0.0875 0.0875*/ 0.0995];
    tryield{i} := [0.099 0.0999 0.0999 0.1006 0.1009
        0.1017 /*0.1018 0.1025*/ 0.105];
```

```
    trlowWal{n}:= [1.00 2.00 3.00 4.00 5.00 7.00 /*10.00
        15.00*/];
    trupWal{n} := [1.24 2.44 3.44 4.44 5.65 7.94 /*10.94
        20.00*/];
    tryr       := 0.1285;
    rev{tp}    "reverse order addresses"  := #tp-2*tp + 1;
    cd         "periodic gross coupon"  := (1+cg/12)^(12/d
        )-1;
    sd         "periodic servicing rate" := (1+s/12)^(12/d
        )-1;
    po{tp}     "outstanding collateral principal at zero
        prepayment" := nom*(1-1/(1+cd)^(tmax+1-tp))
        /(1-1/(1+cd)^tmax);
    a          "zero prepayment annuity" := nom*cd
        /(1-1/(1+cd)^tmax);
    b{tp}      "prepayment rate"  := if(tp>1, 0.06*psa
        /100*Min((tp+age*d)/(d*2.5) , 1));
    coupon{i} "periodic coupon by tranche" := (1+trcoupon
        /12)^(12/d)-1;
    yield{i}  "periodic yield by tranche" := (1+tryield
        /12)^(12/d)-1;
    yr         "periodic residual yield"  := (1+tryr/12)
        ^(12/d)-1;
    cmax       "maximum cmo coupon" := max{i} coupon;
    wallo{i}  "lower weighted average life in periodicity
        " := trlowWal*d;
    walup{i}  "upper weighted average life in periodicity
        " := trupWal*d;
    bv{tp}     "temporary factor for bvf calculation";;
  bv[#tp]:=0, {tp} (bv[tp+rev-1]:=(bv[tp+rev]+a-sd*po[tp+
      rev-1])/(1+cmax));
  parameter
    bvf{tp}    "bond value factor" := if(po,Min(bv/po,1),
        #tp,1);
    pe{tp}     "expected outstanding collateral principal"
        ;
    dum1 := pe[1]:=nom,  for{tp|tp>1} (pe:=pe[tp-1]*(1-b)
        ^(1/d)*po/po[tp-1]);
    cflow{tp} "expected collateral payments" := if(tp
        >1,(1+cd-sd)*pe[tp-1] - pe[tp]);
    prin{tp}  "structuring principal payments from
        collateral" := if(tp>1,pe[tp-1]*bvf[tp-1] - pe*
        bvf);
    sump       "sum of prin" := sum{tp} prin;
    psum{tp}  "sum of principal payments on collateral
        till tp";
    tpsum{tp} "sum of time principal payment products";
    dum2 := tpsum[1]:=0, psum[1]:=0, for{tp|tp>1} (psum:=
        psum[tp-1]+prin, tpsum:=tpsum[tp-1]+(tp-1)*prin);
    walp{tp}  "wal on principal payments from collateral"
```

```
        := if(tp>1,tpsum/psum);
    bign "big number":= sump*0.7;
    smalln "small number" := sump*0.03;
  set
    zpos{i,tp} "possible payment periods by tranche" :=
        if(walp>walup and i within n,0,1);
  variable
    x{i,tp}      "outstanding principal in each tranche";
    c{i,tp}      "cashflow in each tranche";
    y{i,tp}      "upper triangular structure";
    r{tp}        "residual payments";
    pv{i}        "def pv tranches";
    p{i,tp} [-1e18..1e18] "principal payments on tranches
        ";
    tpp{i}  [-1e18..1e18] "product of time and principal
        payment";
    pvres   [-1e18..1e18] "def pv residuals";
    binary z{i,tp} "tranche utilization variable (1=usage
        )";
     tin{i}        "tranche in the structure";
  constraint
    defpv{i} "def pv tranches":  pv = sum{t|zpos} 1/(1+
        yield)^(t-1)*c;
    defpvres "def pv residuals":  pvres = sum{t} 1/(1+yr)
        ^(t-1)*r;
    pdef{zpos[i,t]|t>1} "definition of principal payments
        ":  p = x[i,t-1] - x;
    cdef{zpos[i,t]|t>1} "cashflow accounting":  c =
        coupon*x[i,t-1] + p;
    retiren1{t} "retirement of normal tranches":  sum{n|
        zpos} p = prin + sum{m} (x[m,tp-1]*coupon-c);
    retire{zpos[i,t]|i within n} "retirement of tranches"
        : p <= cflow*z;
    retirem{m,t} "retirement of m tranche":  c <= cflow*z
        ;
    retirem1{m,t} "retirement of m tranche": p <= prin*z;
    cbal{t} "cashflow balance":       sum{i|zpos} c + r =
         cflow;
    tppdef{n} "definition of tpp":     tpp = sum{t|zpos}
        (t-1)*p;
    lowal{n} "lower bound on weighted average life":
        wallo*(sum{ts} x) <= tpp;
    upwal{n} "upper bound on weighted average life":
        walup*(sum{ts} x) >= tpp;
    seq1{t} "sequencing constraint 1":       sum{i|zpos}
        z = 1;
    seq2{zpos[i,t]|t>1} "sequencing constraint 2": y >= y
        [i,t+1];
    ydef{zpos[i,t]|t>1} "definition of y": y = y[i-1,t] +
         z;
```

```
    tindef1{i} "tranche in definition":   sum{ts} x <=
        tin*bign;
    tindef2{i} "tranche in definition":   sum{ts} x >=
        tin*smalln;
    ncon "constraint on number of tranches":  sum{n} tin
        >= minn;
    mcon "constraint on number of m tranches": sum{m} tin
         >= minm;
    pLow{n,tp}: p>=0;
    xfx1{i,tl}: x=0;
    xfx2{i,t|~zpos[i,t+1]}: x=0;
    zfx1{i,tp|~zpos}: z=0;
    zfx2{m,tp|~(sum{j in n} zpos[j,tp])}: z=1;
  maximize proceeds: sum{i} pv + pvres;
  Writep(proceeds,z,x,c,r);
end
```

4.12. Peacefully Coexisting Armies of Queens (coex)

—- Run LPL Code , HTML Document –

Problem: Two armies of queens (black and white) peacefully coexist on a chessboard when they are placed on the board in such a way that no two queens from opposing armies can attack each other. The problem is to find the maximum two equal-sized armies.

This model is from the GAMS model library (GAMS SEQ=219). See also [19].

Modeling Steps:

Listing 4.12: The Complete Model implemented in LPL [22]

```
model COEX "Peacefully Coexisting Armies of Queens";
  set  i ,j,ii "size of chess board" := [ 1..8 ];
    m{i,j,ii,jj in i} "shared positions on the board" :=
        (i=ii) or j=jj or Abs(i-ii)=Abs(j-jj);
  binary variable  b{i,j} "square occupied by a black
      queen";
    w{i,j} "square occupied by a white queen";
  variable  tot "total queens in each army";
  constraint
    eq1{m[i,j,ii,jj]} "keeps armies at peace": b[i,j] + w
        [ii,jj] <= 1;
    eq2 "add up all the black queens": tot = sum{i,j} b;
    eq3 "add up all the white queens": tot = sum{i,j} w;
  maximize obj: tot;
  Draw.Scale(50,50);
  for{i,j} do Draw.Rect(i,j,1,1,if((i+j)%2,0,1),0); end
  for{i,j|b} do Draw.Circle(j+.5,i+.5,.3,5); end
  for{i,j|w} do Draw.Circle(j+.5,i+.5,.3,4); end
end
```

Solution:

opt: solution reported in OPTIMA

```
b.fx('6','2') = 1;
b.fx('7','2') = 1;
b.fx('8','2') = 1;
b.fx('7','1') = 1;
b.fx('8','1') = 1;
b.fx('7','6') = 1;
b.fx('8','6') = 1;
b.fx('7','7') = 1;
b.fx('8','7') = 1;
```

4.13. Alcuin's River Crossing (cross)

—- Run LPL Code , HTML Document –

Problem: A farmer carrying a bushel of corn and accompanied by a goose and a wolf came to a river. He found a boat capable of transporting himself plus one of his possessions - corn, goose, or wolf - but no more. Now, he couldn't leave the corn alone with the goose, nor the goose alone with the wolf, else one would consume the other. Nevertheless, he succeeded in getting himself and his goods across the river safely.

This model is from the GAMS model library (GAMS SEQ=191) (Contributed by Soren Nielsen, Institute for Mathematical Sciences University of Copenhagen). See also [2].

Modeling Steps:

Listing 4.13: The Complete Model implemented in LPL [22]

```
model CROSS "Alcuin's River Crossing";
  set  i "items" := [goose, wolf, corn];
       t "time" := [1..10];
  parameter dir{t}  "crossing - near to far is +1 - far
      to near -1" := -1^(t-1);
  variable
    cross{i,t}   "crossing the river";
    done{t}      "all items in far side";
    binary y{i,t} "1 iff the item is on the far side at
        time t";
  constraint
    DefCross{i,t|t>1} "crossing":    y = y[i,t-1] + dir[t
        -1]*cross[i,t-1];
    DefDone{i,t}  "everything on far side": done <= y;
    limCross{t|t>1}: sum{i} cross[i,t-1] <= 1;
    EatNone1{t}:  dir*(y['goose',t] + y['wolf',t] - 1) <=
         done;
    EatNone2{t}:  dir*(y['goose',t] + y['corn',t] - 1) <=
         done;
    yFx{i}: y[i,1] = 0;
 maximize nocross "number of non crossing periods": sum{t
     } done;
Writep(nocross,y,cross);
end
```

4.14. 3-dimensional Noughts and Crosses (cube)

— Run LPL Code , HTML Document –

Problem: White and black balls are to be arranged in a cube one to a cell in such a way as to minimize the number of lines with balls of equal color. For this example the length of the cube is three. a total of 49 lines exist in a cube of 3x3x3. 27 lines by changing only one coordinate, 18 diagonals within a plane, and 4 diagonals going across planes. (see Will16.lpl)

This model is from the GAMS model library (GAMS SEQ=42). See also [13] and te model WILL16.

Modeling Steps:

Listing 4.14: The Complete Model implemented in LPL [22]

```
model CUBE "3-dimensional Noughts and Crosses";
 set s,sp,spp "domain for line" := [a b c incr decr];
     x{s}   "coordinate labels"     := [ a, b, c ];
     y{sp}  "coordinate labels"     := [ a, b, c ];
     z{spp} "coordinate labels"     := [ a, b, c ];
     d{s}   "directions"            := [ incr, decr ];
     dp{sp} "directions"            := [ incr, decr ];
     b        "bounds"              := [ low, high ];
   ld{s,sp,spp}  "line definition" :=
     (s='incr' and sp within x   and spp within x or
      s within x   and sp='incr' and spp within x  or
      s within x   and sp within x   and spp='incr'  or
      s='incr' and sp within d   and spp within x  or
      s within x   and sp='incr' and spp within d  or
      s within d   and sp within x   and spp='incr'  or
      s='incr' and sp within d   and spp within d);
  parameter
    ls{b} "sign for line definitions" :=/low 1 high -1/;
    lr{b} "rhs for line definitions"  :=/low 2 high -1/;
    df{x,s} "line definition function" := if(s='decr',1+#
        x-2*x, s='incr',0, s-x);
  variable
    line{s,sp,spp}      "line identification";
    binary core{x,y,z}  "placement of balls (white 0
        black 1)";
  constraint
    nbb "total number of balls definition":  sum{x,y,z}
        core = Floor(#x^3/2);
    ldef{ld[s,sp,spp],b} "line definitions": ls*(sum{v in
         x} core[v+df[v,s],v+df[v,sp],v+df[v,spp]]) <=
        line+lr;
```

```
  minimize num "number of lines of equal color": sum{ld[s
      ,sp,spp]} line;
  Writep(num,core,line);
end
```

4.15. Simple Farm Level Model (demo1)

— Run LPL Code , HTML Document –

Problem: This is the first in a series of single farm models . This simplest version has only 7 principal crops and 2 basic inputs, land and labor, which are specified on a monthly basis.

This model is from the GAMS model library (GAMS SEQ=91). See also [16]

Modeling Steps:

Listing 4.15: The Complete Model implemented in LPL [22]

```
model DEMO1 "Simple Farm Level Model";
  set c "crops" := [ wheat, clover, beans, onions, cotton
      , maize, tomato ];
      t "period" := [ jan, feb, mar, apr, may, jun, jul,
          aug, sep, oct, nov, dec ];
  parameter landreq{t,c}  "months of land occupation by
      crop (hectares)" := [
      1.0    1.0     1.0    1.0     .      .      .
      1.0    1.0     1.0    1.0     .      .      .
      1.0    0.5     1.0    1.0    0.5     .      .
      1.0      .     1.0    1.0    1.0     .      .
      1.0      .       .    0.25   1.0   0.25     .
       .       .       .      .    1.0   1.0      .
       .       .       .      .    1.0   1.0    0.75
       .       .       .      .    1.0   1.0    1.0
       .       .       .      .    1.0   1.0    1.0
       .       .       .      .    1.0    .5    1.0
      0.5    0.25    0.25   0.5    0.75    .     0.75
      1.0    1.0     1.0    1.0     .      .      .  ];
  laborreq{t,c} "crop labor requirements (man days per
      hectare)" := [
      1.72   4.5    0.75   5.16     .      .      .
      0.5    1.0    0.75   5.0      .      .      .
      1.0    8.0    0.75   5.0     5.0     .      .
      1.0      .   16.    19.58    5.0     .      .
     17.16     .      .    2.42    9.0    4.3     .
      2.34     .      .      .     2.0    5.04    .
       .       .      .      .    1.5    7.16   17.0
       .       .      .      .    2.0    7.97   15.0
       .       .      .      .    1.0    4.41   12.0
       .       .      .      .   26.0    1.12    7.0
      2.43   2.5    7.5    11.16   12.0     .     6.0
      1.35   7.5    0.75   4.68     .      .      .  ];
  yield{c} "crop yield (tons per hectare)" := /
```

```
  wheat   1.5, clover  6.5,  beans    1, onions  3
  cotton  1.5, maize   2  ,  tomato   3  /;
price{c} "crop prices (dollars per ton)"       := /
  wheat   100, beans   200,  onions  125
  cotton  350, maize    70,  tomato  120 /;
miscost{c} "misc cash costs (dollars per hectare)" := /
  wheat    10, beans     5,  onions   50
  cotton   80, maize     5,  tomato   50 /;
/*   farm data, size labor availability etc. */
land   "farmsize (hectares)"  :=   4.;
famlab "family labor available (days per month)" := 25;
owage  "hire-out  wage rate (dollars per day)" := 3.;
twage  "temporary labor wage  (dollars per day)" := 4;
dpm    "number of working days per month"  := 25;
/*  endogenous variables and equations */
variable
  xcrop{c}   "cropping activity          (hectares)";
  flab{t}    "family labor use           (days)";
  fout{t}    "hiring out                 (days)";
  tlab{t}    "temporary labor            (days)";
  revenue    "value of production        (dollars)";
  mcost      "misc cash cost             (dollars)";
  labcost    "labor cost                 (dollars)";
  labearn    "labor income               (dollars)";
constraint
  landbal{t} "land balance (hectares)": sum{c} xcrop*
      landreq  <= land;
  laborbal{t} "labor balance (days)": sum{c} xcrop*
      laborreq <= flab + tlab;
  flabor{t} "family labor balance (days)": famlab  =
      flab + fout;
  arev "revenue accounting (dollars)": revenue = sum{c}
       xcrop*yield*price;
  acost "cash cost accounting (dollars)": mcost   = sum
      {c} xcrop*miscost;
  alab "labor cost accounting (dollars)": labcost = sum
      {t} tlab*twage;
  aout "labor income accounting (dollars)": labearn =
      sum{t} fout*owage;
maximize yfarm "farm income": revenue+labearn-labcost-
    mcost;
/*  report on solution */
set crep := [ landuse, output, revenue ] "crop report
    summary";
    lrep := [ demand,  family, temporary unused,
        hire_out ] "labor report summary(days)";
parameter
  croprep{crep,c}:=if(crep=1, xcrop,
                      crep=2, xcrop*yield,
                      crep=3, croprep['output',c]*price
```

```
                              );
    labrep{t,lrep}:= if(lrep=1, sum{c} xcrop*laborreq,
                        lrep=2, flab,
                        lrep=3, tlab,
                        lrep=4, -GetValue(laborbal,5),
                        lrep=5, fout);
  Writep(yfarm,croprep,labrep);
end
```

4.16. Non-transitive Dice Design (dice1)

—- Run LPL Code , HTML Document –

Problem: Probabilistic dice - an example of a non-transitive relation. We want to design a set of dice with an integer number on each face such that on average dice1 beats dice2, and dice2 on average beats dice3 etc, but diceN has to beat dice1. MIP codes behave very erratic on such a problem and slight reformulations can result in dramatic changes in performance. Also note the face value will be integers automatically.

This model is from the GAMS library (GAMS SEQ=176). See [4].

Modeling Steps:

Listing 4.16: The Complete Model implemented in LPL [22]

```
model DICE "Non-transitive Dice Design";
  set f ,fp   "faces on a dice" := [1..6];
      dice ,d "number of dice" := [dice1 dice2 dice3];
  parameter
    flo  "lowest face value" :=  1;
    fup  "highest face value" := #dice*#f;
    wn   "wins needed - possible bound" := Floor(0.5*#f
        ^2)+1;
  variable
    wnx "number of wins";
    fval{d,f} [flo..fup]   "value of dice - will be
        integer";
    binary comp{d,f,fp}   "one if f beats fp";
  constraint
    ffx: fval['dice1','1'] = flo;
    equ1{d} "count the wins": sum{f,fp} comp = wnx;
    eq3{d,f,fp} "definition of non-transitive relation":
        fval+(fup-flo)*(1-comp) >= fval[d%#d+1,fp]+1;
    eq4{d,f|f<#f}  "different face values for a single
        dice": fval+1 <= fval[d,f+1];
  maximize obj: wnx;
  Writep(wnx);
end
```

4.17. Stigler's Nutrition model (diet1)

—- Run LPL Code , HTML Document –

Problem: This model determines a least cost diet which meets the daily allowances of nutrients for a moderately active man weighing 154 lbs.

This model is from the GAMS library (GAMS SEQ=7) and from [10], Chap 27.1.

Modeling Steps:

Listing 4.17: The Complete Model implemented in LPL [22]

```
model DIET "Stigler's Nutrition model";
  set
    n "nutrients" := [ calorie /*thousands*/, protein /*
        grams*/ , calcium /*grams*/, iron /*milligrams*/
        vitamin_a /*thousand ius*/, vitamin_b1  /*
        milligrams*/, vitamin_b2  /*milligrams*/, niacin
        /*milligrams*/, vitamin_c /*milligrams*/ ];
    f "foods" := [ wheat, cornmeal, cannedmilk, margarine
        , cheese, peanut_b, lard liver, porkroast, salmon
        , greenbeans, cabbage, onions, potatoes spinach,
        sweet_pot, peaches, prunes, limabeans, navybeans
        ];
  parameter b{n} "required daily allowances of nutrients"
      :=
    / calorie 3, protein 70, calcium 0.8, iron 12
        vitamin_a 5, vitamin_b1 1.8, vitamin_b2 2.7,
        niacin  18, vitamin_c  75 /;
    a{f,n} "nutritive value of foods (per dollar spent)"
        := [
/*(1000)   (g)   (g)    (mg)(1000iu)(mg)     (mg)    (mg)  (mg)
    */
   44.7  1411  2.0  365     .      55.4   33.3   441     .
   36     897  1.7   99   30.9   17.4    7.9   106     .
   8.4    422 15.1    9   26       3    23.5    11     60
   20.6    17  0.6    6   55.8     0.2     .     .      .
    7.4   448 16.4   19   28.1     0.8   10.3     4     .
   15.7   661  1     48     .      9.6    8.1   471     .
   41.7     .  .       .    0.2      .     0.5     5     .
    2.2   333  0.2  139  169.2     6.4   50.8   316   525
    4.4   249  0.3   37     .     18.2    3.6    79     .
    5.8   705  6.8   45    3.5     1      4.9   209     .
    2.4   138  3.7   80   69       4.3    5.8    37   862
    2.6   125  4     36    7.2     9      4.5    26  5369
    5.8   166  3.8   59   16.6     4.7    5.9    21  1184
```

```
   14.3   336  1.8  118    6.7   29.4    7.1   198  2522
    1.1   106  .    138  918.4    5.7   13.8    33  2755
    9.6   138  2.7   54  290.7    8.4    5.4    83  1912
    8.5    87  1.7  173   86.8    1.2    4.3    55    57
   12.8    99  2.5  154   85.7    3.9    4.3    65   257
   17.4  1055  3.7  459    5.1   26.9   38.2    93     .
   26.9  1691 11.4  792     .    38.4   24.6   217     .
       ];
  variable x{f}  "dollars of food f to be purchased daily
       (dollars)";
  constraint nb{n} "nutrient balance  (units)": sum{f} (a
      [f,n]*x) >= b;
  minimize cost "total food bill (dollars)": sum{f} x;
  Writep(cost,x);
end
```

4.18. Fertilizer Production (egypt)

—- Run LPL Code , HTML Document –

Problem: This is a one-period model of the Egyptian fertilizer industry developed at the World Bank. (for a GAMS representation see also: BISSCHOP J., MEERAUS A., On the Development of a General Algebraic Modeling System in a Strategic Planning Environment. Development Research Department, Mathematical Programming Study 20, pp.1-29, 1982. and CHOKSI A.M., MEERAUS A., STOUTJESDIJK, The Planning of Investment Programs in the Fertilizer Industry, Johns Hopkins University Press, Batimore, 1980.

Cited also in: FOURER R., A Modeling Language for Mathematical Programming, 1987

A model formulation is as follows:

Listing 4.18: The Complete Model implemented in LPL [22]

```
model Egypt "Fertilizer Production";
  set
    Center,Ce
        := [ Aswan  Helwan  Assiout  Kafr_el_Zt
             Abu_Zaabal  Abu_Kir  Talkha  Suez   ];
    P,i,j "Plants"
        := [ Aswan  Helwan  Assiout  Kafr_el_Zt
            Abu_Zaabal  ];
    Port  ,Po := [ Abu_Kir  ];
    Region,R
        := [ Alexandria  Behera  Gharbia  Kafr_el_Sh
             Dakahlia  Damietta  Sharkia  Ismailia
             Suez  Menoufia  Kalubia  Giza  Beni_Suef
             Fayoum  Minia  Assiout  New_Valley
             Sohag  Quena  Aswan  ];
    Units ,Un
        := [ Sulf_A_S  Sulf_A_P  Nitr_Acid  Amm_Elec
             Amm_C_Gas  C_Amm_Nitr  Amm_Sulf  SSP  ];
    Process,Pr
        := [ Sulf_A_S  Sulf_A_P  Nitr_Acid  Amm_Elec
             Amm_C_Gas  Can_310  Can_335  Amm_Sulf
             SSP_155  ];
    Nutr  ,N  := [ N  P205  ];
    Commod,C,c
        := [ El_Aswan  Coke_Gas  Phos_Rock  Limestone
             El_Sulfur  Pyrites  Electric  Bf_Gas
             Water  Steam  Bags  Ammonia  Sulf_Acid
             Nitr_Acid  Urea  Can_260  Can_310  Can_335
             Amm_Sulf  Dap  SSP_155  C_250_55  C_300_100
                 ];
```

```
  cRaw,r  := [ El_Aswan  Coke_Gas  Phos_Rock  Limestone
           El_Sulfur  Pyrites  Electric  Bf_Gas
           Water  Steam  Bags  ];
  cInter:= [ Ammonia  Sulf_Acid  Nitr_Acid  ];
  cShip,s := [ Ammonia  Sulf_Acid  ];
  cFinal,f:= [ Urea  Can_260  Can_310  Can_335
           Amm_Sulf  Dap  SSP_155  C_250_55  C_300_100
               ];
parameter
  exch := 0.4;     utilPct := 0.85;
  /*cf75{R,cFinal};
  fn{cFinal,N};
  road{R,Ce};
  railHalf{P,P};
  impdBarg{P}; impdRoad{P};
  io{C,Pr};
  pImp{C};
  pR{cRaw};
  pPr{P,cRaw};
  dcap{P,Un};
  util{Un,Pr};*/
 /*$I 'egypt.inc'*/  -- data is here
  rail{i,j}  := if(railHalf[i,j] , railHalf[i,j] ,
      railHalf[j,i]);
  tranFinal{P,R} := .5 + .0144*road[R,P within Ce];
  tranImport{R,Po} := .5 + .0144*road[R,Po within Ce];
  tranInter{i,j} := 3.5 + .03*rail[i,j];
  tranRaw{P}  := if(impdBarg , 1+.003*impdBarg)
                 + if(impdRoad , .5+.144*impdRoad);
  pDom{P,cRaw} := if(pR , pR , pPr);
  icap{Un,P} := .33*dcap;
set  --- some derived entities
  mPos{P,Un} := icap>0;
  pCap{P,Pr} := forall{Un| util>0} mPos;
  pPos{P,Pr} := pCap and ~pExcept;
  cpPos{C,P} := sum{Pr | pPos} io>0;
  ccPos{C,P} := sum{Pr | pPos} io<0;
  cPos{C,P}  := cpPos or ccPos;
variable
  Z {P,Pr | pPos};
  Xf{cFinal,P,R | cpPos[cFinal within C,P]};
  Xi{s,i,j | cpPos[s within C,i] and ccPos[s within C,j
      ]};
  Vf{cFinal,R,Po};
  Vr{cRaw,P | ccPos[cRaw within C,P]};
  U{cRaw,P | ccPos[cRaw within C,P] and pDom>0};
  Psip; Psil; Psii;
constraint
  mbd{N,R}:
         sum{cFinal,Po} fn*(sum{Po} Vf + sum{P} Xf)
```

```
            >= sum{cFinal} fn*cf75;
    mbdb{cFinal,R | cf75>0}:
            sum{Po} Vf + sum{P} Xf  >= cf75;
    mb{c,i}: sum{Pr} io*Z
                     + sum{j} Xi[c within cShip,i,j]
                     + sum{j} Xi[c within cShip,j,i]
                     + if(pImp,Vr[c within cRaw,i]) + U[c
                        within cRaw,i]
                     >=  sum{R} Xf[c within cFinal,P,R];
    cc{P,Un| mPos}:  sum{Pr} util*Z  <=  utilPct*icap;
    ap:  Psip = sum{cRaw,P} pDom*U;
    al:  Psil = sum{cFinal,P,R}  tranFinal*Xf
              + sum{cFinal,Po,R}  tranImport*Vf
              + sum{s,i,j} tranInter[i,j]*Xi[s,i,j]
              + sum{cRaw,P}  tranRaw*Vr;
    ai:  Psii/exch = sum{f,R,Po}  pImp[f within C]*Vf
                   + sum{r,P} pImp[r within C]*Vr;
  minimize Psi:  Psip + Psil + Psii;
  Writep(Psi,Z,Xf,Xi,Vf,Vr,U,Psip,Psil,Psii);
  --- print some other tables ---
  Writep(mPos,pCap,pExcept,pPos,cpPos,ccPos,cPos);
end
```

4.19. House Plan Design (house)

—- Run LPL Code , HTML Document –

Problem: This problem designs the size of an L shaped house meeting limitations imposed by city codes and aesthetic considerations.

This model is from the GAMS library (GAMS SEQ=99) and from Borland, Eureka: The Solver. Tech. rep., Borland International, 1987.

Modeling Steps:

Listing 4.19: The Complete Model implemented in LPL [22]

```
model HOUSE "House Plan Design";
  variable
    x  := 30 "width of front wing (ft)";
    y  "length of front wing (ft)";
    z  "length of second story (ft)";
    b  [40..68] := 68 "total house width (ft)";
    a  "length of back wing (ft)";
    l  [56..100] "total house length (ft)";
    a1 [0..3000] "area of first floor (sq ft)";
    a2 "area of the second floor (sq ft)";
  constraint
    defa1 "definition of first floor area (sq ft)": a1 =
        x*y+a*b;
    defa2 "definition of second floor area (sq ft)": a2 =
         x*z;
    maxw "front wing maximum width (ft)": b/3<=x<=b/2;
    minp "minimum pool area (sq ft)":  y*(b-x) >= 1500;
    defl "definition of total house length(ft)": l=y+a;
    balk "balkoney size restriction (ft)":  z = a+y/2;
    prop "wing proportionality (ft)": a >= y/2;
  maximize obj "total area of the house  (sq ft)": a1+a2;
  Writep(obj);
  Draw.Scale(7,7);
  Draw.Line(0,0,b,0); Draw.Text('b='&Round(b,-2),b/3,2);
  Draw.Line(0,0,0,l); Draw.Text('l='&Round(l,-2),1,l/2);
  Draw.Line(b,0,b,a); Draw.Text('a='&Round(a,-2),b+1,a/2)
      ;
  Draw.Line(0,l,x,l); Draw.Text('x='&Round(x,-2),x/3,l-2)
      ;
  Draw.Line(x,l-y,x,l);Draw.Text('y='&Round(y,-2),x/2,a+(
      l-y)/1.5);
  Draw.Line(x,l-y,b,a);
  Draw.Line(0,l-y/2,x,l-y/2); Draw.Text('Balcony',2,l-y
      /2+2);
```

```
  Draw.Text('Swimming Pool',x+2,l-y/2+2);
end
```

4.20. A Transportation Problem (trnsport)

—- Run LPL Code , HTML Document –

Problem: This problem finds a least cost shipping schedule that meets requirements at markets and supplies at factories. From Gams model library SEQ=1. [10], chap 3.3, and [21], chap 2.

Listing 4.20: The Complete Model implemented in LPL [22]

```
model trnsport "A Transportation Problem";
  set
    i  "canning plants" :=  [ seattle, 'san-diego' ];
    j  "markets" :=  [ 'new-york', chicago, topeka ];
  parameter
    a{i}  "capacity of plant i in cases" :=
      / seattle   350, 'san-diego' 600 /;
    b{j}  "demand at market j in cases" :=
      / 'new-york' 325, chicago 300, topeka 275 /;
    d{i,j} "distance in thousands of miles" := [
              2.5           1.7          1.8
              2.5           1.8          1.4 ];
    f "freight in dollars per case per thousand miles" :=
        90;
    c{i,j} :=  f*d/1000  "transport cost in thousands of
        dollars per case";
  variable x{i,j} "shipment quantities in cases";
  constraint
    supply{i}: sum{j} x <= a  "observe supply limit at
        plant i";
    demand{j}: sum{i} x >= b  "satisfy demand at market j
        ";
  minimize cost: sum{i,j} c*x "total transportation costs
      in thousands of dollars";
  Writep(cost,x);
end
```

CHAPTER 5

MISC

5.1. Production Planning (aggrplan)

—- Run LPL Code , HTML Document –

Problem: Various products have to be manufactured over several time periods. The demand of each product in each time period is given. The company has to decide in which time period it is more cost-effective to produce under overtime regime or to store the product. (see [11], p.153).

Modeling Steps:

1. We have two sets: products and time periods p and t.
2. The unknown quantity to produce is $X_{p,t}$, the unknown quantity to store is $L_{p,t}$, and the unknown overtime amount is U_t in each period.
3. The data are: demand $d_{p,t}$, technical and workforce coefficients a_p, b_p, Storage and overtime cost lo_p, u_t, the initial product storage $L0_p$, and the different maximal capacities for overtime, workpower and techical capacity.
4. We minimize the overtime and the storage costs.
5. The balance in product quantity between two period must hold.
6. We have also technical and workforce capacity.

Listing 5.1: The Main Model implemented in LPL [22]

```
model aggrplan "Production Planning";
  set p "A number of products";
      t "A number of time periods";
  parameter
     lo{p} "Storage cost";
     a{p} "Production coefficient workforce";
     b{p} "Production coefficient material";
```

```
    u{t} "Cost of overtime workforce";
    L0{p} "Initial stock of product at period 0";
    d{p,t} "Demand of product p at time period t";
    Nmax{t} "Max. workforce capacity in time period t";
    Umax{t} "Max. workforce overtime capacity";
    Cmax{t} "Maximal technical capacity";
  variable X{p,t} "Product quantity to produce";
    L{p,t}        "Product quantity to store";
    U{t} [0..Umax] "Used overtime workforce in time
        period t";
  minimize costs: sum{p,t} lo*L + sum{t} u*U "Minimize
      storage costs and overtime workforce";
  constraint
    Balance{p,t}: X + if(t=1,L0,L[p,t-1]) - L = d;
    TecCapacity{t}: sum{p} b*X <= Cmax;
    ManCapacity{t}: sum{p} a*X - U <= Nmax;
  Writep(costs,X,U);
end
```

A small instance with two products and 4 time periods is given in the data model:

Listing 5.2: The Data Model

```
model data "A Production plan";
  p := [A B];
  t := [1..4];
  Nmax{t} := [160 160 160 160];
  Cmax{t} := [200 200 200 200];
  Umax{t} := [100 100 100 100];
  lo{p}   := [4  4];
  L0{p}   := [ 36  220];
  a{p}    := [1.0  0.5];
  b{p}    := [0.5  1.0];
  u{t}    := [5  5  5  5];
  d{p,t}  := [100  90 110 100 , 200 190 210 200];
  --d{p,t} := 1.1*d[p,t];
end
```

Questions

1. How does the solution change if the technical capacity is unlimited?
2. Demand is uncertain, how does the production plan change if demand augements/drops by 10%? Model these requirements.

Answers

1. Remove the constraints `TecCapacity` and the costs drop from 1467.50 to 615.

2. Add a new last instruction in the data model:

```
d{p,t} := 1.1*d[p,t];   (d{p,t} := 0.9*d[p,t];)
```

   Note that at an addition 10% of the demand the problem becomes infeasible, because the technical capacity is too small to generate the demand.

5.2. A simple blending problem (alloy)

— Run LPL Code , HTML Document –

Problem: A small blending problem

Listing 5.3: The Complete Model implemented in LPL [22]

```
model ALLOYING "A simple blending problem";
  set
    metals,m := [ A,B,C1,C2,D ];
    ores, o := [ ore1 ore2 ore3 ore4 ore5 ore6 ];
  parameter
    bound{o} "availability of ore"
       := [1000   3000   2000   800  2600 1700];
    boundu{o} "availability of ore"
       :=0;
    price {o} := [25     10     20    18    22   12  ];
    purity{o} := [ 0.8    0.4    0.7   0.6   0.7  0.4];
    alloy{m}  := [ 0.18  -0.30  -0.40  0.60  0.02];
    obtain{o,m} := [
        0.20  -0.20  -0.40  0.40  0.00
        0.15  -0.00  -0.20  0.20  0.05
        0.00  -0.40  -0.30  0.30  0.00
        0.10  -0.20  -0.30  0.30  0.00
        0.10  -0.25  -0.25  0.25  0.10
        0.05  -0.08  -0.17  0.17  0.10];
    quantity,Q := 3000;
  variable AMOUNT,X{o} [boundu..bound];
  constraint
    quareq: sum{o} purity*X = Q   "Quantity to produce";
    metreq{m}: sum{o}(alloy*purity-obtain)*X >= 0  "Metal
        requirements";
  minimize cost: sum{o} price*X  "Minimize costs";
  Writep(cost,X);
end
```

5.3. Assign Consulters (assign1)

—- Run LPL Code , HTML Document –

Problem: A consulter company plans to distribute its consulter with different knowledge into the different projects over the periods. Each project needs various knowledge in the different time periods.

Modeling Steps:

Listing 5.4: The Complete Model implemented in LPL [22]

```
model assign1 "Assign Consulters";
  set c "Consulters";
    k   "knowledge";
    t   "time periods";
    p   "projects";
    CK{c,k} "c has the knowledge k";
    PK{p,k,t} "project uses knowledge in time";
  parameter level{c};
  binary variable x{c,p,t};
  constraint
    R{c,t}: sum{p} x <=1 "In each period, a consulter can
         only be with one project";
    S{p,t,k|PK}: sum{c|CK} x>=1 "The project must be
        adequatly accompanied";
  minimize  obj: sum{c,p,t} level*x "minimize the pool of
       consulter";
  model data;
    c := [1..5];
    k := [1..3];
    t := [1..20];
    p := [1..3];
    CK{c,k} := [1 1, 1 2, 2 2, 2 3, 3 3, 4 1, 5 2, 5 1];
    PK{p,k,t} := if(Rnd(0,1)>0.2,Trunc(Rnd(0,2)),0);
    level{c} := [1 2 3 4 5];
  end
  model output;
    Writep(PK,x);
  end
end
```

5.4. The Beer Game (beergame)

— Run LPL Code , HTML Document –

The Beer Game perhaps is one of the best known demonstrations in management. It was designed by John Sterman (Teaching Takes Off, Flight Simulators for Management Education, or/MS Today, October 1992, pp. 40-44).

The purpose of the Beer Game is to introduce the concepts of system dynamics in the context of supply chain management. The supply chain contains five roles:

1. Customer Demand (from the model)
2. Retailer
3. Wholesaler
4. Distributor
5. Factory

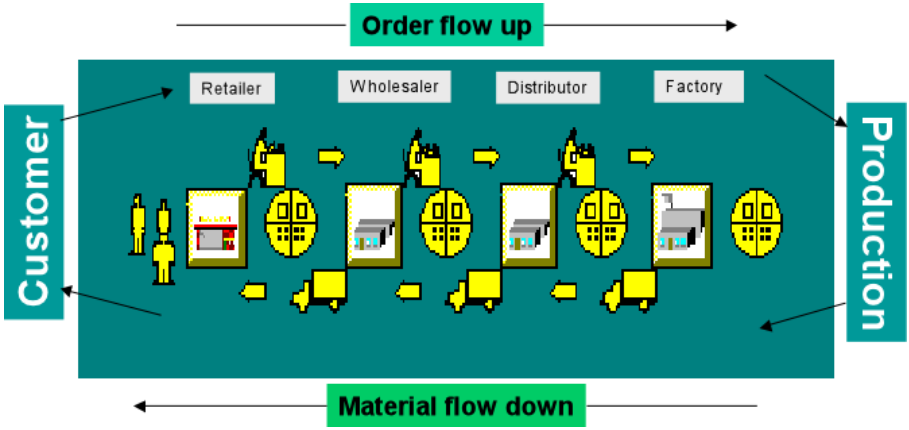

Figure 5.1: The Supply Chain

Problem: This is a version of the famous beer game problem: Consider a simplified beer supply chain, consisting of a single retailer, a single wholesaler which supplies the retailer, a single distributor which supplies the wholesaler, and a single factory with unlimited raw materials which makes (brews) the beer and supplies the distributor. Each component in the supply chain has unlimited storage capacity, and there is a fixed supply lead time and order delay time between each component.

Each week, each component in the supply chain tries to meet the demand of the downstream component. Any orders which cannot be met are recorded

as backorders, and are met as soon as possible. No orders will be ignored, and all orders must eventually be met. At each period, each component in the supply chain is charged a (1.00) shortage cost per backordered item. Also, at each period, each location is charged (0.50) inventory holding cost per inventory item that it owns. Each component owns the inventory at that facility. In addition, the wholesaler owns inventory in transit to the retailer; the distributor owns inventory in transit to the wholesaler; the factory owns both items being manufactured and items in transit to the distributor.

Each supply chain member orders some amount from its upstream supplier. It takes one week for this order to arrive at the supplier. Once the order arrives, the supplier attempts to fill it with available inventory, and there is an additional two week transportation delay before the material being shipped by the supplier arrives at the customer who placed the order.

The goal of the retailer, wholesaler, distributor, and factory, is to minimize total cost, either individually, or for the system. (see: http://beergame.mit.edu).

Modeling Steps:

1. First we identify the players of the system and fix the time horizon: There are four players: Retailer WholeSaler Distributor Factory, in this order defining the down- and up-flow stream of placing orders and delivery of the products. The time horizon defines the length of the "game".
2. In each period the costumer demand an amount of the product (`demand`).
3. Then we define a minimal stock at which an order should be placed, if the Inventory fall beneath that point (`s`) and an amount of stock that is desirable (`S`).
4. Further data are the initial stock at the first period for each player (`Init`).
5. The main data the be calculated are: the Order issues in each period from each player the up-stream (`O`) and the resulting Inventory level in each period for each player (`I`).
6. The Orders are decisions to take at each period: How much should I order to minimize my costs? If each player knows the end demand from the customer, then he can take the politic to order the average demand in each period. This would probably minimize the Inventory overall and for each player.
7. However, normally the players use their own strategies to fill the stock. One such strategy could be the s-S-rule: "If the stock is below the amount of `s` then issue an order that fill it up to `S`, with $s \leq S$.
8. In our case, we also consider back log orders, that is orders that cannot be fulfilled in a period must be fulfilled at a later time period (at heavy

costs). (we can model this fact as negative Inventories.)

9. The rule implemented in the model is: use the s-S rule, however look at the order issues in the previous period and deduce it from the present order (this avoids to have to much stock after the time lag. That is, $O_{p,i} = \Longleftrightarrow\ (I_{p,i} + O_{p,i-1} \leq s, S - I_{p,i} - O_{p,i-1})$.
10. The Inventory then is calculated as follows: $I_{p,i} = \Longleftrightarrow\ (i = 1, Init_p, I_{p,i-1}) + O_{p,i-3} - demand_i$ for the retailer and $I_{p,i} = \Longleftrightarrow\ (i = 1, Init_p, I_{p,i-1}) + O_{p,i-3} + O_{p-1,i-1}$ where we think of $p - 1$ the next player up-stream.

Listing 5.5: The Main Model implemented in LPL [22]

