DUBLIN CITY UNIVERSITY

CA4011 Operations Research

A. Mathematical Programming & related topics

Contents

1. Linear Programming

1.1 Introduction

1.1.1 Overview

Linear Programming (LP) is concerned with problems of allocation of scarce resources in the "best" possible manner. Typically,

EITHER • Maximise profit

OR • Minimise cost

Conditions and elements of a LP problem:-

1. Decision variables ≥ 0 (*usually*)
2. The "Best" criterion is described by a linear function of the decision variables. This is called the Objective Function
3. Rules (e.g. scarcity of resources) are described by linear functions of the decision variables. These are called the Constraints (or Restrictions)

Advantages

- Efficient solution methods available
- Data variation is easily handed (*comes under "sensitivity analysis"*)
- Often possible to approximate non-linearities.

We present the classic Simplex Method of solution. When using the ***ampl*** tool (Fourer, et al., 2003) for large-scale problems various solution methods (solvers) may be used but we will not go into the details of these solvers.

Extensions to Non-linear programming include

- Quadratic programming
- Convex programming
- Geometric programming
- Integer programming
- Dynamic programming

We will introduce some of these in this module as well as Goal Programming where there is more than one criterion – involves deciding a priority order for the goals.

Typical Areas of Application

- Petro-chemical industries, often now with an environmental focus (see e.g. (Al-Sharrah, et al., 2006))
- Agri-food industries (see e.g. (Ahumada & Villalobos, 2009))
  - farm planning
  - feed mix
  - product mix
- Distribution
  - warehouse location
  - minimising transport costs
- Public Services (see e.g. (Jacovkis, et al., 1989)
  - water supply
  - roads

1.1.2 Formulation of LP problems, with introductory examples

There are three steps in formulation, of which Step 1 is probably the most crucial:

1. Identify decision variables (what can one decide about or control?)
2. Identify restrictions or constraints
3. Identify "best" criterion

Example 1: Product mix:

The Handy-Dandy Company makes three types of kitchen appliances (*A*, *B* and *C*). To make each of these appliance types, just two inputs are required - labour and materials. Each unit of *A* made requires 7 hours of labour and four Kg of materials; for each unit of *B* made the requirements are 3 hours of labour and 4 Kg of materials, while for *C* the unit requirements are 6 hours of labour and 5 Kg of material. The company expects to make a profit of €40 for every unit of *A* sold, and the profit per unit for *B* and *C* are €20 and €30 respectively. Given that the company has available to it 150 hours of labour and 200 Kg of material each day, formulate this as a linear programming problem.

Formulation:

Step 1: Identify decision variables (what decisions can be made?)

For this production mix problem the company can decide on how many units of appliances A, B and C to make. Therefore, the decision variables are, say,

Let X_1 = number of units of appliance A to make per day

Let X_2 = number of units of appliance B to make per day

Let X_3 = number of units of appliance C to make per day

Step 2: Identify restrictions or constraints

Evidently, $X_1 \geq 0$, $X_2 \geq 0$, $X_3 \geq 0$ [*This is often the case but not always*]

Also, there are limitations on labour and materials:

Labour (per day): $7X_1 + 3X_2 + 6X_3 \leq 150$

Materials (per day): $4X_1 + 4X_2 + 5X_3 \leq 200$

Note: Here we have assumed linearity (scaling and additivity).

Step 3: Identify "best" criterion

It seems clear that profit should be maximised, that is the objective function is

$$\text{Maximise } 40X_1 + 20X_2 + 30X_3$$

Note: Here also we have assumed linearity (scaling and additivity).

Example 2: Inspection problem

A company employs two grades of quality control inspector to examine pieces being produced on a production line. A grade one inspector can inspect at the rate of 25 pieces per hour, with 98% accuracy and for this they are paid €16 per hour. Grade two inspectors inspect pieces at the slower rate of 15 per hour and with 95% accuracy, and they are paid €12 per hour. The company can call on up to eight grade one inspectors and up to 10 grade two inspectors, but inspectors who are not called upon do not have to be paid. Any errors which are made in the inspection process cost the company €8 each. The company requires that at least 1800 pieces must be inspected each day (8 hours). Formulate this as a linear programming problem.

Formulation:

Step 1: Identify decision variables (what decisions can be made?)

For this problem the company can decide on how many of each grade of inspector to call up. Therefore, the decision variables are, say,

Let X_1 = no. of grade 1 inspectors to call up per day (*appropriate time horizon here*)

Let X_2 = no. of grade 2 inspectors to call up per day

Step 2: Identify restrictions or constraints

We have, $0 \leq X_1 \leq 8$

$0 \le X_2 \le 10$

Also, there is the constraint,

No. of pieces (per day): $8*25X_1 + 8*15X_2 = 200X_1 + 120X_2 \ge 1800$

Step 3: Identify "best" criterion

Here the emphasis is on costs, which must be minimised. The costs are for labour and for (average) number of errors. Thus, per day,

Labour: Inspectors grade 1: $16X_1$ per hour or $128X_1$ per day

Inspectors grade 2: $12X_2$ per hour or $96X_2$ per day

Average no. errors: Inspectors grade 1: $0.02*200X_1 = 4X_1 \Rightarrow 32X_1$ cost

Inspectors grade 2: $0.05*120X_1 = 6X_1 \Rightarrow 48X_1$ cost

Therefore, the objective is to minimise the total cost per day, that is,

$$\text{Minimise } 160X_1 + 144X_2$$

Note: Here, we have assumed linearity (scaling and additivity) in establishing both the "No. of pieces" constraint and the objective function.

Note: This is really an integer LP problem as X_1 and X_2 must be whole numbers.

Example 3: A pair of related problems

Problem I: The following nutrient content and cost data are given for cereals A & B:

	Cereal A	**Cereal B**	**Mean Daily Req't-MDR**
Thiamin	0.20 (mg/oz)	0.10 (mg/oz)	1.0 (mg)
Iron	0.80 (mg/oz)	1.20 (mg/oz)	7.2 (mg)
Price/ounce	2	5/3	

The problem is to satisfy the MDR **at minimum cost** by using some Cereal A and some Cereal B.

Problem II: A salesman has Thiamin and Iron in pill form and would like the customer to take pills instead of cereal to satisfy the MDR. The salesman's objective is to maximise profit at a competitive price.

Formulation of the problems in a "mathematical model":

Problem I: We must introduce decision variables to state the problem mathematically. What we want to decide on is the (unknown) amounts of Cereal A and Cereal B so we start by letting

y_1 = number of ounces the customer should use of cereal A

y_2 = number of ounces the customer should use of cereal B

Then the total cost is going to be $2y_1 + 5/3y_2$ so that our objective is to

Minimise $z = 2y_1 + 5/3y_2$

The customer must also satisfy MDR for both Thiamin and Iron which translates to satisfying the constraints

$0.2y_1 + 0.1y_2 \geq 1$ or $2y_1 + y_2 \geq 10$ (Thiamin MDR)

$0.8y_1 + 1.2y_2 \geq 7.2$ or $8y_1 + 12y_2 \geq 72$ (Iron MDR)

Finally, we must have $y_1 \geq 0$ and $y_2 \geq 0$.

Problem II: In this case the salesman must decide on the price and this leads to introducing the decision variables

x_1 = price per mg of Thiamin

x_2 = price per mg of Iron

Then, assuming the customer buys no more than is required of each nutrient, the total price is going to be $1.0x_1 + 7.2x_2$ so that the objective is to

Maximise $w = x_1 + 7.2x_2$

In this case the constraints arise from needing to be "competitive" with the price for the nutrients taken in cereal form. For y_1 ounces of cereal A one gets $0.2y_1$ mg of Thiamin and $0.8y_1$ mg of Iron at a cost of $2y_1$. The corresponding cost of these amounts of nutrients in pill form is $(0.2y_1)x_1 + (0.8y_1)x_2$ and so we must have the "competitive" constraint

$(0.2y_1)x_1 + (0.8y_1)x_2 \leq 2y_1$ or $0.2x_1 + 0.8x_2 \leq 2$

Similarly, for Cereal B, we have

$(0.1y_2)x_1 + (1.2y_2)x_2 \leq 5/3y_2$ or $0.1x_1 + 1.2x_2 \leq 5/3$

For convenience, we can re-write the constraints as

$2x_1 + 8x_2 \leq 20$

$x_1 + 12x_2 \leq 50/3$

Finally, we also have $x_1 \geq 0$ and $x_2 \geq 0$.

Note: One problem is called the Primal & the Dual; the Dual of the Dual is the Primal.