```
model BeerGame "The Beer Game";
  set p := [Retailer WholeSaler Distributor Factory ];
      i := [1..100]; k:=[Order Inv];
  parameter demand{i} "Market Demand at a period i";
            s{p}      "Place order if Inventory is below"
                ;
            S{p}      "Order to get that of Inventory";
            Init{p}   "Stock at the first period";
            O{p,i}    "Order issues";
            I{p,i}    "Inventory of p at period i";
            T{p,k,i}  "Collect all data in one table";
            dem       "Average demand in a period";;
    data;
    {i,p}(I[p,i] := if(i=1,Init,I[p,i-1]) + O[p,i-3] - if
        (p=1,demand,O[p-1,i-1]) ,
          O[p,i] := if(I+O[p,i-1]<s,S-I-O[p,i-1]));
          --O[p,i] := 2); -- alternative strategy
    {p,i} (T[p,'Order',i] := O,
           T[p,'Inv'  ,i] := I);
    dem:= sum{i}demand/#i "Average demand in a period";
end
```

Solution: The solutions to the two order strategies are given in the following figures.

Figure 5.2: Local s-S strategy of ordering

Figure 5.3: Demand estimated strategy of ordering

Further Comments: The beer game show very well that a supply chain is not inherently stable, as one can see from the results. Demand variability in-

creases as one moves up the supply chain away from the retail customer, and small changes in consumer demand can result in large variations in orders placed upstream. Eventually, the network can oscillate in very large swings as each organization in the supply chain seeks to solve the problem from its own perspective. This phenomenon is known as the **bullwhip effect** and has been observed across most industries, resulting in increased cost and poorer service.

The following reasons contribute to the bullwhip effect:

1. Overreaction to backlogs
2. No communication nor coordination up and down the supply chain
3. Local decision in ordering in order to reduce inventory
4. Order batching, that is, larger orders in an effort to reduce ordering costs
5. Demand forecast inaccuracies.
6. Larger delay in delivery down the chain, an delay placing orders up the chain.

One way to reduce the bullwhip effect is through better information, either in the form of improved communication along the supply chain or (presumably) better forecasts. Because managers realize that end-user demand is more predictable than the demand experienced by factories, they attempt to ignore signals being sent through the supply chain and instead focus on the end-user demand. This approach ignores day-to-day fluctuations in favor of running level. (In the present model, for example, it would substantially reduce the inventories for **all** players, if each player only orders up-stream the mean demand from costumers, independently what the successors or follower order. Try: $O_{p,i} = 2$.)

Another solution is to reduce or eliminate the delays along the supply chain. In both real supply chains and simulations of supply chains, cutting order-to-delivery time by half can cut supply chain fluctuations by 80reduced inventory carry costs, operating costs also decline because less capacity is needed to handle extreme demand fluctuations.

5.5. Agriculture Production (crop)

— Run LPL Code , HTML Document –

Problem: see Katz S. (IBM Systems Journal 19:4,1980, p.505-520, Appendix B)

Listing 5.6: The Complete Model implemented in LPL [22]

```
model CROP "Agriculture Production";
  set
    Month,m := [ May June July ];
    Crop,c  := [ Cotton Onion Pear Avoccado ];
    Field,f := [ Cotton  Onion ];
  parameter
    WaterBND{m} := [200000 260000 270000];
    Labor{c} := [2.9  2.7 1.0 1.5];
    Profit{c} := [6453 6110 4814 8813];
    Ceils{c}:= [2000 250 500 800];
    Land:=2700; FieldLand:=1850; LaborTot:=5850;
    Water{c,m}:=[65 80 90 . 60 . . 53 64 . 75 85];
  variable  CropVar{c} [0..Ceils];
  constraint
    r1: sum{c} CropVar <= Land;
    r2: sum{f} CropVar[f within Crop] <= FieldLand;
    wat{m}: sum{c}  Water*CropVar <= WaterBND;
    r3: sum{c}  Labor*CropVar <= LaborTot;
  maximize prof: sum{c}  Profit*CropVar;
  Writep(prof,CropVar);
end
```

5.6. Import and Export Goods (export)

—- Run LPL Code , HTML Document –

Problem: A country has four industries: Steel, Automotive, Electronics and Plastics. There unit prices are: 500, 1500, 300 and 1200. The total available Manpower is 830,000 year-hours. No more than 650,000 units of Automotive and 60,000 units of Plastic can be produced. Production of 1 unit of Steel requires 0.02 unit of Automotive, 0.01 units of Plastics, 250 currency units of imported raw matrials and 0.5 manpower unit. Production of 1 unit of Automotive reqires 0.8 unit of steel, 0.15 units of electronics 0.11 units of plastic, 300 units of imported materials and 1 manpower unit. Production of 1 unit of Electronics reqires 0.01 unit of steel, 0.01 units of automotive, 0.05 units of plastic half a man years of labor and 50 units of imported materials. Production of 1 unit of plastics requires .03 units of automotive production,

1. units of steel, 0.05 units of electronics, 2 man years of labor and
2. units of imported materials. The economic minister would like to maximize the exceeding export value.

Schrage L.,LINDO, 1986, p.116f."; *References*

Listing 5.7: The Complete Model implemented in LPL [22]

```
model EXPORT "Import and Export Goods";
  set i,j := [ steel automotive electronic plastics];
  parameter prices{i}:=[500 1500 300 1200];
    ImportCosts{i}:=[250 300 50 300];
    ManPowerUsed{i}:=[.5 1 .5 2];
    Limits{i}:=[. 650000 . 60000];
    ManPower := 830000;
    Inputs{i,j} :=
       [. .8 .01 .2 , .02 . .01 .03 , . .15 . .05 , .01
          .11 .05  . ];
  variable P{i} "PRODUCTION"; E{i} "EXPORTS";
  constraint
    A{i}: P = E + sum{j} Inputs[i,j]*P[j];
    B: sum{i} ManPowerUsed*P = ManPower;
    C{i|Limits}: P <= Limits;
  maximize obj: sum{i} ( prices*E - ImportCosts*P );
  Writep(obj,P,E);
end
```

5.7. A Small Portfolio Model (lin193)

— Run LPL Code , HTML Document –

Model from SCHRAGE, Lindo , page 193.

Listing 5.8: The Complete Model implemented in LPL [22]

```
model LIN193 "A Small Portfolio Model";
  variable x; y; z; budget; yield; xfrac; yfrac; zfrac;
  constraint
    R1: 6*x+2*y-z + budget-1.3*yield+xfrac >= 0;
    R2: 2*x+4*y-0.8*z + budget-1.2*yield+yfrac >= 0;
    R3: -x-0.8*y+2*z + budget-1.08*yield+zfrac >= 0;
    R4: x+y+z=1;
    R5: 1.3*x+1.2*y+1.08*z >= 1.12;
    R6: x <= 0.75;
    R7: y <= 0.75;
    R8: z <= 0.75;
  minimize obj: x+y+z+budget+yield+xfrac+yfrac+zfrac;
  Writep(obj,x,y,z,budget,yield,xfrac,yfrac,zfrac);
end
```

5.8. Exposures in Media Vehicles (media)

—- Run LPL Code , HTML Document –

Problem: An advertising agency plans to select the media mix that will maximize the number of effective exposures subject to a set of constraints. There are three media vehicles A, B and C. Vehicle A gives 3,100 (in thousands) effective exposures per issue , vehicle B gives 2,000, and vehicle C gives 2,400. The media planner has an advertising budget of $500,000, which cannot be exceeded. Vehicle A costs $15,000 per issue, vehicle B, $4,000, and vehicle C, $5,000. Furthermore, the media planner wants to spend at least $250,000 on vehicle A. Vehicle A puts out 52 issues a year, vehicle B, eight issues, and vehicle C, twelve issues. The media planner wants to buy at least one issue of vehicle B and six issues of vehicle C.

Listing 5.9: The Complete Model implemented in LPL [22]

```
model MEDIA "Exposures in Media Vehicles";
  set  media :=  [A B C ];
  parameter ExposureCost{media} := [ 15000 4000 5000 ];
    Issues{media} := [ 3100 2000 2400 ];
    PutsOutPerYear{media}   := [ 52 8 12 ];
    MinimalExposures{media} := [  . 1  6 ];
    Budget := 500000;
    MinimalBudgetForA := 250000;
  variable  NumberOfExposures{media} [MinimalExposures..
      PutsOutPerYear];
  constraint
    r1: sum{media} ExposureCost*NumberOfExposures <=
        Budget;
    r2: ExposureCost['A']*NumberOfExposures['A'] >=
        MinimalBudgetForA;
  maximize number: sum{media} Issues*NumberOfExposures;
  Writep(number,NumberOfExposures);
end
```

5.9. Mexican Steel Industry (mexican)

— Run LPL Code , HTML Document –

Model from MATURANA S. V., Survey and Analysis of LINGO, Mgt 298D Course Project; John E. Anderson Graduate School of Management, University of California, Los Angeles, CA 90024, p.23f

Listing 5.10: The Complete Model implemented in LPL [22]

```
model Mexican "Mexican Steel Industry";
set
  plants := [AHMSA FUNDIDOR SICARTSA HYLSA HYLSAP];
  markets := [MexicoDF Monterey Guadalaj];
  cf := [STEEL];
  ci := [ SPONGE PIGIRON ];
  cr := [PELLETS COKE NATGAS ELECTRIC SCRAP];
  process :=[pigironp spongep steeloh steelel steelbof];
  produnit := [blastfrn ofenhrth bof directed elecarc];
parameter
  rd2{plants}:=[739 521 . 521 315];
  dd{markets}:=[55 30 15];
  rd3{markets}:=[428 521 300];
  pv{cf}:=[150]; pe{cf}:=[140]; eb{cf}:=[1];
  pd{cr}:=[18.7 52.17 14 24 105];
parameter
  DT  := 5.209;
  rse := 40;
  k{produnit,plants} := [ 3.25 1.40 1.1 0    0
                          1.5  0.85 0   0    0
                          2.07 1.5  1.3 0    0
                          0    0    0   0.98 1
                          0    0    0   1.13 0.56 ];
  rd1{plants,markets} :=
       [ 1204  218 1125,  1017    . 1030,
          819 1305  704,  1017    . 1030,
          185 1085  760                   ];
  b{produnit,process} :=
    [ 1 0 0 0 0,0 0 1 0 0,0 0 0 0 1,0 1 0 0 0,0 0 0 1 0
      ];
  a1{cr,process} := [ -1.58 -1.38     0    0     0
                      -0.63  0        0    0     0
                       0    -0.57     0    0     0
                       0     0        0   -0.58  0
                       0     0    -0.33    0    -0.12 ];
  a2{ci,process} := [  1.00  0    -0.77    0    -0.95
                       0     1.00  0      -1.09  0    ];
  a3{cf,process} := [  0     0     1.0     1.0   1.0  ];
variable
  phipsi; philam; phipi; phieps;
```

```
  z{process,plants};
  u{cr,plants};
  e{cf,plants};
  d{cf,markets};
  v{cf,markets};
  x{cf,plants,markets};
constraint
  r1{cf,markets}: d = DT*(1+rse/100)*dd/100;
  mbf{cf,plants}: sum{process} a3*z >= sum{markets} x +e;
  mbi{ci,plants}: sum{process} a2*z >= 0;
  mbr{cr,plants}: sum{process} a1*z + u >= 0;
  cc{produnit,plants}:  sum{process} b*z <= k;
  mr{cf,markets}:  sum{plants} x+v >= d;
  me{cf}:  sum{plants} e <= eb;
  r2: phipsi = sum{cr,plants} pd*u;
  r3: philam = sum{cf,plants,markets} if(rd1 , (0.0084*
      rd1+2.48)*x)
           + sum{cf,plants} if(rd2,(0.0084*rd2+2.48)*e)
           + sum{cf,markets} if(rd3,(0.0084*rd3+2.48)*v);
  r4: phipi = sum{cf,markets} pv*v;
  r5: phieps = sum{cf,plants} pe*e;
minimize phi: phipsi + philam + phipi - phieps;
Writep(phi);
end
```

5.10. A Production Planning Model (omp)

— Run LPL Code , HTML Document –

Problem: An application of aggregate multiperiod production planning: A company has 5 products to manufacture on 5 machine types. How much should be manufactured, stored, and sold in each period over the next 6 periods (6 months) in order to maximize the contribution. The production capacity on each machine is limited, as well as the sales. The prices of the products are given. At the end of the six periods the company wants a stock of 50 units of each product in its inventory. The inventory is empty at the beginning of the first period. The only costs are the inventory: each unit of each products costs 0.5 (money-unit) in each period. (The model is from: O.M.P. Optimisation, an Application of the Model Generator to Aggregate Production Planning, A brochure of Beyers and Partners, pp. 3-11.)

Modeling Steps: The data of the problem are given as follows.

1. There are three sets: the time periods (t), the products (p), and the machines (m).
2. The prices for each product is $Price_p$.
3. $Time_{m,p}$ is the time (in hours) that a machine m takes to manufacture one unit of product p.
4. $Limit_{t,p}$ is the upper limit of sales of a product p in period t.
5. $Capacity_{t,m}$ is the quantity of any product a machine m can manufacture in period t.
6. We introduce three variables: the quantity of products p that are manufactured in period t: $Manuf_{p,t}$; the quantity sold of product p in period t: $Sales_{p,t}$, and the quantity stored of product p at the end of time period t: $Stock_{p,t}$.
7. The inventory at the end of the last period must be 50:

$$Stock_{p,JUN} = 50 \quad \text{forall Periods} \quad p$$

8. The proction should not exceed the capacities of the machines:

$$\sum_p Time_{m,p} \cdot Manuf_{p,t} \leq Capacity_{t,m} \quad \texttt{forall} \quad (m,t)$$

9. The product balance over the periods must hold:

$$Manuf_{p,t} + Stock_{p,t-1} = Stock_{p,t} + Sales_{p,t} \quad \texttt{forall} \quad (p,t)$$

10. The contribution (to be maximized) is:

$$\sum_{p,t} (\text{Price}_p \cdot \text{Sales}_{p,t} - 0.5 \cdot \text{Stock}_{p,t})$$

Listing 5.11: The Complete Model implemented in LPL [22]

```
model OMP "A Production Planning Model";
  set
    t  := [ JAN FEB MAR APR MAY JUN ] "Time periods";
    p  := [ Prod1 Prod2 Prod3 Prod4 Prod5 ] "Products";
    m  := [ MACHA MACHB MACHC MACHD MACHE ] "machines";
  parameter
    Price{p}   := [ 10  6  8  4 11 ];
    Time {m,p} "Production time unit of a product at a
        machine" :=
          [ 0.5   0.7     .     .    0.3
            0.1   0.2     .   0.3     .
            0.2     .   0.8     .     .
            0.05  0.03    .   0.07   0.1
              .     .  0.01     .   0.05 ];
    Limit{t,p} "Upper bound on sales of each product in
        period" :=
          [ 500 1000 300 300  800
            600  500 200 100  400
            300  600 200 200  500
            200  300 400 500  200
            400  100 500 100 1000
            500  500 100 300 1100 ];
     Capacity{t,m} "Available capacity (in hours) on a
         machine" :=
          [ 1152  768 1152  384  384
            1536  768  384  384  384
            1536  768 1152    0  384
            1536  384 1152  384  384
            1152  768  768  384  384
            1536  768 1152  384    0 ];
  variable
    Manuf{p,t} "Quantity manufactured";
    Stock{p,t} [0..100] "Quantity in stock (at the end of
         period)";
    Sales{p,t} [0..Limit] "Quantity to sold";
  constraint
    tjune{p}: Stock[p,'JUN'] = 50;
    c1{m,t}:  sum{p} Time*Manuf <= Capacity;
    c2{p,t}: Manuf + Stock[p,t-1] = Stock + Sales;
  maximize z: sum{p,t} (Price*Sales - 0.50*Stock);
  Write('Total contribution  is : %9.2f\n', z);
  for{p} do
```

```
    Write('  Product: %5s
    /-------------------------------------------------/
    / Period |   Manufact. |   Stock     |    Sales   /
    /-------------------------------------------------/\n
        ',p);
    Write{t}('    /   %3s  |  %9.2f  | %8.2f    |  %8.1f
         /\n',
      t,Manuf,Stock,Sales) , ' ');
    Write('    /--
        ------------------------------------------------/\
        n');
  end
  parameter A{m,p,t}:=Time*Manuf;
end
```

Questions

1. Suppose that machine MACHE has broken down at the end of the first period and cannot be replaced. What is the main consequence for the production? What is the new contribution?
2. The model did not consider any production costs (only inventory cost were considered). How could we take into account production cost, say, to manufacture one unit of a product costs 5 (for all products). Model it!

Answers

1. Product 3 and five are not manufactured anymore after period 1. Why? The contribution is: 54592.14. To achieve this, the capacities of MACHE must be set to zero after the first period. Just set:

```
{t|t>1} (Capacity[t,'MACHE']:=0);
```

2. All we need is to modify the objective function as follows:

```
maximize z: sum{p,t} (Price*Sales - 5*Manuf -
    0.50*Stock);
```

5.11. One Machine Scheduling (ordon)

—- Run LPL Code , HTML Document –

ref Groeflin H. Course DSS II

Listing 5.12: The Complete Model implemented in LPL [22]

```
model ordon "One Machine Scheduling";
  set i,j := [1..7]  "jobs";
  parameter a{i} := [10 15 8 20 30 0 30] "earliest time";
            d{i} := [5 6 7 4 3 6 2]  "duration";
            q{i} := [7 26 24 21 6 17 0] "tail time";
  variable t{i} [a..1000] "starting time";
  constraint A{i,j|i<j}: t[j]-t[i] >= d[i] or t[i]-t[j]
      >= d[j];
  minimize obj: max{i} (t+d+q);
  Draw.Scale(10,10);
  {i} Draw.Rect(i&'',t,1,d,2);
  {i|t+d+q=obj} Draw.Line(t+d,2.7,t+d+q,2.7);
  Draw.Text(obj&'',obj,1);
end
```

5.12. A Transport Model Fragment (pam)

— Run LPL Code , HTML Document –

Model from Creegan J.B.jr., PAM Overview";

Listing 5.13: The Complete Model implemented in LPL [22]

```
model PAM "A Transport Model Fragment";
  set
    source,s      := [ SS  HD ];
    destination,d := [ BA  PH  WA  RI ];
    product,p     := [ P1  P2 P3 ];   --not specified in
        the example
   parameter
    trcost{s,d} := [ 3.52  9.47  0.38  8.63
                     2.04  6.61  7.22  9.97 ];
    prcost{s,p} :=  [ 4.22  5.05  4.60 , 1.45  2.45  2.03
         ];
    prCap{s}    :=  [ 5000  6000 ];
    orders{d}   :=  [ 120  100   30  234 ];
  variable
    Produce,PR{p,s};
    Ship,SH{p,s,d};
  constraint
    Capacity{s} : sum{p}  PR <= prCap;
    Balance{p,s}: PR = sum{d} SH;
    Demand{p,d} : sum{s} SH >= orders;
  minimize costs: sum{p,s}(prcost*PR + sum{d} trcost*SH);
  Writep(costs,PR,SH);
end
```

5.13. Production Planning (planning)

—- Run LPL Code , HTML Document –

Further Comments: Compare with MPL

Listing 5.14: The Complete Model implemented in LPL [22]

```
model PLANNING "Production Planning";
set product,p := [1..3];
    month,m   := [January,February,March,April,May,June,
        July,
                  August,September,October,November,
                      December];
parameter
    price{p}       :=  [105.09, 234.00, 800.00];
    Demand {m,p}   := [ -- insert them here instead of
        extra file
  50 24  5, 60 30  6, 70 36  7, 80 42  8, 90 48  9,
 100 52 10,110 50 11,120 48 12,120 44 13,120 40 14,
 110 36 14,100 32 13 ];
    ProductionCapacity{p}  :=  [10000, 42000, 14000 ] ;
    ProductionCost{p}      :=  [64.30, 188.10, 653.20] ;
    InventoryCost          :=  8.8 ;
variable
    Inventory,I{p,m};
    Production,P{p,m};
    Sales,S{p,m};
constraint
   InventoryBalance{p,m}: I = I[p,m-1] + P - S;
   Bn1{p,m}: S <= 1000*Demand ;
   Bn2{p,m}: P  <=  ProductionCapacity ;
   MaxInventory{p,m | m<>#m}: I <= 90000 ;
   CloseInventory{p}: I[p,'December'] = 20000 ;
maximize Profit: sum{p,m} (price*S-InventoryCost*I-
    ProductionCost*P);
Writep(Profit,I,P,S);
end
```

5.14. Multiperiod Production (prod)

—- Run LPL Code , HTML Document –

This model is a multiperiod production model from [Fourer al. 1990]. The formulation was motivated by the experiences of a large producer in the United States. There exists an AMPL formulation and the reader may compare both, the LPL and the AMPL formulation. This model, with three products and 13 periods, consists of 235 variables, constraints and 896 non-zero entries.

Problem: This model determines a series of workforce levels that will most economically meet demands and inventory requirements over time, while minimizing overall costs. There are wage costs (regular and overtime), hiring costs, layoff costs, demand shortage costs, and inventory costs. How many workers should be employed, on a regular or overtime basis, hired or fired at each period?

Modeling Steps:

Listing 5.15: The Complete Model implemented in LPL [22]

```
model Production "Multiperiod Production";
set
  t,q "Periods"  := [ p0 p1 p2 p3 p4 p5 p6 p7 p8 p9 p10
      p11 p12 p13 p14];
  t1{t}          := [    p1 p2 p3 p4 p5 p6 p7 p8 p9 p10
      p11 p12 p13    ];
  t2{q}          := [    p1 p2 p3 p4 p5 p6 p7 p8 p9 p10
      p11 p12 p13    ];
  p "Products"   := [reg18 reg24 pro24];
  life,l         := [1..2];
parameter
  dpp{t1} "working days" := [0 19.5 19 20 19 19.5 19 19
      20 19 20 20 18 18];
  ol{t1}  "overtime limit" := 96;
  cmin{t1}"min crew size"  := 0;
  cmax{t1}"max crew size"  := 8;
  hc{t1}  "hiring costs" := [0 7500 7500 7500 7500 15000
      15000 15000 15000 15000 15000 7500 7500 7500];
  lc{t1}  "layoff costs per period" := hc;
  pt{p}:= [1.194 1.509 1.509] "production time";
  pc{p}:= [2304 2920 2910] "costs";
  cri{p}:= [0.015  0.015  0.015] "inventory costs";
  crs{p}:= [1.1  1.1  1.1] "shortage costs";
  iinv{p}:= [82  792.2  0] "initial inventory";
  rtr "wage per hour in regular time" := 16;
  otr "wage per hour in overtime" := 43.85;
  rir "regular inventory ratio" :=  0.75;
```

```
  pir "promotional inventory ratio" :=  0.8;
  sl  "regular-time hours per shift" :=  8;
  cs  "crew size" := 18;
  iw  "initial workforce" :=  8;
  dem{t,p} "demand" := /
        :          reg18  reg24  pro24    :
    p1              63.8   1212      .
    p2              76      306.2    .
    p3              88.4    319      .
    p4             913.8    208.4    .
    p5             115      298      .
    p6             133.8    328.2    .
    p7              79.6    959.6    .
    p8             111      257.6    .
    p9             121.6    335.6    .
    p10            470      118     1102
    p11             78.4    284.8    .
    p12             99.4    970      .
    p13            140.4    343.8    .
    p14             63.8   1212      .     /;
  pro{t,p} "promoted products in a period (=1)" :=
   / p1  reg24  1
     p4  reg18  1
     p7  reg24  1
     p10 reg18  1
     p10 pro24  1
     p13 reg24  1
     p14 reg24  1 /;
  iil{t1,p}:=Max(iinv - sum{t2|t2<=t1} dem[t2,p],0)  "
      initial inventory left";
  minv{t in t1,p} := dem[t+1,p]*if(pro[t+1,p],pir,rir)  "
      minimum inventory";
variable
  Crews{t|t<#t} "Average number of crews employed";
  Hire{t1}      "Crews hired from previous to current
      period";
  Layoff{t1}    "Crews laid off from previous to current
      period";
  Rprd{t1,p}    "Regular production";
  Oprd{t1,p}    "Overtime production";
  Inv{t1,p,life|life<t1}  "Inventory";
  Shortage{t1,p}  "Accumulated unsatisfied demand";
constraint
  rlim{t1}: sum{p} pt*Rprd <= sl*dpp[t1]*Crews[t1]  "
      Regular time limits";
  olim{t1}: sum{p} pt*Oprd <= ol[t1]  "Overtime limits";
  empl0: Crews['p0'] = iw  "initial crew level";
  empl{t1}: Crews[t1] = Crews[t1-1] + Hire[t1] - Layoff[
      t1]  "Crew levels";
  Bounds{t1}: cmin[t1] <= Crews[t1] <= cmax[t1]  "Crew
```

```
      limits";
  dreq{t1,p}: Rprd + Oprd + Shortage[t1,p] - Shortage[t1
      -1,p]
    + sum{l} ( Inv[t1-1,p,l] - Inv[t1,p,l] ) = dem[t1,p]-
       Max(iil[t1,p],0)
      "Demand requirement";
  ireq{t1,p}: sum{life}  Inv + iil >= minv   "Inventory
      requirements";
  ilim1{t1,p}: Inv[t1,p,1] <= Rprd + Oprd   "New-
      inventory limits";
  alim{t1,p,l | t1>1 and l>1}: Inv <= Inv[t1-1,p,l-1] "
      Inventory limits";
minimize cost: sum{t1} (rtr*sl*dpp[t1]*cs*Crews[t1] + hc[
    t1]*Hire[t1]
    + lc[t1]*Layoff[t1]) + sum{t1,p} (otr*cs*pt*Oprd +
       crs*pc*Shortage)
    + sum{t1,p,life} cri*pc*Inv
      "Regular wages+hired and layoff costs+overtime
          wages+shortage and inventory costs";
Write('\nperiod   Crews     Hire        Layoff\n');
Write{t1}(' %5s     %3d      %3d          %4d\n', t1,
    Crews[t1],Hire[t1],Layoff[t1]);
Writep(cost,Rprd,Oprd,Shortage,Inv);
end
```

Model from: FOURER R. and GAY D.M. and KERNIGHAN B.W., [1990], A Modeling Languagef for Mathematical Programming, in: Management Science, Vol. 36, p. 519ff.

5.15. Production and Planning (prodplan)

—- Run LPL Code , HTML Document –

Problem: A small prodution problem: ...

Modeling Steps:

Listing 5.16: The Complete Model implemented in LPL [22]

```
model PRODPLAN "Production and Planning";
set
    product,p :=  [1..5];
    month  ,m :=  [Jan,Feb,Mar,Apr,May,Jun];
    machine,r :=  [Grind,Vdrill,Hdrill,Boring,Planing];
parameter
    price {p}  := [ 10.00   6.00   8.00   4.00  11.00 ];
    time {r,p} := [
            50   70    0    0   30
            10   20    0   30    0
            20    0   80    0    0
             5    3    0    7   10
             0    0    1    0    5];
    MaxSales {m,p} := [
             500  1000   300   300   800
             600   500   200   100   400
             300   600   200   200   500
             200   300   400   500   200
             400   100   500   100  1000
             500   500   100   300  1100];
    DaysPerMonth   :=  6*4;
    HoursPerDay    :=  2*8;
    HoursPerMonth  :=  HoursPerDay * DaysPerMonth ;
    MachineHours {m,r}  := [ 3, 2, 3, 3, 3,
                             4, 2, 1, 3, 3,
                             4, 2, 3, 2, 3,
                             4, 1, 3, 3, 3,
                             3, 1, 3, 3, 3,
                             4, 2, 2, 3, 2 ];
    InventoryCost   :=   0.50;
variable
    Sale {p,m};
    Produ {p,m};
    Invt {p,m};
constraint
  Capacity{m,r}: sum{p} time/100*Produ <= HoursPerMonth*
     MachineHours;
  Balance{p,m}:  Invt = Invt[p,m-1] + Produ - Sale;
```

```
  Bound {p,m} :  Sale <= MaxSales;
  InvtCapacity {p,m} : if(m='Jun' , Invt=50 , Invt<=100);
maximize Contribution: sum{p,m} (price*Sale-InventoryCost
    *Invt);
Writep(Contribution,Sale,Produ,Invt);
end
```

5.16. Production and Shipment Planning (prodship)

—- Run LPL Code , HTML Document –

Problem: There are a number of plants, each manufacturing several products. The products are shipped to various warehouses from where they are distributed to demand-centers or customers. Each demand-center is supplied from only one warehouse. The question is how much, and at which plants, to produce a given commodity in order to meet the demand requirements at the demand-centers while minimizing production and transportation costs. Figure 5.4 gives a overview of the overall product flow, (The model was designed by: [7]).

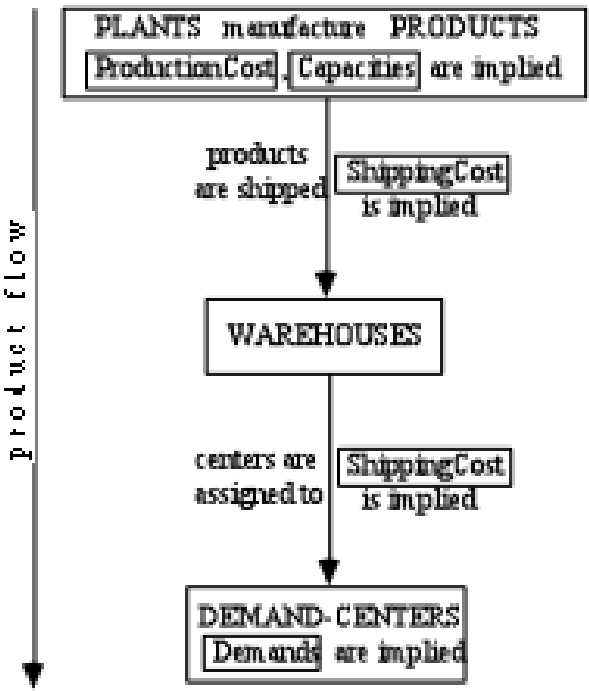

Figure 5.4: Product Flow Description

Modeling Steps:

1. ...

Listing 5.17: The Complete Model implemented in LPL [22]

```
model ProdShip "Production and Shipment Planning";
  set
    plants,f      "A set of plants (factories)";
    products,p    "A set of products";
    warehouses,w  "A set of warehouses";
    centers,c     "A set of demand-centers";
  parameter
    CAPACITY{f,p} "The quantity that can maximally be
        produced";
    PCOST {f,p} "The production cost";
    SCOST{f,w}  "The shipping cost (plant to warehouse)";
    TCOST{w,c}  "The transshipping cost (warehouse to
        centers)";
    DEMAND{c,p} "Demands at the centers";
  variable
    PRODUCE{f,p|PCOST} [0..CAPACITY] "The quantity to
        produce (cannot exceed the capacity)";
    SHIP{f,w,p|PCOST and SCOST}     "The quantity to be
        shipped";
  binary ASSIGN{w,c|TCOST}          "=1, if the center is
      assigned to the warehouse, otherwise 0";
  constraint
    PRODROW{f,p|PCOST} :sum{w} SHIP - PRODUCE = 0
      "All produced quantities must be shipped to the
          warehouses";
  SHIPROW{f,w,p|SCOST}: sum{c} DEMAND*ASSIGN - sum{f}
      SHIP = 0
      "The demands must be satisfied (nothing must be
          stored)";
  CENTROW{c}: sum{w} ASSIGN = 1
      "A center can be supplied from only one warehouse";
  minimize COST: sum{f,p} PCOST*PRODUCE + sum{f,w,p}
      SCOST*SHIP
                + sum{w,c} TCOST*(sum{p} DEMAND)*ASSIGN
      "The overall cost, i.e. production, shipping and
          transshipping cost";
  ----- data sets
  model data1;
  /*  data1: OPT=3290215.695, Var:720, Const:601,Ints
      :169,
              NonZeroes:12751, Density:2.9467% */
    f  :=  [1..5];
    p  :=  [1..11];
    w  :=  [1..10];
    c  :=  [1..20];
    CAPACITY{f,p} := if(Rnd(0,1)<0.95 , Rnd(500,2200));
    PCOST{f,p}    := if(CAPACITY , Rnd(200,300));
    SCOST{f,w}    := if(Rnd(0,1)<0.95 , Rnd(1,20));
    TCOST{w,c}    := if(Rnd(0,1)<0.90 , Rnd(15,70));
    DEMAND{c,p}   := Rnd(20,100);
```

```
end
model data2;
  /* data2: OPT= , Var:1613, Const:1521, Ints:181,
            NonZeroes: 44071, density: 1.7963% */
  f  :=  [1..13];
  p  :=  [1..11];
  w  :=  [1..10];
  c  :=  [1..20];
  CAPACITY{f,p} := if(Rnd(0,1)<0.95 , Rnd(500,2200));
  PCOST{f,p}    := if(CAPACITY , Rnd(200,300));
  SCOST{f,w}    := if(Rnd(0,1)<0.95 , Rnd(1,20));
  TCOST{w,c}    := if(Rnd(0,1)<0.90 , Rnd(15,70));
  DEMAND{c,p}   := Rnd(20,100);
  /* different parameters
  CAPACITY{f,p} := if(Rnd(0,1)<0.95 , Rnd(500,2200));
  PCOST{f,p}    := if(CAPACITY , Rnd(200,300));
  SCOST{f,w}    := if(Rnd(0,1)<0.65 , Rnd(1,20));
  TCOST{w,c}    := if(Rnd(0,1)<0.75 , Rnd(15,70));
  DEMAND{c,p}   := Rnd(20,100); */
end
model data3;
  f  :=  [1..13];
  p  :=  [1..20];
  w  :=  [1..10];
  c  :=  [1..20];
  CAPACITY{f,p} := if(Rnd(0,1)<0.95 , Rnd(500,2200));
  PCOST{f,p}    := if(CAPACITY , Rnd(200,300));
  SCOST{f,w}    := if(Rnd(0,1)<0.95 , Rnd(1,20));
  TCOST{w,c}    := if(Rnd(0,1)<0.90 , Rnd(15,70));
  DEMAND{c,p}   := Rnd(20,100);
end
model data4;
  f := [ topeka ny ];
  p := [ chips nachos ];
  w := [ topeka ny ];
  c := [ east south west ];
  CAPACITY{f,p} := /
     topeka   chips   200
     topeka   nachos  800
     ny       chips   600/;
  PCOST{f,p}:=/
     topeka   chips   230
     topeka   nachos  280
     ny       chips   255  /;
  SCOST{f,w} := /
     topeka   topeka   1
     topeka   ny       45
     ny       ny       2
     ny       topeka   45  /;
  TCOST{w,c} := /
```

```
      topeka       east    60
      topeka       south   30
      topeka       west    40
      ny           east    10
      ny           south   30
      ny           west    80  /;
    DEMAND{c,p} := /
      east   chips    200
      east   nachos    50
      south  chips    250
      south  nachos    80
      west   chips    150
      west   nachos   300  /;
  end
  model data;
    f :=  [1..10];
    p :=  [1..7];
    w :=  [1..20];
    c :=  [1..20];
    CAPACITY{f,p} := if(Rnd(0,1)<0.85 , Rnd(200,800));
    PCOST{f,p}    := if(CAPACITY , Rnd(200,300));
    SCOST{f,w}    := if(Rnd(0,1)<0.65 , Rnd(1,20));
    TCOST{w,c}    := if(Rnd(0,1)<0.75 , Rnd(15,70));
    DEMAND{c,p}   := Rnd(50,250);
  end
  model output;
    Write('Total costs are:     %15.2f\n', COST);
    Write('Assignment:\n   warehouse    assigned centers\
        n\
            ------------------------------------\n');
    Write{w}('   %6s       %5s\n', w, {c|ASSIGN}c);
    Writep(PRODUCE,SHIP,ASSIGN);
  end
end
```

5.17. A Production Model (produ)

—- Run LPL Code , HTML Document –

Problem:

Model from: FOURER R., A Modeling Language for Mathematical Programming, 1987

Modeling Steps:

Listing 5.18: The Complete Model implemented in LPL [22]

```
model PRODU "A Production Model";
  set
    p  "products";
    r  "raw materials";
    t  "number of storage periods";
  parameter
    initStock{r} "maximum initial stock";
    cost{r}      "raw material cost";
    value{r}     "estimated residual value or a disposal
        cost (if < 0)";
    maxPrd       "maximum unit of production";
    units{r,p}   "quantity of raw material needed to
        produce a product";
    profit{p,t|t<>#t} "estimated profit or disposal cost
        (if < 0) of a unit of product in each period
        except the last one";
  variable
    X{p,t | t<>#t} "number of units of product
        manufactured in each period except the last one";
    S{r,t}         "number of units of raw material in
        storage at the beginning of each period";
  constraint
    limit{t | t<>#t}: sum{p} X <= maxPrd
      "total production in each period is limited";
    start{r}: S[r,1] <= initStock
      "storage on raw materials in first period is
          limited";
    balance{r,t | t<>#t}: S[r,t+1] = S - sum{p} units*X
      "storage of each raw material in a period must
          equal storage in the last period plus the used
          units for production";
  maximize totalProfit: sum{t|t<>#t} (sum{p} profit*X -
      sum{r} cost*S) + sum{r} value*S[r,#t]
     "maximize the total over all periods of the
         estimated profit minus the storage costs plus
```

```
        the value of remaining raw materials after the
        last period";
  Writep(totalProfit,X,S);
  model data "an model instance";
    t :=  [1..5];
    p := [nuts bolts washers];
    r := [nickel iron];
    initStock{r}:=[7.32 35.8];
    value{r}:=[-0.01 0.025];
    cost{r}:=[0.025 0.03];
    maxPrd := 123.7;
    profit{p,t} :=[1.73  1.8  1.6  2.2  .
                   1.82  1.9  1.7  2.5  .
                   1.05  1.1  0.95 1.33 .];
    units{r,p} := [0.21  0.17  0.08
                   0.79  0.83  0.92];
  end
end
```

5.18. Multiperiod Production Scheduling (schedule-multi)

—- Run LPL Code , HTML Document –

Problem: Solve the following production planning problem by specifying how many units of product should be manufactured each period in order to minimize the sum of inventory holding c and production costs. .. There are 20 units of product in inventory at the start of period 1.

Model from: Smythe W.R.,Johnson L.A..,1966,p.202

Listing 5.19: The Complete Model implemented in LPL [22]

```
model SCHEDULE "Multiperiod Production Scheduling";
set
  t := [1..5];        -- periods
  c := [ regular,forallTime,inventory ];
parameter
  rc{t} := [120  100   90  140   90];  -- Regular Time
      Capacity, Units
  ov{t} := [ 30   40   20   40   60];  -- Overtime
      Capacity, Units
  pr{t} := [110  160  100  100  150];  -- Production
      Requirements, Units
  co{c,t} :=
      [12   12   16   10   11   -- Regular Time Cost per
          Unit
       15   16   18   14   14   -- Overtime Cost per Unit
        1    1    1    1    1]; -- Inventory costs from
            period i to i+1
variable
  PRODUCT{c,t};
constraint
  init: PRODUCT['inventory',1] = 20  "initial inventory";
  prodcapa{t}: PRODUCT['regular',t] <= rc "production
      capacity";
  capa{t}: PRODUCT['forallTime',t] <= ov  "forall time
      capacity";
  const{t}: sum{c} PRODUCT - PRODUCT['inventory',t+1] =
      pr;
minimize costs: sum{c,t} co * PRODUCT;
Writep(costs,PRODUCT,co);
end
```

5.19. Tanglewood Chair Manufacturing (tangle)

— Run LPL Code , HTML Document –

Model from : MATURANA S. V., Survey and Analysis of LINGO, Mgt 298D Course Project; John E. Anderson Graduate School of Management, University of California, Los Angeles, CA 90024, p.17f

Problem: The Tanglewood Chair Manu. Co. has four plants located around the country. The fabrication data are the following:

Plant	Cost per chair	Max production	Min production
P1	5	500	0
P2	7	750	400
P3	3	1000	500
P4	4	250	250

The wood required to make each chair is twenty pounds and is obtained from two suppliers. The company 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 dollar per pound of wood from each supplier to each plant is shown below:

	P1	P2	P3	P4
sup1	0.01	0.02	0.04	0.04
sup2	0.04	0.03	0.02	0.02

The chairs are sold in New York, Houston, San Francisco and Chicago. The transportation costs are shown below:

	NY	H	SF	C
P1	1	1	2	0
P2	3	6	7	3
P3	3	1	5	3
P4	8	2	1	4

Following data are also available:

City	Selling price	maximum demand	minimum demand
NY	20	2000	500
H	15	400	100
SF	20	1500	500
C	18	1500	500

Where should each plant buy its raw materials? How many chairs should be made at each plant? How many chairs should be sold at each city? Where should each plant ship its product?

5.20. A Travel Optimization Problem (travel)

— Run LPL Code , HTML Document –

Model fromSCICON Ltd. (or Spectrum, 1984, 6, p.125-130

Listing 5.20: The Complete Model implemented in LPL [22]

```
model TRAVEL "A Travel Optimization Problem";
set
  f := [Air Bus Train];
  s := [Air Bus Train];
  g := [Manager Stuff];
parameter
  Time1{f}:=[.8 6 10];
  Maxtr1{f}:=[10 15 100];
  Cost1{f}:=[300 100 60];
  Time2{s}:=[2.1 12 24];
  Maxtr2{s}:=[20 8 100];
  Cost2{s} :=[800 200 160];
  Numgrd{g}:=[12 18];
  Valtim{g}:=[20 15];
  Wait{f,s} := [ 7.1  1.5  2 , 0.9  5  1 , 1.3  0.6  1.5
      ];
  fareMx := 20000;
variable
  X{g,f,s | ~( g=1 and (f=3 or s=3)) };
  hours{g};   tfare;
constraint
  R2{g}: sum{f,s}  X = Numgrd;
  R3{f}: sum{g,s}  X <= Maxtr1;
  R4{s}: sum{g,f}  X <= Maxtr2;
  R5: sum{g,f,s}  (Cost1+Cost2)*X = tfare;
  R6{g}: sum{f,s}  (Time1+Wait+Time2)*X = hours;
  R7 : tfare  <= fareMx;
  /* Note, that instead of the above condition, one could
       also write:
     R8(s):  X('Manager','Train',s) = 0;  !ensures that
        manager do!
     R9(f):  X('Manager',f,'Train') = 0;  !not take the
        train!  */
minimize hMin: sum{g}  Valtim*hours + tfare;
Writep(hMin,X,hours,tfare);
end
```

5.21. A Round Trip Problem (vacation)

—- Run LPL Code , HTML Document –

Problem: Visit all towns at least once at minimal costs. (Note: this is an *incomplete* TSP (Traveling Salesman Problem)).

Listing 5.21: The Complete Model implemented in LPL [22]

```
model VACATION "A Round Trip Problem";
set
  i,j:=[GraniteCity StGeorges Cutler Chester Denver];
parameter
  Miles{i,j} "distances between towns in miles"  :=
        [   .   1600    90    85  1000
          1620    .   2000  2300   800
            95  2005    .     35  1300
           100  2400    40    .   1800
          1500   825  1250  2000    .  ];
  UnitCost "costs per unit/mile" := 1;
integer variable VISITS{i,j | i<>j};
constraint
  V{j} "Visit each location at least once" : sum{i}
      VISITS[i,j] >= 1;
  L{i} "Leave all location at most once"   : sum{j}
      VISITS[i,j] <= 1;
  T{i,j | j>i} "Insure a unique cycle tour" : VISITS[i,j]
       + VISITS[j,i] <= 1;
minimize cost: sum{i,j} Miles[i,j]*UnitCost*VISITS[i,j];
Writep(VISITS,cost);
end
```

5.22. Multi-period Blending Problem (xpress)

— Run LPL Code , HTML Document –

Problem: The following is a multi-period blending model. Production of a metal smelter is to be planned over each of the 30 weeks. Unit production costs increase with production rate. The smelted metal must satisfy certain quality requirements and metal produced in one week may be stored for sale in subsequent weeks. Objective: find the cheapest blend of ores to smelt and plan production to trade off production and storage cost.

Modeling Steps:

Listing 5.22: The Complete Model implemented in LPL [22]

```
model xpress "Multi-period Blending Problem";
   set
     i,nweeks :=  [1..30];
     nores    :=  [1.. 5];
     nlin     :=  [1..10];
     nqual    :=  [1..10];
   variable
     metal{nweeks};
     diff{nweeks};
     ore{nores,nweeks};
     lin{nlin,nweeks};
     stock{nweeks};
   parameter   -- choose random data
      demand{nweeks}    := Rnd(1,10);
      Myield{nores}     := Rnd(2,2000);
      qual{nqual,nores} := Rnd(1000,3000);
      lowerq{nqual}     := Rnd(20,30);
      upperq{nqual}     := Rnd(1000,5000);
      orecst{nores}     := Rnd(1,90);
      orebnd{nores}     := Rnd(1000,2000);
     prdlvl{nlin} := [0,10,20,30,40,50, 70,100,200, 300];
     prdcst{nlin} := [0, 5,15,30,50,80,160,295,795,1345];
     D{nweeks}    := / 1  1 /;
     maxstock     := 1000;
     maxdiff      :=  300;
     curstock     :=  10;
     curmetal     := 100;
   constraint
     yield{nweeks}:  sum{nores} Myield*ore - metal = 0;
     lqual{nqual,nweeks}: sum{nores} (qual-lowerq)*ore >=
          0;
     uqual{nqual,nweeks}: sum{nores} (qual-upperq)*ore <=
          0;
```

```
    produ{nweeks}: sum{nlin} prdlvl*lin - metal = 0;
    convex{nweeks}: sum{nlin}  lin = 1;
    smooth{nweeks }:
       metal - sum{i} metal[i-1] + diff = D*curmetal;
    malbal{nweeks}:
       -stock + metal + sum{i} stock[i-1]
            >= demand - D*curstock;
    StockBound{nweeks}: stock <= maxstock;
    DiffBounds{nweeks}: max(0,-maxdiff) <= diff  <=
        maxdiff;
    OreBound{nores,nweeks}: ore  <= orebnd;
  minimize TotalCost: sum{nweeks,nores} orecst/1000*ore
               + sum{nweeks,nlin} prdcst/1000*lin;
  Writep(TotalCost,metal,diff,stock,ore,lin);
end
```

Further Comments: The XPRESS-LP formulation: (copied from: "XPRESS-LP, a prospectus of Dash Associates").

```
. Multiperiod production/inventory furnace blending problem
.
constraint smelter   . gives the model a name
.
LET nweeks = 30    . number of periods
LET nores = 5      . number of ores can buy
LET nqual = 10     . number of ore qualities
LET nlin = 10      . number of linearisation points
.
VARIABLES          . define decision variables
metal(nweeks)      . final metal production
diff(nweeks)       . change in prod level one week to next
ore(nores,nweeks)  . raw ores to buy
lin(nlin,nweeks)   . production linearisation variables
stock(nweeks)      . stocks of smelted metal
.
TABLES             . define main data tables
demand(nweeks)     . weekly demand
qual(nqual,nores)  . ore qualities (first is metal yield)
lowerq(nqual)      . lower limits on metal qualities
upperq(nqual)      . upper limits on metal qualities
orecst(nores)      . ore cost
prdlvl(nlin)       . metal production level
prdcst(nlin)       . production cost at prdlvl
orebnd(nores)      . upper bounds on ore purchases
maxstock           . maximum metal stock level
```

```
D(nweeks)          . switch array
.
DISKDATA           . input data to major tables
demand(1) = xpress.dem
qual(1,1) = xpress.qul
lowerq(1) = xpress.lqu
upperq(1) = xpress.uqu
orecst(1) = xpress.ocs
orebnd(1) = xpress.orb
DATA                 . data assigned explicitly
prdlvl(1) = 0,10,20,30,40,50, 70,100,200, 300
prdcst(1) = 0, 5,15,30,50,80,160,295,795,1345
D(1)      = 1     . switch - all other D(t) are zero
maxstock = 100    . maximum stock level
maxdiff = 30      . maximum change in production level
curstock = 10     . current stock level
curmetal = 100    . current metal production level
.
CONSTRAINTS       . total cost and problem constraints
.
.      total cost (to be minimized)
.
totalcst: sum(t=1:nweeks,j=1:nores) &
   (orecst(j)/1000*ore(j,t)         &
   + sum(t=1:nweeks,i=1:nlin) (prdcst(i)/1000*lin(i,t) $
.
.      metal yield
.
yield(t=1:nweeks): sum(j=1:nores) qual(1,j)*ore(j,t) &
  - metal(t) = 0
.
.      lower quality limits (where applicable)
.
qual(k=2:nqual,t=1:nweeks | lowerq(k).ne.0):         &
     sum(j=1:nores) (qual(k,j)-lowerq(k))*ore(j,t)  > 0
.      upper quality limits (where applicable)
.
uqual(k=2:nqual,t=1:nweeks) | upperq(k).ne.0):       &
     sum(j=1:nores) (qual(k,j)-upperq(k))*ore(j,t)  < 0
.
.   production level (so can have piecewise linear cost)
.
prod(t=1:nweeks):  sum(i=1:nlin) prdlvl(i)*lin(i,t) &
  - metal(t)  = 0
.
```

```
convex(t=1:nweeks): sum(i=1:nlin) lin(i,t)     = 1
.
.      production smoothing
.
smooth(t=1:nweeks):                              &
  metal(t) - sum(|t.ne.1) metal(t-1) + diff(t)  &
  = D(t)*curmetal
.
.      material balance and satisfy demand
.
matbal(t=1:nweeks):                               &
  -stock(t) + metal(t) + sum(|t.ne.1) stock(t-1) &
  > demand(t) - D(t)*curstock
.
BOUNDS     . simple bounds
.
stock(t=1:nweeks)   < maxstock   . stock level
diff(t=1:nweeks)    < maxdiff    . decrease in production
diff(t=1:nweeks)    > -maxdiff   . increase in production
ore(j=1:nores,t=1:nweeks) < ore purchases
.
GENERATE     . generate the matrix
. end of XPRESS-LP formulation
```

Model from: DASH ASS. (XPRESS-LP is a trademark of DASH ASSOCIATES, Northants, UK).

5.23. Production of Chip Types (potato)

— Run LPL Code , HTML Document –

Problem:

Model from: BISSCHOP J. and ENTRIKEN R., AIMMS, The Modeling System, 1993, p.301.

Modeling Steps:

Listing 5.23: The Complete Model implemented in LPL [22]

```
model POTATO "Production of Chip Types";
  set c:= [ pla mex ];
      p:= [ slicing frying packing ];
  parameter av{p} := [345 480 330];
            ne{c}:=[2 1.5];
            pr{p,c}:=[2 4 4 5 4 2];
  variable X{c}; profit;
  constraint Req{p}: sum{c} pr*X <= av;
  maximize po: sum{c} ne*X;
  Writep(po,X);
end
```

5.24. A Distribution Problem (plants)

—- Run LPL Code , HTML Document –

Problem: A company has three plants, all of which make the same product. The product is sold to four different customers. Each customer has a given demand, the production costs at the three plants are different and each plant has also a given capacity of production. Furthermore, the transportation cost from a plant to a customer are given. The problem is to determine at which plant an order should be made. Build a model for this problem. The model is from: Smythe W.R.,Johnson L.A.,1966,p.205

Modeling Steps: Let the set of plants be $p \in P$ and the set of customers $c \in C$. The demand (order) of a customer c is dem_c, the production cost and capacity at a plant p are $cost_p$ and $capa_p$. The transportation cost from p to c is $tcost_{p,c}$.

The unknown quantity is the unique variable $Q_{p,c}$ which is the quantity to produce at plant p for customer c. Then the model is easy to write down :

$$\begin{array}{lll} \text{min} & \sum_{p\in P}\sum_{c\in C}(tcost_{p,c}+cost_p)Q_{p,c} & \\ \text{subject to} & \sum_{p\in P} Q_{p,c} = dem_c & \text{forall } c \in C \\ & \sum_{c\in C} Q_{p,c} <= capa_p & \text{forall } p \in P \end{array}$$

The objective function sums the transportation and the production costs. The first constraint fulfills the demand for each customer. The second constraint limits the production to the capacity for each plant.

(Note that the first constraint here is an equality. This is not necessary. The constraint could also be formulated as a "greater equal" inequality. Since the objective function is a minimizing function, the left hand side of constraint one will never be larger than dem_c. Inequalities are in general better than equalities for the solver.)

Listing 5.24: The Complete Model implemented in LPL [22]

```
model PLANTS "A Distribution Problem";
  set
    c "customers" := [W,X,Y,Z];
    p "plants"    := [A,B,C];
  parameter
    dem{c}  :=  [700 1500 400 500] "order size" ;
    cost{p} :=  [ 45   40  50] "production cost";
```

```
    capa{p} :=  [1000 800 1500] "available capacity";
    tcost{c,p} "transportation cost" :=
         [ 8  4  6 , 11  10  8 , 6  12  7 , 9  5  14];
  variable  Q{p,c} "quantity produced";
  constraint
    D{c} "Demand" :   sum{p} Q  = dem;
    S{p} "Supply" :   sum{c} Q <= capa;
  minimize   Cost : sum{c,p} (tcost+cost)*Q;
  Writep(Q);
end
```

5.25. Diet (hogfeed)

—- Run LPL Code , HTML Document –

Problem:

Modeling Steps:

Listing 5.25: The Complete Model implemented in LPL [22]

```
model Diet "Diet";
  set i; j;
  parameter c{j}; b{i}; A{i,j};
  variable x{j};
  constraint
    N{i}: sum{j} A*x >= b;
    P: sum{j} x = 1;
  minimize cost: sum{j} c*x;
  Writep(x);
  model data;
    i:=[1..4]; j:=[1..4];
    c{j}:=[35,50,80,95];
    b{i}:=[2.4, 0.7, 5, 21];
    A{i,j} := [2.2, 3.4, 7.2, 1.5,
               1.4, 1.1, 0.0, 0.8,
               2.3, 5.6, 11.1, 1.3,
               12.0, 11.9, 41.8, 52.1];
  end
end
```

Model from : LINGO the problem: SWINE ROSES Hog Farm (WB)

5.26. Quantrix Example (finance)

— Run LPL Code , HTML Document –

Problem: A small finance model from the Quatrix documentation

Listing 5.26: The Complete Model implemented in LPL [22]

```
model finance "Quantrix Example";
  set y:=[year0 year1 year2 year3 year4];
      f,y1{y} := y>1;
      i:=[Revenue COGS 'Gross Profit' 'SG&A' Depreciation
           'Other Expenses'
          EBIT 'D&A' EBITDA Interest EBT Tax 'Net Income
              '];
      it := [Cash 'Acc receivable' Inventory 'Net PP&E' '
          Acc Payable' 'Short-term Dept'
             'Long-term Dept' 'Paid-in Capital' 'Retained
                 Earnings'];
      as := [DSO 'Inv Days' DPO];
      ip :=['Net Income' 'D&A' 'Change In Acc Rec' '
          Change in Inventories' 'Change in Acc Pay'
            'Operating Cash Flow' 'Investments in PP&E' '
               Debt Financing' 'Equity Financing'
            'Free Cash Flow' 'Starting Cash Balance' '
               Ending Cash Balance'];
  parameter
    Income{i,y}:=/ [*,year0]
      Revenue 1000 COGS 500 'SG&A' 250 Depreciation 50 '
          Other Expenses' 100 Interest 10 Tax 15 /;
    Cases{i,f}:=/[Revenue,*] year1 0.05 year2 0.05 year3
        0.05 year4 0.05/;
    Balance{i,y}:=/[*,year0]
      Cash 1000 'Acc receivable' 100 Inventory 100 'Net
          PP&E' 550 'Acc Payable' 150 'Short-term Dept'
          50
      'Long-term Dept' 300 'Paid-in Capital' 510 '
          Retained Earnings' 740/;
    Assumptions{as} := [36 72 100];
    Cash{ip,y};
    {i,f|i>1} (Cases[i,f] := Income[i,1]/Income[1,1]),
              Cases['COGS','year2']:=0.55,
        {i,f} (Income[i,f] := if(i=1,(1+Cases)*Income[i,f
            -1],Cases*Income[1,f])),
          {y} (Income['Gross Profit',y]:=Income['Revenue
              ',y]-Income['COGS',y],
               Income['EBIT',y]:=Income['Gross Profit',y
                   ]-Income[4,y]-Income[5,y]-Income[6,y],
```

```
        Income['EBITDA',y]:=Income['EBIT',y]-
            Income['D&A',y],
        Income['EBT',y]:=Income['EBIT',y]-Income['
            Interest',y],
        Income['Net Income',y]:=Income['EBT',y]-
            Income['Tax',y]),
    {f} (Balance['Acc receivable',f]:=Income[1,f]*
    Assumptions['DSO']/365,
        Balance['Inventory',f]:=Income[2,f]*
            Assumptions['Inv Days']/365,
        Balance['Net PP&E',f]:=Balance['Net PP&E',
            f-1]+Cash['Investments in PP&E',f]-
            Income['Depreciation',f],
        Balance['Acc Payable',f]:=Income[2,f]*
            Assumptions['DPO']/365,
        Balance['Short-term Dept',f]:=Balance['
            Short-term Dept',f-1],
        Balance['Long-term Dept',f]:=Balance['Long
            -term Dept',f-1]+Cash['Debt Financing
            ',f],
        Balance['Paid-in Capital',f]:=Balance['
            Paid-in Capital',f-1]+Cash['Equity
            Financing',f],
        Balance['Retained Earnings',f]:=Balance['
            Retained Earnings',f-1]+Income['Net
            Income',f]),
        Cash['Ending Cash Balance',1]:=Balance['
            Cash',1],
    {y} (Cash['Net Income',y]:=Income['Net Income',
    y],
        Cash['D&A',y]:=Income['Depreciation',y]),
    {f} (Cash['Change In Acc Rec',f]:=Balance['Acc
    receivable',f-1]-Balance['Acc receivable',f
    ],
        Cash['Change in Inventories',f]:=Balance['
            Inventory',f-1]-Balance['Inventory',f
            ],
        Cash['Change in Acc Pay',f]:=Balance['Acc
            Payable',f]-Balance['Acc Payable',f
            -1]),
    {y} (Cash['Operating Cash Flow',y]:=sum{ip|ip
    <=5}Cash,
        Cash['Free Cash Flow',y]:=sum{ip|ip<=7 and
             ip<>6}Cash),
    {f} (Cash['Ending Cash Balance',f]:=Cash['
    Ending Cash Balance',1]+sum{y1[y2]|y2<=f}
    Cash['Free Cash Flow',y2],
        Cash['Starting Cash Balance',f]:=Cash['
            Ending Cash Balance',f-1],
        Balance['Cash',f]:=Cash['Ending Cash
```

```
                    Balance',f]);
    Writep(Income,Balance,Cash);
end
```

5.27. Lindo's QP Example (qp-lindo1)

—- Run LPL Code , HTML Document –

Problem: A small QP model from the Lindo documentation

Listing 5.27: The Complete Model implemented in LPL [22]

```
model qpLindo1 "Lindo's QP Example";
  --SetSolver(lindoLSol);
  variable x0; x1;
  constraint A: x0+x1 >= 10;
  minimize obj: .5*x0^2 + .75*x0*x1 + x1^2 + x0 + x1;
  Writep(obj,x0,x1);
end
```

5.28. Lindo's QCP Example (qp-lindo2)

— Run LPL Code , HTML Document –

Problem: A small QCP model from the Lindo documentation

Listing 5.28: The Complete Model implemented in LPL [22]

```
model qcpLindo2 "Lindo's QCP Example";
 --SetSolver(lindoLSol);
  variable x0; x1; x2;
  constraint A: (x0-1)^2 + (x1-1)^2 <= 1;
             B: (x1-3)^2 + (x2-1)^2 <= 2;
  maximize obj: 3*x0 + 10*x1 - 2*x0^2 - 3*x1^2 - 4*x2^2 +
       2*x0*x2 + 5*x1*x2;
  Writep(obj,x0,x1,x2);
end
```

5.29. Lindo's QP Example (qp-lindo3)

—- Run LPL Code , HTML Document –

Problem: A small QP model from the Lindo documentation

Listing 5.29: The Complete Model implemented in LPL [22]

```
model qpLindo3 "Lindo's QP Example";
  --SetSolver(lindoLSol);
  --SetSolver(gurobiLSol);
  --SetSolver(cplexLSol);
  set i,j:= [1..7];
  parameter r{i} := [.14 .77 .28 .17 .56 .18 .70];
            u{i} := [.04 .56 .37 .32 .52 .38 .25];
            R := .3;
            K := 3;
            Q{i,j} :=
  [ 1.00 0.11 0.04 0.02 0.08 0.03 0.10
    0.11 1.00 0.21 0.13 0.43 0.14 0.54
    0.04 0.21 1.00 0.05 0.16 0.05 0.20
    0.02 0.13 0.05 1.00 0.10 0.03 0.12
    0.08 0.43 0.16 0.10 1.00 0.10 0.40
    0.03 0.14 0.05 0.03 0.10 1.00 0.12
    0.10 0.54 0.20 0.12 0.40 0.12 1.00 ];
  variable w{i}; binary x{i};
  constraint A: sum{i} w = 1;
             B: sum{i} r*w >= R;
             C{i}: w <= u*x;
             D: sum{i} x <= K;
  maximize obj: sum{i,j} -Q[i,j]*w[i]*w[j]/2;
  Write('obj=%8.6f\n', obj);
  Write{i}(' Invest %5.2f in asset %1s\n', w*100,i);
end
```

5.30. Lindo's qcp example (qp-lindo4)

—- Run LPL Code , HTML Document –

Problem: A small QP model from the Lindo documentation

Listing 5.30: The Complete Model implemented in LPL [22]

```
model qpLindo4 "Lindo's qcp example";
  set i,j:=[1..4];
  parameter r{i} := [.3 .2 -.4 .2];
            K := .4;
            Q{i,j} := [ 1    .64  .27 0
                        .64  1    .13 0
                        .27  .13  1   0
                        0    0    0   1];
  variable w{i};
  constraint A: sum{i,j} Q[i,j]*w[i]*w[j] <= K;
              B: sum{i} w = 1;
  maximize obj: sum{i} r*w;
  Write('obj=%8.6f\n', obj);
  Write{i}(' Invest %5.2f in asset %1s\n', w*100,i);
end
```

5.31. Gurobi's QP Example (qp-gur)

—- Run LPL Code , HTML Document –

Problem: A small QP (quadratic) model from the Gurobi documentation

Listing 5.31: The Complete Model implemented in LPL [22]

```
model qpGur "Gurobi's QP Example";
  variable x; y; z;
  minimize obj: x^2 + x*y + y^2 + y*z + z^2;
  constraint A: x + 2*y + 3*z >= 4;
              B: x + y >= 1;
  Writep(obj,x,y,z);
end
```

5.32. Gurobi's qcp example (qcp-gur)

— Run LPL Code , HTML Document –

Problem: A small QCP model from the Gurobi documentation

Listing 5.32: The Complete Model implemented in LPL [22]

```
model qcpGur "Gurobi's qcp example";
  variable x; y; z;
  maximize obj: x;
  constraint A: x + y + z = 1;
              B: x^2 + y^2 <= z^2;
              C: x^2 <= y*z;
  Writep(obj,x,y,z);
end
```

CHAPTER 6

OTHERBOOKS

6.1. Forestry Production Planning (forestry)

—- Run LPL Code , HTML Document –

Problem: Wood-D Industries has just signed contracts for the clear felling of three large forest tracts of second-growth radiata pines. The harvested trees will supply the firm's sawmill and chipboard plant. Some of the logging output is also available for export. All trees harvested are cut on location into sections 20 feet long, referred to as first cuts, second cuts, etc. On the basis of a detailed survey of each forest, the firm's chief forester estimated the average composition of each forest's total output as shown below:

Log Cuts	Forest A	Forest B	Forest C
First **and** Second	42%	46%	45%
Third **and** Fourth	40%	41%	45%
Fifth **and** over	18%	13%	10%

The average daily output is 128 HC for forest A, 192 HC for forest B, and 200 HC for forest C (1 HC = 100 cubic feet = 2.832 cubic meter). The log cuts are sorted onto logging trucks for transportation to either the sawmill or the chipboard plant, the two facilities being at different locations. Transportation costs per HC are given in the following table:

Transportation	Sawmill	Chipboard
Forest A	$4	$7
Forest B	$3	$5
Forest C	$6	$3

Handling costs per HC is given in the table below:

Handling	Sawmill	Chipboard

First **and** Second	$2.5	$1.2
Third **and** Fourth	$3.5	$1.5
Fifth **and** over	$5.0	$2.0

At the sawmill, logs are sawn into three grades of finished products: clear grade, dressing grade, and construction grade. A substantial fraction of the incoming volume of wood ends up as scrap and sawdust. The table below shows the average log conversion factors at the sawmill, as well as the average processing rates. Excluding breakdowns, the productive capacity at the mill averages 360 minutes per day.

LogCuts	Sawn Timber MBF/HC	Scraps HC/HC	Sawdust HC/HC	Processing Time per HC
1-2	0.72	0.26	0.14	1.8
3-4	0.66	0.30	0.15	2.6
5-	0.54	0.39	0.16	3.9

From sample logs processed at the sawmill, average yields for each grade of sawn timber were determined. They are summarized in the following table.

	Clear			Dressing			Construction		
LogCuts /Forest	A	B	C	A	B	C	A	B	C
1-2	35%	28%	20%	48%	42%	30%	17%	30%	50%
3-4	10%	3%	25%	18%	9%	30%	72%	88%	45%
5-	0	0	5%	5%	0	30%	95%	100%	65%

The ex-mill wholesale price per MBF is $150 for clear grades, $110 for dressing grades and $80 for construction grades. Scraps at the sawmill are transferred by truck to the chipboard plant for chipping. The transportation cost is $4 per HC. Sawdust is used as fuel in the mill's drying kiln and saves $12 in other fuel costs per HC. At the chipboard plants, logs and scraps are chipped. The chips are then mixed with additive glues, filled into 4' by 8' forms and then compressed into boards of various thicknesses. The whole process is highly automated. Each HC of wood yields 0.76M3/4 of chipboard (1M3/4 = 1000 square feet of 3.4 thickness = 1.77 cubic meters). The plant can produce upto 112M3/4 of chipboard per day. Chipboard prices ex-factory are $105 per M3/4. In light of predicted demand and desired stock levels, certain minimum daily output rates of finished product are set up by Wood-D management for given planning period. These are 31 MBF of clear grades, 35 MBF of dressing grade, 48 MBF of construction grade, and 96 M3/4 of chipboard. Export prices valid during the same planning period are $95 per HC

for first and second cuts, and $88 per HC for third and fourth cuts. Fifth or higher cuts are not exported. The problem is to determine the optimal daily operating policy during the planning period in question.

Project Formulate the production planning problem as a linear program using MPL. Summarize the optimal daily operating policy for the planning problem.

Reference: Prof. Ariela Sofer, Operations Research Dept., George Mason University Based on a problem from the book 'Introduction to Operations Research Techniques' by Hans G. Daellenbach, John A. George, and Donald C. McNickle, 1983

6.2. Wyndor Glass Company (hill03-1-1)

— Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12], Example 3.1-1, Page 25. A mathematical model formulation is:

Listing 6.1: The Complete Model implemented in LPL [22]

```
model hill03_1_1 "Wyndor Glass Company";
  set
    p := [Door, Window] "Products";
    f := [1..3] "Plants";
  parameter
    TimeAvail{f}  := [4, 12, 18];
    ProdTime{f,p} := [1, 0,  0, 2,  3, 2];
    Profit{p}     := [3.00, 5.00];
  variable Produce{p};
  constraint
    TimeCapacity{f}: sum{p} ProdTime*Produce <= TimeAvail
        ;
  maximize TotalProfit: sum{p} Profit*Produce;
  Writep(TotalProfit,Produce);
end
```

6.3. Mary's Radiation Therapy (hill03-4-1)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 3.4-1, Page 44 A mathematical model formulation is:

Listing 6.2: The Complete Model implemented in LPL [22]

```
model hill03_4_1 "Mary's Radiation Therapy";
  set
    b   := [Beam1, Beam2] "Beam";
    a   := [CriticalTissues, TumorRegion, CenterTumor] "
        BodyArea";
  parameter
    HealthyAnatomy{b}    := [0.4, 0.5];
    DosageAbsorbed{b,a} := [0.3, 0.5, 0.6,   0.1, 0.5,
        0.4];
    Restriction{a}       := [2.7, 6.0, 6.0];
    K{a}                 := [1  2  3];
  variable D{b} "Dosage";
  constraint
    Dosage1{a|K=1}: sum{b} DosageAbsorbed*D <=
        Restriction;
    Dosage2{a|K=2}: sum{b} DosageAbsorbed*D  =
        Restriction;
    Dosage3{a|K=3}: sum{b} DosageAbsorbed*D >=
        Restriction;
  minimize TotalHealthyDosage: sum{b} HealthyAnatomy*D;
  Writep(TotalHealthyDosage,D);
end
```

6.4. Kibbutzim Crop Allocation (hill03-4-2)

— Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 3.4-2, Page 46 A mathematical model formulation is:

Listing 6.3: The Complete Model implemented in LPL [22]

```
model hill03_4_2 "Kibbutzim Crop Allocation";
  set
    k :=  [1..3] "Kibbutz";
    c :=  [SugarBeets, Cotton, Sorghum] "Crops";
  parameter
    UsableLand{k} :=  [400, 600, 300];          -- {Acres}
    WaterAlloc{k} :=  [600, 800, 375];          -- {
        AcreFeet}
    WaterUsed{c}  :=  [3, 2, 1];                -- {
        AcreFeet/Acre}
    NetReturn{c}  :=  [1000, 750, 250];         -- {$/Acre}
    MaxQuota{c}   :=  [600, 500, 325];          -- {Acres}
  variable AllocAcre,A{c,k};
  constraint
    MaxLandUse{k}: sum{c} A <= UsableLand;
    MaxWaterAlloc{k}: sum{c} WaterUsed*A <= WaterAlloc;
    TotalAcreage{c}: sum{k} A <= MaxQuota;
    EqProp{k}: sum{c} A/UsableLand = sum{c} A[c,k%#k+1]/
        UsableLand[k%#k+1];
  maximize TotalProfit: sum{c,k} NetReturn*A;
  Writep(TotalProfit,A);
end
```

6.5. Nori/Leets Air Pollution (hill03-4-3)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 3.4-3, Page 50 A mathematical model formulation is:

Listing 6.4: The Complete Model implemented in LPL [22]

```
model hill03_4_3 "Nori/Leets Air Pollution";
  set
    pollutant,p  := [Particulates, SulfurOxides,
        Hydrocarbons];
    furnace,f    := [Blast, OpenHearth];
    abatementa,a := [TallerSmokestacks, Filters,
        BetterFuels];
  parameter
    RequiredReduction{p} := [60, 150, 125];
    ReducedEmissionRate,rer{p,a,f} :=
       [12,   9,   25,  20,   17,  13,
        35,  42,   18,  31,   56,  49,
        37,  53,   28,  24,   29,  20];
    AnnualCost{a,f} := [8,  10,   7,   6,  11,   9];
  variable U{a,f} [0..1] "Used Method";
  constraint
    EmissionReduction{p}: sum{f,a} rer*U >=
        RequiredReduction;
  minimize  TotalCost: sum{a,f} AnnualCost*U;
  Writep(TotalCost,U);
end
```

6.6. Save-It Company (hill03-4-4)

— Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 3.4-4, Page 53 A mathematical model formulation is:

Listing 6.5: The Complete Model implemented in LPL [22]

```
model hill03_4_4 "Save-It Company";
  set
    g     := [A, B, C] "Grade";
    m := [1..4] "Material";
    GradeMat{g,m} := [A 1, A 2, A 3, A 4, B 1, B 2, B 4,
        C 1];
  parameter
    MinMixSpecs{g,m} :=
      /A 1  0,  A 2  0.40,  A 3  0  ,  A 4  0.2,
       B 1  0,  B 2  0.1,   B 4  0.1,  C 1  0/;
    MaxMixSpecs{g,m}  :=
     /A 1  0.3, A 2  0, A 3  0.5,  A 4  0.2,
      B 1  0.5, B 2  0, B 4  0.1,  C 1  0.7/;
    AmalgamationCost{g} := [3.00, 2.50, 2.00];
    SellingPrice{g}     := [8.50, 7.00, 5.50];
    MaterialAvail{m}    := [3000, 2000, 4000, 1000];
    TreatmentCost{m}    := [3.00, 6.00, 4.00, 5.00];
    MinimumTreated      := 0.5;
    TreatmentCashAvail  := 30000;
  variable M{g,m} "MixMaterial";
  constraint
    MinMixtureSpecs{g,m|MinMixSpecs>0}: M >= MinMixSpecs
        *(sum{m} M);
    MaxMixtureSpecs{g,m|MaxMixSpecs>0}: M <= MaxMixSpecs
        *(sum{m} M);
    MaterialLimit{m}: sum{g} M <=  MaterialAvail;
    ReqAmountTreated{m}: sum{g} M >= MinimumTreated*
        MaterialAvail;
    TreatCostRestrict: sum{g,m} TreatmentCost*M =
        TreatmentCashAvail;
  maximize TotalProfit: sum{g,m} SellingPrice*M - sum{g,m
      } AmalgamationCost*M;
  Writep(TotalProfit,M);
end
```

6.7. Save-It Company (hill03-4-4b)

—- Run LPL Code , HTML Document –

Problem: lternative formulation for the same model, adds summary variables.

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 3.4-4b, Page 53 A mathematical model formulation is:

Listing 6.6: The Complete Model implemented in LPL [22]

```
model hill03_4_4b "Save-It Company";
  set  grade,g    := [A, B, C];
    material,m := [1..4];
    GradeMat{g,m}:=[A 1, A 2, A 3, A 4, B 1, B 2,B 4,C
        1];
  parameter
    MinMixSpecs{g,m}  :=
      /A 1  0,  A 2  0.40,  A 3  0,     A 4  0.20,
       B 1  0,  B 2  0.10,  B 4  0.10,  C 1  0/;
    MaxMixSpecs{g,m}  :=
      /A 1 0.3,  A 2  0,  A 3 0.50,  A 4 0.20,
       B 1 0.50, B 2  0,  B 4 0.10,  C 1 0.70/;
    AmalgamationCost{g} := [3.00, 2.50, 2.00];
    SellingPrice{g}     := [8.50, 7.00, 5.50];
    MaterialAvail{m}    := [3000, 2000, 4000, 1000];
    TreatmentCost{m}    := [3.00, 6.00, 4.00, 5.00];
    MinimumTreated      := 0.5;
    TreatmentCashAvail  := 30000;
  variable
    G{g} "ProduceGrade";
    U{m} "MaterialUsed";
    M{g,m} "MixMaterial";
  constraint
    GradeProd{g}: G = sum{m} M ;
    MatUse{m} :  U = sum{g} M ;
    MinMixtureSpecs{g,m|MinMixSpecs>0}:M>=MinMixSpecs*G;
    MaxMixtureSpecs{g,m|MaxMixSpecs>0}:M<=MaxMixSpecs*G;
    MaterialLimit{m}: U <= MaterialAvail;
    ReqAmountTreated{m}: U >= MinimumTreated*
        MaterialAvail;
    TreatCostRestrict: sum{m} TreatmentCost*U =
        TreatmentCashAvail;
  maximize TotalProfit: sum{g} SellingPrice*G - sum{g}
     AmalgamationCost*G;
  Writep(TotalProfit,G,U,M);
```

```
end
```

6.8. Union Airways Personnel (hill03-4-5)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 3.4-5, Page 57 A mathematical model formulation is:

Listing 6.7: The Complete Model implemented in LPL [22]

```
model hill03_4_5 "Union Airways Personnel";
  set
    s := [1..5] "Shifts";
    p := [AM_6_8,  AM_8_10,  AM_10_12,
          PM_12_2, PM_2_4,   PM_4_6,   PM_6_8,
          PM_8_10, PM_10_12, AM_12_6]  "Periods";
    ShiftSchedule{s,p} :=
        [1 AM_6_8,   1 AM_8_10,  1 AM_10_12, 1 PM_12_2,
         2 AM_8_10,  2 AM_10_12, 2 PM_12_2,  2 PM_2_4,
         3 PM_12_2,  3 PM_2_4,   3 PM_4_6,   3 PM_6_8,
         4 PM_4_6,   4 PM_6_8,   4 PM_8_10,  4 PM_10_12,
         5 PM_10_12, 5 AM_12_6];
  parameter
    MinAgentsNeeded{p}   := [48, 79, 65, 87, 64, 73, 82,
        43, 52, 15];
    DailyCostPerAgent{s} := [170, 160, 175, 180, 195];
  variable A{s} "Assign agents to s";
  constraint
    MeetRequirements{p}: sum{s|ShiftSchedule} A >=
        MinAgentsNeeded;
  minimize TotalCost: sum{s} DailyCostPerAgent*A;
  Writep(TotalCost,A);
end
```

6.9. Distribution Unlimited (hill03-4-6)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 3.4-6, Page 59 A mathematical model formulation is:

Listing 6.8: The Complete Model implemented in LPL [22]