1.1.3 Graphical solution of simple problems; Some Terminology

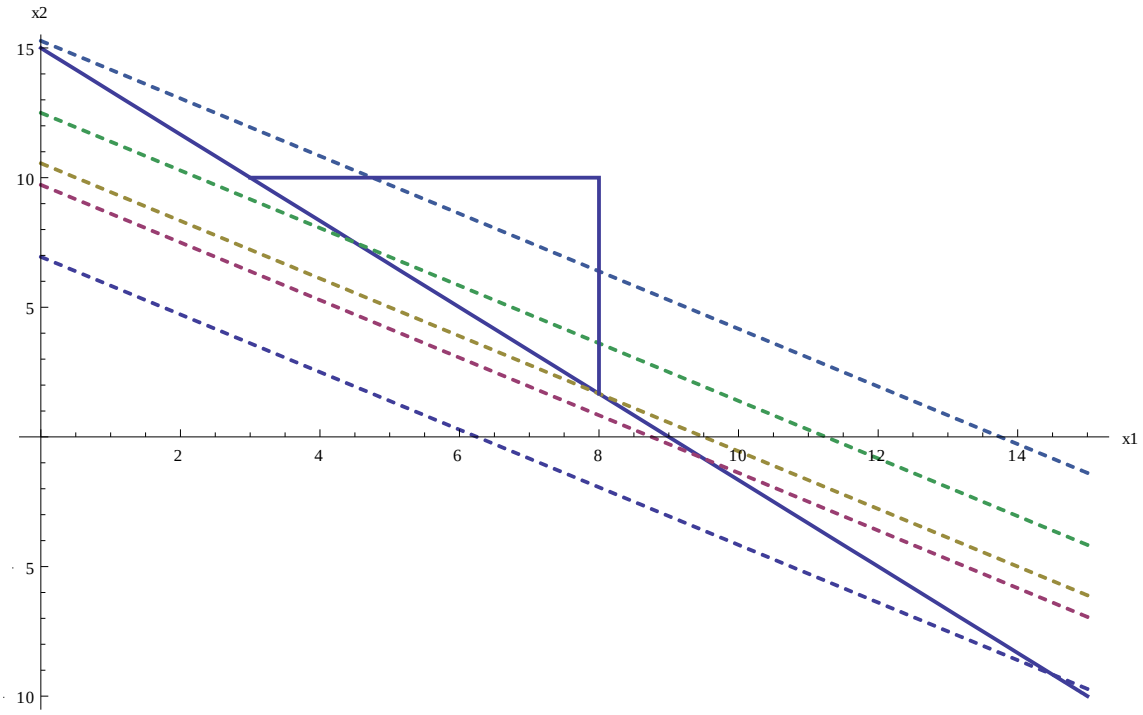
If a LP problem involves just two decision variables, as in Examples 2 and 3 of section 1.1.2, then it can be solved graphically.

Example 2: Minimise $160X_1 + 144X_2$

`subject to` $200X_1 + 120X_2 \geq 1800$

and $0 \leq X_1 \leq 8$

$0 \le X_2 \le 10$



In this figure, the solid sloped line is $200X_1 + 120X_2 = 1800$ and we know that the solution must lie on or above this line. The triangular region, bounded by this line and the lines $X_1 = 8$ and $X_2 = 10$, is the set of ***FEASIBLE*** solutions of the problem that is those solutions for which the constraints are satisfied.

The dashed lines, from higher to lower, are the lines $160X_1 + 144X_2 = 2200$, $160X_1 + 144X_2 = 1800$, $160X_1 + 144X_2 = 1520$, $160X_1 + 144X_2 = 1400$ and $160X_1 + 144X_2 = 1000$. These illustrate the behaviour of the objective function. Looking at the objective "Minimise $160X_1 + 144X_2$" we can see that the minimum occurs with value 1520 when $X_1 = 8$ and $X_2 = 5/3$.

Note: This means that 8 (the maximum) Grade 1 inspectors and 5/3 Grade 2 inspectors should be called up. *In practice, one might need to call up a whole number*.

Exercise on Example 3:

(a) Show that the corners of the feasible region for Problem I are (0,10), (3, 4) and (9,0) and that the minimum of z is 38/3.

(b) For Problem II show that the feasible region is bounded and that the maximum of w occurs at corner (20/3, 5/6) and that its value is (also) 38/3

Some terminology: A number of situations that may arise can be illustrated graphically, as follows.

(1) Alternative/Multiple Optimal Solutions

Consider the problem

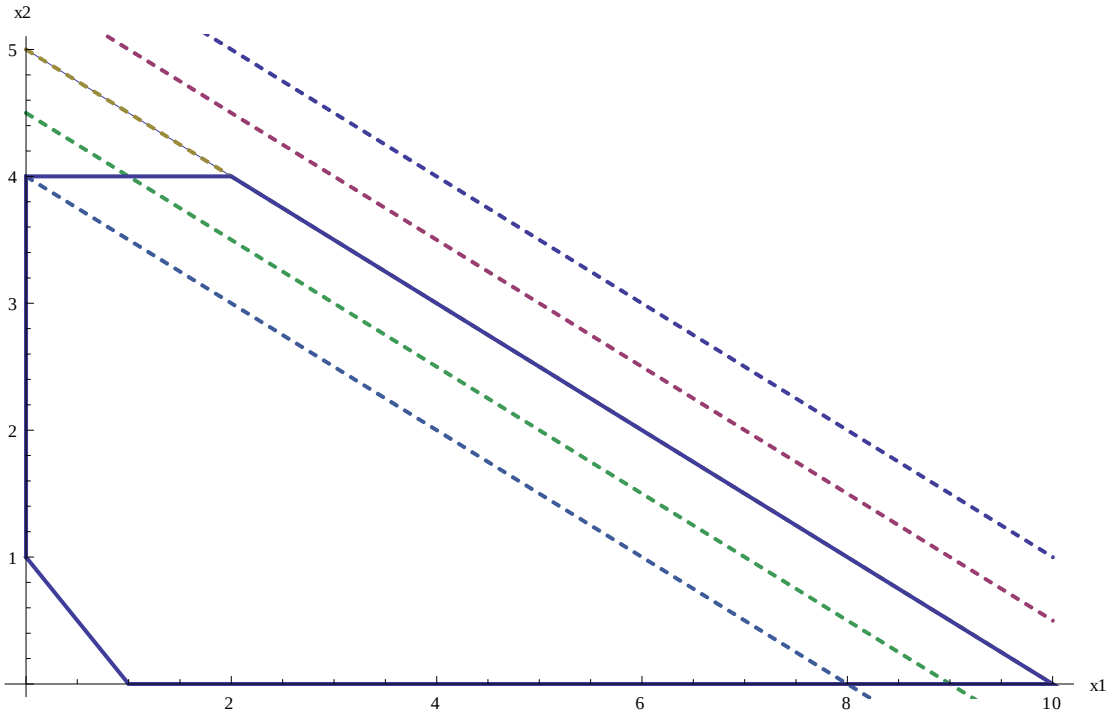
Maximise $X_1 + 2X_2$

subject to $2X_1 + 4X_2 \leq 20$

$X_1 + X_2 \geq 1$

$X_2 \leq 4$

and $X_1, X_2 \geq 0$



The feasible region is on the boundary and inside the polygon. The dashed lines indicate the behaviour of the objective function. As $X_1 + 2X_2$ = Constant and $2X_1 + 4X_2 = 20$ are parallel, the maximum is at any point on the line joining (2,4) to (10,0) and has value 10.