```
model hill03_4_6 "Distribution Unlimited";
  set i,j := [F1, F2, DC, W1, W2] "Locations";
  parameter
    ShipCost{i,j} :=
     /F1 F2 200, F1 DC 400, F1 W1 900, F2 DC 300,
      DC W2 100, W1 W2 300, W2 W1 200/;
    Supply{i} := /F1 50, F2 40/;
    Demand{i} := /W1 30, W2 60/;
    MaxFlow{i,j} := /F1 F2 10, DC W2 80/;
  variable Ship{i,j|ShipCost};
  constraint
    NetFlow{i}: sum{j} Ship[j,i] + Supply
                  = sum{j} Ship[i,j] + Demand;
    MaxShipment{i,j|MaxFlow}: Ship <= MaxFlow;
  minimize TotalCost: sum{i,j} ShipCost*Ship;
  Writep(TotalCost,Ship);
end
```

6.10. Wyndor Glass (hill07-1-1)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 7.1-1, Page 311 A mathematical model formulation is:

Listing 6.9: The Complete Model implemented in LPL [22]

```
model hill07_1_1 "Wyndor Glass";
  set
    p :=  [Door, Window] "Products";
    f :=  [1..3] "Plants";
  parameter
    TimeAvail{f}   := [4, 12, 18];
    ProdTime{f,p}  := [1, 0,  0, 2, 3, 2];
    Profit{p}      := [3.00, 5.00];
  variable T{f} "Assign agents to f";
  constraint ProduceDual{p}: sum{f} ProdTime*T >= Profit;
  minimize TProfit: sum{f} TimeAvail*T;
  Writep(TProfit,T);
end
```

6.11. Upper Bound (hill07-3-1)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] „ Example 7.3-1, Page 319 A mathematical model formulation is:

Listing 6.10: The Complete Model implemented in LPL [22]

```
model hill07_3_1 "Upper Bound";
  set
    activity1,a    :=  [Activity1,Activity2,Activity3];
    constraint1,c  :=  [constraint1,constraint2];
  parameter
    ResourceAvail{c} := [12,4];
    ResourceUse{c,a} := [4,1,0, -2,0,1];
    UnitProfit{a}    := [2,1,2];
    Upper{a}         := [4,15,6];
  variable Produce{a} [0..Upper];
  constraint
    TimeCapacity{c}: sum{a} ResourceUse*Produce <=
        ResourceAvail;
  maximize TotalProfit: sum{a} UnitProfit*Produce;
  Writep(TotalProfit,Produce);
end
```

6.12. Goal Programming (hill07-5-1)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 7.5-1, Page 333 A mathematical model formulation is:

Listing 6.11: The Complete Model implemented in LPL [22]

```
model hill07_5_1 "Goal Programming";
  set
    p := [1..3] "Products";
    f := [Profit, Employment, Investment] "Factors";
    d := ['Over', 'Under'] "Deviations";
  parameter
    Goal{f}             :=  [125, 40, 55];
    PenaltyWeight{d,f} := [0, 2, 3,   5, 4, 0];
    UnitContrib{f, p}  :=  [12, 9,15,  5, 3, 4,   5, 7,
        8];
  variable
    ProdRate{p};
    Amount{f,d|PenaltyWeight>0};
  constraint
    Calc{f}: sum{p} ProdRate*UnitContrib
           = Goal + Amount[f,'Over'] - Amount[f,'Under'];
  minimize WeightedSum: sum{d,f} PenaltyWeight*Amount;
  Writep(WeightedSum,Amount);
end
```

6.13. P-T Company (hill08-1-1)

— Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 8.1-1, Page 351 A mathematical model formulation is:

Listing 6.12: The Complete Model implemented in LPL [22]

```
model hill08_1_1 "P-T Company";
  set
    cannery,c := [C1, C2, C3];
    warehouse,w := [W1, W2, W3, W4];
  parameter
    ShipCost{c,w} := [464, 513, 654, 867,
                      352, 416, 690, 791,
                      995, 682, 388, 685];
    Supply{c} := [75,125,100];
    Demand{w} := [80, 65, 70, 85];
  variable Ship{c,w};
  constraint
    Output{c}: sum{w} Ship = Supply;
    Allocation{w}: sum{c} Ship = Demand;
  minimize TotalCost: sum{c,w} ShipCost*Ship;
  Writep(TotalCost,Ship);
end
```

6.14. Northern Airplane (hill08-1-2)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] „ Example 8.1-2, Page 359 A mathematical model formulation is:

Listing 6.13: The Complete Model implemented in LPL [22]

```
model hill08_1_2 "Northern Airplane";
  set
    produce,p := [1..4];
    install,i := [1..5];
  parameter
    M := 999;
    DistCost{p,i} := [1.080, 1.095, 1.110, 1.125, 0,
                      999,   1.110, 1.125, 1.140, 0,
                      999, 999,     1.100, 1.115, 0,
                      999, 999,999, 1.130, 0];
    Supply{p} := [25, 35, 30, 10];
    Demand{i} := [10, 15, 25, 20, 30];
  variable Engines{p,i};
  constraint
    Production{p}: sum{i} Engines = Supply;
    Installations{i}: sum{p} Engines = Demand;
  minimize TotalCost: sum{p,i} DistCost*Engines;
  Writep(TotalCost,Engines);
end
```

6.15. Metro Water (hill08-1-3)

— Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 8.1-3, Page 362 A mathematical model formulation is:

Listing 6.14: The Complete Model implemented in LPL [22]

```
model hill08_1_3 "Metro Water";
  set
    r := [Colombo, Sacron, Calorie, Dummy] "Rivers";
    c := [Berdoo_min, Berdoo_extra, LosDevils, SanGo,
       Hollyglass] "Cities";
  parameter
    M := 999;
    DistCost{r, c} := [16, 16, 13, 22, 17,
                       14, 14, 13, 19, 15,
                       19, 19, 20, 23, 999,
                      999,  0,999,  0,  0];
    Supply{r} := [50, 60, 50, 50];
    Demand{c} := [30, 20, 70, 30, 60];
  variable Distribute{r,c};
  constraint
   Source{r}: sum{c} Distribute = Supply;
   Destination{c}: sum{r} Distribute = Demand;
  minimize TotalCost: sum{r,c} 10000000*DistCost*
      Distribute;
  Writep(TotalCost,Distribute);
end
```

6.16. Job Shop Company (hill08-3-1)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 8.3-1, Page 381 A mathematical model formulation is:

Listing 6.15: The Complete Model implemented in LPL [22]

```
model hill08_3_1 "Job Shop Company";
  set
    m := [m1, m2, m3, Dummy] "Machines";
    p := [1..4] "Locations";
  parameter
    M := 999;
    DistCost{m, p} := [13, 16, 12, 11,
                       15,999, 13, 20,
                        5,  7, 10,  6,
                        0,  0,  0,  0];
    Supply{m} := [1, 1, 1, 1];
    Demand{p} := [1, 1, 1, 1];
  variable Assign{m,p};
  constraint
    Source{m}: sum{p} Assign = Supply;
    Destination{p}: sum{m} Assign = Demand;
  minimize TotalCost: sum{m,p} DistCost*Assign;
  Writep(TotalCost,Assign);
end
```

6.17. Better Products Company (hill08-3-2a)

— Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 8.3-2a, Page 386 A mathematical model formulation is:

Listing 6.16: The Complete Model implemented in LPL [22]

```
model hill08_3_2a "Better Products Company";
  set
    f := [1..3] "Plants";
    p := [1..5] "Products";
  parameter
    M := 999;
    DistCost{f, p} := [41, 27, 28, 24, 0,
                       40, 29,999, 23, 0,
                       37, 30, 27, 21, 0];
    Supply{f} := [75, 75, 45];
    Demand{p} := [20, 30, 30, 40, 75];
  variable Assign{f,p};
  constraint
    Source{f}: sum{p} Assign = Supply;
    Destination{p}: sum{f} Assign = Demand;
  minimize TotalCost: sum{f, p} DistCost*Assign;
  Writep(TotalCost,Assign);
end
```

6.18. Better Products Company (hill08-3-2b)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 8.3-2b, Page 386 A mathematical model formulation is:

Listing 6.17: The Complete Model implemented in LPL [22]

```
model hill08_3_2b "Better Products Company";
  set
    f := [p1a, p1b, p2a, p2b, p3] "Plants";
    p := [1..5]  "Products";
  parameter
    M := 99999;
    Cost{f, p} := [820, 810, 840, 960, 0,
                   820, 810, 840, 960, 0,
                   800, 870,99999, 920, 0,
                   800, 870,99999, 920, 0,
                   740, 900, 810, 840,99999];
  binary variable Assign{f, p};
  constraint
    Assignee{f}: sum{p} Assign = 1;
    Task{p}: sum{f} Assign = 1;
  minimize TotalCost: sum{f, p} Cost*Assign;
  Writep(TotalCost,Assign);
end
```

6.19. Shortest Path (hill09-3-1)

— Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 9.3-1, Page 411 A mathematical model formulation is:

Listing 6.18: The Complete Model implemented in LPL [22]

```
model hill09_3_1 "Shortest Path";
  set i,j := [O, A, B, C, D, E, T];
  parameter
    Distance{i,j} :=
       /O A 2,  O B 5,  O C 4,  A B 2,  A D 7,  B C 1,
        B D 4,  B E 3,  C E 4,  D E 1,  D T 5,  E T 7/;
  variable Path{i,j|Distance>0};
  constraint
    FlowBalance{i}: if(i='O',1) + sum{j} Path[j,i]
                  = if(i='T',1) + sum{j} Path[i,j];
  minimize TotalDistance: sum{i,j} Distance*Path;
  Writep(TotalDistance,Path);
end
```

6.20. Maximum Flow (hill09-5-1)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 9.5-1, Page 420 A mathematical model formulation is:

Listing 6.19: The Complete Model implemented in LPL [22]

```
model hill09_5_1 "Maximum Flow";
  set
    node,i,j := [O, A, B, C, D, E, T];
    SourceNode{i}  := [O];
    DestNode{i}    := [T];
  parameter
    Capacity{i,j} :=
       /O A 5,  O B 7,  O C 4,  A B 1,  A D 3,  B C 2,
        B D 4,  B E 5,  C E 4,  D T 9,  E D 1,  E T 6/;
  variable
    Flow{i,j|Capacity>0} [0..Capacity];
    Entrance{node|node='O'};
    Destination{node|node='E'};
  constraint
    FlowBal{node}: Entrance + sum{i} Flow[i,node]
                 = Destination + sum{j} Flow[node,j];
  maximize TotalFlow: Entrance['O'];
  Writep(TotalFlow,Flow);
end
```

6.21. Minimum Cost (hill09-6-1)

— Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 9.6-1, Page 429 A mathematical model formulation is:

Listing 6.20: The Complete Model implemented in LPL [22]

```
model hill09_6_1 "Minimum Cost";
  set
    node,i,j := [A, B, C, D, E];
  parameter
    UnitCost{i,j} :=
      /A B 2,  A C 4,  A D 9,  B C 3,  C E 1,  D E 3, E D
          2/;
    Capacity{i,j} := /A  B  10, C  E  80/;
    SupplyDemand{node} := [50, 40, 0, -30, -60];
  variable Ship{i,j|UnitCost>0};
  constraint
    FlowBalance{node}: sum{i} Ship[i,node] + SupplyDemand
                       = sum{j} Ship[node,j];
    MaxCapacity{i,j|Capacity>0}: Ship <= Capacity;
  minimize TotalCost: sum{i,j} UnitCost*Ship;
  Writep(TotalCost,Ship);
end
```

6.22. Critical Path (hill10-1-1)

—- Run LPL Code , HTML Document –

Problem: RT/PERT Method

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 10.1-1, Page 469 A mathematical model formulation is:

Listing 6.21: The Complete Model implemented in LPL [22]

```
model hill10_1_1 "Critical Path";
  set
    n,m := [A,B,C,D,E,F,G,H,I,J,K,L,M,N,Finish];
    Path{n,m} :=
        [ A B, B C, C D, C E, C I, D G, E F, E H, F J,
         G H, H M, I J, J K, J L, K N, L N, M Finish, N
             Finish];
  parameter Duration{n} :=
      [2,4,10,6,4,5,7,9,7,8,4,5,2,6,0];
  variable S{n} "StartTime";
  constraint SeqRelation{Path[n,m]}: S[n] + Duration[n]
      <= S[m];
  minimize FinishTime: S['Finish'];
  Writep(FinishTime,S);
end
```

6.23. Crashing (hill10-5-1)

— Run LPL Code , HTML Document –

Problem: ritical Path

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 10.5-1, Hillier and Lieberman, Page 494 A mathematical model formulation is:

Listing 6.22: The Complete Model implemented in LPL [22]

```
model hill10_5_1 "Crashing";
  set
    n,m := [A,B,C,D,E,F,G,H,I,J,K,L,M,N];
    Path{n,m} :=
       [A B, B C, C D, C E, C I, D G, E F, E H, F J,
        G H, H M, I J, J K, J L, K N, L N];
  parameter
    NormalTime{n} := [2,4,10,6,4,5,7,9,7,8,4,5,2,6];
    CrashTime{n}  := [1,2,7,4,3,3,4,6,5,6,3,3,1,3];
    NormalCost{n} := [180, 320, 620, 260, 410, 180, 900,
                      200, 210, 430, 160, 250, 100, 330];
    CrashCost{n} := [280, 420, 860, 340, 570, 260, 1020,
                     380, 270, 490, 200, 350, 200, 510];
    MaxReduce{n} := NormalTime - CrashTime;
    CostWeekSaved{n} := CrashCost/MaxReduce - NormalCost/
        MaxReduce;
  variable
    S{n}               "StartTime";
    F    [0..40]        "Finished";
    R{n} [0..MaxReduce] "ReducedTime";
  constraint
    SeqRelation{Path[n,m]}: S[n] + NormalTime[n] - R[n]
        <=  S[m];
    M: S['M'] + NormalTime['M'] - R['M'] <= F;
    N: S['N'] + NormalTime['N'] - R['N'] <= F;
  minimize TotalCost: sum{n} NormalCost + sum{n}
     CostWeekSaved*R;
  Writep(TotalCost,S,F,R);
end
```

6.24. Calif. Manufacturing (hill12-1-1)

—- Run LPL Code , HTML Document –

Problem: acility Location

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 12.1-1, Page 577 A mathematical model formulation is:

Listing 6.23: The Complete Model implemented in LPL [22]

```
model hill12_1_1 "Calif. Manufacturing";
  set
    location,lo:= [WLA, FLA, WSF, FSF];
  parameter
    NPV{lo}      := [6, 9, 4, 5];  -- In millions
    Capital{lo} := [5, 6, 2, 3];  -- In millions
  binary variable x{lo};
  constraint
    Budget: sum{lo} Capital*x <= 10;
    Warehouse: sum{lo|lo<#lo} x <= 1;
    Cnt{lo|lo>1}: -x + x[lo+2] <= 0;
  maximize TotalNPV: sum{lo} NPV*x;
  Writep(TotalNPV,x);
end
```

6.25. Production Rates (hill12-4-1)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 12.4-1, , Page 592 A mathematical model formulation is:

Listing 6.24: The Complete Model implemented in LPL [22]

```
model hill12_4_1 "Production Rates";
    set
    plant  ,f := [1..2];
    product,p := [1..3];
  parameter
    Hours{f,p} := [3, 4, 2,  4, 6, 2];
    Profit{p} := [5, 7, 3];      -- Per unit
    Sales{p}  := [7, 5, 9];
    M := 999;
  variable rate{p} [0..Sales];
  binary variable
    y{p};    -- Equals 1 if product i is produced
    P;       -- Equals 1 if we produce in Plant 2
  constraint
    Produce{p}: rate <= M*y;
    NumPrd: sum{p} y  <=2;
    Plnt{f|f=1}: sum{p} Hours*rate <= if(f=1,30+M*P, 40+M
        *(1-P));
  maximize TotProfit: sum{p} Profit*rate ;
  Writep(TotProfit,y,P);
end
```

6.26. TV Spots (hill12-4-2a)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 12.4-2a, Page 594 A mathematical model formulation is:

Listing 6.25: The Complete Model implemented in LPL [22]

```
model hill12_4_2a "TV Spots";
  set
    product,p := [1..3];
    spots  ,s := [0..3];
  parameter
    Profit{s,p}:= [0 0 0, 1 0 -1, 3 2 2, 3 3 4];
    J{s} := [1, 2, 3, 0];
  binary variable
    y{s,p|s>0};   // = 1 if product i fills j spots
  constraint
    NumProd{p}: sum{s} y <= 1;
    NumSpots: sum{p,s} J*y = 5;
  maximize TotProfit: sum{p,s} Profit*y;
  Writep(TotProfit,y);
end
```

6.27. TV Spots (hill12-4-2b)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 12.4-2b, Hillier and Lieberman, Page 595 A mathematical model formulation is:

Listing 6.26: The Complete Model implemented in LPL [22]

```
model hill12_4_2b "TV Spots";
  set
    product,p := [1..3];
    spots  ,s := [0..3];
  parameter
    Slope{p,s} := [0 1 2 0, 0 0 2 1, 0 -1 3 2];
  binary variable y{p,s};
  constraint
    Cnst {p,s|s>1}: y - y[p,s-1] <=0;
    Tot: sum{p,s|s>0} y = 5;
  maximize TotProfit: sum{p,s} Slope*y;
  Writep(TotProfit,y);
end
```

6.28. Crew Assignments (hill12-4-3)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 12.4-3, Page 598 A mathematical model formulation is:

Listing 6.27: The Complete Model implemented in LPL [22]

```
model hill12_4_3 "Crew Assignments";
  set
    j       := [1..12];
    flights := [1..11];
  parameter
    Cost{j} :=  [2, 3, 4, 6, 7, 5, 7, 8, 9, 9, 8, 9];
    Feas{flights,j} :=
            [1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0,
             0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0,
             0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1,
             0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1,
             1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0,
             0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1,
             0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0,
             0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0,
             0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1,
             0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1];
  binary variable x{j};
  constraint
    Sched{flights}: sum{j} Feas*x >= 1;
    Numb: sum{j} x = 3;
  minimize TotCost: sum{j} Cost*x ;
  Writep(TotCost,x);
end
```

6.29. Nonlinear constraint (hill13-2-1)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 13.2-1, Page 659 A mathematical model formulation is:

Listing 6.28: The Complete Model implemented in LPL [22]

```
model hill13_2_1 "Nonlinear constraint";
  set
    product,p :=  [Door, Window];
  parameter
    TimeAvail  := 4;
    ProdTime{p} := [1, 0];
    Profit{p}   := [3, 5];
  variable P{p};
  constraint
    TimeCapacity: sum{p} ProdTime*P <= TimeAvail;
    NLConstr: 9*P['Door']^2 + 5*P['Window']^2 <= 216;
  maximize TotalProfit: sum{p} Profit*P;
  Writep(TotalProfit,P);
end
```

6.30. Nonlinear Objective (hill13-2-2)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 13.2-2, Page 659 A mathematical model formulation is:

Listing 6.29: The Complete Model implemented in LPL [22]

```
model hill13_2_2 "Nonlinear Objective";
  set
    product,p :=  [Door, Window];
    plant  ,f :=  [1..3];
  parameter
    TimeAvail{f}    := [4, 12, 18];
    ProdTime{f,p}   := [1, 0,  0, 2,  3, 2];
  variable P{p} "Produce";
  constraint
    TimeCapacity{f}: sum{p} ProdTime*P <= TimeAvail;
  maximize TotalProfit: 126*P['Door'] - 9*P['Door']^2
                  +182*P['Window'] - 13*P['Window']^2;
  Writep(TotalProfit,P);
end
```

6.31. Nonlinear Objective 2 (hill13-2-3)

— Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 13.2-3, Page 661 A mathematical model formulation is:

Listing 6.30: The Complete Model implemented in LPL [22]

```
model hill13_2_3 "Nonlinear Objective 2";
  set
    product :=  [Door, Window];
    plant   :=  [1..3];
  parameter
    TimeAvail{plant}          := [4, 12, 18];
    ProdTime{plant, product} := [1, 0,  0, 2,  3, 2];
  variable Produce{product};
  constraint
    TimeCapacity{plant}: sum{product} ProdTime*Produce <=
         TimeAvail;
  maximize TotalProfit: 54*Produce['Door'] - 9*Produce['
      Door']^2
                 + 78*Produce['Window'] - 13*Produce['
                     Window']^2;
  Writep(TotalProfit,Produce);
end
```

6.32. Convex Programming (hill13-9-1)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] :, Example 13.9-1, Page 698 A mathematical model formulation is:

Listing 6.31: The Complete Model implemented in LPL [22]

```
model hill13_9_1 "Convex Programming";
  set        i := [1..2];
  parameter  c{i} := [3, 2];
  variable   x{i};
  constraint Constr: sum{i} c*x <= 6;
  maximize   Z: 5*x[1] - x[1]^2 + 8*x[2] - 2*x[2]^2;
  Writep(Z,x);
end
```

6.33. Odds And Even (hill14-1-1)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 14.1-1, Page 726 A mathematical model formulation is:

Listing 6.32: The Complete Model implemented in LPL [22]

```
model hill14_1_1 "Odds And Even";
  set strategy,s1,s2:= [1..2];
  parameter
    Payoff{s1, s2} := [ 1, -1,   -1,  1];
  variable
    P{s1} "Probabilities";
    ValueOfGame;
  constraint
    TotalProbabilities: sum{s1} P = 1;
    ExpectedPayoff{s2}: sum{s1} Payoff*P >= ValueOfGame;
  maximize GameValue: ValueOfGame;
  Writep(GameValue,P,ValueOfGame);
end
```

6.34. Political Variaton 1 (hill14-2-1)

—- Run LPL Code , HTML Document –

Problem: ame Theory

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 14.2-1, Page 728 A mathematical model formulation is:

Listing 6.33: The Complete Model implemented in LPL [22]

```
model hill14_2_1 "Political Variaton 1";
  set strategy,s1,s2 := [1..3];
  parameter Payoff{s1,s2} := [1,1,4,1,0,5,0,1,-1];
  variable
    Probabilities,P{s1};
    ValueOfGame;
  constraint
    TotalProbabilities: sum{s1} P = 1;
    ExpectedPayoff{s2}: sum{s1} Payoff*P >= ValueOfGame;
  maximize GameValue: ValueOfGame;
  Writep(GameValue,P);
end
```

6.35. Political Variation 2 (hill14-2-2)

— Run LPL Code , HTML Document –

Problem: ame theory

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 14.2-2, Page 731 A mathematical model formulation is:

Listing 6.34: The Complete Model implemented in LPL [22]

```
model hill14_2_2 "Political Variation 2";
  set strategy1,s1,s2 := [1..3];
  parameter Payoff{s1,s2}:=[-3,-2,6,2,0,2,5,-2,-4];
  variable
    Probabilities,P{s1};
    ValueOfGame;
  constraint
    TotalProbabilities: sum{s1} P = 1;
    ExpectedPayoff{s2}: sum{s1} Payoff*P >= ValueOfGame;
  maximize GameValue: ValueOfGame;
  Writep(GameValue,P);
end
```

6.36. Political Variation 3 (hill14-2-3)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Example 14.2-3, Page 732 A mathematical model formulation is:

Listing 6.35: The Complete Model implemented in LPL [22]

```
model hill14_2_3 "Political Variation 3";
  set strategy,s1,s2 := [1..3];
  parameter Payoff{s1,s2}:=[0,-2,2,5,4,-3,2,3,-4];
  variable
    Probabilities,P{s1};
    ValueOfGame;
  constraint
    TotalProbabilities: sum{s1} P = 1;
    ExpectedPayoff{s2}: sum{s1} Payoff*P >= ValueOfGame;
  maximize GameValue: ValueOfGame;
  Writep(GameValue,P);
end
```

6.37. School Board Scheduling (hillC3-1)

— Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Chapter 3, Case 1, Page 79 A mathematical model formulation is:

Listing 6.36: The Complete Model implemented in LPL [22]

```
model hillC3_1 "School Board Scheduling";
  set
    g := [6,7,8] "Grade";
    s := [1,2,3] "School";
    a := [1..6]   "Area";
  parameter
    NrStudents{a} := [450, 600, 550, 350, 500, 450];
    PctGrade{a,g} := [0.32 0.38 0.30,  0.37 0.28 0.35,
                      0.30 0.32 0.38,  0.28 0.40 0.32,
                      0.39 0.34 0.27,  0.34 0.28 0.38];
    StudentGrade{a,g} := PctGrade*NrStudents;
    MinStudGradePct := 0.30;
    MaxStudGradePct := 0.35;
    BusingCost{a,s} := [300   0 700,  -1 400 500,
                        600 300 200, 200 500  -1,
                          0  -1 400, 500 300   0];
    SchoolCapacity{s}  := [900 1100 1000];
  variable
    AssignStudent,A{a,s,g|BusingCost>0};
  constraint
    StudentDistr{a,g}: sum{s} A = PctGrade*NrStudents;
    MaxSchool{s}: sum{a,g} A <= SchoolCapacity;
    MinGrade{s,g}: sum{a} A >= MinStudGradePct*(sum{a,g}
        A);
    MaxGrade{s,g}: sum{a} A <= MaxStudGradePct*(sum{a,g}
        A);
  minimize  TotalCost: sum{a,s,g} BusingCost*A;
  Writep(TotalCost,A);
end
```

6.38. Manufacturing (hillP3-1-3)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Chapter 3, Case 1, Page 79 A mathematical model formulation is:

Listing 6.37: The Complete Model implemented in LPL [22]

```
model hillP3_1_3 "Manufacturing";
  set
    p  := [ 1..3]  "Products";
    m  := [Milling, Lathe, Grinder] "Machines";
  parameter
    TimeAvail{m} := [500, 350, 150];          --hours
    Productivity{m,p} := [9, 3, 5,  5, 4, 0,  3, 0, 2];
          --hours
    Profit{p} := [50, 20, 25];                --dollars
    MaxSales  := 20;
  variable Produce{p};
  constraint
    ProdCap{m}: sum{p} Productivity*Produce <= TimeAvail;
    SalesCap: Produce[3] <= MaxSales;
  maximize TotalProfit: sum{p} Profit*Produce ;
  Writep(TotalProfit,Produce);
end
```

6.39. TV Manufacturing (hillP3-1-4)

— Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Chapter 3.1, Problem 4, Page 69 A mathematical model formulation is:

Listing 6.38: The Complete Model implemented in LPL [22]

```
model hillP3_1_4 "TV Manufacturing";
  set tvset := [i27, i20];
  parameter
    MaxDemand{tvset} := [40, 10];
    WorkHoursAvail   := 500;
    WorkReq{tvset}   := [20, 10];
    Profit{tvset}    := [120, 80];
  variable Produce{tvset} [0..MaxDemand];
  constraint
    WorkLimit: sum{tvset} WorkReq*Produce <=
        WorkHoursAvail;
  maximize TotalProfit: sum{tvset} Profit*Produce ;
  Writep(TotalProfit,Produce);
end
```

6.40. Resource PQ (hillP3-1-5)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Chapter 3.1, Problem 5, Page 69 A mathematical model formulation is:

Listing 6.39: The Complete Model implemented in LPL [22]

```
model hillP3_1_5 "Resource PQ";
  set
    product   := [1..2];
    resource  := [P, Q];
  parameter
    ResourceReq{product,resource} :=  [1, 2,  3, 2];
    Profit{product} := [1, 2];
    ResourceAvail{resource} :=  [200, 300];
    MaxProduce2 := 60;
  variable Produce{product};
  constraint
    ResourceLimit{resource}: sum{product} ResourceReq*
        Produce <= ResourceAvail;
    ProduceLimit: Produce[2] <= MaxProduce2;
  maximize TotalProfit: sum{product}  Profit*Produce ;
  Writep(TotalProfit,Produce);
end
```

6.41. Insurance (hillP3-1-6)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Chapter 3.1, Problem 6, Page 70 A mathematical model formulation is:

Listing 6.40: The Complete Model implemented in LPL [22]

```
model hillP3_1_6 "Insurance";
  set
    p  := [SpecialRisk, Mortage] "product";
    d  := [Underwriting, Administration, Claims] "
        department";
  parameter
    Profit{p}         :=  [5, 2];
    HoursAvail{d}     :=  [2400, 800, 1200];
    HoursUnit{p, d}   :=  [3, 0, 2,  2, 1, 0];
  variable Produce{p};
  constraint WorkReq{d}: sum{p} HoursUnit*Produce <=
      HoursAvail;
  maximize TotalProfit: sum{p} Profit*Produce ;
  Writep(TotalProfit,Produce);
end
```

6.42. Auto Spare Parts (hillP3-1-7)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Chapter 3.1, Problem 7, Page 70 A mathematical model formulation is:

Listing 6.41: The Complete Model implemented in LPL [22]

```
model hillP3_1_7 "Auto Spare Parts";
  set
    m   :=  [1, 2]    "Machines";
    p   :=  [A, B, C] "Parts";
  parameter
    ProcessTime{m,p} := [0.02, 0.03, 0.05, 0.05, 0.02,
        0.04];
    TimeAvail{m}  :=  [40, 40];
    Profit{p}     :=  [50, 40, 30];
  variable Produce{p};
  constraint Capacity{m}: sum{p} ProcessTime*Produce <=
      TimeAvail;
  maximize TotalProfit: sum{p} Profit*Produce ;
  Writep(TotalProfit,Produce);
end
```

6.43. Resource QRS (hillP3-2-1)

— Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Chapter 3.2, Problem 1, Page 71 A mathematical model formulation is:

Listing 6.42: The Complete Model implemented in LPL [22]

```
model hillP3_2_1 "Resource QRS";
  set
    p  :=  [A, B] "Products";
    r  :=  [Q, R, S] "Resources";
  parameter
    Profit{p}   :=  [3, 2];
    ResAvail{r} :=  [2, 2, 4];
    ResUsed{p,r}:=  [2, 1, 3,  1, 2, 3];
  variable Produce{p};
  constraint Capacity{r}: sum{p} ResUsed*Produce <=
      ResAvail;
  maximize TotalProfit: sum{p} Profit*Produce ;
  Writep(TotalProfit,Produce);
end
```

6.44. Invest Venture (hillP3-2-3)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Chapter 3.2, Problem 3, Page 72 A mathematical model formulation is:

Listing 6.43: The Complete Model implemented in LPL [22]

```
model hillP3_2_3 "Invest Venture";
  set v := [1..2] "Ventures";
  parameter
    CashOnHand     := 6000;
    InvestCash{v} := [5000, 4000];
    WorkReq{v}     := [400, 500];
    Profit{v}      := [4500, 4500];
    WorkHoursAvail:= 600;
  variable InvestShare{v} [0..1];
  constraint
    CashLimit: sum{v} InvestCash*InvestShare <=
        CashOnHand;
    WorkLimit: sum{v} WorkReq*InvestShare <=
        WorkHoursAvail;
  maximize TotalProfit: sum{v} Profit*InvestShare ;
  Writep(TotalProfit,InvestShare);
end
```

6.45. Investment (hillP3-4-10)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Chapter 3.4, Problem 10, Page 76 A mathematical model formulation is:

Listing 6.44: The Complete Model implemented in LPL [22]

```
model hillP3_4_10 "Investment";
  set
   a  :=  [1, 2, 3] "assets";
   y  :=  [5, 10, 20] "years";
  parameter
    AssetCost := 100;
    AssetIncome{a,y}:= [200,  50,   0,  100,  50, 150,
        50, 100, 200];
    MinCashFlowReq{y} :=  [4000, 0, 3000];
  variable Invest{a};
  constraint Requirement{y}: sum{a} AssetIncome*Invest >=
      MinCashFlowReq;
  minimize TotalAmount: sum{a} AssetCost*Invest ;
  Writep(TotalAmount,Invest);
end
```

6.46. Investor ABCD (hillP3-4-11)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Chapter 3.4, Problem 11, Page 76 A mathematical model formulation is:

Listing 6.45: The Complete Model implemented in LPL [22]

```
model hillP3_4_11 "Investor ABCD";
  set
    i := [A, B, C, D] "investments";
    y :=[ 1..5] "years";
    InvestAvail{i,y} :=
      [A 1, A 2, A 3, A 4, B 1, B 2, B 3, C 2, D 5];
  parameter
    TotalInvest := 60000;
    InvestReturn{i} := [1.40, 1.70, 1.90, 1.30];
  variable
    MakeInvest,I{i,y|InvestAvail};
    MoneyAvailEnd,M{y};
  constraint
    MoneyBalance{y}: M = if(y=1,TotalInvest,M[y-1])
       + sum{i} InvestReturn*I[(#y-2+y)%#y+1] - sum{i} I;
    MoneyLimit{y}: sum{i} I <= if(y=1,TotalInvest,M[y-1])
        ;
  maximize  MoneyAccumulated: MoneyAvailEnd[#y];
  Writep(MoneyAccumulated,I,M);
end
```

6.47. Alloy Blending (hillP3-4-12)

— Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Chapter 3.4, Problem 12, Page 76 A mathematical model formulation is:

Listing 6.46: The Complete Model implemented in LPL [22]

```
model hillP3_4_12 "Alloy Blending";
  set
    a   := [1..5] "alloy";
    e := [Tin, Zinc, Lead] "elements";
  parameter
    NewAlloyReq{e} := [40, 35, 25];
    ElementPct{e,a}:= [60, 25, 45, 20, 50,
                       10, 15, 45, 50, 40,
                       30, 60, 10, 30, 10];
    AlloyCost{a} := [22, 20, 25, 24, 27];
  variable Blend{a};
  constraint BlendReq{e}: sum{a} ElementPct*Blend >=
      NewAlloyReq;
  minimize TotalCost: sum{a} AlloyCost*Blend ;
  Writep(TotalCost,Blend);
end
```

6.48. Computer Fac Oper Assignment (hillP3-4-13)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Chapter 3.4, Problem 13, Page 76 A mathematical model formulation is:

Listing 6.47: The Complete Model implemented in LPL [22]

```
model hillP3_4_13 "Computer Fac Oper Assignment";
  set
    w  :=  [Mon, Tues, Wed, Thurs, Fri] "Workdays";
    o  :=  [KC, DH, HB, SC, KS, NK] "Operators";
    Male{o}   := [KC, DH, HB, SC] "Males";
    Female{o} := [KS, NK] "Females";
  parameter
    WageRate{o} :=  [10.00, 10.10, 9.90, 9.80, 10.80,
        11.30];
    HoursAvail{o,w}  :=  [6, 0, 6, 0, 6,
                          0, 6, 0, 6, 0,
                          4, 8, 4, 0, 4,
                          5, 5, 5, 0, 5,
                          3, 0, 3, 8, 0,
                          0, 0, 0, 6, 2];
  variable AssignHours{o,w|HoursAvail>0} [0..HoursAvail];
  constraint
    AssignDay{w}: sum{o} AssignHours >= 14;
    MinHours{Male}: sum{w} AssignHours  >= if(Male,8,7);
  minimize TotalCost: sum{o,w} WageRate*AssignHours;
  Writep(TotalCost,AssignHours);
end
```

6.49. Warehouse Storage (hillP3-4-14)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Chapter 3.4, Problem 14, Page 77 A mathematical model formulation is:

Listing 6.48: The Complete Model implemented in LPL [22]

```
model hillP3_4_14 "Warehouse Storage";
  set
    m,m1 := [ 1..5] "Months";
    p := [ 1..5] "Periods";
  parameter
    SpaceReq{m} :=[30 20 40 10 50];      -- 1000sf
    LeaseCost{p}:=[650 1000 1350 1600 1900];  -- $00/1000
        sf
  variable SpaceLeased{m,p|p<=6-m};
  constraint MinSpace{m1}: sum{m,p|m<=m1 and p>m1-m}
      SpaceLeased >= SpaceReq;
  minimize TotalCost: sum{m,p} LeaseCost*SpaceLeased;
  Writep(TotalCost,SpaceLeased);
end
```

6.50. Paper Manufacturing (hillP3-4-15)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Chapter 3.4, Problem 15, Page 78 A mathematical model formulation is:

Listing 6.49: The Complete Model implemented in LPL [22]

```
model hillP3_4_15 "Paper Manufacturing";
  set
    mill,i       :=  [p1,p2,p3,p4];
    machine,m    :=  [m1,m2,m3];
    rawmat,r     :=  [r1,r2,r3,r4];
    papertype,p  :=  [t1,t2,t3,t4,t5];
    customer,c   :=  [c1,c2,c3,c4,c5,c6,c7,c8];
  parameter
    PaperDemand{c,p}      := Rnd(1,20);
    RawMatUsed{p,m,r}     := Rnd(1,4);
    RawMatAvail{i,r}      := Rnd(100,1000);
    MachineUse{p,m}       := Rnd(1,4);
    MachineAvail{i,m}     := Rnd(100,1000);
    ProduceCost{i,p,m}    := Rnd(1,10);
    TransportCost{i,c,p}  := Rnd(1,10);
  variable
    Produce,P{i,m,p|ProduceCost};
    Ship,S{i,c,p|TransportCost};
  constraint
    RawMat{i,r}: sum{m,p} RawMatUsed*P <= RawMatAvail;
    ProdCap{i,m}: sum{p} MachineUse*P <= MachineAvail;
    MillBalance{i,p}: sum{m} P = sum{c} S;
    MeetDemand{c,p}: sum{i} S >= PaperDemand;
  minimize TotalCost: sum{i,m,p} ProduceCost*P + sum{i,c,
      p} TransportCost*S;
  Writep(TotalCost,P,S);
end
```

6.51. Steak-Potato Diet (hillP3-4-6)

— Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Chapter 3.4, Problem 6, Page 74 A mathematical model formulation is:

Listing 6.50: The Complete Model implemented in LPL [22]

```
model hillP3_4_6 "Steak-Potato Diet";
  set
    food       ,f := [Steak, Potatoes];
    ingredient,i := [Carbohydrates, Protein, Fat];
  parameter
    IngredPerServing{i,f} := [ 5, 15,  20,  5,   15,  2];
    MinDailyReq{i} := [50, 40,  0];
    MaxDailyReq{i} := [ 0,  0, 60];
    CostPerServing{f}    := [4, 2];
  variable Servings{f};
  constraint
    MinRequirement{i|MinDailyReq}: sum{f}
        IngredPerServing*Servings >= MinDailyReq;
    MaxRequirement{i|MaxDailyReq}: sum{f}
        IngredPerServing*Servings <= MaxDailyReq;
  minimize  TotalCost: sum{f} CostPerServing*Servings ;
  Writep(TotalCost,Servings);
end
```

6.52. Pig Feed Blending (hillP3-4-7)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Chapter 3.4, Problem 7, Page 74 A mathematical model formulation is:

Listing 6.51: The Complete Model implemented in LPL [22]

```
model hillP3_4_7 "Pig Feed Blending";
  set
    f := [Corn, Tankage, Alfalfa] "feed";
    i := [Carbohydrates, Protein, Vitamins] "ingredient";
  parameter
    Ingredient{f, i}:=[90 30 10, 20 80 20, 40 60 60];
    MinDailyReq{i}  :=[200, 180, 150];
    Cost{f}         :=[0.84, 0.72, 0.60];
  variable FeedUse,F{f};
  constraint Requirement{i}: sum{f} Ingredient*F >=
      MinDailyReq;
  minimize  TotalCost: sum{f} Cost*F;
  Writep(TotalCost,F);
end
```

6.53. Plant Production Planning (hillP3-4-8)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Chapter 3.4, Problem 8, Page 75 A mathematical model formulation is:

Listing 6.52: The Complete Model implemented in LPL [22]

```
model hillP3_4_8 "Plant Production Planning";
  set
    plant ,p := [1..3];
    size  ,s := [Large, Medium, Small];
  parameter
    Profit{s}          := [420, 360, 300];
    PlantCapacity{p}   := [750, 900, 450];
    SpaceAvail{p}      := [13000, 12000, 5000];
    SpaceReq{s}        := [20, 15, 12];
    Demand{s}          := [900, 1200, 750];
  variable Produce{p,s};
  constraint
    CapacityLimit{p}: sum{s} Produce <= PlantCapacity;
    SpaceLimit{p}: sum{s} SpaceReq*Produce <= SpaceAvail;
    MaxDemand{s}: sum{p} Produce <= Demand;
    EqualCapUse{p|p<#p}: 1/PlantCapacity*(sum{s} Produce)
          = 1/PlantCapacity[p+1]*(sum{s} Produce[p+1,s]);
  maximize  TotalProfit: sum{p,s} Profit*Produce ;
  Writep(TotalProfit,Produce);
end
```

6.54. Cargo Plane Planning (hillP3-4-9)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [12] , Chapter 3.4, Problem 9, Page 75 A mathematical model formulation is:

Listing 6.53: The Complete Model implemented in LPL [22]

```
model hillP3_4_9 "Cargo Plane Planning";
  set
    c := [Front, Center, Back] "compartment";
    g :=[ 1..4] "cargo";
  parameter
    WeightCapacity{c} := [12, 18, 10];
    SpaceCapacity{c}  := [7000, 9000, 5000];
    CargoAvail{g} := [20, 16, 25, 13];
    Volume{g}  := [500, 700, 600, 400];
    Profit{g}  := [320, 400, 360, 290];
  variable AcceptCargo{c,g};
  constraint
    CargoLimit{g}: sum{c} AcceptCargo <= CargoAvail;
    WeightLimit{c}: sum{g} AcceptCargo <= WeightCapacity;
    SpaceLimit{c}: sum{g} Volume*AcceptCargo <=
        SpaceCapacity;
  maximize TotalProfit: sum{c,g} Profit*AcceptCargo;
  Writep(TotalProfit,AcceptCargo);
end
```

6.55. MidWest Grain Elevator (midwest)

— Run LPL Code , HTML Document –

Problem:

Midwest Grain Elevator: Case Summary Statement: Midwest Grain Elevator performs storage and handling services for buyers and sellers of grain on a service fee basis. To achieve maximum profit Midwest must maximize the amount of grain it handles. In order to accomplish this objective, Midwest must build customer loyalty on the part of both buyers and sellers of grain. Thus, Midwest's preferred policy is to encourage repeat business by satisfying buyers' orders at minimum cost. This policy benefits sellers as well as by increasing the number of orders for high quality grain that can be filled.

Currently Midwest is storing 507,900 bushels of corn for Saw Mill River Feed & Grain Company. This corn has been classified into 26 different types, which differ by moisture content, weight per bushel, amount of damage, and amount of foreign material. Saw Mill has limited quantities of each of the 26 types of corn. The ability to meet an order is obviously restricted by this supply. The quantities and cost of each type of corn is detailed in the table shown below. Corn Cost and Supply Table

Corn Type	Supply	Cost
1	15000	1.50
2	30000	1.45
3	45000	1.44
4	25000	1.45
5	40000	1.42
6	20000	1.38
7	30000	1.37
8	75000	1.37
9	15000	1.39
10	50000	1.39
11	16000	1.27
12	20000	1.28
13	10000	1.17
14	12000	1.22
15	6000	1.12
16	2000	1.18
17	5000	1.42
18	4000	1.42
19	5000	1.42
20	6000	1.36
21	1300	1.29

22	29000	1.29
23	1900	1.42
24	33000	1.40
25	6700	1.22
26	5000	1.30

Saw Mill has now received an order for 6 different grades of corn totaling

1. bushels which it intends to fill from the grain stored with Midwest. The Saw Mill must meet the stated demand for each of the 6 grades. Exceeding demand would be ill-advised, since there is no guarantee that the next order filled will require the same characteristics. The demand for each type of grade is shown in the table below.

Corn Grade	Demand
Gr1	40000
Gr2	32000
Gr3	50000
Gr4	20000
Gr5	30000
Gr6	100000

Midwest's handling fee will be based on supplying the 272,000 bushels of corn, regardless of the manner in which the various corns are blended to meet the specifications. However, in keeping with the policy of providing quality service to other buyers and sellers, the problem Midwest faces is that of determining the least cost blend of available grains that will satisfy the specified criteria of the current order. Each of the 6 grades of corn demanded must meet specified criteria for moisture content, minimum weight per bushel, percentage damaged, and percentage of foreign material, which are detailed in the table below. Corn Grade Characteristics:

Grade	MaxMoisture	MinWeight	MaxDamage	MaxForeign
Gr1	13.0	56.0	2.0	2.0
Gr2	15.5	54.0	5.0	3.0
Gr3	15.0	56.0	2.0	4.0
Gr4	16.0	54.0	9.0	4.0
Gr5	23.0	54.0	10.0	6.0
Gr6	20.0	54.0	9.0	4.0

For each of the 6 grades the four quality requirements are as follows: The weighted average moisture content cannot exceed the maximum

acceptable moisture level for that grade. The weighted average weight per bushel of the corns used in the blend of the grades must meet or exceed the minimum level. The weighted average percent damaged of the 26 types of corn used in the blend cannot exceed the maximum level. The weighted average percentage of foreign material that could be used in the blend cannot exceed the maximum requirement. The quality characteristics for each of the 26 corn types is given in the table below. Corn Quality Characteristics:

Corn Type	Moisture	Weight	Damage	Foreign
1	11.0	58.0	3.0	1.0
2	12.0	57.0	2.0	1.5
3	15.0	57.0	2.0	1.0
4	12.0	58.0	3.0	3.0
5	13.0	56.0	4.0	2.0
6	15.0	54.0	4.0	2.0
7	15.0	55.0	5.0	3.0
8	18.0	57.0	5.0	1.0
9	14.0	58.0	2.0	4.0
10	15.0	55.0	3.0	2.0
11	17.0	53.0	7.0	5.0
12	15.0	55.0	8.0	3.0
13	22.0	56.0	8.0	5.0
14	18.0	54.0	13.0	5.0
15	17.0	55.0	20.0	8.0
16	13.5	57.0	30.0	2.0
17	13.5	57.0	3.0	2.0
18	13.5	57.5	3.0	2.0
19	13.5	57.0	5.0	3.0
20	15.0	56.5	10.0	4.0
21	13.5	58.0	15.0	4.0
22	13.0	57.0	15.0	5.0
23	13.0	58.0	3.0	2.0
24	13.0	56.0	5.0	3.0
25	14.0	56.0	20.0	5.0
26	15.0	56.5	10.0	4.0

The objective for this problem is to determine the number of bushels used in the blending for each of the 26 types of corn so the the total cost of the 6 desired grades of corn is minimized.

Reference: Prof. Ramesh Sharda, Operations Research Dept., Oklahoma State University

Modeling Steps:

Listing 6.54: The Complete Model implemented in LPL [22]

```
model midwest "MidWest Grain Elevator";
  set
    type  := [1..26];
    grade := [Gr1, Gr2, Gr3, Gr4, Gr5, Gr6];
  parameter
    -- Supply characteristics
    Supply{type} := [15000, 30000, 45000, 25000, 40000,
        20000, 30000, 75000, 15000,
       50000, 16000, 20000, 10000, 12000,  6000,  2000,
           5000,  4000, 5000,  6000, 1300, 29000, 1900,
           33000, 6700, 5000];
    Cost{type} := [1.50, 1.45, 1.44, 1.45, 1.42, 1.38,
        1.37, 1.37, 1.39
     1.39, 1.27, 1.28, 1.17, 1.22, 1.12, 1.18, 1.42, 1.42
     1.42, 1.36, 1.29, 1.29, 1.42, 1.40, 1.22, 1.30 ];
    Moisture{type} := [11, 12, 15, 12, 13, 15, 15, 18,
        14,
      15, 17, 15, 22, 18, 17, 13.5, 13.5 13.5,
      13.5, 15, 13.5, 13, 13, 13, 14, 15];
    Weight{type} := [58.0, 57.0, 57.0, 58.0, 56.0, 54.0,
        55.0, 57.0, 58.0,
     55.0, 53.0, 55.0, 56.0, 54.0, 55.0, 57.0, 57.0,
         57.5,
     57.0, 56.5, 58.0, 57.0, 58.0, 56.0, 56.0, 56.5];
    Damage{type} := [ 3, 2, 2, 3, 4, 4, 5, 5, 2,
      3, 7, 8, 8, 13, 20, 30, 3, 3,
      5, 10, 15, 15, 3, 5, 20, 10];
    Foreign {type} := [1, 1.5, 1, 3, 2, 2, 3, 1, 4,
      2, 5, 3, 5, 5, 8, 2, 2, 2,
      3, 4, 4, 5, 2, 3, 5, 4];
    -- Demand Characteristics
    Demand{grade} := [40000, 32000, 50000, 20000, 30000,
        100000];
    MaxMoisture{grade} := [13, 115, 15, 16, 23, 20];
    MinWeight{grade}   := [56, 54, 56, 54, 54, 54];
    MaxDamage{grade}   := [2, 5, 2, 9, 10, 9];
    MaxForeign{grade}  := [2, 3, 4, 4,  6, 4];
  variable Bushels{type,grade};
  constraint
    QuantitySupplied{type}: sum{grade} Bushels <= Supply;
    QuantityDemanded{grade}: sum{type} Bushels  = Demand;
    MoistureContent{grade}: sum{type} Moisture*Bushels <=
         MaxMoisture*(sum{type} Bushels);
    WeightPerBushel{grade}: sum{type} Weight*Bushels >=
        MinWeight*(sum{type} Bushels);
    DamagePercent{grade}: sum{type} Damage*Bushels <=
        MaxDamage*(sum{type} Bushels);
    ForeignMaterial{grade}: sum{type} Foreign*Bushels <=
```

```
        MaxForeign*(sum{type} Bushels);
  minimize totalcost: sum{type,grade} Cost*Bushels;
  Writep(totalcost,Bushels);
end
```

Solution:

opt: 366863;

6.56. Fertilizer (murty2-1)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [15], Example 2.1, Page 23 A mathematical model formulation is:

Listing 6.55: The Complete Model implemented in LPL [22]

```
model murty2_1 "Fertilizer";
  set
    fertilizer,f :=  ['Hi-ph', 'Lo-ph'];
    material  ,m :=  [RM1, RM2, RM3];
  parameter
    NetProfit{f}    :=  [15, 10];
    RawMatReq{f,m}  :=  [2, 1, 1, 1, 1, 0];
    RawMatAvail{m}  :=  [1500, 1200, 500];
  variable Produce{f};
  constraint Req{m}: sum{f} RawMatReq*Produce <=
      RawMatAvail;
  maximize TotalProfit: sum{f} NetProfit*Produce;
  Writep(TotalProfit,Produce);
end
```

6.57. Gasoline Blending (murty2-2)

— Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [15], Example 2.2, Page 28 A mathematical model formulation is:

Listing 6.56: The Complete Model implemented in LPL [22]

```
model murty2_2 "Gasoline Blending";
  set
    rawgas,r := [1..4];
    fueltype,f := [1..3];
  parameter
    OctaneRating{r}     := [68, 86, 91, 99];
    RawGasAvail{r}      := [4000, 5050, 7100, 4300];
    RawGasCost{r}       := [31.02, 33.15, 36.35, 38.75];
    RawGasPrice{r}      := [36.85, 36.85, 38.95, 38.95];
    MinOctaneFuel{f}  := [95, 90, 85];
    FuelPrice{f}      := [45.15, 42.95, 40.99];
    MinimumDemand{f}  := [0, 0, 15000];
    MaximumDemand{f}  := [10000, 0, 0];
  variable
    FuelComposition,FC{f,r};
    RawGasSold,RG{r};
  expression
    TotalFuelRevenue  : sum{r,f} FuelPrice*FC;
    TotalRawGasRevenue: sum{r} RawGasPrice*RG;
    TotalRawGasCost   : sum{r,f} RawGasCost*FC + sum{r}
        RawGasCost*RG;
  constraint
    RawGasLimit{r}: sum{f} FC + RG <= RawGasAvail;
    OctaneReq{f}: sum{r} OctaneRating*FC >= MinOctaneFuel
        *(sum{r} FC);
    MinDemand{f|MinimumDemand}: MinimumDemand <= sum{r}
        FC;
    MaxDemand{f|MaximumDemand}: sum{r} FC <=
        MaximumDemand;
  maximize TotalProfit: TotalFuelRevenue +
     TotalRawGasRevenue - TotalRawGasCost;
  Writep(TotalProfit,FC,RG);
end
```

6.58. Diet Problem (murty2-3)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [15], Example 2.3, Page 31 A mathematical model formulation is:

Listing 6.57: The Complete Model implemented in LPL [22]

```
model murty2_3 "Diet Problem";
  set
    grain  := [1..2];
    nutri  := [Starch, Protein, Vitamins];
  parameter
    NutriUnits{nutri,grain} := [5, 7,  4, 2,  2, 1];
    MinDailyReq{nutri} := [8, 15, 3];
    GrainCost{grain}   := [0.60, 0.35];
  variable Amount{grain};
  constraint
    Requirement{nutri}: sum{grain} NutriUnits*Amount >=
        MinDailyReq;
  minimize TotalCost: sum{grain} GrainCost*Amount;
  Writep(TotalCost,Amount);
end
```

6.59. Transportation (murty2-4)

— Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [15],Example 2.4, Page 33 A mathematical model formulation is:

Listing 6.58: The Complete Model implemented in LPL [22]

```
model murty2_4 "Transportation";
  set
    mine  := [1..2];
    plant := [1..3];
  parameter
    Availability{mine}   := [800, 300];
    Requirement{plant}   := [400, 500, 200];
    ShipCost{mine,plant} := [11, 8, 2,   7, 5, 4];
  variable Ship{mine,plant};
  constraint
    OreLimit{mine}: sum{plant} Ship <= Availability;
    OreReq{plant}: sum{mine} Ship >=  Requirement;
  minimize TotalCost: sum{mine,plant} ShipCost*Ship;
  Writep(TotalCost,Ship);
end
```

6.60. Marriage Problem (murty2-5)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [15],Example 2.5, Page 36 A mathematical model formulation is:

Listing 6.59: The Complete Model implemented in LPL [22]

```
model murty2_5 "Marriage Problem";
  set
    man   := [1..5];
    woman := [1..5];
  parameter
    Happyness{man,woman} := [ 78, -16,  19,  25,  83,
                              99,  98,  87,  16,  92,
                              86,  19,  39,  88,  17,
                             -20,  99,  88,  79,  65,
                              67,  98,  90,  48,  60];
  variable TimeTogether{man,woman};
  constraint
    MenMonogamous{man}: sum{woman} TimeTogether = 1;
    WomenMonogamous{woman}: sum{man} TimeTogether = 1;
  maximize TotalHappyness: sum{man,woman} Happyness*
      TimeTogether;
  Writep(TotalHappyness,TimeTogether);
end
```

6.61. Multi-Period Planning (murty2-6)

— Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [15], Example 2.6, Page 40 A mathematical model formulation is:

Listing 6.60: The Complete Model implemented in LPL [22]

```
model murty2_6 "Multi-Period Planning";
  set period,p := [1..6];
  parameter
    ProdCost{p}     := [20, 25, 30, 40, 50, 60];
    ProdCapacity{p}:= [1500 2000 2200 3000 2700 2500];
    Demand{p}       := [1100 1500 1800 1600 2300 2500];
    SellingPrice{p}:= [180, 180, 250, 270, 300, 320];
    InventoryCost   := 2;
    InitialStock    := 500;
    FinalStock      := 500;
  variable
    Produce,P{p} [0..ProdCapacity];
    Inventory,I{p};
    Sales,S{p} [0..Demand];
  expression
    TotalRevenue : sum{p} SellingPrice*S;
    TotalProdCost: sum{p} ProdCost*P;
    TotalInvtCost: sum{p} InventoryCost*I;
    TotalCost    : TotalProdCost + TotalInvtCost;
  constraint
    InventoryBalance{p}: I = if(p=1,InitialStock,I[p-1])
        + P - S;
    F: I[6] = FinalStock;
  maximize NetProfit: TotalRevenue - TotalCost;
  Writep(NetProfit,P,I,S);
end
```

6.62. Breck and Dapper (shapiro1-1)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [20], Example 1.1, Page 12. A mathematical model formulation is:

Listing 6.61: The Complete Model implemented in LPL [22]

```
model shapiro1_1 "Breck and Dapper";
  set drill  :=  [1..4];
  parameter
    PlastReq{drill}  := [0.82, 0.62, 1.42, 2.03];  -- lb
    CopperReq{drill} := [0.43, 0.69, 0.33, 0.20];  -- lb
    WireReq{drill}   := [15, 16, 9, 9];            -- yd
    Contrib{drill}   := [12.5 11.3 17.2 19.9]; -- dollars
    PlastAvail       :=  16000;      -- lb
    CopperAvail      :=   5000;      -- lb
    WireOnHand       :=   8000;      -- yd
    ProdWireCap      :=  80000;      -- yd
    ProdWireRate     :=  3.6/100;    -- lb/yd
    WireProdCost     :=  0.14;
    WirePurchaseCost :=  0.29;
  variable
    Produce,x{drill};
    WirePurchase,Wp;
    WireProduce,Wm [0..ProdWireCap];
  constraint
    PlastLimit: sum{drill} PlastReq*x <= PlastAvail;
    CopperLimit: sum{drill} CopperReq*x + ProdWireRate*Wm
        <=  CopperAvail;
    WireLimit: sum{drill} WireReq*x <=  WireOnHand + Wm +
        Wp;
    Marketing: x[1] + x[2] >= x[3] + x[4];
  maximize TotalProfit: sum{drill} Contrib*x -
     WireProdCost*Wm - WirePurchaseCost*Wp;
  Writep(TotalProfit,x,Wp,Wm);
end
```

6.63. DowPont Chemical (shapiro1-2)

— Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [20], Example 1.2, Page 17. A mathematical model formulation is:

Listing 6.62: The Complete Model implemented in LPL [22]

```
model shapiro1_2 "DowPont Chemical";
  set
    chemical,c :=  [R,A,B,C,D];
    process,p  :=  [1..3];
  parameter
    RawMatCost         := 1;
    ProcessCapacity{p} := [56, 25, 46];
    MarketPrice{c}     := [0.00, 2.40, 3.20, 6.40, 12.20];
    ProcessCost{p,c} := /1 R 1.50, 2 A 3.50, 3 B 4.20, 3
        C 4.20/;
    ProcessOutput{p,c} := /1 A 0.45, 1 B 0.55, 2 C 0.80,
        3 D 0.75/;
  variable
   ProdSale,S{c};
   ProdUse,U{c};
  constraint
    r1: U['A']+S['A'] = ProcessOutput['1','A'] * U['R'];
    r2: U['B']+S['B'] = ProcessOutput['1','B'] * U['R'];
    r3: U['C']+S['C'] = ProcessOutput['2','C'] * U['A'];
    r4: S['D'] = ProcessOutput['3','D']*(U['B'] + U['C'])
        ;
    r5: U['B']  =  1.5*U['C'];
    r6: U['R']        <=  ProcessCapacity['1'];
    r7: U['A']        <=  ProcessCapacity['2'];
    r8: U['B'] + U['C'] <= ProcessCapacity['3'];
  maximize TotalProfit:
    sum{c} MarketPrice*S - RawMatCost*U['R'] - sum{c,p}
        ProcessCost*U;
  Writep(TotalProfit,S,U);
end
```

6.64. Portfolio Selection (shapiro1-3)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [20], Example 1.3, Page 21. A mathematical model formulation is:

Listing 6.63: The Complete Model implemented in LPL [22]

```
model shapiro1_3 "Portfolio Selection";
  set
    bond  :=  [A, B, C, D, E];
    type  :=  [Municipal, Agency, Government];
    BondType{bond,type} := [A Municipal,
                            B Agency,
                            C Government,
                            D Government,
                            E Municipal];
  parameter
    TotalAvailForInvest   := 10;
    QualityBank{bond}     := [2, 2, 1, 1, 5];
    YearsToMaturity{bond} := [9, 15, 4, 3, 2];
    Yield{bond}           := [0.086, 0.108, 0.1, 0.088,
        0.09];
    YieldAfterTax{bond}   := [0.086, 0.054, 0.05, 0.044,
        0.09];
    MaxMunicipal          := 3;
    MaxAvgQuality         := 1.4;
    MaxAvgMaturity        := 5;
    TaxRate               := 0.5;
  variable Invest,x{bond};
  constraint
    AmountInvested: sum{bond} x <= TotalAvailForInvest;
    MaxInvestType{type|type='Municipal'}: sum{bond|
        BondType} x <= MaxMunicipal;
    QualityReq: sum{bond} QualityBank*x <= MaxAvgQuality
        *(sum{bond} x);
    MaturityReq: sum{bond} YearsToMaturity*x <=
        MaxAvgMaturity*(sum{bond} x);
  maximize AfterTaxEarnings:  sum{bond} YieldAfterTax*x;
  Writep(AfterTaxEarnings,x);
end
```

6.65. Portfolio Selection (shapiro1-3b)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [20], Example 1.3b, Page 21. A mathematical model formulation is:

Listing 6.64: The Complete Model implemented in LPL [22]

```
model shapiro1_3b "Portfolio Selection";
  set bond  :=  [A, B, C, D, E];
      type  :=  [Municipal, Agency, Government];
      BondType{bond,type} := [A Municipal,
                              B Agency,
                              C Government,
                              D Government,
                              E Municipal];
  parameter
    TotalAvailForInvest   := 10;
    QualityBank{bond}     := [2, 2, 1, 1, 5];
    YearsToMaturity{bond} := [9, 15, 4, 3, 2];
    Yield{bond}         := [0.086 0.108 0.1 0.088 0.09];
    YieldAfterTax{bond} := [0.086 0.054 0.05 0.044 0.09];
    MaxMunicipal          := 3;
    MaxAvgQuality         := 1.4;
    MaxAvgMaturity        := 5;
    TaxRate               := 0.5;
    BorrowInterestRate    := 0.11;
  variable
    Invest,x{bond};
    Borrow,y;
  constraint
    AmountInvested: sum{bond} x <= TotalAvailForInvest+y;
    MaxInvestType{type|type='Municipal'}: sum{bond|
        BondType} x <= MaxMunicipal;
    QualityReq: sum{bond} QualityBank*x <= MaxAvgQuality
        *(sum{bond} x);
    MaturityReq: sum{bond} YearsToMaturity*x <=
        MaxAvgMaturity*(sum{bond} x);
    MaxBorrow: Borrow <= 1;
  maximize AfterTaxEarnings: sum{bond} YieldAfterTax*x -
      BorrowInterestRate*TaxRate*y;
  Writep(AfterTaxEarnings,x,y);
end
```

6.66. Transportation (shapiro1-4)

—- Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [20], Example 1.4, Page 24. A mathematical model formulation is:

Listing 6.65: The Complete Model implemented in LPL [22]

```
model shapiro1_4 "Transportation";
  set
    mill     :=  [1..2];
    market   :=  [Northburg, Midburg, Southburg];
  parameter
    WheatAvail              := 50000;
    MillingCost{mill}       := [0.13, 0.16];
    MarketDemand{market}    := [10000, 15000, 25000];
    ShipCostMill{mill}      := [0.31, 0.33];
    ShipCostMarket{market, mill} := [0.18, 0.25,  0.22,
        0.23,  0.27, 0.19];
  variable
    ShipMill{mill};
    ShipMarket{mill, market};
  constraint
    WheatToMills: sum{mill} ShipMill = WheatAvail;
    MillBalance{mill}: ShipMill = sum{market} ShipMarket;
    MarketReq{market}: sum{mill} ShipMarket =
        MarketDemand;
  maximize TotalCost:
    sum{mill} MillingCost*ShipMill
    + sum{mill} ShipCostMill*ShipMill
    + sum{mill,market} ShipCostMarket*ShipMarket;
  Writep(TotalCost,ShipMill,ShipMarket);
end
```

6.67. Multi-Period Scheduling (shapiro1-5)

— Run LPL Code , HTML Document –

Problem:

A complete problem description and how to model it mathematically is clearly explained in [20], Example 1.5, Page 28. A mathematical model formulation is:

Listing 6.66: The Complete Model implemented in LPL [22]

```
model shapiro1_5 "Multi-Period Scheduling";
  set  m,month   := [1..12];
  parameter
    Demand{m} := [3800, 4100, 4700, 5900, 7200, 7900,
       6700, 5100, 5300, 6400, 5800, 4800];
    ProductionCost  := 100;   InventoryCost   := 7;
    LaborCost       := 1600;  HiringCost      := 300;
    LayoffCost      := 500;   StockOutCost    := 500;
    RecordersPerDay := 8;     AssemblyCapacity:= 7000;
    MaxNewWorkers   := 50;    MaxLayoffPercent:= 0.1;
    InitInventory   := 750;   InitWorkforce   := 500;
  variable
    Produce,P{m};
    Inventory,I{m};  Ie "final Inventory";
    Workers,W{m};    Hired,H{m};
    LaidOff,L{m};    StockOut,S{m};
  expression
    TotalProductionCost: sum{m} ProductionCost*P ;
    TotalInventoryCost : sum{m} InventoryCost*1/2*(I+if(m
       =#m,Ie,I[m+1]));
    TotalLaborCost     : sum{m} LaborCost*W ;
    TotalHiringCost    : sum{m} HiringCost*H ;
    TotalLayoffCost    : sum{m} LayoffCost*L ;
    TotalStockOutCost  : sum{m} StockOutCost*S ;
  constraint
    InvBalance{m}: P+I-if(m=#m,Ie,I[m+1])+S = Demand;
    WorkforceLimit{m}: P <= RecordersPerDay*W;
    AssemblyLimit{m}: P <= AssemblyCapacity;
    WorkforceBalance{m}: W = if(m=1,InitWorkforce,W[m-1])
        + H - L;
    MaxHired{m}: H <= MaxNewWorkers;
    MaxLaidOff{m}: L <= MaxLayoffPercent * if(m=1,
       InitWorkforce,W[m-1]);
    II: I[1] = InitInventory;
  minimize TotalCost:  TotalProductionCost +
     TotalInventoryCost
```

```
    + TotalLaborCost + TotalHiringCost + TotalLayoffCost
        + TotalStockOutCost;
  Writep(TotalCost,P,I,W,H,L,S);
end
```

CHAPTER 7

WINSTON

7.1. Giapetto Wood Carving (winst3-1-1)

—- Run LPL Code , HTML Document –

Problem: "Giapetto's Woodcarving, Inc., manufactures two types of wooden toys: soldiers and trains. A soldier sells for $27 and uses $10 worth of raw materials. Each soldier that is manufactured increases Giapetto's variable labor and overhead costs by $14. A train sells for $21 and uses $9 worth of raw materials. Each train built increases Giapetto's variable labor and overhead costs by $10. The manufacture of wooden soldiers and trains requires two types of skilled labor: carpentry and finishing. A soldier requires 2 hours of finishing labor and 1 hour of carpentry labor. A train requires 1 hour of finishing and 1 hour of carpentry labor. Each week, Giapetto can obtain all the needed raw material but only 100 finishing hours and 80 carpentry hours. Demand for trains is unlimited, but at most 40 soldiers are bought each week. Giapetto wants to maximize weekly profit (revenues - costs). Formulate a mathematical model of Giapetto's situation that can be used to maximize Giapetto's weekly profit."This problem is from [24] , Example 3.1-1, Page 45.

Modeling Steps:

Listing 7.1: The Complete Model implemented in LPL [22]

```
model winst3_1_1 "Giapetto Wood Carving";
  set
    t   := [Soldier, Train] "Toys";
    w := [Carpentry, Finishing] "Labor";
  parameter
    Price{t}        := [27, 21];
    VariableCost{t} := [14, 10];
    RawMatCost{t}   := [10, 9];
    Profit{t}       := Price - VariableCost - RawMatCost;
    LaborHours{t,w} := [1, 2,  1, 1];
    LaborAvail{w} := [80, 100];
```

```
    MaxSoldier        := 40;
  variable Produce,P{t};
  constraint
    LaborCapacity{w} : sum{t} LaborHours*P <= LaborAvail;
    MaxDemand:  P['Soldier'] <= MaxSoldier;
  maximize TotalProfit : sum{t} Profit*P;
  Writep(TotalProfit,P);
end
```

7.2. Giapetto Wood Carving II (winst3-1-1s)

—- Run LPL Code , HTML Document –

Problem: "Giapetto's Woodcarving, Inc., manufactures two types of wooden toys: soldiers and trains. A soldier sells for $27 and uses $10 worth of raw materials. Each soldier that is manufactured increases Giapetto's variable labor and overhead costs by $14. A train sells for $21 and uses $9 worth of raw materials. Each train built increases Giapetto's variable labor and overhead costs by $10. The manufacture of wooden soldiers and trains requires two types of skilled labor: carpentry and finishing. A soldier requires 2 hours of finishing labor and 1 hour of carpentry labor. A train requires 1 hour of finishing and 1 hour of carpentry labor. Each week, Giapetto can obtain all the needed raw material but only 100 finishing hours and 80 carpentry hours. Demand for trains is unlimited, but at most 40 soldiers are bought each week. Giapetto wants to maximize weekly profit (revenues - costs). Formulate a mathematical model of Giapetto's situation that can be used to maximize Giapetto's weekly profit."This problem is from [24] , Example 3.1-1, Page 45.

Modeling Steps:

Listing 7.2: The Complete Model implemented in LPL [22]

```
model winst3_1_1s "Giapetto Wood Carving II";
  variable  soldier; train;
  constraint
    carpentry: 1*soldier + 1*train <= 80;
    finishing: 2*soldier + 1*train <= 100;
    maxsoldier: soldier <=40;
  maximize Profit: 27*soldier + 21*train - 24*soldier -
     19*train;
  Writep(Profit);
end
```

7.3. Sailco (winst3-10-12)

—- Run LPL Code , HTML Document –

Problem: "Sailco Corporation must determine how many sailboats should be produced during each of the next four quarters (one quarter = three months). The demand during each of the next four quarters is as follows: first quarter, 40 sailboats; second quarter, 60 sailboats; third quarter, 75 sailboats; fourth quarter, 25 sailboats. Sailco must meet demands on time. At the beginning of the first quarter, Sailco has an inventory of 10 sailboats. At the beginning of each quarter, Sailco must decide how many sailboats should be produced during that quarter. For simplicity , we assume that sailboats manufactured during a quarter can be used to meet demand for that quarter. During each quarter, Sailco can produce up to 40 sailboats with regular-time labor at a total cost of $400 per sailboat. By having employees work overtime during a quarter, Sailco can produce additional sailboats with overtime labor at a total cost of $450 per sailboat.

At the end of each quarter (after production has occurred and the current quarter's demand has been satisfied), a carrying or holding cost of $20 per sailboat is incurred. Use linear programming to determine a production schedule to minimize the sum of production and inventory costs during the next four quarters." This probelm is from [24] , Example 3.10-12, Page 95.

Modeling Steps:

Listing 7.3: The Complete Model implemented in LPL [22]

```
model winst3_10_12 "Sailco";
  set
    quarter,q := [1, 2, 3, 4];
    labor,a := [reg, OT];
  parameter
    Demand{q}         := [40, 60, 75, 25];
    LaborCost{a}      := [400.00, 450.00];
    HoldCost{q}       := [20.00, 20.00, 20.00, 20.00];
    StartInven        := 10;
  variable
    Produce,P{q,a};
    Inventory,I{q};
  constraint
    ProdCapacity{q}: sum{a} P + if(q=1,StartInven,I[q-1])
         = I + Demand;
    up{q}: Produce[q,'reg'] <= 40;
  minimize TotalCost: sum{q,a} LaborCost*P + sum{q}
      HoldCost*I;
  Writep(TotalCost);
```

```
end
```

7.4. Dorian Advertising (winst3-2-2)

— Run LPL Code , HTML Document –

Problem: "Dorian Auto manufactures luxury cars and trucks. The company believes that its most likely customers are high-income women and men. To reach these groups, Dorian Auto has embarked on an ambitious TV advertising campaign and has decided to purchase 1-minute commercial spots on two types of programs: comedy shows and football games. Each comedy commercial is seen by 7 million high-income women and 2 million high-income men. Each football commercial is seen by 2 million high-income women and 12 million high-income men. A 1-minute comedy ad costs $50,000, and a 1-minute football ad costs $100,000. Dorian would like the commercials to be seen by at least 28 million high-income women and 24 million high-income men. Use linear programming to determine how Dorian Auto can meet its advertising requirements at minimum cost." This probelm is from [24] , Example 3.2-2, Page 57.

Modeling Steps:

Listing 7.4: The Complete Model implemented in LPL [22]

```
model winst3_2_2 "Dorian Advertising";
  set
    a    := [Comedy, Football] "adtype";
    c := [HIW, HIM] "customers";
  parameter
    MinuteCost{a}       := [50, 100];
    SeenBy{a,c} := [7,  2,   2, 12];
    TargetReach{c}   := [28, 24];
  variable Purchase{a};
  constraint
    MustReach{c}: sum{a} SeenBy*Purchase >= TargetReach;
  minimize TotalCost: sum{a} MinuteCost*Purchase ;
  Writep(TotalCost,Purchase);
end
```

7.5. Dorian Advertising (winst3-2-2s)

—- Run LPL Code , HTML Document –

Problem: "Dorian Auto manufactures luxury cars and trucks. The company believes that its most likely customers are high-income women and men. To reach these groups, Dorian Auto has embarked on an ambitious TV advertising campaign and has decided to purchase 1-minute commercial spots on two types of programs: comedy shows and football games. Each comedy commercial is seen by 7 million high-income women and 2 million high-income men. Each football commercial is seen by 2 million high-income women and 12 million high-income men. A 1-minute comedy ad costs $50,000, and a 1-minute football ad costs $100,000. Dorian would like the commercials to be seen by at least 28 million high-income women and 24 million high-income men. Use linear programming to determine how Dorian Auto can meet its advertising requirements at minimum cost." This probelm is from [24] , Example 3.2-2, Page 57.

Modeling Steps:

Listing 7.5: The Complete Model implemented in LPL [22]

```
model winst3_2_2s "Dorian Advertising";
  variable Comedy; Football;
  constraint
    Women:    7*Comedy + 2*Football  >= 28;
    Men:      2*Comedy + 12*Football >= 24;
  minimize Cost: 50*Comedy + 100*Football;
  Writep(Cost);
end
```

7.6. Auto Company (winst3-3-3)

— Run LPL Code , HTML Document –

Problem: "An auto company manufactures cars and trucks. Each vehicle must be processed in the paint shop and body assembly shop. If the paint shop were only painting trucks, 40 per day could be painted. If the paint shop were only painting cars, 60 per day could be painted. If the body shop were only producing cars, it could process 50 per day. If the body shop were only producing trucks, it could process 50 per day. Each truck contributes $300 to profit, and each car contributes $200 to profit. Use linear programming to determine a daily production schedule that will maximize the company's profits." This probelm is from [24] , Example 3.3-3, Page 61.

Modeling Steps:

Listing 7.6: The Complete Model implemented in LPL [22]

```
model winst3_3_3 "Auto Company";
  set
    vehicle := [Cars, Trucks];
    shop    := [Paint, BodyAssembly];
  parameter
    Profit{vehicle}          := [200, 300];
    CapPerDay{shop, vehicle} := [60, 40,  50, 50];
  variable Produce{vehicle};
  constraint
    ProdCapacity{shop}: sum{vehicle} 1/CapPerDay*Produce
        <= 1;
  maximize TotalProfit: sum{vehicle} Profit*Produce ;
  Writep(TotalProfit,Produce);
end
```

7.7. Auto Company (winst3-3-3s)

—- Run LPL Code , HTML Document –

Problem: "An auto company manufactures cars and trucks. Each vehicle must be processed in the paint shop and body assembly shop. If the paint shop were only painting trucks, 40 per day could be painted. If the paint shop were only painting cars, 60 per day could be painted. If the body shop were only producing cars, it could process 50 per day. If the body shop were only producing trucks, it could process 50 per day. Each truck contributes $300 to profit, and each car contributes $200 to profit. Use linear programming to determine a daily production schedule that will maximize the company's profits." This probelm is from [24] , Example 3.3-3, Page 61.

Modeling Steps:

Listing 7.7: The Complete Model implemented in LPL [22]

```
model winst3_3_3s "Auto Company";
  variable Cars; Trucks;
  constraint
     PaintShop:    1/60*Cars + 1/40*Trucks <= 1;
     BodyAssembly: 1/50*Cars + 1/50*Trucks <= 1;
  maximize Profit: 300*Trucks + 200*Cars;
  Writep(Profit);
end
```

7.8. Unbounded Problem (winst3-3-5)

—- Run LPL Code , HTML Document –

Problem: "Graphically solve the following LP". This probelm is from [24] , Example 3.3-5, Page 64.

$$\begin{array}{ll} \max & z = 2x_1 - x_2 \\ \text{subject to} & x_1 - x_2 \leq 1 \\ & 2x_1 + x_2 \geq 6 \\ & x_1, x_2 \geq 0 \end{array}$$

Modeling Steps:

Listing 7.8: The Complete Model implemented in LPL [22]

```
model winst3_3_5 "Unbounded Problem";
  variable x1; x2;
  constraint r1: x1-x2<=1 and 2*x1+x2>=6;
  maximize  z: 2*x1-x2;
  Writep(z);
  Write('the problem is unbounded\n');
end
```

7.9. Diet Problem (winst3-4-6)

—- Run LPL Code , HTML Document –

Problem: "My diet requires that all the food I eat come from one of the four "basic food groups" (chocolate cake, ice cream, soda, and cheesecake). At present, the following four foods are available for consumption: brownies, chocolate ice cream, cola, and pineapple cheesecake. Each brownie costs 50 cents, each scoop of chocolate ice cream costs 20 cents, each bottle of cola costs 30 cents, and each piece of pineapple cheesecake costs 80 cents. Each day, I must ingest at least 500 calories, 6 oz of chocolate, 10 oz of sugar, and 8 oz of fat. The nutritional content per unit of each food is shown in Table 2. Formulate a linear programming model that can be used to satisfy my daily nutritional requirements at minimum cost." This probelm is from [24] , Example 3.4-6, Page 66.

Modeling Steps:

Listing 7.9: The Complete Model implemented in LPL [22]

```
model winst3_4_6 "Diet Problem";
  set
     foods     := [Brownie, Icecream, Cola, Cheesecake];
     nutrients := [Calories, Chocolate, Sugar, Fat];
  parameter
    FoodCost{foods}     := [0.50, 0.20, 0.30, 0.80];
    NutriReq{nutrients} := [500, 6, 10, 8];
    NutriValues{foods,nutrients} := [400,   3,  2,  2,
                                     200,   2,  2,  4,
                                     150,   0,  4,  1,
                                     500,   0,  4,  5];
  variable Consume{foods};
  constraint
    Required{nutrients}: sum{foods} NutriValues*Consume
        >= NutriReq;
  minimize Cost: sum{foods} FoodCost*Consume;
  Writep(Cost,Consume);
end
```

7.10. Diet Problem (winst3-4-6s)

—- Run LPL Code , HTML Document –

Problem: "My diet requires that all the food I eat come from one of the four "basic food groups" (chocolate cake, ice cream, soda, and cheesecake). At present, the following four foods are available for consumption: brownies, chocolate ice cream, cola, and pineapple cheesecake. Each brownie costs 50 cents, each scoop of chocolate ice cream costs 20 cents, each bottle of cola costs 30 cents, and each piece of pineapple cheesecake costs 80 cents. Each day, I must ingest at least 500 calories, 6 oz of chocolate, 10 oz of sugar, and 8 oz of fat. The nutritional content per unit of each food is shown in Table 2. Formulate a linear programming model that can be used to satisfy my daily nutritional requirements at minimum cost." This problem is from [24] , Example 3.4-6, Page 66.

Modeling Steps:

Listing 7.10: The Complete Model implemented in LPL [22]