(2) Unbounded Solutions

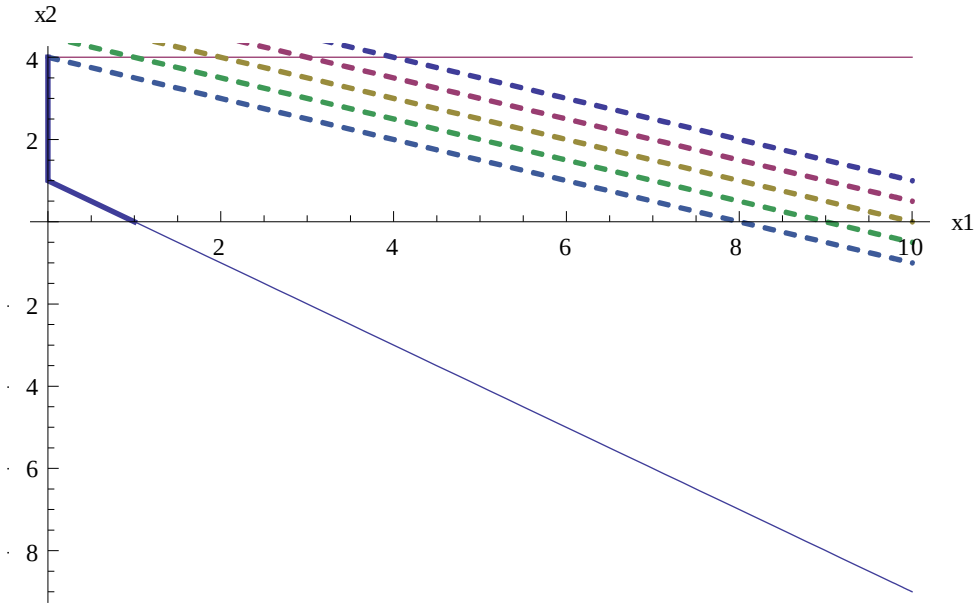
Consider the problem (same as previous except one constraint is omitted)

Maximise $X_1 + 2X_2$

subject to $X_1 + X_2 \geq 1$

$X_2 \leq 4$

and $X_1, X_2 \geq 0$



The feasible region is clearly unbounded (which might mean an error in the formulation!).

(3) Infeasible Problem

Consider the problem (similar to previous but with two inequality signs reversed)

Maximise $X_1 + 2X_2$

subject to $X_1 + X_2 \leq 1$

$X_2 \geq 4$

and $X_1, X_2 \geq 0$

The diagram of (2) can be referred to except that in this case a solution must both lie in the small triangular region near the origin and above the line $X_2 = 4$, which is impossible. Probably means an error in the formulation.

1.1.4 Layout of the remainder of Section 1

Section 1 (Linear Programming) is made up of this introductory section followed by four sections

1.2 Simplex Method

1.3 Large Scale problems – Tool usage (***ampl***)

1.4 Multiple Criteria Decision Making, including Goal Programming

1.5 Duality and Dual Simplex Method

The Simplex Method is the classical method for solving LP problems and we will define it and use it to solve small scale problems. As part of this we will introduce various concepts and terminologies which have become an accepted part of the treatment of LP problems.

In practice, the real difficulties in LP are often in formulation rather than in solving as there are various tools available for solving, which is essentially part of Numerical Analysis. We will introduce the ***ampl*** software tool (Fourer, et al., 2003) and apply it to various classes of problems.

We present Multiple Criteria Decision Making (with Goal Programming), including relevant ***ampl*** facilities. Finally, we present the Dual Simplex method and a more thorough (than section 1.2.10) treatment of sensitivity analysis.

1.1.5 Some references

There are various books on Operations Research in general, and on Linear Programming in particular in DCU library or available on-line. The books by Taha and Luenberger listed below are well known, especially.

The following are all in the DCU library: (Chvatal, 1983), (Williams, 2013), (Applegate, et al., 2007), (Hillier & Lieberman, 1974), (Papadimitriou & Steiglitz, 1982), (Larson & Odoni, 1981), (Rader, 2010), (Schrage, 2000), (Senge, 2006), (Bueno de Mesquita, 2009), (Minieka, 1978), (Taha, 2007), (Luenberger, 1984), (Woolsey, 2003) and (Fourer, et al., 2003).

Some or all of the following appear to be downloadable for free; however, be careful not to violate any copyright laws if you make use of them:

(Hastie, et al., 2008) (*"If, by OR, you also want to include modern-day and the future that is data-driven analytics, then [this] is a must .."*), (Bradley, et al., 1977), and (Bazaraa, et al., 2010).

This list is intended to give a flavour of what is "out there". Some of the books are straightforward technical books while some are more for the general reader or as interesting background (e.g. (Schrage, 2000) and (Bueno de Mesquita, 2009)).

(Taha, 2007) is a good general book. It is recommended especially to go to the web-site associated with the ***AMPL*** book (Fourer, et al., 2003) – note that one of the authors is the same as for the well known Kernighan & Ritchie book on C programming.

Finally, be aware that there is a lot of useful material on-line.

1.2 Simplex method

1.2.1 A fundamental property of linear programming problems

The optimum solution of a Linear Programming problem always occurs at a corner (or vertex) of the feasible region. This may be proved mathematically but we do not do this in CA4011.

We will see that the Simplex Method is a procedure that starts at a known vertex of the feasible region (called a basic feasible solution – see below) and then proceeds step by step from one vertex to another making sure that the objective function value is improved at each step. The procedure stops when no further improvement is possible

1.2.2 Standard form of linear programming problem

Representation of a problem with *m* constraints and *n* variables:

Maximise (or Minimise) $Z = c_1X_1 + c_2X_2 + \ldots\ldots\ldots\ldots + c_nX_n$

subject to

$$a_{11}X_1 + a_{12}X_2 + \ldots\ldots\ldots\ldots + a_{1n}X_n = b_1$$
$$a_{21}X_1 + a_{22}X_2 + \ldots\ldots\ldots\ldots + a_{2n}X_n = b_2$$
$$\ldots$$
$$\ldots$$
$$a_{m1}X_1 + a_{m2}X_2 + \ldots\ldots\ldots\ldots + a_{mn}X_n = b_m$$

and

$$X_1 , X_2 , X_3 , \ldots\ldots\ldots\ldots X_n \geq 0$$
$$b_1 , b_2 , b_3 , \ldots\ldots\ldots\ldots b_m \geq 0 \textbf{ [Note this]}$$

Note:

1. Objective function is maximise or minimise.
2. Constraints are expressed as equations
3. All variables are ≥ 0.
4. All right-hand-side (RHS) constants are ≥ 0.

Sometimes vector-matrix notation is used:

Max (Min) $Z = \underline{c}\,\underline{x}$

subject to $\underline{A}\,\underline{x} = \underline{b}$

$\underline{x} \geq \underline{0}$

$\underline{b} \geq \underline{0}$

$\underline{A}$ (m x n) coefficient matrix

$\underline{x}$ (n x 1) decision vector

$\underline{b}$ (m x 1) requirement vector

$\underline{c}$ (1 x n) cost (profit) vector

Usually, a formulated problem is not in standard firm so that it is necessary to transform it. In particular, inequality constraints must be transformed to equations and unsigned decision variables must be replaced by signed ones.

Handling Inequality Constraints

Inequalities need to be converted into equations by introducing **slack** or **surplus** variables.

e.g. $X_1 \leq 8$ becomes $X_1 + X_3 = 8$

where $X_3 \geq 0$ *is a* Slack variable

e.g. $200X_1 + 120X_2 \geq 1800$

becomes $200X_1 + 120X_2 - X_4 = 1800$

where $X_4 \geq 0$ *is a* Surplus variable

Handling Variables with Unrestricted Sign

Example: an investor has €100 cash and can borrow to invest.

Let X_1 = amount invested

Investment constraint: $X_1 + X_2 = 100$

where X_2 can be positive or negative i.e. is unrestricted

Replace X_2 by $(X_2' - X_2'')$

where $X_2' \geq 0,\ X_2'' \geq 0$

At solution either $X_2' > X_2''$

Or $X_2' < X_2''$

Note: When a LP problem is in standard form then we can carry out such operations (called "elementary row operations) as dividing an equation by a constant and subtracting an equation from another one.

1.2.3 Recap on Linear Equations; Canonical form

Consider the pair of equations

$$X_1 - 2X_2 + X_3 - 4X_4 = 2 \quad (1)$$
$$X_1 - X_2 - X_3 - 3X_4 = 4 \quad (2)$$

S_1

There are 4 unknowns and 2 equations which means (normally) that there is more than one solution. (For example, see our previous examples showing feasible solution regions or sets).

We can transform (S_1) to an equivalent equation pair using the elementary row operations (pivoting)

- Multiply an equation by a number
- Add to any equation a constant multiple of any other equation

Note: This process will be referred to as **pivoting** in the Simplex Method.

For example, multiply (1) by -1 and add it to (2) to get

$$X_1 - 2X_2 + X_3 - 4X_4 = 2 \quad (3)$$
$$X_2 - 2X_3 + X_4 = 2 \quad (4)$$

S_2

Next, multiply (4) by 2 and add to (3) to get

$$X_1 - 3X_3 - 2X_4 = 6 \quad (5)$$
$$X_2 - 2X_3 + X_4 = 2 \quad (6)$$

S_3

Note that the use of elementary row operations guarantees that S_3 has the same solution set as S_1.

The form of S_3 allows us to easily write down a possible solution, namely

$X_1 = 6, X_2 = 2, X_3 = 0, X_4 = 0$

This is called a **basic feasible** solution where

X_1, X_2 - Basic variables

X_3 , X_4 - Non-basic variables

S_3 is said to be a canonical system or to be in canonical form.

We will see that to apply the Simplex Method a system must first be in canonical form.

Example: In Example 1 of section 1.1.2 we had the inequalities,

Labour (per day): $7X_1 + 3X_2 + 6X_3 \leq 150$

Materials (per day): $4X_1 + 4X_2 + 5X_3 \leq 200$

(A) Transform to **standard form** using slack variables $X_4 \geq 0$ and $X_5 \geq 0$,

Labour (per day): $7X_1 + 3X_2 + 6X_3 + X_4 = 150$ (1)

Materials (per day): $4X_1 + 4X_2 + 5X_3 + X_5 = 200$ (2)

In fact this is also in **canonical form** with basic variables X_4 and X_5. We can write down a solution immediately, namely

$$X_4 = 150, X_5 = 200$$

$$X_1 = X_2 = X_3 = 0$$

This would mean simply that that we made no units of any appliance and therefore used no labour or materials.

(B) Suppose now we first divide (1) by 7 to get

Labour (per day): $X_1 + (3/7)X_2 + (6/7)X_3 + (1/7)X_4 = 150/7$ (1')

Materials (per day): $4X_1 + 4X_2 + 5X_3 + X_5 = 200$ (2)

And then subtract 4 times (1') from (2) to get

Labour (per day): $X_1 + (3/7)X_2 + (6/7)X_3 + (1/7)X_4 = 150/7$ (1')

Materials (per day): $(16/7)X_2 + (11/7)X_3 - (4/7)X_4 - + X_5 = 800/7$ (2')

This is also in canonical form with basic variables X_1 and X_5. We can write down a solution immediately, namely

$$X_1 = 150/7, X_5 = 800/7$$

$$X_4 = X_2 = X_3 = 0$$

This would mean that that we made 150/7 units of appliance A but none of appliances B or C. Also, we have used all the labour (no slack) but have not used 800/7 units of material. We can see that the objective function "$40X_1 + 20X_2 + 30X_3$" has the value $40(150/7) = 600/7$ which is greater than the zero value of the first basic solution.

Note: We can proceed in the same way to find other basic solutions but it would really be guesswork as to which would maximise the objective function. The Simplex Method provides a systematic approach.

Note: A point of detail to note – the Simplex Method is such that the right hand side of any equation never becomes negative.

1.2.4 Simplex Method: A first example

In brief:

- **Requires initial basic feasible solution (B.F.S.)**
- **Improve by finding another B.F.S. with better objective function value**
- **Continue until no improvement possible**

Our first LPP has been formulated as

Max $Z = 3X_1 + 2X_2$

subject to $2X_1 + X_2 \leq 6$

$X_1 + 2X_2 \leq 8$

$X_1 , X_2 \geq 0$

The first step is to express it in standard form, by introducing slack variables:

Max $Z = 3X_1 + 2X_2$

subject to $2X_1 + X_2 + S_1 = 6$

$X_1 + 2X_2 + S_2 = 8$

$X_1 , X_2 , S_1 , S_2 \geq 0$

S_1 , S_2 - slack variables

Next we would need to express the problem in canonical form. However, in this example, the above standard form is also canonical, having the basic feasible solution (B.F.S.)

$S_1 = 6$ $S_2 = 8$ $X_1 = 0$ $X_2 = 0$ where S_1 , S_2 are the basic variables

The next step is to form an initial Simplex Tableau, which we lay out as

	X_1	X_2	S_1	S_2	**b**
Z					
S_1					
S_2					

We fill in the given data as follows,

	X_1	X_2	S_1	S_2	**b**
Z	-3	-2	0	0	0
S_1	**2**	1	1	0	6
S_2	1	2	0	1	8

Next, we look for another B.F.S. (if there is one) that has a higher Z value (currently it has value 0). There are two elements to this namely (a) select a new variable to **enter** the basis and (b) select an existing basic variable to **leave** the basis.

(a) **Entering variable**

Method is to scan the Z row for negative entries (for a maximisation problem) and choose the largest (in absolute value). For our problem this means to make X_1 basic.

(b) **Leaving variable (Minimum ratio test)**

Method is to take the ratio of the entering variable's values (but considering only non-negative ones) and the corresponding **b** values, and to select to leave the corresponding current basic variable. For our problem, this test leads to

Min (6/2, 8/1) = 6/2 and the corresponding basic variable is S_1.

Thus,

X_1 replaces S_1 as a basic variable

Note that the highlighted entry in the above tableau is called the **pivot** element.

Next we must calculate a new Simplex Tableau, using elementary operations, so that X_1 is basic:

We show the individual steps in this:

	X_1	X_2	S_1	S_2	**b**
Z	-3	-2	0	0	0
X_1	1	1/2	1/2	0	3
S_2	1	2	0	1	8

Next get a 0 in the Z row of X_1:

	X_1	X_2	S_1	S_2	**b**
Z	0	-1/2	3/2	0	9
X_1	1	1/2	1/2	0	3
S_2	1	2	0	1	8

Finally, get a 0 in the Z row of X_1:

	X_1	X_2	S_1	S_2	**b**
Z	0	-1/2	3/2	0	9
X_1	1	1/2	1/2	0	3
S_2	0	**3/2**	-1/2	1	5

This is our second tableau with B.F.S. $X_1 = 3$, $S_2 = 5$, $X_2 = 0$, $S_1 = 0$

Next, we look for another B.F.S. (if there is one) that has a higher Z value (currently it has value 0). There are the two elements to this namely (a) select a new variable to **enter** the basis and (b) select an existing basic variable to **leave** the basis.

(a) **Entering variable**

Method is to scan the Z row for negative entries (for a maximisation problem) and choose the largest (in absolute value). For our problem this means to make X_2 basic.

(b) **Leaving variable (Minimum ratio test)**

Method is to take the ratio of the entering variable's values (but considering only non-negative ones) and the corresponding **b** values, and to select to leave the corresponding current basic variable. For our problem, this test leads to

Min (3/(1/2), 5/(3/2)) = 10/3 and the corresponding basic variable is S_2.

Thus,

X_2 replaces S_2 as a basic variable

Note that the highlighted entry in the above tableau is again the **pivot** element.

	X_1	X_2	S_1	S_2	**b**
Z	0	0	4/3	1/3	32/3
X_1	1	0	2/3	-1/3	4/3
X_2	0	1	-1/3	2/3	10/3

There are no negatives in *Z* row, so solution is optimal - * added to emphasise:

$X_1^* = 4/3; \quad X_2^* = 10/3; \quad Z^* = 32/3$

1.2.5 Summary of the Simplex Method algorithm

(**Maximisation with ≤ constraints**)

Step 0: Use standard form to obtain initial B.F.S. (as this also gives canonical form)

Step 1: Select variable to enter basis

(largest negative entry in z-row)

If none, STOP

Step 2: Select variable to leave basis

(minimum ratio test)

Step 3: Calculate new B.F.S.

(pivot operations)

Go to step 1.

Minimisation:- Modify step 1 → Select variable with largest positive entry.

Note: However, we will see there are other issues that may arise particularly that we may not have a canonical form to begin with.

1.2.6 Simplex Method: Computational problems that may arise

- Ties in selection of non-basic variable to enter basis. Resolve by either of
  - arbitrary choice
  - minimum subscript rule

- Ties in ratio rule (to decide on current basic variable to leave the basis). This situation arises when there is a degenerate solution meaning that one or more basic variable takes on a value of zero.

Example: Suppose we have the simplex tableau

	X_1	X_2	X_3	X_4	X_5	X_6	**b**
Z	0	0	0	-2	0	-1	4
X_1	1	0	0	**1**	-1	0	2
X_2	0	1	0	2	0	1	4
X_3	0	0	1	1	1	1	3

The current solution is $X_1 = 2$, $X_2 = 4$, $X_3 = 3$, $X_4 = X_5 = X_6 = 0$ with $Z = 4$.

(a) **Entering variable**

Method is to scan the Z row for negative entries (for a maximisation problem) and choose the largest (in absolute value). For our problem this means to make X_4 basic.

(b) **Leaving variable (Minimum ratio test)**

Method is to take the ratio of the entering variable's values (but considering only non-negative ones) and the corresponding **b** values, and to select to leave the current basic variable corresponding to the minimum ratio. For our problem, this test leads to

Min (2/1, 4/2, 3/1) = 2 **but** there are two corresponding basic variable (X_1., X_2).