```
model winst3_4_6s "Diet Problem";
  variable Brownie,B; Icecream,I; Cola,C; Cheesecake,CC;
  constraint
     Calories:  400*B + 200*I + 150*C + 500*CC >= 500;
     Chocolate:   3*B +   2*I                  >=  6;
     Sugar:       2*B +   2*I +   4*C +   4*CC >= 10;
     Fat:         2*B +   4*I +     C +   5*CC >=  8;
  minimize Cost: 0.50*B + 0.20*I + 0.30*C + 0.80*CC;
  Writep(Cost);
end
```

7.11. Star Oil (winst3-6-8)

—- Run LPL Code , HTML Document –

Problem: "Star Oil Company is considering five different investment opportunities. The cash outflows and net present values (in millions of dollars) are given in Table 7.22. Star Oil has $40 million available for investment at the present time (time 0); it estimates that one year from now (time 1) $20 million will be available for investment. Star Oil may purchase any fraction of each investment. In this case, the cash outflows and NPV are adjusted accordingly. For example, if Star Oil purchases one fifth of investment 3, then a cash outflow of 5 (5) - $1 million would be required at time 0, and a cash outflow of -Is– (5) $1 million would be required at time 1. The one-fifth share of investment 3 would yield an NPV of I (16) $3.7 million. Star Oil wants to maximize the NPV that can be obtained by investing in investments 1-5. Formulate an LP that will help achieve this goal. Assume that any funds left over at time 0 cannot be used at time 1. " This probelm is from [24] , Example 3.6-8, Page 74.

	Inv.1	Inv.2	Inv.3	Inv.4	Inv.5
Time 0 cash outflow	$11	$53	$5	$5	$29
Time 1 cash outflow	$3	$6	$5	$1	$34
NPV	$13	$16	$16	$14	$39

Table 7.1:

Modeling Steps:

Listing 7.11: The Complete Model implemented in LPL [22]

```
model winst3_6_8 "Star Oil";
  set
    invest := [1..5];
    time   := [0..1];
  parameter
    CashAvail{time} := [40, 20];
    CashOutflow{time,invest} := [11, 53,  5,  5, 29,
                                  3,  6,  5,  1, 34];
    NPV{invest}     := [13, 16, 16, 14, 39];
  variable Purchase,P{invest} [0..1];
  constraint
    InvestLimit{time}: sum{invest} CashOutflow*P <=
        CashAvail;
  maximize TotalNetPresentValue: sum{invest} NPV*P;
  Writep(TotalNetPresentValue,P);
```

```
end
```

7.12. Semicond Eletronics (winst3-7-9)

—- Run LPL Code , HTML Document –

Problem: "Semicond is a small electronics company that manufactures tape recorders and radios. The per-unit labor costs, raw material costs, and selling price of each product aregiven in Table 7.22. On December 1, 1998, Semicond has available raw material that is sufficient to manufacture 100 tape recorders and 100 radios. On the same date, the company's balance sheet is as shown in Table 7.17, and Semicond's asset/liability ratio (called the current ratio) is 20,000/10,000 = 2.

Semicond must determine how many tape recorders and radios should be produced during December. Demand is large enough to ensure that all goods produced will be sold. All sales are on credit, however, and payment for goods produced in December will not be received until February 1, 1999. During December, Semicond will collect $2000 in accounts receivable, and Semicond must pay off $1000 of the outstanding loan and a monthly rent of $1000. On January 1, 1999, Semicond will receive a shipment of raw material worth $2000, which will be paid for on February 1, 1999. Semicond's management has decided that the cash balance on January 1, 1999 must be at least $4000. Also, Semicond's bank requires that the current ratio at the beginning of January be at least 2. To maximize the contribution to profit from December production, (revenues to be received) - (variable production costs), what should Semicond produce during December?" This probelm is from [24] , Example 3.7-9, Page 77.

	Tape Recorder	Radio
Selling Price	$100	$90
Labor cost	$ 50	$35
Raw material cost	$ 30	$40

Table 7.2:

	Assets	Liabilities
Cash	$10,000	
Accounts receivable*	$ 3,000	
Inventory outstanding**	$ 7,000	
Bank loan		$ 10,000

Table 7.3:

* Accounts receivable is money owed to Semicond by customers who have previously purchased Semicond products.

** Value of December 1, 1998 inventory = 30(100)+ 40(100) = $7000

Modeling Steps:

Listing 7.12: The Complete Model implemented in LPL [22]

```
model winst3_7_9 "Semicond Eletronics";
  set p := [TapeRec, Radio] "products";
  parameter
    SellingPrice{p}  := [100, 90];
    LaborCost{p}     := [50, 35];
    RawMatCost{p}    := [30, 40];
    ContribProfit{p} := SellingPrice - LaborCost -
        RawMatCost;
    RawMatAvail{p}   := [100, 100];
    CashOnHandDec1      := 10000;
    AccountsRecDec1     :=  3000;
    InventoryValueDec1  :=  7000;
    LiabilitiesDec1     := 10000;
    RawMatReceivedJan1  :=  2000;
    AccRecCollectDec    :=  2000;
    LoanPaymentDec      :=  1000;
    MonthlyRentDec      :=  1000;
    MinCashOnHandJan    :=  4000;
    MinAssetLiabRatio   :=  2;
  variable ProduceDec,P{p};
  expression
    CashPositionJan1: CashOnHandDec1 + AccRecCollectDec -
         LoanPaymentDec
      - MonthlyRentDec - sum{p} LaborCost * P;
    AccountsRecJan1: AccountsRecDec1 + sum{p}
        SellingPrice*P - AccRecCollectDec;
    InventoryValueJan1: InventoryValueDec1 - sum{p}
        RawMatCost*P + RawMatReceivedJan1;
    LiabilitiesJan1: LiabilitiesDec1 - LoanPaymentDec +
        RawMatReceivedJan1;
    AssetsJan1: CashPositionJan1 + AccountsRecJan1 +
        InventoryValueJan1;
  constraint
    RawMatLimit{p}: P <= RawMatAvail;
    CashBalance: CashPositionJan1 >= MinCashOnHandJan;
    AssetLiabRatio: AssetsJan1 >= MinAssetLiabRatio*
        LiabilitiesJan1;
  maximize TotalProfit: sum{p} ContribProfit*ProduceDec ;
  Writep(TotalProfit,P);
end
```

7.13. Rylon Corporation (winst3-9-11)

—- Run LPL Code , HTML Document –

Problem: "Rylon Corporation manufactures Brute and Chanelle perfumes. The raw material needed to manufacture each type of perfume can be purchased for $3 per pound. Processing 1 lb of raw material requires 1 hour of laboratory time. Each pound of processed raw material yields 3 oz of Regular Brute Perfume and 4 oz of Regular Chanelle Perfume. Regular Brute can be sold for $7/oz and Regular Chanelle for $6/oz. Rylon also has the option of further processing Regular Brute and Regular Chanelle to produce Luxury Brute, sold at $18/oz, and Luxury Chanelle, sold at $14/oz. Each ounce of Regular Brute processed further requires an additional 3 hours of laboratory time and $4 processing cost and yields 1 oz of Luxury Brute. Each ounce of Regular Chanelle processed further requires an additional 2 hours of laboratory time and $4 processing cost and yields 1 oz of Luxury Chanelle. Each year, Rylon has 6000 hours of laboratory time available and can purchase up to 4000 lb of raw material. Formulate an LP that can be used to determine how Rylon can maximize profits. Assume that the cost of the laboratory hours is a fixed cost." This probelm is from [24] , Example 3.9-11, Page 90.

Modeling Steps:

Listing 7.13: The Complete Model implemented in LPL [22]

```
model winst3_9_11 "Rylon Corporation";
  set  p  := [Brute, Chanele] "perfumes";
       t  := [Regular, Luxury] "types";
  parameter
    RawMatCost      :=  3.00;     -- $/lb
    RawMatYield{p}  :=  [3, 4];   -- oz/lb
    LabTimeRaw      :=  1;        -- Hour/lb
    LabTimeLux{p}   :=  [3, 2];   -- Hour/oz
    Price{p, t}  := [7, 18,  6, 14];  -- $/oz
    ProcessCost{p}  := [4, 4];        -- $/oz
    LabTimeAvail := 6000;   RawMatAvail := 4000;
  variable
    Sales,S{p, t};       -- oz
    RawMatPurchased,R;   -- lb
  expression
    TotalRevenue    : sum{p, t}  Price*S;
    TotalProcessCost: sum{p,t|t='Luxury'} ProcessCost*S;
    TotalRawMatCost : RawMatCost*R;
  constraint
    RawMatLimit: R <= RawMatAvail;
    LabTimeLimit: LabTimeRaw*R
      + sum{p,t|t='Luxury'} LabTimeLux*S <= LabTimeAvail;
```

```
    MaterialBalance{p}: sum{t} S = RawMatYield*R;
  maximize TotalProfit: TotalRevenue - TotalProcessCost -
      TotalRawMatCost;
  Writep(TotalProfit,S,R);
end
```

7.14. Farmer Jones (winstP3-1-1)

—- Run LPL Code , HTML Document –

Problem: "Farmer Jones must detertnme how many acres of corn and wheat to plant this year. An acre of wheat yields bushels of wheat and requires 10 hours of labor per week. An acre of corn yields 10 bushels of com and requires 4 hours of labor per week. All wheat can be sold at $4 a bushel, and all corn can be sold at $3 a bushel. Seven acres of land and 40 hours per week of labor are available. Government regu-lations require that at least 30 bushels of corn be produced during the current year. Let x1 = number of acres of corn planted, and x2 = number of acres of wheat planted. Using these decision variables, formulate an LP whose solution will tell Farmer Jones how to maximize the total revenue from wheat and corn." This probelm is from [24] , Problem 3.1-1, Page 52.

Modeling Steps:

Listing 7.14: The Complete Model implemented in LPL [22]

```
model winstP3_1_1 "Farmer Jones";
  set grain := [Corn, Wheat];
  parameter
    YieldBushel{grain} := [10, 25];    -- bushels/acre
    ReqLabor{grain}    := [4, 10];     -- hours/acres/week
    Price{grain}       := [3, 4];      -- $/bushel
    LandAvail           := 7;           -- acres
    LaborAvail          := 40;          -- hours/week
    MinBushelsCorn      := 30;          -- bushels
  variable PlantAcres {grain};          -- acres
  constraint
    GovReq{grain|grain='Corn'}: YieldBushel*PlantAcres >=
        MinBushelsCorn;
    Acres: sum{grain} PlantAcres <= LandAvail;
    HoursPerWeek: sum{grain} ReqLabor*PlantAcres <=
       LaborAvail;
  maximize Revenue: sum{grain} Price*YieldBushel*
     PlantAcres;
  Writep(Revenue);
end
```

7.15. Farmer Jones (winstP3-1-1s)

—- Run LPL Code , HTML Document –

Problem: "Farmer Jones must detertnme how many acres of corn and wheat to plant this year. An acre of wheat yields bushels of wheat and requires 10 hours of labor per week. An acre of corn yields 10 bushels of com and requires 4 hours of labor per week. All wheat can be sold at $4 a bushel, and all corn can be sold at $3 a bushel. Seven acres of land and 40 hours per week of labor are available. Government regu-lations require that at least 30 bushels of corn be produced during the current year. Let x1 = number of acres of corn planted, and x2 = number of acres of wheat planted. Using these decision variables, formulate an LP whose solution will tell Farmer Jones how to maximize the total revenue from wheat and corn." This probelm is from [24] , Problem 3.1-1, Page 52.

Modeling Steps:

Listing 7.15: The Complete Model implemented in LPL [22]

```
model winstP3_1_1s "Farmer Jones";
  variable Corn; Wheat;   --acres
  constraint
    Acres:         Corn + Wheat <= 7;
    HoursPerWeek:  4*Corn + 10*Wheat <= 40;
    GovReq:        10*Corn >= 30;
  maximize Revenue: 3*10*Corn + 4*25*Wheat;
  Writep(Revenue);
end
```

7.16. Truck Corporation (winstP3-1-4)

—- Run LPL Code , HTML Document –

Problem: "Truckco manufactures two types of trucks: 1 and 2. Each truck must go through the painting shop and assembly shop. If the painting shop were completely devoted to painting type 1 trucks, 800 per day could be painted, whereas if the painting shop were completely devoted to painting type 2 trucks, 700 per day could be painted. If the assembly shop were completely devoted to assembling truck 1 engines, 1500 per day could be assembled, and if the assembly shop were completely devoted to assembling truck 2 engines, 1200 per day could be assembled. Each type 1 truck contributes $300 to profit; each type 2 truck contributes $500. Formulate an LP that will maximize Truckco's profit." This probelm is from [24] , Problem 3.1-4, Page 52.

Modeling Steps:

Listing 7.16: The Complete Model implemented in LPL [22]

```
model winstP3_1_4 "Truck Corporation";
  set
    truck := [1, 2];
    shop  := [Painting, Assembly];
  parameter
    Profit{truck} := [300, 500];
    CapPerDay{shop, truck} := [ 800,  700,   1500, 1200];
  variable Produce{truck};
  constraint ProdCapacity{shop}: sum{truck} 1/CapPerDay*
     Produce <= 1;
  maximize TotalProfit: sum{truck} Profit*Produce ;
  Writep(TotalProfit);
end
```

7.17. Product-Mix (Truck Co) (winstP3-1-4s)

— Run LPL Code , HTML Document –

Problem: "Truckco manufactures two types of trucks: 1 and 2. Each truck must go through the painting shop and assembly shop. If the painting shop were completely devoted to painting type 1 trucks, 800 per day could be painted, whereas if the painting shop were completely devoted to painting type 2 trucks, 700 per day could be painted. If the assembly shop were completely devoted to assembling truck 1 engines, 1500 per day could be assembled, and if the assembly shop were completely devoted to assembling truck 2 engines, 1200 per day could be assembled. Each type 1 truck contributes $300 to profit; each type 2 truck contributes $500. Formulate an LP that will maximize Truckco's profit." This probelm is from [24] , Problem 3.1-4, Page 52.

Modeling Steps:

Listing 7.17: The Complete Model implemented in LPL [22]

```
model winstP3_1_4s "Product-Mix (Truck Co)";
  variable Truck1; Truck2;
  constraint
    Painting:  1/800*Truck1 +  1/700*Truck2 <= 1;
    Body:     1/1500*Truck1 + 1/1200*Truck2 <= 1;
  maximize Profit: 300*Truck1 + 500*Truck2;
  Writep(Profit);
end
```

7.18. Leary Chemical (winstP3-2-3)

—- Run LPL Code , HTML Document –

Problem: "Leary Chemical manufactures three chemicals: A, B, and C. These chemicals are produced via two production processes: l and 2. Running process for an hour costs S4 and yields 3 units of A, 1 of B, and 1 of C. Running process 2 for an hour costs S1 and produces 1 unit of A and 1 of B. To meet customer demands, at least 10 units of A, 5 of B, and 3 of C must be produced daily. Graphically determine a daily production plan that minimizes the cost of meeting Leary Chemical's daily demands." This probelm is from [24] , Problem 3.2-3, Page 60.

Modeling Steps:

Listing 7.18: The Complete Model implemented in LPL [22]

```
model winstP3_2_3 "Leary Chemical";
  set
    process  :=  [P1, P2];
    chemical := [A, B, C];
  parameter
    ProcessCost{process} := [4.00, 1.00];
    ChemYield{chemical, process} :=   --  Units/hour
         [3, 1,  1, 1,  1, 0];
    DailyDemand{chemical} := [10, 5, 3];  --  Units
  variable RunHours{process};      --  hours
  constraint
    MeetDemand{chemical}: sum{process} ChemYield*RunHours
        >= DailyDemand;
  minimize TotalCost: sum{process} ProcessCost*RunHours;
  Writep(TotalCost);
end
```

7.19. Chemical Product Processes (winstP3-2-3s)

— Run LPL Code , HTML Document –

Problem: "Leary Chemical manufactures three chemicals: A, B, and C. These chemicals are produced via two production processes: l and 2. Running process for an hour costs S4 and yields 3 units of A, 1 of B, and 1 of C. Running process 2 for an hour costs S1 and produces 1 unit of A and 1 of B. To meet customer demands, at least 10 units of A, 5 of B, and 3 of C must be produced daily. Graphically determine a daily production plan that minimizes the cost of meeting Leary Chemical's daily demands." This probelm is from [24] , Problem 3.2-3, Page 60.

Modeling Steps:

Listing 7.19: The Complete Model implemented in LPL [22]

```
model winstP3_2_3s "Chemical Product Processes";
  variable Process1; Process2;
  constraint
    ChemicalA:  3*Process1 + Process2 >= 10;
    ChemicalB:    Process1 + Process2 >= 5;
    ChemicalC:    Process1 >= 3;
  minimize Cost: 4*Process1 + 1*Process2;
  Writep(Cost);
end
```

7.20. Furniture Corporation (winstP3-2-5)

—- Run LPL Code , HTML Document –

Problem: "Furnco manufactures desks and chairs. Each desk uses 4 units of wood, and each chair uses 3. A desk contributes $40 to profit, and a chair contributes $25. Marketing restntions require that the number of chairs produced be at least twice the number of desks produced. If 20 units of wood are available, formulate an LP to maximize Furnco's profit. Then graphically solve the LP." This probelm is from [24] , Problem 3.2-5, Page 60.

Modeling Steps:

Listing 7.20: The Complete Model implemented in LPL [22]

```
model winstP3_2_5 "Furniture Corporation";
  set furniture := [Desks, Chairs];
  parameter
    Profit{furniture}  := [40, 25];
    WoodUse{furniture} := [4, 3];
    WoodAvail          := 20;
  variable Produce{furniture};
  constraint
    WoodLimit:  sum{furniture} WoodUse*Produce <=
        WoodAvail;
    Marketing:  2*Produce['Chairs'] >= Produce['Desks'];
  maximize TotalProfit: sum{furniture} Profit*Produce ;
  Writep(TotalProfit);
end
```

7.21. Product-Mix (Furniture Corporation) (winstP3-2-5s)

—- Run LPL Code , HTML Document –

Problem: "Furnco manufactures desks and chairs. Each desk uses 4 units of wood, and each chair uses 3. A desk contributes $40 to profit, and a chair contributes $25. Marketing restntions require that the number of chairs produced be at least twice the number of desks produced. If 20 units of wood are available, formulate an LP to maximize Furnco's profit. Then graphically solve the LP." This probelm is from [24] , Problem 3.2-5, Page 60.

Modeling Steps:

Listing 7.21: The Complete Model implemented in LPL [22]

```
model winstP3_2_5s "Product-Mix (Furniture Corporation)";
  variable Desks; Chairs;
  constraint
   WoodAvail:  4*Desks + 3*Chairs <= 20;
   Marketing:  2*Chairs >= Desks;
  maximize Profit:  40*Desks + 25*Chairs;
  Writep(Profit);
end
```

7.22. Momiss River Pollution (winstP3-4-1)

—- Run LPL Code , HTML Document –

Problem: "There are three factories on the Morriss River (1, 2, and 3). Each emits two types of pollutants and 2) into the river. If the waste from each factory is processed, the pollution in the river can be reduced. It costs S15 to process a ton of factory 1 waste, and each ton processed reduces the amount of pollutant 1 by 0.10 ton and the amount of pollutant 2 by 0.45 ton. It costs $10 to process a ton of factory 2 waste, and each ton processed will reduce the amount of pollutant 1 by 0.20 ton and the amount of pollutant 2 by 0.25 ton. It costs S20 to process a ton of factory 3 waste, and each ton processed will reduce the amount of pollutant 1 by 0.40 ton and the amount of pollutant 2 by 0.30 ton. The state wants to reduce the amount of pollutant 1 in the river by at least 30 tons and the amount of pollutant 2 in the river by at least 40 tons. Formulate an LP that will minimize the cost of reducing pollution by the desired amounts. Do you think that the LP assumptions (Proportionality, Additivity, Divisibility, and Certainty) are reasonable for this problem?" This probelm is from [24] , Problem 3.4-1, Page 69.

Modeling Steps:

Listing 7.22: The Complete Model implemented in LPL [22]

```
model winstP3_4_1 "Momiss River Pollution";
  set
    factory    :=[ 1..3];
    pollutant  :=[ 1..2];
  parameter
    ProcessCost{factory}           := [15.00, 10.00,
        20.00];
    ReduceRate{factory,pollutant}  := [0.10, 0.45,  0.20,
         0.25,  0.40, 0.30];
    MinReduce{pollutant}           := [30, 40];
  variable ProcessWaste,W{factory};
  constraint RequiredReduce{pollutant}: sum{factory}
      ReduceRate*W >= MinReduce;
  minimize TotalCost: sum{factory} ProcessCost*W;
  Writep(TotalCost);
end
```

7.23. US Lab (winstP3-4-2)

— Run LPL Code , HTML Document –

Problem: "U.S. Labs manufactures mechanical heart valves from the heart valves of pigs. Different heart operations require valves of different sizes. U.S. Labs purchases pig valves from three different suppliers. The cost and size mix of the valves purchased from each supplier are given in Table 7.22. Each month, U.S. Labs places one order with each supplier. At least 590Jarge, 300 medium, and 300 small valves must be purchased each month. Because of limited availability of pig valves, at most 500 valves per month can be purchased from each supplier. Formulate an LP that can be used to minimize the cost of acquiring the needed valves." This problem is from [24] , Problem 3.4-2, Page 69.

	Cost Per Value	Percent Large	Percent Medium	Percent Small
Supplier 1	$5	40	40	20
Supplier 2	$4	30	35	35
Supplier 3	$3	20	20	60

Table 7.4:

Modeling Steps:

Listing 7.23: The Complete Model implemented in LPL [22]

```
model winstP3_4_2 "US Lab";
  set  supplier     :=[ 1..3];
       valve        := [Large, Medium, Small];
  parameter CostValve{supplier}  := [5.00, 4.00, 3.00];
    PurchaseMix{supplier, valve} := [0.40, 0.40, 0.20,
                0.30, 0.35, 0.35,   0.20, 0.20, 0.60];
    --MaxPurchase{supplier}  := [500, 500, 500]; --not
        feasibe with this data
    MaxPurchase{supplier}        := [600, 600, 600];
    ReqPurchase{valve}           := [500, 300, 300];
  variable Purchase,P{supplier} [0..MaxPurchase];
  constraint
    RequiredPurchase{valve}: sum{supplier} PurchaseMix*P
        >= ReqPurchase;
  minimize TotalCost: sum{supplier,valve} CostValve*P;
  Writep(TotalCost);
end
```

7.24. Diet Problem (winstP3-4-3)

—- Run LPL Code , HTML Document –

Problem: "Peg and Al Fundy have a limited food budget, so eg is trying to feed the family as cheaply as possible. However, she still wants to make sure her family members meet their daily nutritional requirements. Peg can buy two foods. Food 1 sells for $7 per pound, and each pound contains 3 units of vitamin A and 1 unit of vitamin C. Food 2 sells for $1 per pound, and each pound contains 1 unit of each vitamin. Each day, the family needs at least 12 units of vitamin A and 6 units of vitamin C.

a) Verify that Peg should purchase 12 units of food 2 each day and thus oversatisfy the vitamin C requirement by 6 units.

b) Al has put his foot down and demanded that Peg fulfill the family's daily nutritional requirement exactly by obtaining precisely 12 units of vitamin A and 6 units of vitamin C. The optimal solution to the new problem will involve ingesting less vitamin C, but it will be more expensive. Why?"
This problem is from [24] , Problem 3.4-3, Page 69.

Modeling Steps:

Listing 7.24: The Complete Model implemented in LPL [22]

```
model winstP3_4_3 "Diet Problem";
  set
    f  :=[  1..2]  "Food";
    v  :=  [A, C] "Vitamin";
  parameter
    FoodCost{f}           :=  [7, 1];
    AmountVitamin{f, v}   :=  [3, 1,   1, 1];
    MinDailyReq{v}        :=  [12, 6];
  variable Purchase{f};
  constraint
   Requirement{v}: sum{f} AmountVitamin*Purchase >=
       MinDailyReq;
  minimize TotalCost: sum{f} FoodCost*Purchase;
  Writep(TotalCost);
end
```

7.25. Gold Mining (winstP3-4-4)

— Run LPL Code , HTML Document –

Problem: "Goldilocks needs to find at least 12 lb of gold and at least 18 lb of silver to pay the monthly rent. There are two mines in which Goldilocks can find gold and silver. Each day that Goldilocks spends in mine 1, she finds 2 lb of gold and 2 lb of silver. Each day that Goldilocks spends in mine 2, she finds 1 lb of gold and 3 lb of silver. Formulate an LP to help Goldilocks meet her requirements while spending as little time as possible in the mines. Graphically solve the LP." This probelm is from [24] , Problem 3.4-4, Page 69.

Modeling Steps:

Listing 7.25: The Complete Model implemented in LPL [22]

```
model winstP3_4_4 "Gold Mining";
  set
    mine    :=[ 1..2];
    metals  := [gold, silver];
  parameter
    Discover{mine, metals}  := [2, 2,  1, 3];
    MetalReq{metals}        := [12, 18];
  variable DaysSpent{mine};
  constraint Requirement{metals}: sum{mine} Discover*
      DaysSpent >= MetalReq;
  minimize  TotalDays: sum{mine} DaysSpent;
  Writep(TotalDays);
end
```

7.26. Capital Budget (winstP3-6-2)

—- Run LPL Code , HTML Document –

Problem: "Two investments with varying cash flows (in thousands of dollars) are available, as shown in Table 7.22. At time 0, $10,000 is available for investment, and at time 1, $7000 is available. Assuming that r = 0.10, set up an LP whose so-lution maximizes the NPV obtained from these investments. Graphically find the optimal solution to the LP. (Assume that any fraction of an investment may be purchased)." This probelm is from [24] , Problem 3.6-2, Page 76.

	Cash Flow (in thousands) at Time			
	0	1	2	3
Investment 1	-$6	-$5	$7	$9
Investment 2	-$8	-$3	$9	$7

Table 7.5:

Modeling Steps:

Listing 7.26: The Complete Model implemented in LPL [22]

```
model winstP3_6_2 "Capital Budget";
  set
    time   := [0..3];
    invest := [1..2];
  parameter
    CashAvail{time}       := [10, 7, 0, 0];
    CashFlow{invest,time} := [-6 -5 7 9 -8 -3 9 7];
    InterestRate   := 0.1;
    RateMult       := 1.0 + InterestRate;
    NPV{invest}    := sum{time} CashFlow/RateMult^time;
  variable MakeInvest{invest} [0..1];
  constraint InvestLimit{time}: sum{invest} - CashFlow*
     MakeInvest <= CashAvail;
  maximize TotalProfit: sum{invest} NPV*MakeInvest ;
  Writep(TotalProfit,MakeInvest);
end
```

7.27. Blending Candy (winstP3-8-1)

—- Run LPL Code , HTML Document –

Problem: "You have decided to enter the candy business. You are considering producing two types of candies: Slugger Candy and Easy Out Candy, both of which consist solely of sugar, nuts, and chocolate. At present, you have in stock 100 oz of sugar, 20 oz of nuts, and 30 oz of chocolate. The mixture used to make Easy Out Candy must contain at least 20mixture used to make Slugger Candy must contain at least 10can be sold for 250, and each ounce of Slugger Candy for 200. Formulate an LP that will enable you to maximize your revenue from candy sales." This probelm is from [24] , Problem 3.8-1, Page 87.

Modeling Steps:

Listing 7.27: The Complete Model implemented in LPL [22]

```
model winstP3_8_1 "Blending Candy";
  set
   candy      := [Slugger, EasyOut];
   ingred     := [Sugar, Nuts, Chocolate];
  parameter
   Price{candy}           := [0.20, 0.25];
   AvailIngred{ingred}    := [100, 20, 30];
   MinPercent{candy,ingred}:=[0, 0.1, 0.1, 0, 0.2, 0];
  variable SweetMix{candy,ingred};
  constraint
    MinIngred{candy,ingred|MinPercent>0}: SweetMix >=
        MinPercent*(sum{ingred} SweetMix);
    MaxIngread{ingred}: sum{candy} SweetMix <=
        AvailIngred;
  maximize TotalProfit: sum{candy,ingred} Price*SweetMix;
  Writep(TotalProfit);
end
```

7.28. Blending Oranges (winstP3-8-2)

—- Run LPL Code , HTML Document –

Problem: "O.J. Juice Company sells bags of oranges and cartons of orange juice. O.J. grades oranges on a scale of 1 (poor) to 10 (excellent). At present, Q.J. has on hand 100,000 lb of grade 9 oranges and 120,000 lb of grade 6 oranges. The average quality of oranges sold in bags must be at least 7, and the average quality of the oranges used to produce orange juice must be at least 8. Each pound of oranges that is used for juice yields a revenue of $1.50 and incurs a variable cost (consisting of labor costs, variable overhead costs, inventory costs, and so on) of $1.05. Each pound of oranges sold in bags yields a revenue of 50 cents and incurs a variable cost of 200. Formulate an LP to help O.J. maximize profit." This probelm is from [24] , Problem 3.8-2, Page 87.

Modeling Steps:

Listing 7.28: The Complete Model implemented in LPL [22]

```
model winstP3_8_2 "Blending Oranges";
  set
   grade   := [g6, g9];
   product := [Juice, Bags];
  parameter
   GradeValue{grade}      := [6, 9];
   OrangesOnHand{grade}   := [100000, 120000];
   MinAvgQuality{product} := [8, 7];
   Revenue{product}       := [1.50, 0.50];
   VarCost{product}       := [1.05, 0.20];
   Profit{product}        := Revenue - VarCost;
  variable Produce{grade,product};
  constraint
    LimitOranges{grade}: sum{product} Produce <=
        OrangesOnHand;
    ReqQuality{product}: sum{grade} GradeValue*Produce >=
         MinAvgQuality*(sum{grade} Produce);
  maximize TotalProfit: sum{grade,product} Profit*Produce
      ;
  Writep(TotalProfit);
end
```

7.29. Blending Portfolio (winstP3-8-3)

—- Run LPL Code , HTML Document –

Problem: "A bank is attempting to determine where its assets should be invested during the current year. At present, $500,000 is available for investment in bonds, home loans, auto loans, and personal loans. The annual rate of return on each type of investment is known to be: bonds, 10auto loans, 13bank's portfolio is not too risky, the bank's investment man- ager has placed the following three restrictions on the bank's portfolio:

a) The amount invested in personal loans cannot ex- ceed the amount invested in bonds.

b) The amount invested in home loans cannot exceed the amount invested in auto loans.

c) No more than 25be in personal loans.

The bank's objective is to maximize the annual return on its investment portfolio. Formulate an LP that will enable the bank to meet this goal." This problem is from [24] , Problem 3.8-3, Page 87.

Modeling Steps:

Listing 7.29: The Complete Model implemented in LPL [22]

```
model winstP3_8_3 "Blending Portfolio";
  set portfolio := [Bonds, Home, Auto, Personal];
  parameter
    CashAvail              := 500000;
    ReturnRate{portfolio}  := [0.10, 0.16, 0.13, 0.20];
    PercentPersonal        := 0.25;
  variable Invest{portfolio};
  constraint
    MaxInvestment: sum{portfolio} Invest <= CashAvail;
    PersonalLomit: Invest['Personal'] <= Invest['Bonds'];
    HomeLimit: Invest['Home'] <= Invest['Auto'];
    MaxPersonal: Invest['Personal'] <= PercentPersonal*(
        sum{portfolio} Invest);
  maximize TotalProfit: sum{portfolio} ReturnRate*Invest;
  Writep(TotalProfit);
end
```

7.30. Blending Investments (winstP3-8-4)

—- Run LPL Code , HTML Document –

Problem: "Young MBA Erica Cudahy may invest up to $1000. She can invest her money in stocks and loans. Each dollar invested in stocks yields 10 cents profit, and each dollar invested in a loan yields 15 cents profit. At least 30% of all money invested must be in stocks, and at least $400 must be in loans. Formulate any that can be used to maximize total profit earned from Erica's investment. Then graphically solve the LP." This probelm is from [24] , Problem 3.8-4, Page 87.

Modeling Steps:

Listing 7.30: The Complete Model implemented in LPL [22]

```
model winstP3_8_4 "Blending Investments";
  set invest  := [Stocks, Loans];
  parameter
    MaxAmountAvail    := 1000;
    MinStockPercent   := 0.30;
    MinLoanAmount     := 400.00;
    ProfitPerDollar{invest} := [0.10, 0.15];
  variable Amount{invest};
  constraint
   TotalInvestment: sum{invest} Amount<=MaxAmountAvail;
   MinStocks: Amount['Stocks'] >= MinStockPercent*(sum{
       invest} Amount);
   MinLoans: Amount['Loans'] >= MinLoanAmount;
  maximize TotalProfit: sum{invest} ProfitPerDollar*
      Amount;
  Writep(TotalProfit);
end
```

7.31. Blending Oils (winstP3-8-5)

—- Run LPL Code , HTML Document –

Problem: "Chandler Oil Company has 5000 barrels of oil 1 and 3.8 Blending Problems 87 10,000 barrels of oil 2. The company sells two products: gasoline and heating oil. Both products are produced by combining oil 1 and oil 2. The quality level of each oil is as follows: oil 1, 10; oil 2, 5. Gasoline must have an average quality level of at least 8 and heating oil, at least 6. Demand for each product must be created by advertising. Each dollar spent advertising gasoline creates 5 barrels of demand and each spent on heating oil creates 10 barrels of demand. Gaso- line is sold for $25 per barrel, heating oil for $20. Formulate an LP to help Chandler maximize profit. Assume that no oil of either type can be purchased." This probelm is from [24] , Problem 3.8-5, Page 87.

Modeling Steps:

Listing 7.31: The Complete Model implemented in LPL [22]

```
model winstP3_8_5 "Blending Oils";
  set
    oiltype,o  :=  [Oil1, Oil2];
    product,p  :=  [Gasoline, HeatingOil];
  parameter
    BarrelsAvail{o} := [5000, 10000];
    QualityLevel{o} := [10, 5];
    QualityReq{p}   := [8, 6];
    Price{p}        := [25.00, 20.00];
    DemandPerAdDollar{p}  := [5, 10];
  variable
    Blend,B{p,o};
    AdDollars,A{p};
  constraint
   MinAvgQuality{p}: sum{o} QualityLevel*B >= QualityReq
       *(sum{o} B);
   MaxOilAvail{o}: sum{p} B <= BarrelsAvail;
   AdSpending{p}: sum{o} B = DemandPerAdDollar*A;
  maximize TotalProfit: sum{p,o} Price*B - sum{p} A;
  Writep(TotalProfit);
end
```

7.32. Blending Oils (winstP3-8-5b)

—- Run LPL Code , HTML Document –

Problem: "Chandler Oil Company has 5000 barrels of oil 1 and 3.8 Blending Problems 87 10,000 barrels of oil 2. The company sells two products: gasoline and heating oil. Both products are produced by combining oil 1 and oil 2. The quality level of each oil is as follows: oil 1, 10; oil 2, 5. Gasoline must have an average quality level of at least 8 and heating oil, at least 6. Demand for each product must be created by advertising. Each dollar spent advertising gasoline creates 5 barrels of demand and each spent on heating oil creates 10 barrels of demand. Gaso- line is sold for $25 per barrel, heating oil for $20. Formulate an LP to help Chandler maximize profit. Assume that no oil of either type can be purchased." This probelm is from [24] , Problem 3.8-5, Page 87.

Modeling Steps:

Listing 7.32: The Complete Model implemented in LPL [22]

```
model winstP3_8_5b "Blending Oils";
  set
    oiltype,o  :=  [Oil1, Oil2];
    product,p  :=  [Gasoline, HeatingOil];
  parameter
    BarrelsAvail{o} := [5000, 10000];
    QualityLevel{o} := [10, 5];
    QualityReq{p}   := [8, 6];
    Price{p}        := [25.00, 20.00];
    DemandPerAdDollar{p}  := [5, 10];
  variable
    Blend,B{p,o};
    Produce,P{p};
    AdDollars,A{p};
  constraint
    BlendOil{p}: P = sum{o} B;
    MinAvgQuality{p}: sum{o} QualityLevel*B >= QualityReq
        *P;
    MaxOilAvail{o}: sum{p} B <= BarrelsAvail;
    AdSpending{p}: P = DemandPerAdDollar*A;
  maximize TotalProfit: sum{p} Price*P - sum{p} A;
  Writep(TotalProfit);
end
```

7.33. Blending fertilizer (winstP3-8-6)

—- Run LPL Code , HTML Document –

Problem: "Bullco blends silicon and nitrogen to produce two types of fertilizers. Fertilizer 1 must be at least 40sells for \$70/lb. Fertilizer 2 must be at least 70sells for \$40/lb. Bullco can purchase up to 80 lb of nitrogen at \$15/1b and up to 100 lb of silicon at \$10/1b. Assuming that all fertilizer produced can be sold, formulate an LP to help Bullco maximize profits. " This probelm is from [24] , Problem 3.8-6, Page 87.

Modeling Steps:

Listing 7.33: The Complete Model implemented in LPL [22]

```
model winstP3_8_6 "Blending fertilizer";
  set
    fertilizer  := [f1, f2];
    component   := [Silicon, Nitrogen];
  parameter
    Profit{fertilizer}              := [70.00, 40.00];
    Percent{fertilizer, component} := [0, 0.40, 0.70, 0];
    Cost{component}                 := [15.00, 10.00];
    MaxPurchase{component}          := [80, 100];
  variable Blend{fertilizer,component};
  constraint
    Require{fertilizer,component|Percent>0}: Blend >=
        Percent*(sum{component} Blend);
    MaxComponent{component}: sum{fertilizer} Blend <=
        MaxPurchase;
  maximize TotalProfit: sum{fertilizer,component} (Profit
      *Blend - Cost*Blend);
  Writep(TotalProfit,Blend);
end
```

7.34. Blending chemicals (winstP3-8-7)

—- Run LPL Code , HTML Document –

Problem: "Eli Daisy uses chemicals 1 and 2 to produce two drugs. Drug 1 must be at least 70at least 60at $6 per oz; up to 30 oz of drug 2 can be sold at $5 per oz. Up to 45 oz of chemical 1 can be purchased at $6 per oz, and up to 40 oz of chemical 2 can be purchased at $4 per oz. Formulate an LP that can be used to maximize Daisy's profits. " This probelm is from [24] , Problem 3.8-7, Page 87.

Modeling Steps:

Listing 7.34: The Complete Model implemented in LPL [22]

```
model winstP3_8_7 "Blending chemicals";
  set
    chemical,c  :=  [c1, c2];
    drug    ,d  :=  [d1, d2];
  parameter
    MinChemPercent{d,c} := [0.70,  0   0, 0.60];
    MaxDemand{d}        := [40, 30];
    Price{d}            := [6.00, 5.00];
    MaxChemAvail{c} := [45, 40];
    Cost{c}         := [6.00, 4.00];
  variable MixChem{c,d};
  constraint
    Requirement{d,c|MinChemPercent>0}: MixChem >=
        MinChemPercent*(sum{c} MixChem);
    LimitDemand{d}: sum{c} MixChem <= MaxDemand;
    LimitChemical{c}: sum{d} MixChem <= MaxChemAvail;
  maximize TotalProfit: sum{d} Price*(sum{c} MixChem) -
      sum{c} Cost*(sum{d} MixChem);
  Writep(TotalProfit);
end
```

7.35. Highland TV and Radio (winstP3-8-8)

— Run LPL Code , HTML Document –

Problem: "Highland's TV-Radio Store must determine how many TVs and radios to keep in stock. A TV requires 10 sq ft of floorspace, whereas a radio requires 4 sq ft; 200 sq ft of floorspace is available. A TV will earn Highland $60 in profits, and a radio will earn $20. The store stocks only TVs and radios. Marketing requirements dictate that at least 60of all appliances in stock be radios. Finally, a TV ties up $200 in capital, and a radio, $50. Highland wants to have at most $3000 worth of capital tied up at any time. Formulate an LP that can be used to maximize Highland's profit." This probelm is from [24] , Problem 3.8-8, Page 87.

Modeling Steps:

Listing 7.35: The Complete Model implemented in LPL [22]