Suppose we arbitrarily choose X_1 as the leaving variable => get new tableau

	X_1	X_2	X_3	X_4	X_5	X_6	b
Z	2	0	0	0	-2	-1	8
X_4	1	0	0	1	-1	0	2
X_2	-2	1	0	0	**2**	1	0
X_3	-1	0	1	0	2	1	1

The new B.F.S. is $X_4 = 2$, $X_2 = 0$, $X_3 = 1$, $X_1 = X_5 = X_6 = 0$ with $Z = 8$.

(a) **Entering variable:** Usual rule => X_5 enters

(b) **Leaving variable (Min. ratio test)** leads to Min (-, 0/2, 1/2) = 0 => X_2 leaves

Then, the next tableau is

	X_1	X_2	X_3	X_4	X_5	X_6	b
Z	0	1	0	0	0	0	8
X_4	0	1/2	0	1	0	1/2	2
X_5	-1	1/2	0	0	**1**	1/2	0
X_3	1	-1	1	0	0	0	1

The new B.F.S. is $X_4 = 2$, $X_5 = 0$, $X_3 = 1$, $X_1 = X_2 = X_6 = 0$ with $Z = 8$.

So, we see that the value of the objective function does not improve.

Normally degeneracy rectifies itself. However, if there is a re-appearance of a previous b.f.s. then Cycling has occurred (rare in practice).

• If all coefficients in column of entering variable are negative or zero (i.e. minimum ratio test fails) => solution is unbounded.

Example (maximisation): Suppose we have the tableau

	X_1	X_2	S_1	S_2	b
Z	-2	-3	0	0	0
S_1	1	-1	1	0	10
S_2	2	0	0	1	40

The current B.F.S. is $S_1 = 10$, $S_2 = 40$, $X_1 = X_2 = 0$ with $Z = 0$.

(a) **Entering variable:** Usual rule => X_2 enters

(b) **Leaving variable (Min. ratio test)** fails as the coefficients (-1 and 0) of entering variable X2 are negative and zero, respectively.

• Zero entry for non-basic variable in Z row => an alternative solution exists.

Example (maximisation): Say we have the tableau

	X_1	X_2	S_1	S_2	S_3	**b**
Z	0	0	0	1	0	14
S_1	0	0	1	-1/5	8/5	6
X_2	0	1	0	1/5	-3/5	1
X_1	1	0	0	1/5	2/5	4

The current B.F.S. is $S_1 = 6$, $X_2 = 1$, $X_1 = 4$, $S_2 = = S_3 = 0$ with $Z = 14$

Normally, as there are no negative entries in the Z row we would conclude that we have reached the optimal B.F.S..
However, because it has a zero entry in the Z row we could make S_3 basic without changing value of Z and this would give an alternative alternative optimum.

Thus if S_3 enters then by the min. ratio rule min{6/(8/5), -, 4/(2/5)} = 15/4 S_1 leaves.

This leads to the new tableau

	X_1	X_2	S_1	S_2	S_3	**b**
Z	0	0	0	1	0	14
S_3	0	0	5/8	-1/8	1	15/4
X_2	0	1	3/8	1/8	0	13/4
X_1	1	0	-1/4	1/4	0	5/2

This gives [***check arithmetic!***] the alternative solution $X_1 = 5/2 \quad X_2 = 13/4$
$S_1 = 0 \quad S_2 = 0 \quad S_3 = 15/4$
(with unchanged Z value).

- If a negative value appears in **b** column, then the problem is infeasible.

1.2.7 Artificial variable technique (to "invent" basic variable(s))

Summary of rationale for "artificial" variables:

- Needed when there is no obvious starting solution.
- If no basic variable for an equation, then 'invent' an "artificial" one.
- Values of artificial variables are driven to zero by the simplex method.

Example:

Min $Z = 4X_1 + X_2$

s.t. $3X_1 + X_2 = 3$

$4X_1 + 3X_2 \geq 6$

$X_1 + 2X_2 \leq 4$

$X_1, X_2 \geq 0$

Standard form (by intoducing a surplus variable S_1 and a slack variable S_2):

Min $Z = 4X_1 + X_2 + 0.S_1 + 0.S_2$

s.t. $3X_1 + X_2 = 3$

$4X_1 + 3X_2 - S_1 = 6$

$X_1 + 2X_2 + S_2 = 4$

$X_1, X_2, S_1, S_2 \geq 0$

However, unlike our first example, this is not in canonical form as there are no basic variables for the first and second equations.

The artificial variable technique is

(A) to augment these equations with artificial variables R_1 and R_2 (*the choice of variable name is unimportant*) as follows:

$$3X_1 + X_2 + R_1 = 3$$
$$4X_1 + 3X_2 - S_1 + R_2 = 6$$

Note: *Unlike slack and surplus variables, artificial variables have no meaning for the original problem and so need to be eliminated (that is, driven to become zero, once they serve their purpose of getting the Simplex Method started*.

So, the artificial variable technique also

(B) requires the objective function to be modified so the values of R_1 and R_2 are driven to zero (i.e. eliminated from the basis) as the Simplex Method proceeds. To do this, the artificial variables are "penalised" in the objective function as follows:

Min $Z = 4X_1 + X_2 + 0.S_1 + 0.S_2 + MR_1 + MR_2$, where $M > 0$ is some very large constant. This means that as long as R_1 and/or R_2 remain in the basis the value of Z will be arbitrarily large.

The complete set of constraints is

s.t. $3X_1 + X_2 + R_1 = 3$

$4X_1 + 3X_2 - S_1 + R_2 = 6$

$X_1 + 2X_2 + S_2 = 4$

$X_1, X_2, S_1, S_2, R_1, R_2 \geq 0$

Note: We can see that an initial solution (to this modified problem) is $R_1 = 3$, $R_2 = 6$, $S_2 = 4$ *and* $X_1 = X_2 = S_1 = 0$. *The corresponding value of Z is 9M.*

Next, the method proceeds by

(a) A preliminary tableau modification to yield an initial Simplex Tableau

(b) Normal Simplex Method – usually artificial variables will be eliminated and when they are all eliminated we will have a B.F.S. for the original problem.

The **preliminary** tableau is

	X_1	X_2	S_1	R_1	R_2	S_2	***b***
Z	-4	-1	0	-M	-M	0	0
R_1	3	1	0	1	0	0	3
R_2	4	3	-1	0	1	0	6
S_2	1	2	0	0	0	1	4

Remember that basic variables must have zeros in all columns except for one, including in the Z row. Therefore, we must eliminate M's from R_1 and R_2 columns by pivot operations.

This results in the initial tableau (now in canonical form) for the Simplex Method:

	X_1	X_2	S_1	R_1	R_2	S_2	b
Z	-4 + 7M	-1 + 4M	- M	0	0	0	9M
R_1	3	1	0	1	0	0	3
R_2	4	3	-1	0	1	0	6
S_2	1	2	0	0	0	1	4

Note that this is a minimisation problem so the entering variable is the one with the largest positive entry in the Z row. As M is arbitrarily large and positive, the entering variable is the one with the largest positive coefficient of M.

First iteration (X_1 enters and R_1 leaves, by Min. Ratio rule):

	X_1	X_2	S_1	R_1	R_2	S_2	***b***
Z	0	$\frac{(1 + 5M)}{3}$	- M	$\frac{(4 - 7M)}{3}$	0	0	4 + 2M
X_1	1	1/3	0	1/3	0	0	1
R_2	0	5/3	-1	-4/3	1	0	2
S_2	0	5/3	0	-1/3	0	1	3

Note: As soon as R1 leaves we could remove it from our calculations but we retained it here for explanatory purposes.

Second iteration (X_2 enters (the only candidate!) and R_2 leaves, by Min. Ratio rule):

	X_1	X_2	S_1	R_1	R_2	S_2	***b***
Z	0	0	1/5	8/5 - M	-1/5 – M	0	18 / 5
X_1	1	0	1/5	3/5	-1/5	0	3/5
X_2	0	1	-3/5	-4/5	3/5	0	6/5
S_2	0	0	1	1	-1	1	1

The B.F.S. is $X_1 = 3/5$, $X_2 = 6/5$ and $S_2 = 1$, $S_2 = 0$ (and $R_1 = R_2 = 0$) and $Z = 18/5$.

Note: *This is a feasible solution of the original problem and is, essentially, the initial tableau for that problem*.

Third iteration (S_1 enters (the only candidate!) and S_2 leaves, by Min. Ratio rule):

	X_1	X_2	S_1	R_1	R_2	S_2	b
Z	0	0	0	7/5 - M	– M	-1/5	17 / 5
X_1	1	0	0	2/5	0	-1/5	2/5
X_2	0	1	0	-1/5	0	3/5	9/5
S_1	0	0	1	1	-1	1	1

This is the optimum B.F.S. ($X_1 = 2/5$, $X_2 = 9/5$ and $S_1 = 1$, $S_2 = 0$) and $Z = 17/5$.

We can see also that $R_1 = R_2 = 0$ but these are no longer relevant as their only purpose was in getting the method started.

Note: If artificial variables remain in the basis then the original problem is infeasible.

1.2.8 Some worked examples

Example 1: A company involved in the assembly and distribution of printers in concerned with two types – laser and inkjet. Assembly of each laser printer takes two hours, while each inkjet printer takes one hour to assemble, and the staff can provide a total of 40 person-hours of assembly time per day. In addition, warehouse space must be available for the assembly and distribution of the printers, 1 square metre for each laser printer and 3 square metres for each inkjet printer; the company has a total of 45 square metres of storage space available for assembled printers each day. Laser printers can be sold for a profit of €30 per unit and inkjet printers earn a profit of €25 each, but the market in which the company is operating can absorb a maximum of 12 laser printers per day. (There is no such limitation on the market for inkjet printers).

Formulate this as a linear programming problem and determine, using the simplex method, the number of each type of printer the company should assemble and distribute in order to maximise daily profit.

Solution: Let X_1 = number of laser printers assembled
X_2 = number of inkjet printers assembled

Maximise $30X_1 + 25X_2$
Subject to $2X_1 + X_2 <= 40$
$X_1 + 3X_2 <= 45$
$X_1 <= 12$
$X_1 , X_2 >= 0$

Standard form:

Maximise $30X_1 + 25X_2$
Subject to $2X_1 + X_2 + S_1 = 40$
$X_1 + 3X_2 + S_2 = 45$
$X_1 + S_3 = 12$
$X_1 , X_2 , S_1 , S_2 , S_3 >= 0$

	X_1	X_2	S_1	S_2	S_3	**b**
Z	-30	-25	0	0	0	0
S_1	2	1	1	0	0	40
S_2	1	3	0	1	0	45
S_3	1	0	0	0	1	12

	X_1	X_2	S_1	S_2	S_3	**b**
Z	0	-25	0	0	30	360
S_1	0	1	1	0	-2	16
S_2	0	3	0	1	-1	33
X_1	1	0	0	0	1	12

	X_1	X_2	S_1	S_2	S_3	**b**
Z	0	0	0	25/3	20	635
S_1	0	0	1	-1/3	-5/3	5
X_2	0	1	0	1/3	-1/3	11
X_1	1	0	0	0	1	12

Optimal tableau. Solution $X_1^* = 12$(# of laser printers), $X_2^* = 11$ (# of inkjet printers).
Total daily profit = €635

Note: You should check that you can do all the intermediate calculations.

Example 2: (Big M method – Use of artificial variables)

Note: This involves just 2 variables so you could solve it graphically.

Maximise $3X_1 + 4X_2$

Subject to $2X_1 + 3X_2 <= 600$
$X_1 + X_2 <= 225$
$5X_1 + 4X_2 <= 1000$
$X_1 + 2X_2 >= 150$

$X_1, X_2 >= 0$

Solution:

Standard form:

Maximise $3X_1 + 4X_2$

Subject to $2X_1 + 3X_2 + S_1 = 600$
$X_1 + X_2 + S_2 = 225$
$5X_1 + 4X_2 + S_3 = 1000$
$X_1 + 2X_2 - S_4 = 150$

$X_1, X_2, S_1, S_2, S_3, S_4 >= 0$

Not in canonical form because there is no basic variable in the fourth equation. Therefore we add an artificial variable to that equation (R_1) and give it a large **negative** coefficient (*because a maximisation problem*) in the objective function, to penalise it:

Maximise $3X_1 + 4X_2 - MR_1$

Subject to $2X_1 + 3X_2 + S_1 = 600$
$X_1 + X_2 + S_2 = 225$
$5X_1 + 4X_2 + S_3 = 1000$
$X_1 + 2X_2 - S_4 + R_1 = 150$

$X_1, X_2, S_1, S_2, S_3, S_4, R_1 >= 0$

	X_1	X_2	S_4	S_1	S_2	S_3	R_1	b
Z	-3	-4	0	0	0	0	+M	
S_1	2	3	0	1	0	0	0	600
S_2	1	1	0	0	1	0	0	225
S_3	5	4	0	0	0	1	0	1000
R_1	1	2	-1	0	0	0	1	150

Not yet in Canonical form because of +M entry on Z row for one basic variable (R_1). Pivot to replace +M on Z row by zero - Z row – M*R_1 row:

	X_1	X_2	S_4	S_1	S_2	S_3	R_1	b
Z	(-3-M)	(-4-2M)	M	0	0	0	0	-150M

S_1	2	3	0	1	0	0	0	600
S_2	1	1	0	0	1	0	0	225
S_3	5	4	0	0	0	1	0	1000
R_1	1	2	-1	0	0	0	1	150

Now start the Simplex Method "proper":

	X_1	X_2	S_4	S_1	S_2	S_3	R_1	b
Z	-1	0	-2	0	0	0	M	300
S_1	½	0	3/2	1	0	0	-3/2	375
S_2	½	0	½	0	1	0	-½	150
S_3	3	0	2	0	0	1	-2	700
X_2	½	1	-½	0	0	0	½	75

	X_1	X_2	S_4	S_1	S_2	S_3	R_1	b
Z	-1/3	0	0	4/3	0	0	M	800
S_4	1/3	0	1	2/3	0	0	-1	250
S_2	1/3	0	0	-1/3	1	0	0	25
S_3	7/3	0	0	-4/3	0	1	0	200
X_2	2/3	1	0	1/3	0	0	0	200

	X_1	X_2	S_4	S_1	S_2	S_3	R_1	b
Z	0	0	0	1	1	0	M	825
S_4	0	0	1	1	-1	0	-1	225
X_1	1	0	0	-1	3	0	0	75
S_3	0	0	0	1	-7	1	0	25
X_2	0	1	0	1	-2	0	0	250

Optimal tableau: Solution: $X_1^* = 75$ $X_2^* = 250$ $Z^* = 825$

Example 3: (*Exercise to carry out the Simplex Method calculations!*)

A machine tool company conducts a job-training program for machinists. Trained machinists are used as teachers in the program at a ratio of one for every ten trainees. The training program lasts for one month. From past experience it has been found that out of ten trainees hired, only seven complete the program successfully (the unsuccessful trainees are released).

Trained machinists are also needed for machining and the company's requirements for the next three months are as follows:

January	100
February	150
March	200

In addition, the company requires 250 trained machinists by April. There are 130 trained machinists available at the beginning of the year.

Payroll costs per month are:

Each trainee	€400
Each trained machinist (machining or teaching)	€700
Each trained machinist idle (Union forbids firing them!)	€500

Set up the linear programming problem (LPP) that will produce the minimum cost hiring and training schedule and meet the company's requirements. Then solve the LPP using the Simplex Method.

Solution:

Each trained machinist can:

(i) work a machine
(ii) teach
(iii) stay idle

The number of trained machinists is fixed; therefore the decision variables are the number teaching each month and the number idle each month.

Let X_1 = number of trained machinists teaching in January
X_2 = number of trained machinists idle in January
X_3 = number of trained machinists teaching in February
X_4 = number of trained machinists idle in February
X_5 = number of trained machinists teaching in March
X_6 = number of trained machinists idle in March

Each month the total number of trained machinists = number machining + number teaching + number idle.

January: $100 + X_1 + X_2 = 130 \rightarrow X_1 + X_2 = 30$

February: $150 + X_3 + X_4 = 130 + 7X_1 \rightarrow 7X_1 - X_3 - X_4 = 20$
(Note that each machinists teaching trains 10 with 70% success)

March: $200 + X_5 + X_6 = 130 + 7X_1 + 7X_3 \rightarrow 7X_1 + 7X_3 - X_5 - X_6 = 70$

April: 250 trained machinists required in April:
$130 + 7X_1 + 7X_3 + 7X_5 = 250 \rightarrow 7X_1 + 7X_3 + 7X_5 = 120$

The cost of machinists operating is constant and therefore not required in the objective function.