```
model winstP3_8_8 "Highland TV and Radio";
  set p :=  [TV, Radio] "products";
  parameter
    ReqSpace{p}     := [10, 4];
    SpaceAvail      := 200;
    Profit{p}       := [60.00, 20.00];
    MinStockRadioPct  := 0.60;
    CapitalTied{p} := [200.00, 50.00];
    CapitalAvail    := 3000;
  variable InStock{p};
  constraint
    LimitSpace: sum{p} ReqSpace*InStock  <= SpaceAvail;
    MinimumStock{p|p='Radio'}: InStock >=
        MinStockRadioPct*(sum{p} InStock);
    LimitCapital: sum{p} CapitalTied*InStock <=
        CapitalAvail;
  maximize TotalProfit: sum{p} Profit*InStock ;
  Writep(TotalProfit);
end
```

7.36. Production Process (Sunco Oil) (winstP3-9-1)

—- Run LPL Code , HTML Document –

Problem: "Sunco Oil has three different processes that can be used to manufacture various types of gasoline. Each process in-volves blending oils in the company's catalytic cracker. Run-ning process 1 for an hour costs $5 and requires 2 barrels of crude oil 1 and 3 barrels of crude oil 2. The output from running process 1 for an hour is 2 barrels of gas 1 and 1 barrel of gas 2. Running process 2 for an hour costs $4 and requires 1 barrel of crude 1 and 3 barrels of crude 2. The output from running process 2 for an hour is 3 barrels of gas 2. Running process 3 for all hour costs $1 and requires 2 barrels of crude 2 and 3 barrels of gas 2. The output from running process 3 for an hour is 2 barrels of gas 3. Each week , 200 barrels of crude 1, at $2/barrel, and 300 barrels of crude 2, at $3/bar-rel, may be purchased. All gas produced can be sold at the following per-barrel prices: gas 1, $9; gas 2, $10; gas 3, $24. Formulate an LP whose solution will maximize revenues less costs. Assume that only 100 hours of time on the catalytic cracker are available each week." This probelm is from [24] , Problem 3.9-1, Page 93.

Modeling Steps:

Listing 7.36: The Complete Model implemented in LPL [22]

```
model winstP3_9_1 "Production Process (Sunco Oil)";
  set  process := [1..3];
       crude   := [1..2];   gas := [1..3];
  parameter RunCost{process}  := [5, 4, 1];
    CrudeReq{process,crude}:= [2, 3,  1, 3,  0, 2];
    GasReq{process, gas}   := [0 0 0 0 0 0 3 0];
    OutputGas{process,gas} := [2 1 0 0 3 0 0 0 2];
    MaxPurchase{crude}     := [200, 300];
    CrudeCost{crude}       := [2.00, 3.00];
    GasPrice{gas}          := [9.00, 10.00, 24.00];
    MaxCrackerHours        := 100;
  variable
    SalesGas{gas};  RunProcess{process};
    PurchaseCrude{crude};
  constraint
    CrackerLimit: sum{process} RunProcess <=
        MaxCrackerHours;
    CrudeBalance{crude}: PurchaseCrude = sum{process}
        CrudeReq*RunProcess;
    PurchaseLimit{crude}: PurchaseCrude <= MaxPurchase;
    GasBalance{gas}: sum{process} OutputGas*RunProcess
      = sum{process} GasReq*RunProcess + SalesGas;
  maximize TotalProfit: sum{gas} GasPrice*SalesGas
```

```
    - sum{process} RunCost*RunProcess
    - sum{crude} CrudeCost*PurchaseCrude;
  Writep(TotalProfit);
end
```

7.37. Production Process (winstP3-9-2)

—- Run LPL Code , HTML Document –

Problem: "Furnco manufactures tables and chairs. A table requires 40 board ft of wood, and a chair requires 30 board ft of wood. Wood may be purchased at a cost of $1 per board ft, and 40,000 board ft of wood are available for purchase. It takes 2 hours of skilled labor to manufacture an unfinished table or an unfinished chair. Three more hours of skilled labor will turn an unfinished table into a finished table, and 2 more hours of skilled labor will turn an unfinished chair into a finished chair. A total of 6000 hours of skilled labor are available (and have already been paid for). All furniture produced can be sold at the following unit prices: unfinished table, $70; finished table, $140; unfinished chair, $60; finished chair, $110. Formulate an LP that will maximize the contribution to profit from manufacturing tables and chairs. " This probelm is from [24] , Problem 3.9-2, Page 93.

Modeling Steps:

Listing 7.37: The Complete Model implemented in LPL [22]

```
model winstP3_9_2 "Production Process";
  set
    f := [Desks, Chairs] "furniture";
    s    := [Unfinished, Finished] "status";
  parameter
    WoodUse{f} := [40, 30];
    WoodCost   := 1;
    WoodAvail  := 40000;
    LaborNeeded{f,s} := [2, 5,  2, 4];
    HoursAvail       := 6000;
    Price{f,s} := [70, 140,  60, 110];
  variable
    Produce{f,s};
    WoodPurchase [0..WoodAvail];
  constraint
    WoodLimit: sum{f, s} WoodUse*Produce <= WoodPurchase;
    LaborLimit: sum{f, s} LaborNeeded*Produce <=
        HoursAvail;
  maximize TotalProfit: sum{f,s} Price*Produce
                      - WoodCost*WoodPurchase;
  Writep(TotalProfit);
end
```

7.38. Production Process (Rylon Corp.) (winstP3-9-3)

— Run LPL Code , HTML Document –

Problem: "Rylon Corporation manufactures Brute and Chanelle perfumes. The raw material needed to manufacture each type of perfume can be purchased for $3 per pound. Processing 1 lb of raw material requires 1 hour of laboratory time. Each pound of processed raw material yields 3 oz of Regular Brute Perfume and 4 oz of Regular Chanelle Perfume. Regular Brute can be sold for $7/oz and Regular Chanelle for $6/oz. Rylon also has the option of further processing Regular Brute and Regular Chanelle to produce Luxury Brute, sold at $18/oz, and Luxury Chanelle, sold at $14/oz. Each ounce of Regular Brute processed further requires an additional 3 hours of laboratory time and $4 processing cost and yields 1 oz of Luxury Brute. Each ounce of Regular Chanelle processed further requires an additional 2 hours of laboratory time and $4 processing cost and yields 1 oz of Luxury Chanelle. Each year, Rylon has 6000 hours of laboratory time available and can purchase up to 4000 lb of raw material. Formulate an LP that can be used to determine how Rylon can maximize profits. Assume that the cost of the laboratory hours is a fixed cost.

Suppose that in Example (3.9-11), 1 lb of raw material could be used to produce either 3 oz of Brute or 4 oz of Chanelle. How would this change the formulation? " This probelm is from [24] , Problem 3.9-3, Page 93.

Modeling Steps:

Listing 7.38: The Complete Model implemented in LPL [22]

```
model winstP3_9_3 "Production Process (Rylon Corp.)";
  set
    p  := [Brute, Chanele] "perfumes";
    t     := [Regular, Luxury] "types";
  parameter
    RawMatCost          :=  3.00;              --$/lb
    RawMatYield{p}      :=  [3, 4];            --oz/lb
    LabTimeRaw          :=  1;                 --Hours/lb
    LabTimeLux{p}       :=  [3, 2];            --Hours/oz
    Price{p, t}         := [7, 18,
                            6, 14];            --$/oz
    ProcessCost{p}      := [4, 4];             --$/oz
    LabTimeAvail        := 6000;
    RawMatAvail         := 4000;
  variable
    Sales{p, t};                               --oz
    RawMatPurchased{p};                        --lb
```

```
  expression
    TotalRevenue    : sum{p,t}  Price*Sales;
    TotalProcessCost: sum{p,t|t='Luxury'} ProcessCost*
        Sales;
    TotalRawMatCost: RawMatCost*(sum{p} RawMatPurchased);
  constraint
    RawMatLimit: sum{p} RawMatPurchased <= RawMatAvail;
    LabTimeLimit: LabTimeRaw*(sum{p} RawMatPurchased)
      + sum{p,t|t='Luxury'} LabTimeLux*Sales <=
          LabTimeAvail;
    MaterialBalance{p}: sum{t} Sales = RawMatYield*
        RawMatPurchased;
  maximize TotalProfit: TotalRevenue - TotalProcessCost -
      TotalRawMatCost;
  Writep(TotalProfit);
end
```

7.39. Production Process (winstP3-9-5)

— Run LPL Code , HTML Document –

Problem: "A company produces A, B, and C and can sell these products in unlimited quantities at the following unit prices: A, $10; B, $56; C, $100. Producing a unit of A requires 1 hour of labor; a unit of B, 2 hours of labor plus 2 units of A; and a unit of C, 3 hours of labor plus 1 unit of B. Any A that is used to produce B cannot be sold. Similarly, any B that is used to produce C cannot be sold. A total of 40 hours of labor are available. Formulate an LP to maximize the company's revenues. " This probelm is from [24] , Problem 3.9-5, Page 93.

Modeling Steps:

Listing 7.39: The Complete Model implemented in LPL [22]

```
model winstP3_9_5 "Production Process";
  set product,p,p1 := [A, B, C];
  parameter
    Price{p}     := [10, 56, 100];
    ProdHours{p} := [1, 2, 3];
    BillOfMat{p,p1}  := /A B 2,  B C 1/;
    HoursAvail       := 40;
  variable
    Produce{p};
    Sales{p};
  constraint
    LimitProduction: sum{p} ProdHours*Produce <=
        HoursAvail;
    ProdBalance{p1}: Produce = sum{p} BillOfMat*Produce +
         Sales;
  maximize Revenues: sum{p} Price*Sales ;
  Writep(Revenues);
end
```

7.40. Production Process (Daisy Drug) (winstP3-9-6)

—- Run LPL Code , HTML Document –

Problem: "Daisy Drugs manufactures two drugs: 1 and 2. The dru2s are produced by blending two chemicals: 1 and 2. By weigh-t, drug 1 must contain at least 65contain at least 55drug 2 sells for $4/oz. Chemicals 1 and 2 can be produced by one of two production processes. Running process 1 for an hour requires 3 oz of raw material and 2 hours skilled labor and yields 3 oz of each chemical. Running process 2 for an hour requires 2 oz of raw material and 3 hours of skilled labor and yields 3 oz of chemical 1 and 1 oz of chemical 2. A total of 120 hours of skilled labor and 100 oz of raw material are available. Formulate an LP that can be used to maximize Daisy's sales revenues. " This probelm is from [24] , Problem 3.9-6, Page 93.

Modeling Steps:

Listing 7.40: The Complete Model implemented in LPL [22]

```
model winstP3_9_6 "Production Process (Daisy Drug)";
  set
    drug      :=[ 1..2];
    chemical  :=[ 1..2];
    process   :=[ 1..2];
  parameter
    PercentReq{drug, chemical} := [0.65,  0,    0, 0.55];
    Price{drug}              := [6, 4];
    LaborUse{process}        := [2, 3];
    RawMatUse{process}       := [3, 2];
    ChemicalYield{process, chemical} := [3, 3,   3, 1];
    LaborAvail           := 120;
    RawMatAvail          := 100;
  variable
    RunProcess,R{process};
    Blend,B{chemical,drug};
  constraint
    LaborLimit: sum{process} LaborUse*R <= LaborAvail;
    RawMatLimit: sum{process} RawMatUse*R <= RawMatAvail;
    ChemBalance{chemical}: sum{process} ChemicalYield*R
        >= sum{drug} B;
    RequireChem{drug,chemical|PercentReq>0}: B >=
        PercentReq*(sum{chemical} B);
  maximize TotalRevenue: sum{chemical,drug}  Price*B;
  Writep(TotalRevenue);
end
```

7.41. Production Process (Lizzies Dairy) (winstP3-9-7)

—- Run LPL Code , HTML Document –

Problem: " Lizzie's Dairy produces cream cheese and cottage cheese. Milk and cream are blended to produce these two products. Both high-fat and low-fat milk can be used to pro- duce cream cheese and cottage cheese. High-fat milk is 60% fat; low-fit milk is 30% fat. The milk used to produce cream cheese must average at least 50% fat and that for cottage cheese, at least 35% fat. At least 40% (by weight) of the inputs to cream cheese and at least 20% (by weight) of the inputs to cottage cheese must be cream. Both cottage cheese and cream cheese are produced by putting milk and cream through the cheese machine. It costs 40c to process 1 lb of inputs into a pound of cream cheese. It costs 40 cents to produce 1 lb of cottage cheese, but every pound of input for cottage cheese yields 0.9 lb of cottage cheese and 0.1 lb of waste. Cream can be produced by evaporating high-tat and low-fatmilk. It costs 40 cents to evaporate I lb of high-fat milk. Each, pound of high-fat milk that is evaporated yields 0.6 lb of cream. It costs 40 cents to evaporate 1 lb of low-fat milk. Each pound of low-fat milk that is evaporated yields 0.3 lb of cream. Each day, up to 3000 lb of input may be sent through the cheese machine. Each day, at least 1000 lb of cottage cheese and 1000 lb of cream cheese must be produced. Up to 1500 lb of cream cheese and 2000 lb of cottage cheese can be sold each day. Cottage cheese is sold for $1.20/1b and cream cheese for $1.50/1b. High-fat milk is purchased for 80 cents/lb and low-fat milk for 40 cents/lb. The evaporator can process at most 2000 lb of milk daily. Formulate an LP that can he used to maximize Lizzie 's daily profit." This probelm is from [24] , Problem 3.9-7, Page 93.

Modeling Steps:

Listing 7.41: The Complete Model implemented in LPL [22]

```
model winstP3_9_7 "Production Process (Lizzies Dairy)";
  set
    cheese   :=  [CreamCheese, CottageCheese];
    milk     :=  [HiFatMilk, LowFatMilk];
  parameter
    FatContent{milk}       :=  [0.60, 0.30];
    MinFatPct{cheese}      :=  [0.50, 0.35];
    MinCreamPct{cheese}    :=  [0.40, 0.20];
    EvapYield{milk}        :=  [0.6, 0.3];
    EvapCost{milk}         :=  [0.40, 0.40];
    EvapCapacity           :=  2000;
    ProcessYield{cheese}   :=  [1.0, 0.9];
    ProcessCost{cheese}    :=  [0.40, 0.40];
```

```
    ProcessCapacity       :=  3000;
    MinProduce{cheese}    :=  [1000, 1000];
    MaxProduce{cheese}    :=  [1500, 2000];
    MilkCost{milk}        :=  [0.80, 0.40];
    CheesePrice{cheese}   :=  [1.50, 1.20];
  variable
    ProduceCheese,P{cheese}  [MinProduce..MaxProduce];
    PurchaseMilk,Bm{milk};
    EvaporateMilk,Em{milk};
    ProcessMilk,Pm{milk,cheese};
    ProcessCream,Pc{cheese};
  expression
    TotalRevenue    : sum{cheese} CheesePrice*P;
    TotalMilkCost   : sum{milk}   MilkCost*Bm;
    TotalEvapCost   : sum{milk}   EvapCost*Em;
    TotalProcessCost: sum{cheese} ProcessCost*P;
  constraint
    PurchaseMilkBal{milk}: Bm = Em + sum{cheese} Pm;
    EvapBalance: sum{milk} EvapYield*Em = sum{cheese} Pc;
    EvaporateLimit: sum{milk} Em <= EvapCapacity;
    CreamRequirements{cheese}: Pc >= MinCreamPct*(sum{
        milk} Pm + Pc);
    FatRequirements{cheese}: sum{milk} FatContent*Pm >=
        MinFatPct*(sum{milk} Pm);
    ProcessBalance{cheese}: P = ProcessYield*(sum{milk}
        Pm + Pc);
    ProcessLimit: sum{cheese} (sum{milk} Pm  + Pc) <=
        ProcessCapacity;
  maximize TotalProfit: TotalRevenue-TotalMilkCost-
      TotalEvapCost-TotalProcessCost;
  Writep(TotalProfit,P,Bm,Em,Pm,Pc);
end
```

7.42. Production Process (Lizzies Dairy) (winstP3-9-7b)

—- Run LPL Code , HTML Document –

Problem: " Lizzie's Dairy produces cream cheese and cottage cheese. Milk and cream are blended to produce these two products. Both high-fat and low-fat milk can be used to pro- duce cream cheese and cottage cheese. High-fat milk is 60% fat; low-fit milk is 30% fat. The milk used to produce cream cheese must average at least 50% fat and that for cottage cheese, at least 35% fat. At least 40% (by weight) of the inputs to cream cheese and at least 20% (by weight) of the inputs to cottage cheese must be cream. Both cottage cheese and cream cheese are produced by putting milk and cream through the cheese machine. It costs 40c to process 1 lb of inputs into a pound of cream cheese. It costs 40 cents to produce 1 lb of cottage cheese, but every pound of input for cottage cheese yields 0.9 lb of cottage cheese and 0.1 lb of waste. Cream can be produced by evaporating high-tat and low-fatmilk. It costs 40 cents to evaporate I lb of high-fat milk. Each, pound of high-fat milk that is evaporated yields 0.6 lb of cream. It costs 40 cents to evaporate 1 lb of low-fat milk. Each pound of low-fat milk that is evaporated yields 0.3 lb of cream. Each day, up to 3000 lb of input may be sent through the cheese machine. Each day, at least 1000 lb of cottage cheese and 1000 lb of cream cheese must be produced. Up to 1500 lb of cream cheese and 2000 lb of cottage cheese can be sold each day. Cottage cheese is sold for $1.20/1b and cream cheese for $1.50/1b. High-fat milk is purchased for 80 cents/lb and low-fat milk for 40 cents/lb. The evaporator can process at most 2000 lb of milk daily. Formulate an LP that can he used to maximize Lizzie 's daily profit." This probelm is from [24] , Problem 3.9-7, Page 93.

Modeling Steps:

Listing 7.42: The Complete Model implemented in LPL [22]

```
model winstP3_9_7b "Production Process (Lizzies Dairy)";
  set
    cheese  :=  [CreamCheese, CottageCheese];
    milk    :=  [HiFatMilk, LowFatMilk];
  parameter
    FatContent{milk}       :=  [0.60, 0.30];
    MinFatPct{cheese}      :=  [0.50, 0.35];
    MinCreamPct{cheese}    :=  [0.40, 0.20];
    EvapYield{milk}        :=  [0.6, 0.3];
    EvapCost{milk}         :=  [0.40, 0.40];
    EvapCapacity           :=  2000;
    ProcessYield{cheese}   :=  [1.0, 0.9];
    ProcessCost{cheese}    :=  [0.40, 0.40];
```

```
    ProcessCapacity       :=  3000;
    MinProduce{cheese}    :=  [1000, 1000];
    MaxProduce{cheese}    :=  [1500, 2000];
    MilkCost{milk}        :=  [0.80, 0.40];
    CheesePrice{cheese}   :=  [1.50, 1.20];
  variable
    ProduceCheese,P{cheese} [MinProduce..MaxProduce];
    PurchaseMilk,Bm{milk};
    EvaporateMilk,Em{milk};
    ProcessMilk,Pm{milk,cheese};
    ProcessCream,Pc{cheese};
    ProcessInput,Pi{cheese};
  expression
    TotalRevenue    : sum{cheese} CheesePrice*P;
    TotalMilkCost   : sum{milk} MilkCost*Bm;
    TotalEvapCost   : sum{milk} EvapCost*Em;
    TotalProcessCost: sum{cheese} ProcessCost*P;
  constraint
    PurchaseMilkBal{milk}: Bm = Em + sum{cheese} Pm;
    EvapBalance: sum{milk} EvapYield*Em = sum{cheese} Pc;
    EvaporateLimit: sum{milk} Em <= EvapCapacity;
    CreamRequirements{cheese}: Pc >= MinCreamPct*Pi;
    FatRequirements{cheese}: sum{milk} FatContent*Pm >=
        MinFatPct*(sum{milk} Pm);
    ProcessBalance{cheese}: Pi = sum{milk} Pm + Pc;
    OutputCheese{cheese}: P = ProcessYield*Pi;
    ProcessLimit: sum{cheese} Pi <= ProcessCapacity;
  maximize TotalProfit: TotalRevenue-TotalMilkCost-
      TotalEvapCost-TotalProcessCost;
  Writep(TotalProfit);
end
```

7.43. Product-Mix (Bloomington Breweries) (winstR3-01)

—- Run LPL Code , HTML Document –

Problem: "Bloomington Breweries produces beer and ale. Beer sells for $5 per barrel, and ale sells for $2 per barrel. Producing a barrel of beer requires 5 lb of corn and 2 lb of hops. Producing a barrel of ale requires 2 lb of corn and 1 lb of hops. Sixty pounds of corn and 25 lb of hops are available. Formulate an LP that can be used to maximize revenue. Solve the LP graphically. " This probelm is from [24] , Chapter 3 Review Problem 1, page 108

Modeling Steps:

Listing 7.43: The Complete Model implemented in LPL [22]

```
model winstR3_01 "Product-Mix (Bloomington Breweries)";
  set
    product  := [Beer, Ale];
    ingred   := [Corn, Hops];
  parameter
    Price{product}                := [5.00, 2.00];
    IngredUsed{product, ingred}   := [5, 2,  2, 1];
    IngredAvail{ingred}           := [60, 25];
  variable Produce{product};
  constraint IngredReq{ingred}: sum{product} IngredUsed*
      Produce <= IngredAvail;
  maximize TotalProfit: sum{product} Price*Produce;
  Writep(TotalProfit);
end
```

7.44. Product-mix (Farmer Jones Cake) (winstR3-02)

—- Run LPL Code , HTML Document –

Problem: "Farmer Jones bakes two types of cake (chocolate and vanilla) to supplement his income. Each chocolate cake can be sold for $1, and each vanilla cake can be sold for 500. Each chocolate cake requires 20 minutes of baking time and uses 4 eggs. Each vanilla cake requires 40 minutes of baking time and uses 1 egg. Eight hours of baking time and 30 eggs are available. Formulate an LP to maximize farmer Jones's revenue. Then graphically solve the LP. (A fractional number of cakes is okay.) " This probelm is from [24] , Chapter 3 Review Problem 2, page 108.

Modeling Steps:

Listing 7.44: The Complete Model implemented in LPL [22]

```
model winstR3_02 "Product-mix (Farmer Jones Cake)";
  set cake  :=  [Chocolate, Vanilla];
  parameter
    Price{cake}       :=  [1.00, 0.50];
    BakingTime{cake}  :=  [20, 40];
    EggsUsed{cake}    :=  [4, 1];
    BakingTimeAvail   :=  8*60;
    EggsAvail         :=  30;
  variable BakeCakes{cake};
  constraint
    BakingTimeLimit: sum{cake} BakingTime*BakeCakes <=
        BakingTimeAvail;
    EggLimit: sum{cake} EggsUsed*BakeCakes <= EggsAvail;
  maximize  TotalRevenue: sum{cake} Price*BakeCakes;
  Writep(TotalRevenue,BakeCakes);
end
```

7.45. Investment (winstR3-03)

—- Run LPL Code , HTML Document –

Problem: "I now have $100. The following investments are available during the next three years:

Investment A: Every dollar invested now yields $0.10 a year from now and $1.30 three years from now.

Investment B: Every dollar invested now yields $0.20 a year from now and $1.10 two years from now.

Investment C: Every dollar invested a year from now yields $1.50 three years from now.

During each year, uninvested cash can be placed in money market funds, which yield 6may be placed in each of investments A, B, and C. Formulate an LP to maximize my cash on hand three years from now." This problem is from [24] , Chapter 3 Review Problem 3, page 108.

Modeling Steps:

Listing 7.45: The Complete Model implemented in LPL [22]

```
model winstR3_03 "Investment";
  set  invest := [A, B, C];
       year   :=[ 0..3];
  parameter  CashAvail{year} := [100 0 0 0];
    CashFlow{invest,year} := [1.00, -0.10,     0, -1.30,
                              1.00, -0.20, -1.10,     0,
                                 0,  1.00,     0, -1.50];
    MaxInvestment     := 50;
    MoneyInterestRate := 0.06;
    MoneyRateMult     := 1.0 + MoneyInterestRate;
  variable
    Investment{invest} [0..MaxInvestment];
    MoneyMarket{year};
  constraint
    CashBalance{year}: CashAvail + MoneyRateMult*
        MoneyMarket[year-1]
      = sum{invest} CashFlow*Investment  + MoneyMarket;
  maximize CashAtEndOfYear3: MoneyMarket[#year-1];
  Writep(CashAtEndOfYear3);
end
```

7.46. Process Oil (Sunco Oil) (winstR3-04)

—- Run LPL Code , HTML Document –

Problem: "Sunco processes oil into aviation fuel and heating oil. It costs $40 to purchase each 1000 barrels of oil, which is then distilled and yields 500 barrels of aviation fuel and 500 barrels of heating oil. Output from the distillation may be sold directly or processed in the catalytic cracker. If sold after distillation without further processing, aviation fuel sells for $60 per 1000 barrels, and heating oil sells for $40 per 1000 barrels. It takes 1 hour to process 1000 barrels of aviation fuel in the catalytic cracker, and these 1000 barrels can be sold for $130. It takes 45 minutes to process 1000 barrels of heating oil in the cracker, and these 1000 barrels can be sold for $90. Each day, at most 20,000 barrels of oil can be purchased, and 8 hours of cracker time are available. Formulate an LP to maximize Sunco's profits." This probelm is from [24] , Chapter 3 Review Problem 4, page 109.

Modeling Steps:

Listing 7.46: The Complete Model implemented in LPL [22]

```
model winstR3_04 "Process Oil (Sunco Oil)";
  set
    product,p := [AvaiationFuel, HeatingOil];
    process,c := [Distill, Cracker];
  parameter
    RawOilCost            := 40;     -- $/1000 barrels
    DistillYield{p} := [0.5, 0.5];  -- 1000 barrels
    Price{c,p} := [ 60, 40,  -- $/1000 barrels
                  130, 90];
    CrackerTime{p}   := [60, 45];     -- min/1000 barrels
    CrackerTimeAvail := 8*60;
    RawOilAvail      := 20;           -- 1000 barrels
  variable
    Produce{c, p};
    RawOilPurchase [0..RawOilAvail];
  constraint
    DistillBalance{p}: DistillYield*RawOilPurchase = sum{
        c} Produce;
    CrackerLimit: sum{p} CrackerTime*Produce['Cracker',p]
         <= CrackerTimeAvail;
  maximize TotalProfit:  sum{p,c} Price*Produce
                    - RawOilCost*RawOilPurchase;
  Writep(TotalProfit);
end
```

7.47. Investment (Finco) (winstR3-05)

—- Run LPL Code , HTML Document –

Problem: "Finco has the following investments available:

Investment A For each dollar invested at time 0, we re ceive $0.10 at time 1 and $1.30 at time 2. (Time 0 = now; time 1 = one year from now; and so on.)

Investment B For each dollar invested at time 1, we re ceive $1.60 at time 2.

Investment C For each dollar invested at time 2, we re ceive $1.20 at time 3.

At any time, leftover cash may be invested in T-bills, which pay 10be invested in each of investments A, B, and C. Formulate an LP that can be used to maximize Finco's cash on hand at time 3."

This probelm is from [24] , Chapter 3 Review Problem 5, page 109.

Modeling Steps:

Listing 7.47: The Complete Model implemented in LPL [22]

```
model winstR3_05 "Investment (Finco)";
  set  invest := [A, B, C];
       year   := [0..3];
  parameter  CashAvail{year} := [100, 0, 0, 0];
    CashFlow{invest,year} := [1.00, -0.10, -1.30,     0,
                                 0,  1.00, -1.60,     0,
                                 0,     0,  1.00, -1.20];
    MaxInvestment     := 50;
    TbillInterestRate := 0.10;
    TbillRateMult     := 1.0 + TbillInterestRate;
  variable
    Investment{invest} [0..MaxInvestment];
    Tbill{year|year<#year-1};
  constraint
    CashBalance{year|year<#year-1}: CashAvail +
        TbillRateMult*Tbill[year-1]
      = sum{invest} CashFlow*Investment + Tbill;
  maximize CashAtEndOfYear3:   TbillRateMult*Tbill[2]
    - sum{invest} CashFlow[invest,3]*Investment;
  Writep(CashAtEndOfYear3);
end
```

7.48. Blending Steel (winstR3-06)

—- Run LPL Code , HTML Document –

Problem: "All steel manufactured by Steelco must meet the fol- lowing requirements: 3.2-3.5% carbon; 1.8-2.5% silicon; 0.9-1.2% nickel; tensile strength of at least 45,000 pounds per square inch (psi). Steelco manufactures steel by combin- ing two alloys. The cost and properties of each alloy are given in Table 7.22. Assume that the tensile strength of a mixture of the two alloys can be determined by averaging that of the al- loys that are mixed together. For example, a one-ton mixture that is 40% alloy 1 and 60% alloy 2 has a tensile strength of 0.4(42,000) + 0.6(50,000). Use linear programming to determine how to minimize the cost ofproducing a ton of steel." This probelm is from [24] , Chapter 3 Review Problem 6, page 109.

	Alloy 1	Alloy 2
Cost per ton	$190	$200
Percent silicon	2%	2.5%
Percent nickel	1%	1.5%
Percent Carbon	3%	4%
Tensile strenght	42,000 psi	50,000 psi

Table 7.6:

Modeling Steps:

Listing 7.48: The Complete Model implemented in LPL [22]

```
model winstR3_06 "Blending Steel";
  set
    elements := [Carbon, Silicon, Nickel];
    alloy    :=[ 1..2];
  parameter
    MinReq{elements}  := [0.032, 0.018, 0.009];
    MaxReq{elements}  := [0.035, 0.025, 0.012];
    MinTensilStrength := 45000;
    Cost{alloy}       := [190, 200];
    Properties{elements, alloy} := [0.030, 0.040,  0.020,
         0.025,  0.010, 0.015];
    TensilStrength{alloy}       := [42000, 50000];
  variable Produce{alloy};
  constraint
    ElementLimits{elements}: MinReq <= sum{alloy}
        Properties*Produce <= MaxReq;
```

```
    MinTensil: sum{alloy} TensilStrength*Produce >=
        MinTensilStrength;
    TotalProduce: sum{alloy} Produce = 1;
  minimize TotalCost: sum{alloy} Cost*Produce;
  Writep(TotalCost);
end
```

7.49. Production Planning (winstR3-07)

—- Run LPL Code , HTML Document –

Problem: "Steelco manufactures two types of steel at three different steel mills. During a given month, each steel mill has 200 hours of blast furnace time available. Because of differences in the furnaces at each mill, the time and cost to produce a ton of steel differs for each mill. The time and cost for each mill are shown in Table 7.22. Each month, Steelco must manufacture at least 500 tons of steel 1 and 600 tons of steel 2. Formulate an LP to minimize the cost of manufacturing the desired steel." This probelem is from [24] , Chapter 3 Review Problem 7, page 109.

	Steel 1		Steel 2	
	Cost	Time (minutes)	Cost	Time (minutes)
Mill 1	$10	20	$11	22
Mill 2	$12	24	$9	18
Mill 3	$14	28	$10	30

Table 7.7:

Modeling Steps:

Listing 7.49: The Complete Model implemented in LPL [22]

```
model winstR3_07 "Production Planning";
  set  mill      := [1..3];
       steel     := [1..2];
  parameter
    TimeAvail{mill} :=[200, 200, 200]; -- Hours/Month
    ProdCost{mill, steel}:=[10 11 12 9 14 10]; -- $/Ton
    ProdTime{mill,steel}:=[20 22 24 18 28 30]; -- Minutes
        /Ton
    MinutesPerHour          :=  60;    -- Minutes/Hour
    MinProd{steel}      :=  [500, 600]; -- Tons/Month
  variable Produce{mill,steel};  -- Tons/Month
  constraint
    TimeLimit{mill}: sum{steel} ProdTime*Produce <=
        TimeAvail*MinutesPerHour;
    MinProduction{steel}: sum{mill} Produce >= MinProd;
  minimize TotalCost: sum{mill, steel} ProdCost*Produce;
  Writep(TotalCost);
end
```

7.50. Production Planning with Diet (winstR3-08)

—- Run LPL Code , HTML Document –

Problem: "Walnut Orchard has two farms that grow wheat and corn. Because of differing soil conditions, there are differ- ences in the yields and costs of growing crops on the two farms. The yields and costs are shown in Table 7.22. Each farm has 100 acres available for cultivation; 11,000 bushels of wheat and 7000 bushels of corn must be grown. Determine a planting plan that will minimize the cost of meeting these demands. How could an extension of this model be used to allocate crop production efficiently throughout a nation?" This probelm is from [24] , Chapter 3 Review Problem 8, page 109.

	Farm 1	Farm 2
Corn yield/acre	500 bushels	650 bushels
Cost/acre of corn	$100	$120
Wheat yield/acre	400 bushels	350 bushels
Cost/acre of wheat	$90	$80

Table 7.8:

Modeling Steps:

Listing 7.50: The Complete Model implemented in LPL [22]

```
model winstR3_08 "Production Planning with Diet";
  set
    farm  :=[ 1..2];
    crop  := [Wheat, Corn];
  parameter
    CropCost{farm,crop} := [90, 100,  80, 120]; -- $/acre
    CropYield{farm,crop}:= [400, 500, 350, 650]; --
        bushels/acre
    Demand{crop}      := [11000, 7000]; -- bushels
    LandAvail{farm}   := [100, 100];    -- acres
  variable Grow{farm,crop}; -- acres
  constraint
    MinCrop{crop}: sum{farm} CropYield*Grow >= Demand;
    LandCapacity{farm}: sum{crop} Grow <= LandAvail;
  minimize  TotalCost: sum{farm, crop} CropCost*Grow;
  Writep(TotalCost);
end
```

7.51. Process Scheduling (winstR3-09)

—- Run LPL Code , HTML Document –

Problem: "Candy Kane Cosmetics (CKC) produces Leslie Per- fume, which requires chemicals and labor. Two production processes are available: Process 1 transforms 1 unit of la- bor and 2 units of chemicals into 3 oz of perfume. Process 2 transforms 2 units of labor and 3 units of chemicals into 5 oz of perfume. It costs CKC $3 to purchase a unit of la- bor and $2 to purchase a unit of chemicals. Each year, up to 20,000 units of labor and 35,000 units of chemicals can be purchased. In the absence of advertising, CKC believes it can sell 1000 oz of perfume. To stimulate demand for Leslie, CKC can hire the lovely model Jenny Nelson. Jenny is paid $100/hour. Each hour Jenny works for the company is es- timated to increase the demand for Leslie Perfume by 200 oz. Each ounce of Leslie Perfume sells for $5. Use linear programming to determine how CKC can maximize profits." This probelm is from [24] , Chapter 3 Review Problem 9, page 109.

Modeling Steps:

Listing 7.51: The Complete Model implemented in LPL [22]

```
model winstR3_09 "Process Scheduling";
  set process := [1..2];
  parameter
    LaborUnits{process}    := [1, 2];
    ChemicalUnits{process} := [2, 3];
    PerfumeYield{process}  := [3, 5];
    LaborCost      := 3.00;
    ChemicalCost   := 2.00;
    LaborAvail     := 20000;
    ChemicalAvail  := 35000;
    BasicDemand    := 1000;
    Cost      := 100;
    DemandPerHour  := 200;
    PricePerfume   := 5;
  variable
    ProcessUnits{process};
    Hours;
  expression
    TotalRevenue     : sum{process} PricePerfume*
        PerfumeYield*ProcessUnits ;
    TotalLaborCost   : sum{process} LaborCost*LaborUnits*
        ProcessUnits ;
    TotalChemicalCost: sum{process} ChemicalCost*
        ChemicalUnits*ProcessUnits ;
    TotalCost        : Cost*Hours;
```

```
    TotalCost1       : TotalLaborCost + TotalChemicalCost
         + TotalCost;
  constraint
    LaborLimit: sum{process} LaborUnits*ProcessUnits  <=
        LaborAvail;
    ChemicalLimit: sum{process} ChemicalUnits*
        ProcessUnits  <= ChemicalAvail;
    DemandLimit: sum{process} PerfumeYield*ProcessUnits
              <= BasicDemand + DemandPerHour*Hours;
  maximize TotalProfit: TotalRevenue - TotalCost1;
  Writep(TotalProfit);
end
```

7.52. Product-mix (Carco Advertising) (winstR3-10)

—- Run LPL Code , HTML Document –

Problem: "Carco has a $150,000 advertising budget. To increase automobile sales, the firm is considering advertising in news-papers and on television. The more Carco uses a particular medium, the less effective is each additional ad. Table 44 shows the number of new customers reached by each ad. Each newspaper ad costs $1000, and each television ad costs $10,000. At most, 30 newspaper ads and 15 television ads can be placed. How can Carco maximize the number of new customers created by advertising?" This problem is from [24] , Chapter 3 Review Problem 10, page 110.

	Number of Ads	New Customers
Newspaper	1-10	900
	11-20	600
	21-30	300
Television	1-5	10,000
	6-10	5000
	11-15	2000

Table 7.9:

Modeling Steps:

Listing 7.52: The Complete Model implemented in LPL [22]

```
model winstR3_10 "Product-mix (Carco Advertising)";
  set media  := ['NP1-10', 'NP11-20', 'NP21-30', 'TV1-5',
      'TV6-10', 'TV11-15'];
  parameter
    AdvertBudget     := 150000;
    AdvertCost{media}:=[1000 1000 1000 10000 10000
       10000];
    MaxPlacement{media}:= [10, 10, 10, 5, 5, 5];
    NewCustomers{media}:=[900 600 300 10000 5000 2000];
  variable PlaceAds{media} [0..MaxPlacement];
  constraint MaxBudget: sum{media} AdvertCost*PlaceAds <=
      AdvertBudget;
  maximize TotalCust: sum{media} NewCustomers*PlaceAds;
  Writep(TotalCust);
end
```

7.53. Process Oil (winstR3-11)

— Run LPL Code , HTML Document –

Problem: "Sunco Oil has refineries in Los Angeles and Chicago. The Los Angeles refinery can refine up to 2 million barrels of oil per year, and the Chicago refinery up to 3 million. Once refined, oil is shipped to two distribution points: Houston and New York City. Sunco estimates that each distribution point can sell up to 5 million barrels per year. Because of dif-ferences in shipping and refining costs, the profit earned (in dollars) per million barrels of oil shipped depends on where the oil was refined and on the point of distribution (see Table 7.22). Sunco is considering expanding the capacity of each re-finery. Each million barrels of annual refining capacity that is added will cost $120,000 for the Los Angeles refinery and $150,000 for the Chicago refinery. Use linear program-ming to determine how Sunco can maximize its profits less expansion costs over a ten-year period." This probelm is from [24] , Chapter 3 Review Problem 11, page 110.

	Profitt Per Million Barrels	
	To Houston	To New York
From Los Angeles	$20,000	$15,000
From Chicago	$18,000	$17,000

Table 7.10:

Modeling Steps:

Listing 7.53: The Complete Model implemented in LPL [22]

```
model winstR3_11 "Process Oil";
  set  refinery   := [LosAngeles, Chicago];
       distrpoint := [Houston, NewYork];
  parameter
    RefineCap{refinery}  := [2, 3]; --million barrels
    MaxSales{distrpoint} := [5, 5]; --million barrels
    Profit{refinery,distrpoint}:=[20 15 18 17]; -- $1000/
        million barrels
    AddCapacityCost{refinery}  := [120, 150];  -- $1000/
        million barrels
    PlanningYears := 10;
  variable
    Refine{refinery,distrpoint};   --  million barrels/
        year
```