Relevant costs are:

(i) cost of training program i.e. cost of trainees + cost of teachers
(ii) cost of idle machinists

Objective function (cost minimisation):

Minimise Z = 400 (10 X_1 + 10X_3 + 10X_5) (trainees)
+ 700 (X_1 + X_3 + X_5) (teachers)
+ 500 (X_2 + X_4 + X_6) (idle machinists)

Complete problem:

Minimise Z = 4700X_1 + 500 X_2 + 4700X_3 + 500 X_4 + 4700X_5 + 500X_6

Subject to $X_1 + X_2 = 30$
$7X_1 – X_3 – X_4 = 20$
$7X_1 + 7X_3 – X_5 – X_6 = 70$
$7X_1 + 7X_3 + 7X_5 = 120$

$X_1 , X_2 , X_3, X_4 ,X_5 , X_6 >= 0$

1.2.9 Duality in linear programming

- Associated with every L.P. there is another L.P. - The Dual
- Simplex method gives the solution to both problems
- Used in sensitivity analysis

We consider the following pair of Standard problems:

Primal

$$\text{Maximise } Z = \sum_{j=1}^{n} c_j X_j = (c_1, \dots, c_n) \begin{pmatrix} X_1 \\ \cdot \\ \cdot \\ \cdot \\ X_n \end{pmatrix}$$

Subject to

$$\sum_{j=1}^{n} a_{ij} X_j \leq b_i \text{ for } i = 1 \text{ to } m \textbf{ or } \boldsymbol{A_{mxn} X_{nx1} \leq b_{mx1}}$$

and

$$X_j \geq 0 \text{ for } j = 1 \text{ to n}$$

Dual

$$\text{Minimise } W = \sum_{i=1}^{m} Y_i b_i = (Y_1, \dots, Y_m) \begin{pmatrix} b_1 \\ \cdot \\ \cdot \\ \cdot \\ b_m \end{pmatrix}$$

Subject to

$$\sum_{i=1}^{m} Y_i a_{ij} \geq c_j \text{ for } j = 1 \text{ to } n \textbf{ or } \boldsymbol{Y_{1xm} A_{mxn} \geq c_{1xn}}$$

and

$Y_i \geq 0$ for $i = 1$ to m

A represents the "per unit resource use", **b** represents the "resources on hand" and **c** represents the "per unit profits". **X** represents the "units sold" and **Y** represents the "per unit resource value". We will discuss the interpretation of the dual vector **Y** in more detail shortly.

Example 1:

Primal	**Dual**
Max $Z = 2X_1 + X_2$	Min $W = 6Y_1 + 4Y_2 + 3Y_3$
s.t. $X_1 + 2X_2 \leq 6$	s.t. $Y_1 + 3Y_2 + 0Y_3 \geq 2$
$3X_1 - X_2 \leq 4$	$2Y_1 - Y_2 + Y_3 \geq 1$
$X_2 \leq 3$	$Y_1, Y_2, Y_3 \geq 0$
$X_1, X_2 \geq 0$	

Example 2:

Primal

$$\begin{array}{lrcrcrcr} \max & 2000X_{fancy} & + & 1700X_{fine} & + & 1200X_{new} & & \\ \text{s.t.} & X_{fancy} & + & X_{fine} & + & X_{new} & \leq & 12 \\ & 25X_{fancy} & + & 20X_{fine} & + & 19X_{new} & \leq & 280 \\ & X_{fancy} & , & X_{fine} & , & X_{new} & \geq & 0 \end{array}$$

Dual

$$\begin{array}{lrcrcrl} & (\text{van cap}) & & (\text{labor}) & & & (\text{profits}) \\ \min & 12U_1 & + & 280U_2 & & & (\text{Resource Payments}) \\ \text{s.t.} & U_1 & + & 25U_2 & \geq & 2000 & (X_{fancy}) \\ & U_1 & + & 20U_2 & \geq & 1700 & (X_{fine}) \\ & U_1 & + & 19U_2 & \geq & 1200 & (X_{new}) \\ & U_1 & , & U_2 & \geq & & (\text{nonegativity}) \end{array}$$

Note: Last line of Dual should have "≥ 0"

Summary:

- One dual variable for each primal constraint
- Primal objective function → Dual R.H.S.
- Primal R.H.S. → Dual objective function

- Transpose L.H.S. coefficient matrix
- Reverse direction of inequalities
- Reverse optimisation direction

It can be proved mathematically that if both problems have bounded feasible solutions then their optimum objective values are equal. If one problem has an unbounded solution then the other problem is not feasible.

Note: Duality is very closely linked with sensitivity analysis (next section). We will see that one can read off the solution to one problem from the final optimal tableau of the other problem.

1.2.10 Introductory sensitivity analysis

The following example is used to discuss the "dual solution" and its interpretation and subsequently to present a limited form of sensitivity analysis. A more complete treatment is presented in section 1.5.

Example: A company makes two products,

Resources	**Product 1**	**Product 1**	**Available**
Labour(hr/unit)	1	2	10 hours
Material(Kg/unit)	6	6	36 Kg
Storage (M^2 /unit)	8	4	40 M^2
Profit (€/unit)	4	5	

Let X_1, X_2 = number of units of product 1, 2 made

Primal formulation	Dual formulation
Max $Z = 4X_1 + 5X_2$	Min $W = 10Y_1 + 36Y_2 + 40Y_3$
s.t. $X_1 + 2X_2 \leq 10$	s.t. $Y_1 + 6Y_2 + 8Y_3 \geq 4$
$6X_1 + 6X_2 \leq 36$	$2Y_1 + 6Y_2 + 4Y_3 \geq 5$
$8X_1 + 4X_2 \leq 40$	$Y_1, Y_2, Y_1 \geq 0$
$X_1, X_2 \geq 0$	

We can see that the initial simplex method tableau of the primal problem is

	X_1	X_2	S_1	S_2	S_3	***b***
Z	-4	-5	*0*	*0*	*0*	0
S_1	1	2	1	0	0	10
S_2	6	6	0	1	0	36
S_3	8	4	0	0	1	40

Optimal tableau of primal problem is (check!):

	X_1	X_2	S_1	S_2	S_3	***b***
Z	0	0	*1*	*½*	*0*	28
X_2	0	1	1	- 1/6	0	4
X_1	1	0	- 1	1/3	0	2
S_3	0	0	4	- 2	1	8

So the optimal solution to the primal problem is
$X^*_1 = 2$, $X^*_2 = 4$, $S^*_3 = 8$, $S_1^* = S_2^* = 0$ (no slack in labour & materials) and $Z^* = 28$

Notes:
1. It can be shown that **the optimal solution to the dual problem** is $Y_1^* = 1;\quad Y_2^* = 1/2;\ Y_3^* = 0.$
Interpretation: $Y_1^*,\ Y_2^* > 0 \rightarrow$ Labour + materials scarce
$Y_3^* = 0 \rightarrow$ storage abundant.

2. Expression for W*, the minimum value of the (Dual) objective function: For our example, Z* = W* = (**Y***)(**b**) =

$$[1,\ ½,\ 0] \begin{vmatrix} 10 \\ 36 \\ 40 \end{vmatrix}$$

$$10 + 18 + 0 = 28$$

In fact, the formula W* = (**Y***)(**b**) holds generally.

Also, at any stage during the Simplex procedure, the current value of the objective function will be given by (**Y**)(**b**) where **Y** is the current set of dual variable values.

3. Rate of change or sensitivity of objective function to b[1]: From the foot-note it follows that $\delta(Z)/\delta\mathbf{b} = \delta(W)/\delta\mathbf{b} = \delta(\mathbf{Yb})/\delta\mathbf{b} = \mathbf{Y}$. In terms of components of **Y** we have $Y_i = \delta Z / \delta b_i = \delta Z / \delta b_i$.
In particular, for the optimal solution we have $Y_i^* = \delta Z^* / \delta b_i$

Thus, we can see that a dual variable (also called a **reduced cost** or a **shadow price** of a resource) is the sensitivity of the objective function value to changes in the corresponding right-hand-side constant.

Notation: In this example, **c** = (4, 5, 0, 0, 0). It turns out to be convenient to have a special notation ($\mathbf{c_B}$) for those coefficients that correspond to just the basic variables. For the final tableau (above), the basic variables are (***in order***) X_2, X_1 and S_3 so that $\mathbf{c_B} = (5, 4, 0)$.