```
    AddCapacity{refinery};          --  million barrels
  expression
    TotalProfitPerYear: sum{refinery, distrpoint} Profit*
        Refine;
    TotalAddCapCost   : sum{refinery} AddCapacityCost*
        AddCapacity;
  constraint
    RefineryLimit{refinery}: sum{distrpoint} Refine <=
        RefineCap + AddCapacity;
    MaxDemand{distrpoint}: sum{refinery} Refine  <=
        MaxSales;
  maximize TotalProfit: PlanningYears*TotalProfitPerYear
      - TotalAddCapCost;
  Writep(TotalProfit);
end
```

7.54. Telephone Survey (winstR3-12)

— Run LPL Code , HTML Document –

Problem: "For a telephone survey, a marketing research group needs to contact at least 150 wives, 120 husbands, 100 single adult males, and 110 single adult females. It costs $2 to make a daytime call and (because of higher labor costs) $5 to make an evening call. Table 7.22 lists the results. Because of limited staff, at most half of all phone calls can be evening calls. Formulate an LP to minimize the cost of completing the survey." This probelm is from [24] , Chapter 3 Review Problem 12, page 110.

Person Responding	Percent of Daytime Calls	Percent of Evening Calls
Wife	30	30
Husband	10	30
Single male	10	15
Single female	10	20
None	40	5

Table 7.11:

Modeling Steps:

Listing 7.54: The Complete Model implemented in LPL [22]

```
model winstR3_12 "Telephone Survey";
  set time   := [Daytime, Evening];
    people := [Wife, Husband, SingleMale, SingleFemale];
  parameter
    RespondPercent{people,time}:=[0.3 0.3 0.1 0.3
                                  0.1 .15 0.1 0.2];
    MustReach{people} := [150, 120, 100, 110];
    CallCost{time} := [2, 5];
  variable Calls{time};
  constraint
    ContactReq{people}: sum{time} RespondPercent*Calls >=
         MustReach;
    StaffLimit: Calls['Evening']<=1/2*(sum{time} Calls);
  minimize TotalCost: sum{time} CallCost*Calls;
  Writep(TotalCost);
end
```

7.55. Feed Blending (winstR3-13)

—- Run LPL Code , HTML Document –

Problem: "Feedco produces two types of cattle feed, both consist-ing totally of wheat and alfalfa. Feed 1 must contain at least 80% wheat, and feed 2 must contain at least 60% alfalfa. Feed 1 sells for $1.50/1b, and feed 2 sells for $1.30/1b. Feedco can purchase up to 1000 lb of wheat at 500/1b and up to 800 lb of alfalfa at 400/lb. Demand for each type of feed is unlimited. Formulate an LP to maximize Feedco's profit." This probelm is from [24] , Chapter 3 Review Problem 13, page 110.

Modeling Steps:

Listing 7.55: The Complete Model implemented in LPL [22]

```
model winstR3_13 "Feed Blending";
  set
    feed    := [f1, f2];
    grain   := [wheat, alfalfa];
  parameter
    MinGrainPct{feed, grain}  := [0.80,   0,   0, 0.60];
    Price{feed}         := [1.50, 1.30];
    GrainCost{grain}    := [0.50, 0.40];
    GrainAvail{grain}   := [1000, 800];
  variable Produce,P{feed, grain};
  constraint
    MinGrain{feed,grain|MinGrainPct>0}: P >= MinGrainPct
        *(sum{grain} P);
    MaxGrain{grain}: sum{feed} P <= GrainAvail;
  maximize TotalProfit: sum{feed,grain} Price*P - sum{
      feed,grain} GrainCost*P;
  Writep(TotalProfit,P);
end
```

7.56. Feed Blending (winstR3-14)

— Run LPL Code , HTML Document –

Problem: "Feedco (see Problem 13) has decided to give its customer (assume it has only one customer) a quantity discount. If the customer purchases over 300 lb of feed 1, each pound over the first 300 lb will sell for only \$1.25/1b. Similarly, if the customer purchases more than 300 pounds of feed 2, each pound over the first 300 lb will sell for \$1.00/1b. Modify the LP of Problem 13 to account for the presence of quantity discounts. (Hint: Define variables for the feed sold at each price.)" This probelm is from [24] , Chapter 3 Review Problem 14, page 110.

Modeling Steps:

Listing 7.56: The Complete Model implemented in LPL [22]

```
model winstR3_14 "Feed Blending";
  set
    feed   := [f1, f2];
    grain  := [Wheat, AlfAlfa];
    price  := [Standard, Discount];
  parameter
    MinGrainPct{feed, grain}  := [0.80,   0,  0 , 0.60];
    FeedPrice{feed,price}  := [1.50, 1.25,  1.30, 1.00];
    GrainCost{grain}          := [0.50, 0.40];
    GrainAvail{grain}         := [1000, 800];
    DiscountCutoff            := 300;
  variable
    Produce,P{feed,grain};
    Sales,S{feed,price};
  constraint
    MinGrain{feed,grain|MinGrainPct>0}: P >= MinGrainPct
        *(sum{grain} P);
    MaxGrain{grain}: sum{feed} P <= GrainAvail;
    FeedSales{feed}: sum{grain} P >= sum{price} S;
    SBound: S['Standard'] <= DiscountCutoff;
  maximize TotalProfit: sum{feed,price} FeedPrice*S - sum
      {feed,grain} GrainCost*P;
  Writep(TotalProfit,P,S);
end
```

7.57. Feed Blending (winstR3-14b)

—- Run LPL Code , HTML Document –

Problem: "Feedco (see Problem 13) has decided to give its customer (assume it has only one customer) a quantity discount. If the customer purchases over 300 lb of feed 1, each pound over the first 300 lb will sell for only \$1.25/1b. Similarly, if the customer purchases more than 300 pounds of feed 2, each pound over the first 300 lb will sell for \$1.00/1b. Modify the LP of Problem 13 to account for the presence of quantity discounts. (Hint: Define variables for the feed sold at each price.)" This probelm is from [24] , Chapter 3 Review Problem 14, page 110.

Modeling Steps:

Listing 7.57: The Complete Model implemented in LPL [22]

```
model winstR3_14b "Feed Blending";
  set
    feed   := [F1, F2];
    grain  := [Wheat, AlfAlfa];
    price  := [Standard, Discount];
  parameter
    MinGrainPct{feed, grain}  := [0.80,  0,  0 , 0.60];
    FeedPrice{feed,price}  := [1.50, 1.25, 1.30, 1.00];
    GrainCost{grain}          := [0.50, 0.40];
    GrainAvail{grain}         := [1000, 800];
    DiscountCutoff            := 300;
  variable Produce,P{feed, grain, price};
  constraint
    MinGrain{feed,grain|MinGrainPct>0}: sum{price} P >=
        MinGrainPct*(sum{grain,price} P);
    MaxGrain{grain}: sum{feed,price} P <= GrainAvail;
    MaxStandard{feed}: sum{grain,price|price='Standard'}
        P <= DiscountCutoff;
  maximize TotalProfit: sum{feed,grain,price} (FeedPrice*
      P - GrainCost*P);
  Writep(TotalProfit,P);
end
```

7.58. Production Planning (winstR3-15)

— Run LPL Code , HTML Document –

Problem: "Chemco produces two chemicals: A and B. These chemicals are produced via two manufacturing processes. Process 1 requires 2 hours of labor and 1 lb of raw material to produce 2 oz of A and 1 oz of B. Process 2 requires 3 hours of labor and 2 lb of raw material to produce 3 oz of A and 2 oz of B. Sixty hours of labor and 40 lb of raw material are available. Demand for A is unlimited, but only 20 oz of B can be sold. A sells for \$16/oz, and B sells for \$14/oz. Any B that is unsold must be disposed of at a cost of \$2/oz. Formulate an LP to maximize Chemco's revenue less disposal costs." This probelm is from [24] , Chapter 3 Review Problem 15, page 110.

Modeling Steps:

Listing 7.58: The Complete Model implemented in LPL [22]

```
model winstR3_15 "Production Planning";
  set
    chemical  :=  [A, B];
    process   :=  [1, 2];
  parameter
    LaborReq{process}              := [2, 3];
    RawMatReq{process}             := [1, 2];
    ChemYield{process,chemical}    := [2, 1,
                                       3, 2];
    LaborAvail                     := 60;
    RawMatAvail                    := 40;
    Price{chemical}                := [16.00, 14.00];
    Demand{chemical}               := [-1, 20];
    DisposeCost{chemical}          := [-1, 2.00];
  variable
    Produce{process};
    Sales{chemical};
    Dispose{chemical|chemical='B'};
  constraint
    LaborLimit: sum{process} LaborReq*Produce <=
        LaborAvail;
    RawMatLimit: sum{process} RawMatReq*Produce <=
        RawMatAvail;
    ProcessBalance{chemical}: sum{process} ChemYield*
        Produce = Sales + Dispose;
    MaxDemand: Sales['B'] <= Demand['B'];
  maximize TotalProfit: sum{chemical} Price*Sales
                      - sum{chemical} DisposeCost*Dispose;
  Writep(TotalProfit,Produce,Sales,Dispose);
```

```
end
```

7.59. Product Mix (winstR3-17)

— Run LPL Code , HTML Document –

Problem: "Furnco manufactures tables and chairs. Each table and chair must be made entirely out of oak or entirely out of pine. A total of 150 board ft of oak and 210 board ft of pine are available. A table requires either 17 board ft of oak or 30 board ft of pine, and a chair requires either 5 board ft of oak or 13 board ft of pine. Each table can be sold for $40, and each chair for $15. Formulate an LP that can be used to maximize revenue." This probelm is from [24] , Chapter 3 Review Problem 17, page 110.

Modeling Steps:

Listing 7.59: The Complete Model implemented in LPL [22]

```
model winstR3_17 "Product Mix";
  set
    furniture  := [desk, chair];
    materials  := [oak, pine];
  parameter
    MatAvail{materials}              := [150, 210];
    MatUsed{furniture, materials}    := [17, 30,  5, 13];
    Price{furniture}                 := [40.00, 15.00];
  variable Produce{furniture};
  constraint MatReq{materials}: sum{furniture} MatUsed*
      Produce <= MatAvail;
  maximize TotalProfit: sum{furniture} Price*Produce ;
  Writep(TotalProfit);
end
```

7.60. Bus-ville SchoolDistricts (winstR3-18)

—- Run LPL Code , HTML Document –

Problem: "The city of Busville contains three school districts. The number of. minority and nonminority students in each district is given in Table 7.22. Of all students, 25% (200/800) are minority students. The local court has decided that each of the town's two high schools (Cooley High and Walt Whitman High) must have approximately the same percentage of minority students (within +5%) as the entire town. The distances (in miles) between the school districts and the high schools are given in Table 7.17. Each high school must have an enrollment of 300-500 students. Use linear programming to determine an assignment of students to schools that minimizes the total distance students must travel to school." This probelm is from [24] , Chapter 3 Review Problem 18, page 110.

District	Minority Students	Nonminority Students
1	50	200
2	50	250
3	100	150

Table 7.12:

District	Cooley High	Walt Whitman High
1	1	2
2	2	1
3	1	1

Table 7.13:

Modeling Steps:

Listing 7.60: The Complete Model implemented in LPL [22]

```
model winstR3_18 "Bus-ville SchoolDistricts";
  set
    district  :=[ 1..3];
    status    := [Minority, NonMinority];
    schools   := [CooleyHigh, WaltWitmanHigh];
```

```
  parameter
    StudentCount{district,status} := [ 50, 200,  50, 250,
         100, 150];
    Distance{district,schools} := [1, 2,  2, 1,  1, 1];
    MinorityPercent := 0.25;
    MinMinority := MinorityPercent - 0.05;
    MaxMinority := MinorityPercent + 0.05;
    MinStudents := 300;
    MaxStudents := 500;
  variable AssignStudents,A{district,schools,status};
  constraint
    StudentAssignment{district,status}: sum{schools} A =
        StudentCount;
    StudentLimit{schools}: MinStudents <= sum{district,
        status} A <= MaxStudents;
    MinorityLowerLimit{schools}: sum{district,status|
        status='Minority'} A
             >= MinMinority*(sum{district,status} A);
    MinorityUpperLimit{schools}: sum{district,status|
        status='Minority'} A
             <= MaxMinority*(sum{district,status} A);
  minimize TotalDistance: sum{district,schools,status}
      Distance*A;
  Writep(TotalDistance);
end
```

7.61. Brady (winstR3-19)

—- Run LPL Code , HTML Document –

Problem: "Brady Corporation produces cabinets. Each week, they require 90,000 cu ft of processed lumber. They may obtain lumber in two ways. First, they may purchase lumber from an outside supplier and then dry it at their kiln. Sec-ond, they may chop down logs on their land, cut them into lumber at their sawmill, and finally dry the lumber at their kiln. Brady can purchase grade 1 or grade 2 lumber. Grade l lumber costs $3 per cu ft and when dried yields 0.7 cu ft of useful lumber. Grade 2 lumber costs $7 per cubic foot and when dried yields 0.9 cu ft of useful lumber. It costs the Company $3 to chop down a log. After being cut and dried, a log yields 0.8 cu ft of lumber. Brady incurs costs of $4 per cu ft of lumber dried. It costs $2.50 per cu ft of logs sent through the sawmill. Each week, the sawmill can process up to 35,000 cu ft of lumber. Each week, up to 40,000 cu ft of grade 1 lumber and up to 60,000 cu ft of grade 2 lumber can be purchased. Each week, 40 hours of time are available for drying lumber. The time it takes to dry 1 cu ft of grade 1 lumber, grade 2 lumber, or logs is as follows: grade 1-2 seconds; grade 2-0.8 second; log 1.3 seconds. Formulate a.11 LP to help Brady minimize the weekly cost of meeting the demand for processed lumber." This probelm is from [24] , Chapter 3 Review Problem 19, page 111.

Modeling Steps:

Listing 7.61: The Complete Model implemented in LPL [22]

```
model winstR3_19 "Brady";
  set
    lumber    := [outside, own, log];
    grade     := [g1, g2];
  parameter
    Price{grade}     := [3.00, 7.00];
    Yield{lumber}    := [0.7, 0.9, 0.8];
    ChopCost         :=  3.00;
    DryCost          :=  4.00;
-- model not finished
  Write('The model is not complete\n');
end
```

7.62. Canadian Parks Commission (winstR3-20)

—- Run LPL Code , HTML Document –

Problem: "The Canadian Parks Commission controls two tracts of land. Tract 1 consists of 300 acres and tract 2, 100 acres. Each acre of tract 1 can be used for spruce trees or hunting, or both. Each acre of tract 2 can be used for spruce trees or camping, or both. The capital (in hundreds of dollars) and labor (in man-days) required to maintain one acre of each tract, and the profit (in thousands of dollars) per acre for each possible use of land are given in Table 7.22. Capital of $150,000 and 200 man-days of labor are available. How should the land be allocated to various uses to maximize profit received from the two tracts?" This probelm is from [24] , Chapter 3 Review Problem 20, page 111.

	Capital	Labor	Profit
Tract 1 Spruce	3	0.1	0.2
Tract 1 Hunting	3	0.2	0.4
Tract 1 Both	4	0.2	0.5
Tract 2 Spruce	1	0.05	0.06
Tract 2 Camping	30	5	0.09
Tract 2 Both	10	1.01	1.1

Table 7.14:

Modeling Steps:

Listing 7.62: The Complete Model implemented in LPL [22]

```
model winstR3_20 "Canadian Parks Commission";
  set  Land     :=  [Tract1, Tract2];
    UsedFor:=[Spruce Hunting SpruceHunt Camping
       SpruceCamp];
  parameter
    AcresAvail{Land}           :=  [300, 100];
/* CapitalUse{Land,UsedFor} :=
     [300 300 400 0 0 100 0 0 3000 1000];
   LaborUse{Land,UsedFor}  :=
     [.1 .2 .2 0 0 .05 0 0 5 1.01];
   Profit{Land,UsedFor}     :=
     [200 400 500 0 0 60 0 0 90 1100]; */
   CapitalUse{Land,UsedFor}:= /Tract1 Spruce      300,
     Tract1 Hunting  300,      Tract1 SpruceHunt  400,
     Tract2 Spruce   100,      Tract2 Camping    3000,
     Tract2 SpruceCamp 1000/;
```

```
   LaborUse{Land,UsedFor}:=  /Tract1 Spruce      0.1,
     Tract1 Hunting     0.2,  Tract1 SpruceHunt  0.2,
     Tract2 Spruce      0.05, Tract2 Camping     5.0,
     Tract2 SpruceCamp  1.01/;
   Profit{Land,UsedFor}:=  /Tract1 Spruce      200,
     Tract1 Hunting   400,  Tract1 SpruceHunt  500,
     Tract2 Spruce     60,  Tract2 Camping      90,
     Tract2 SpruceCamp 1100/;
   CapitalAvail :=  150000;   LaborAvail   :=  200;
  variable Allocate{Land,UsedFor};
  constraint
    AcresLimit{Land}: sum{UsedFor} Allocate<=AcresAvail;
    CapitalLimit: sum{Land, UsedFor} CapitalUse*Allocate
        <= CapitalAvail;
    LaborLimit: sum{Land, UsedFor} LaborUse*Allocate <=
        LaborAvail;
  maximize TotalProfit:sum{Land,UsedFor} Profit*Allocate;
  Writep(TotalProfit);
end
```

7.63. Chandler Enterprises (winstR3-21)

— Run LPL Code , HTML Document –

Problem: "Chandler Enterprises produces two competing prod-ucts: A and B. The company wants to sell these products to two groups of customers: group 1 and group 2. The value each customer places on a unit of A and B is as shown in Table 7.22. Each customer will buy either product A or product B, but not both. A customer is willing to buy product A if she believes that Value of product A - price of product A $\geq$ Value of product B - price of product B and Value of product A - price of product A ≥ 0. A customer is willing to buy product B if she believes that Value of product B - price of product B $\geq$ value of product A - price of product A and Value of product B - price of product B ≥ 0.

Group 1 has 1000 members, and group 2 has 1500 members. Chandler wants to set prices for each product that ensure that group 1 members purchase product A and group 2 members purchase product B. Formulate an LP that will help Chandler maximize revenues." This problem is from [24] , Chapter 3 Review Problem 21, page 111.

Modeling Steps:

	Group 1 Customer	Group 2 Customer
Value of A to	\$10	\$12
Value of B to	\$8	\$15

Table 7.15:

Listing 7.63: The Complete Model implemented in LPL [22]

```
model winstR3_21 "Chandler Enterprises";
  set  product   := [A, B];
       custgroup := [1..2];
  parameter Value{product,custgroup}:=[10 12 8 15];
    Members{custgroup} := [1000, 1500];
  variable Price{product};
  constraint
    a: Value['A','1'] - Price['A'] >= Value['B','1'] -
       Price['B'];
    b: Value['B','2'] - Price['B'] >= Value['A','2'] -
       Price['A'];
    c: Value['A','1'] - Price['A']  >= 0;
```

```
    d: Value['B','2'] - Price['B']  >= 0;
  maximize  TotalRevenues: Members['1']*Price['A'] +
     Members['2']*Price['B'];
  Writep(TotalRevenues);
end
```

7.64. Alden Enterprices (winstR3-22)

— Run LPL Code , HTML Document –

Problem: "Alden Enterprises produces two products. Each prod-uct can be produced on one of two machines. The length of time needed to produce each product (in hours) on each machine is as shown in Table 7.22. Each month, 500 hours of time are available on each machine. Each month, customers are willing to buy up to the quantities of each product at the prices given in Table 7.17. The company's goal is to maximize the revenue obtained from selling units during the next two months. Formulate an LP to help meet this goal." This problem is from [24] , Chapter 3 Review Problem 22, page 112.

	Machine 1	Machine 2
Product 1	4	3
Product 2	7	4

Table 7.16:

	Demand		Prices	
	Month 1	Month 2	Month 1	Month 2
Product 1	100	190	$55	$12
Product 2	140	130	$65	$32

Table 7.17:

Modeling Steps:

Listing 7.64: The Complete Model implemented in LPL [22]

```
model winstR3_22 "Alden Enterprices";
  set product  := [p1,p2];
      machine  := [m1,m2];
      month    := [Jan,Feb];
  parameter  TimeAvail{machine}    := [500, 500];
    MachineTime{product,machine}   := [4, 3,  7, 4];
    Price{product, month}   := [55 12 65 32];
    Demand{product, month} := [100, 190,  140, 130];
  variable Produce{product, machine, month};
  constraint
    TimeCapacity{machine}: sum{product,month} MachineTime
        *Produce <= TimeAvail;
```

```
    MaxDemand{product,month}: sum{machine} Produce <=
        Demand;
  maximize TotalRevenue: sum{product,machine,month} Price
      *Produce;
  Writep(TotalRevenue);
end
```

7.65. Kiriakis (winstR3-23)

—- Run LPL Code , HTML Document –

Problem: "Kiriakis Electronics produces three products. Each product must be processed on each of three types of ma-chines. When a machine is in use, it must be manned by a worker. The time (in hours) required to process each product on each machine and the profit associated with each product are shown in Table 7.22. At present, five type 1 machines, three type 2 machines, and four type 3 machines are available. The company has ten workers available and must determine how many workers to assign to each machine. The plant is open 40 hours per week, and each worker works 35 hours per week. Formulate an LP that will enable Kiriakis to assign workers to machines in a way that maximizes weekly profits. (Note: A worker need not spend the entire work week manning a single machine.)" This probelm is from [24] , Chapter 3 Review Problem 23, page 112.

	Product 1	Product 2	Product 3
Machine 1	2	3	4
Machine 2	3	5	6
Machine 3	4	7	9
Profit	$6	$8	$10

Table 7.18:

Modeling Steps:

Listing 7.65: The Complete Model implemented in LPL [22]

```
model winstR3_23 "Kiriakis";
  set  p     := [1..3]  "Products";
       m     := [1..3]  "Machines";
       w      := [1..10] "Workers";
  parameter ProcessTime{m,p}:=[2 3 4 3 5 6 4 7 9];
    Profit{p}         := [6, 8, 10];
    MachineAvail{m}   := [5, 3, 4];
    PlantOpenHours    := 40;
    WorkerHours       := 35;
  variable
    Produce,P{p};
    MachineHours,M{m};
  constraint
    ProcessLimit{m}: sum{p} ProcessTime*P<=MachineHours;
    MachineLimit{m}: M <= PlantOpenHours*MachineAvail;
```

```
    WorkerLimit: sum{m} M <= 10*WorkerHours;
  maximize TotalProfit: sum{p} Profit*P;
  Writep(TotalProfit);
end
```

7.66. Multi-Period Planning (winstR3-40)

— Run LPL Code , HTML Document –

Problem: "Alexis Cornby makes her living buying and selling corn. On January 1, she has 50 tons of corn and $1000. On the first day of each month Alexis can buy corn at the following prices per ton: January, $300; February, $350; March, $400; April, $500. On the last day of each month Alexis can sell corn at the following prices per ton: January, $250; February, $400; March, $350; April, $550. Alexis stores her corn in a warehouse that can hold at most 100 tons of corn. She must be able to pay cash for all corn the time of purchase. Use linear programming to determine how Alexis can maximize her cash on hand at the end of April. " This probelm is from [24] , Chapter 3 Review Problem 40, page 114.

Modeling Steps:

Listing 7.66: The Complete Model implemented in LPL [22]

```
model winstR3_40 "Multi-Period Planning";
  set m  :=  [Jan, Feb, Mar, Apr] "Months";
  parameter
    StartCorn        := 50;       -- tons of corm
    StartCash        := 1000;     -- dollars
    PurchasePrice{m} := [300, 350, 400, 500];
    SellPrice{m}     := [250, 400, 350, 550];
    StorageAvail     := 100;      -- tons of corn
  variable
    PurchaseCorn,B{m};
    SellCorn{m} [0..StorageAvail];
    StoreCorn{m};
    CashOnHand,C{m};
  constraint
    CashBalance{m}: C = if(m=1,StartCash,C[m-1]) +
        SellPrice*SellCorn - PurchasePrice*B;
    PurchaseLimit{m}: PurchasePrice*B <= if(m=1,StartCash
        ,C[m-1]);
    SellLimit{m}: SellCorn <= StoreCorn;
    CornBalance{m}: StoreCorn = if(m=1,StartCorn,
        StoreCorn[m-1]) + B - SellCorn[m-1];
  maximize CashOnHandApril: C[#m];
  Writep(CashOnHandApril,B,SellCorn,StoreCorn,C);
end
```

7.67. Production Planning (winstR3-40b)

—- Run LPL Code , HTML Document –

Problem: "Alexis Cornby makes her living buying and selling corn. On January 1, she has 50 tons of corn and $1000. On the first day of each month Alexis can buy corn at the following prices per ton: January, $300; February, $350; March, $400; April, $500. On the last day of each month Alexis can sell corn at the following prices per ton: January, $250; February, $400; March, $350; April, $550. Alexis stores her corn in a warehouse that can hold at most 100 tons of corn. She must be able to pay cash for all corn the time of purchase. Use linear programming to determine how Alexis can maximize her cash on hand at the end of April. " This probelm is from [24] , Chapter 3 Review Problem 40, page 114.

Modeling Steps:

Listing 7.67: The Complete Model implemented in LPL [22]

```
model winstR3_40b "Production Planning";
  set m  :=  [Jan, Feb, Mar, Apr];
  parameter
    StartCorn           := 50;
        -- tons of corm
    StartCash           := 1000;
        -- dollars
    PurchasePrice{m}    := [300, 350, 400, 500];
    SellPrice{m}        := [250, 400, 350, 550];
    StorageAvail        := 100;
        -- tons of corn
  variable
    PurchaseCorn,B{m};
    SellCorn{m};
    StoreCorn{m} [0..StorageAvail];
    CashOnHand,C{m};
  constraint
    CashBalance{m}: C = if(m=1,StartCash,C[m-1]) +
        SellPrice*SellCorn - PurchasePrice*B;
    PurchaseLimit{m}: PurchasePrice*B <= if(m=1,StartCash
        ,C[m-1]);
    SellLimit{m}: SellCorn <= StoreCorn;
    CornBalance{m}: StoreCorn = if(m=1,StartCorn,
        StoreCorn[m-1]) + B - SellCorn[m-1];
  maximize CashOnHandApril: C[#m];
  Writep(CashOnHandApril,B,SellCorn,StoreCorn,C);
end
```

7.68. Multi-Period Finance Planning (winstR3-41)

— Run LPL Code , HTML Document –

Problem: "At the beginning of month 1, Finco has $400 in cash. At the beginning of months 1, 2, 3, and 4. Finco rceceives certain revenues, after which it pays hills (see Table 7.22). Any money left over may be invested for one month at the interest rate of 0.1% per month; for two months at 0.5% per month; for three months at 1 % per month; or for four months at 2% per month. Use linear programming to determine an invest ment strategy that maximizes cash on hand at the beinning of month 5." This probelm is from [24] , Chapter 3 Review Problem 41, page 114.

	Revenue	Bills
Month 1	$400	$600
Month 2	$800	$500
Month 3	$300	$500
Month 4	$300	$250

Table 7.19:

Modeling Steps:

Listing 7.68: The Complete Model implemented in LPL [22]

```
model winstR3_41 "Multi-Period Finance Planning";
  set period ,p :=[ 1..4];
  parameter
    StartCash        := 400;
    Interest{p}      := [0.001, 0.005, 0.01, 0.02];
    Revenues{p}      := [400, 800, 300, 300];
    Bills{p}         := [600, 500, 500, 250];
  variable
    Invest{p};
    Cash{p};
  constraint Balance{p}: Cash + Bills + Invest
      = if(p=1,StartCash,Cash[p-1]) + Revenues + (1+
          Interest)*Invest[p-1];
  maximize EndCash: Cash[#p];
  Writep(EndCash,Invest,Cash);
end
```

7.69. Transporation: Waste disposal (winstR3-42)

—- Run LPL Code , HTML Document –

Problem: "City 1 produces 500 tons of waste per day, and city produces 400 tons of waste per day. Waste must be incinerated at incinerator 1 or 2, and each incinerator can process up to 500 tons of waste per day. The cost to incinerate waste is \$40/ton at incinerator 1 and \$30/ton at 2. Incineration reduces each ton of waste to 0.2 tons of debris, which must be dumped at one of two landfills. Each landfill can receive at most 200 tons of debris per day. It costs \$3 per mile to trans-port a ton of material (either debris or waste). Distances (in miles) between locations are shown in Table 7.22. Formulate an LP that can be used to minimize the total cost of disposing of the waste of both cities." This probelm is from [24] , Chapter 3 Review Problem 42, page 114.

	Incinerator 1	Incinerator 2
City 1	30	5
City 2	36	42
	Landfill 1	Landfill 2
Incinerator 1	5	8
Incinerator 2	9	6

Table 7.20:

Modeling Steps:

Listing 7.69: The Complete Model implemented in LPL [22]

```
model winstR3_42 "Transporation: Waste disposal";
  set
    city      := [1..2];
    incin     := [1..2];
    landfill  := [1..2];
  parameter
    WasteProduced{city}         := [500, 400];
    BurnCapacity{incin}         := [500, 500];
    BurnCost{incin}             := [40, 30];
    ResultingDebris             := 0.20;
    LandfillCapacity{landfill}  := [200, 200];
    CostPerMile                 := 3.00;
    DistToIncin{city,incin}     := [30, 5,  36, 42];
    DistToLandfill{incin,landfill}  := [5, 8,   9, 6];
  variable
```

```
    WasteToIncin,W{city,incin};
    DebrisToLandfill,D{incin,landfill};
  expression
    TotalBurnCost     : sum{city,incin} BurnCost*W;
    IncinTranspCost   : CostPerMile*(sum{city,incin}
        DistToIncin*W);
    LandfillTranspCost: CostPerMile*(sum{incin,landfill}
        DistToLandfill*D);
  constraint
    BurnWaste{city}: sum{incin} W = WasteProduced;
    IncinLimit{incin}: sum{city} W <= BurnCapacity;
    IncinBalance{incin}: sum{city} W = 1/ResultingDebris
        *(sum{landfill} D);
    LandfillLimit{landfill}: sum{incin} D <=
        LandfillCapacity;
  minimize TotalCost: TotalBurnCost + IncinTranspCost +
      LandfillTranspCost;
  Writep(TotalCost);
end
```

7.70. Product-mix with blending (winstR3-45)

—- Run LPL Code , HTML Document –

Problem: "Turkeyco produces two types of turkey cutlets for sale to fast food restaurants. Each type of cutlet consists of white meat and dark meat. Cutlet 1 sells for $4/lb and must consist of at least 70% white meat. Cutlet 2 sells for $3/lb and must consist of at least 60% white meat. At most 50 lb of cutlet 1 and 30 lb of cutlet 2 can be sold. The two types of turkey used to manufacture the cutlets are purchased from the Gob-bleGobble Turkey Farm. Each type 1 turkey costs $10 and yields 5 lb of white meat and 2 lb of dark meat. Each type 2 turkey costs $8 and yields 3 lb of white meat and 3 lb of dark meat. Formulate an LP to maximize Turkeyco's profit." This probelm is from [24] , Chapter 3 Review Problem 45, page 115.

Modeling Steps:

Listing 7.70: The Complete Model implemented in LPL [22]

```
model winstR3_45 "Product-mix with blending";
  set
    c  := [1..2]          "Cutlet";
    m  := [White, Dark]   "Meat";
    t  := [1..2]          "Turkey";
  parameter
    Price{c}              := [4.00, 3.00];
    MinWhitePercent{c}    := [0.70, 0.60];
    MaxDemand{c}          := [50, 30];
    TurkeyCost{t}         := [10.00, 8.00];
    TurkeyMeat{t, m}      := [5, 2,   3, 3];
  variable
    PurchaseTurkey{t};
    CutletMeat{c,m};
  expression
    TotalRevenue: sum{c,m} Price*CutletMeat;
    TotalCost   : sum{t} TurkeyCost*PurchaseTurkey;
  constraint
    MinWhiteMeat{c}: sum{m|m='White'} CutletMeat >=
        MinWhitePercent*(sum{m} CutletMeat);
    MaximumDemand{c}: sum{m} CutletMeat <= MaxDemand;
    MeatLimit{m}: sum{c} CutletMeat <= sum{t} TurkeyMeat*
        PurchaseTurkey;
  maximize TotalProfit: TotalRevenue - TotalCost;
  Writep(TotalProfit);
end
```

7.71. Review Prob (Priceler) (winstR3-46)

—- Run LPL Code , HTML Document –

Problem: "Priceler manufactures sedans and wagons. The number of vehicles that can be sold each of the next three months are listed in Table 7.22. Each sedan sells for $8000, and each wagon sells for $9000. It costs $6000 to produce a sedan and $7500 to produce a wagon. To hold a vehicle in inventory for one month costs $150 per sedan and $200 per wagon. During each month, at most 1500 vehicles can be produced. Production line restrictions dictate that during month 1, at least two thirds of all cars produced must be sedans. At the beginning of month 1, 200 sedans and 100 wagons are available. Formulate an LP that can be used to maximize Priceler's profit during the next three months." This probelm is from [24] , Chapter 3 Review Problem 46, page 116.

	Sedans	Wagons
Month 1	1100	600
Month 2	1500	700
Month 3	1200	500

Table 7.21:

Modeling Steps:

Listing 7.71: The Complete Model implemented in LPL [22]

```
model winstR3_46 "Review Prob (Priceler)";
  set  v   := [Sedan, Wagon] "Vehicles";
       m   :=[ 1..3]          "Months";
  parameter  Price{v} := [8000, 9000];
    ProdCost{v}  := [6000, 7500];
    InvtCost{v}  := [150, 200];
    InvtCapacity := 1500;
    StartInvt{v} := [200, 100];
    Demand{m, v} := [1100,  600, 1500,  700, 1200,  500];
  variable
    Produce,P{v,m};
    Inventory,I{v,m};
    Sales,S{v,m}  [0..Demand];
  expression
    TotalRevenue : sum{v,m} Price*S;
    TotalProdCost: sum{v,m} ProdCost*P;
    TotalInvtCost: sum{v,m} InvtCost*I;
    TotalCost    : TotalProdCost+TotalInvtCost;
```

```
  constraint
    InventoryBalance{v,m}:I=if(m=1,StartInvt,I[m-1])+P-S;
    ProductionLine{m|m=1}: P['Sedan',1]>=2/3*(sum{v} P);
    MaxInventory{m}: sum{v} I <= InvtCapacity;
  maximize TotalProfit: TotalRevenue-TotalCost;
  Writep(TotalProfit,P,I,S);
end
```

7.72. EJ-Korvair DeptStore (winstR3-53)

—- Run LPL Code , HTML Document –

Problem: "E.J. Korvair Department Store has $1000 in available cash. At the beginning of each of the next six months, E.J. will receive revenues and pay bills as shown in Table 7.22. It is clear that E.J. will have a short-term cash flow problem until the store receives revenues from the Christmas shopping season. To solve this problem, E.J. must borrow money. At the beginning of July, E.J. may take out a six-month loan. Any money borrowed for a six-month period must be paid back at the end of December along with 9% interest (early payback does not reduce the interest cost of the loan). E.J. may also meet cash needs through month-to-month bor- rowing. Any money borrowed for a one-month period incurs an interest cost of 4% per month. Use linear programming to determine how E.J. can minimize the cost of paying its bills on time." This probelm is from [24] , Chapter 3 Review Problem 53, page 117.

	Revenues	Bills
July	$1000	$5000
August	$2000	$5000
September	$2000	$6000
October	$4000	$2000
November	$7000	$2000
December	$9000	$1000

Table 7.22:

Modeling Steps:

Listing 7.72: The Complete Model implemented in LPL [22]

```
model winstR3_53 "EJ-Korvair DeptStore";
  set m := [July, August, September, October, November,
      December] "Months";
  parameter
    CashAvail    := 1000;
    Revenues{m}  := [1000, 2000, 2000, 4000, 7000, 9000];
    Bills{m}     := [5000, 5000, 6000, 2000, 2000, 1000];
    LoanInterest    := 0.09;
    BorrowInterest  := 0.04;
    LoanRate        := 1.0 + LoanInterest;
    BorrowRate      := 1.0 + BorrowInterest;
  variable
```

```
    CashLevel,C{m};
    SixMonthLoan,L;
    BorrowMonth,S{m};
  constraint
   CashBalance{m|m=1}:
        C = CashAvail + Revenues - Bills + S + L;
    CashBalance1{m|m>1 and m<#m}:
        C = C[m-1] + Revenues - Bills + S - BorrowRate*S[
            m-1];
    CashBalance2{m|m=#m}:
        C = C[m-1] + Revenues - Bills - BorrowRate*S[m-1]
             - LoanRate*L;
  minimize TotalCost: LoanInterest*L + sum{m}
      BorrowInterest*S;
  Writep(TotalCost);
end
```

BIBLIOGRAPHY

[1] Billionnet A. *Optimisation Discrète*. Dunod, Frech Edition, 2007.

[2] Loebel A. Borndoerfer R., Groetschel M. Alcuin's Transportation Problem and Integer Programming. Konrad Zuse Zentrum for Informationstechnik, Berlin, 1995.

[3] McCormick G.P. Bracken J. *Selected Applications of Nonlinear Programming*. John Wiley and Sons, New York, 1968.

[4] Meeraus A. Brooke A., Kendrick D. *GAMS: A User's Guide*. The Scientific Press, Redwood City, California, 1988.

[5] Zenios S.A. Dahl H., Meeraus A. Some Financial Optimization Models: Risk Management. in: Zenios, S.A., ed, Financial Optimization. Cambridge University Press, New York, 1993.

[6] More J.J. Dolan E.D. Benchmarking Optimization Software with COPS., Tech. Report. Mathematics and Computer Science Division,, 2000.

[7] Johnson E.L. *Modeling and Strong Linear Programs for Mixed Integer Programming*, pages 1–43. NATO ASI Series F, Vol. 51, Springer, 1989.

[8] Kernighan B.W. Fourer R., Gay D.M. *AMPL: A Modelling Language for Mathematical Programming*. Duxbury Press/Brooks/Cole Publ. Co., second edition, 2003.

[9] Polya G. *How To Solve It*. Penguin Books 1990, London, 1957.

[10] Dantzig G.B. *Linear Programming and Extensions*. Princeton University Press, Princeton, New Jersey, 1963.

[11] Tempelmeier H. Günther H-O. *Produktion und Logistik*. Springer, Berlin, 2002.

[12] Liebermann G.J. Hillier F.S. *Introduction to Operations Research*. McGraw-Hill, 6rd ed., 1999.

[13] Williams H.P. *Model Building in Mathematical Programming*. John Wiley, Third Edition Revisted, 1993.

[14] D. Kendrick. Caution and Probing in a Macroeconomic Model. Journal of Economic Dynamics and Control 4, 2, 1982.

[15] Murty K.G. *Operations Research: Deterministic Optimization Models*. Prentice Hall, 1994.

[16] O'Mara G.T. Kutcher G.P., Meeraus A. Agriculture Sector and Policy Models. The World Bank, 1988.

[17] Scandizzo P.L. Kutcher G.P. The Agricultural Economy of Northeast Brazil. The Johns Hopkins University Press, Baltimore and London, 1981.

[18] Cesari L. *Optimization – Theory and Applications*. Springer Verlag, 1983.

[19] Bosch R. Mind Sharpener. OPTIMA MPS Newsletter, January 2000.

[20] Shapiro RD. *Optimization models for planning and allocation*. John Wiley Sons, 1984.

[21] Rosenthal R.E. *Tutorial of GAMS in: GAMS: A User's Guide*. The Scientific Press, Redwood City, California, 1988.

[22] Hürlimann T. Reference Manual for the LPL Modeling Language, most recent version. https://matmod.ch/lpl/doc/manual.pdf.

[23] Wiens T.B. The Economics of High-yield Agriculture, in: China: Triple-Cropping at the Baimano Peoples' Commune. The World Bank, Washington DC, 1985.

[24] Winston W.L. *Introduction to mathematical programming, applications and algorithms*. Duxbury Press, second edition, 1995.