---

[1] (a) Differentiation of a scalar by a vector: If u is a scalar and **v** is a vector, say

$$\mathbf{v} = \begin{vmatrix} v_1 \\ v_2 \\ \dots \\ v_P \end{vmatrix}$$

Then, by definition, $\delta u/\delta\mathbf{v} = [\delta u/\delta v_1, \delta u/\delta v_2, \dots, \delta u/\delta v_P]$. This is called the gradient of u with respect to **v**.
(b) If $\mathbf{t} = [t_1, t_2, \dots, t_P]$ is a row vector then it can be shown that $\delta(\mathbf{tv})/\delta\mathbf{v} = \mathbf{t}$.

We clarify the above interpretation of dual variables further in the following, where it is convenient to introduce matrices to explain.

It is customary to use the symbol **B** to denote the matrix in the *initial* tableau that corresponds to the variables of the final basis (in the final tableau). Mathematically, the steps of the Simplex Method amount to simply multiplying the initial system by the inverse of **B** that is by $\mathbf{B}^{-1}$.
So initially, in the example, we had the system

$$\begin{vmatrix} 1 & 2 & 1 & 0 & 0 \\ 6 & 6 & 0 & 1 & 0 \\ 8 & 4 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} X1 \\ X2 \\ S1 \\ S2 \\ S3 \end{vmatrix} = \begin{vmatrix} 10 \\ 36 \\ 40 \end{vmatrix}$$

Then we (in effect) multiplied by $\mathbf{B}^{-1}$ to get

$$\mathbf{B}^{-1} \begin{vmatrix} 1 & 2 & 1 & 0 & 0 \\ 6 & 6 & 0 & 1 & 0 \\ 8 & 4 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} X1 \\ X2 \\ S1 \\ S2 \\ S3 \end{vmatrix} = \mathbf{B}^{-1} \begin{vmatrix} 10 \\ 36 \\ 40 \end{vmatrix}$$

Or, from final tableau,

$$\begin{vmatrix} 0 & 1 & 1 & -1/6 & 0 \\ 1 & 0 & -1 & 1/3 & 0 \\ 0 & 0 & 4 & -2 & 1 \end{vmatrix} \begin{vmatrix} X1 \\ X2 \\ S1 \\ S2 \\ S3 \end{vmatrix} = \begin{vmatrix} 4 \\ 2 \\ 8 \end{vmatrix}$$

The final basis in our example is $X_2|X_1|S_3$ so from the initial and final tableaux we have, respectively,

$$\mathbf{B} = \begin{vmatrix} 2 & 1 & 0 \\ 6 & 6 & 0 \\ 4 & 8 & 1 \end{vmatrix} \quad \text{and} \quad \mathbf{B}^{-1} = \begin{vmatrix} 1 & -1/6 & 0 \\ -1 & 1/3 & 0 \\ 4 & -2 & 1 \end{vmatrix}$$

We make the following observations:
(A) (1) The optimal (basis) solution to the primal problem is $(X_2^*, X_1^*, S_3^*) = \mathbf{B}^{-1}\mathbf{b} =$

$$\begin{vmatrix} 1 & -1/6 & 0 \\ -1 & 1/3 & 0 \\ 4 & -2 & 1 \end{vmatrix} \begin{vmatrix} 10 \\ 36 \\ 40 \end{vmatrix} = \begin{vmatrix} 4 \\ 2 \\ 8 \end{vmatrix}$$

(2) The optimal solution to the dual problem is $(Y_1^*, Y_2^*, Y_3^*) = \mathbf{c_B B^{-1}} =$

$$\begin{vmatrix} 5 & 4 & 0 \end{vmatrix} \begin{vmatrix} 1 & -1/6 & 0 \\ -1 & 1/3 & 0 \\ 4 & -2 & 1 \end{vmatrix} = \begin{vmatrix} 1 & ½ & 0 \end{vmatrix}$$

where $\mathbf{c_B}$ denotes the cost coefficients corresponding to the basic variables only.

(3) The (optimum) objective function values for primal and dual are the same:

$\underline{\mathbf{c}}_\mathbf{B}.(X_2^*, X_1^*, S_3^*) =$

$$\begin{vmatrix} 5 & 4 & 0 \end{vmatrix} \begin{vmatrix} X_2^* \\ X_1^* \\ S_3^* \end{vmatrix} = 5X_2^*+4X_1^*+0S_3^*=5(4)+4(2)+0(8) = 28= \mathbf{c_B B^{-1} b}$$

$$\begin{vmatrix} (Y_1^*, Y_2^*, Y_3^*) \end{vmatrix} \begin{vmatrix} 10 \\ 36 \\ 40 \end{vmatrix} =10Y_1^*+36Y_2^*+40Y_3^*=10(1)+36(1/2)+40(0)$$

(B) Sensitivity analysis for **b** (example)

$$\begin{vmatrix} 1 & -1/6 & 0 \\ -1 & 1/3 & 0 \\ 4 & -2 & 1 \end{vmatrix} \begin{vmatrix} 10+\Delta_1 \\ 36 \\ 40 \end{vmatrix} = \begin{vmatrix} 4+\Delta_1 \\ 2-\Delta_1 \\ 8+4\Delta_1 \end{vmatrix} \geq \begin{vmatrix} 0 \\ 0 \\ 0 \end{vmatrix}$$

=> $-2 <= \Delta_1 <= 2$. Subject to this restriction, the optimal solution would then be $(X_2^*, X_1^*, S_3^*)= (4+\Delta_1, 2-\Delta_1, 8+4\Delta_1)$

(C) Sensitivity analysis for **c** (example) (in fact, $\mathbf{c_B}$ rather than **c**)

$$\begin{vmatrix} 5+\delta_2 & 4 & 0 \end{vmatrix} \begin{vmatrix} 1 & -1/6 & 0 \\ -1 & 1/3 & 0 \\ 4 & -2 & 1 \end{vmatrix} = \begin{vmatrix} 1+\delta_2 & ½-\delta_2/6 & 0 \end{vmatrix} \geq 0$$

=> $1+\delta_2 >= 0$ and $½-\delta_2/6 >= 0$ => $-1 <= \delta_2 <= 3$. Subject to this restriction the objective function value would be $(1+\delta_2)(10)+(½-\delta_2/6)(36)+(0)(40)=28+4\delta_2$.

Recap and short-cuts for sensitivity analysis:

$Y_1^* = 1 \rightarrow$ optimal profit will change by €1 for every hour of change in labour availability.

$Y_2^* = 1/2 \rightarrow$ optimal profit will change by €0.50 for every Kg of change in material availability.

$Y_3^* = 0 \rightarrow$ optimal profit will not change if storage availability changes.

- Statements valid <u>within limits</u> (as seen above but we recap below without explicit use of $\mathbf{B}^{-1}$):

- <u>Suppose a change in labour availability</u>:

b vector (column) in formulation is $\begin{pmatrix} 10 + \Delta_1 \\ 36 \\ 40 \end{pmatrix}$

instead of $\begin{pmatrix} 10 \\ 36 \\ 40 \end{pmatrix}$

b column in optimal tableau?

$$\begin{pmatrix} 4 + \Delta_1 \\ 2 - \Delta_1 \\ 8 + 4\Delta_1 \end{pmatrix}$$

- result of repeating original pivots.

<u>Short cut</u>:-

From optimal tableau take

b column $\begin{pmatrix} 4 \\ 2 \\ 8 \end{pmatrix}$ and $\mathbf{S}_1$ column $\begin{pmatrix} 1 \\ -1 \\ 4 \end{pmatrix}$

(S_1 associated with labour)

$\mathbf{b} \geq \mathbf{0}$

$$4 + \Delta_1 \geq 0 \quad \rightarrow \Delta_1 \geq -4$$

$$2 - \Delta_1 \geq 0 \quad \rightarrow \Delta_1 \leq 2$$

$$8 + 4\Delta_1 \geq 0 \quad \rightarrow \Delta_1 \geq -2$$

$$-2 \leq \Delta_1 \leq 2$$

$\rightarrow Z^*$ will change by €1 for every 1 hour change in labour availability within the range 8 - 12 hours.

- similarly for material, using **b** column and $\mathbf{S}_2$ column:-

$$-6 \leq \Delta_2 \leq 4$$

i.e. range 30 - 40 Kg.

- similarly for storage, using **b** column and $\mathbf{S}_3$ column:-

$$\Delta_3 \geq -8$$

i.e. range $32 \rightarrow \infty$

Changes in Objective Function Coefficients ($\mathbf{c_B}$ only as non-basic variables do not affect the objective function value; sensitivity can be derived mathematically in a similar fashion as for **b**):
Suppose change in profit per unit for product 1.

Objective function:-

$$\text{Max } Z = (4 + \delta_1)X_1 + 5X_2$$

Can iterate to give new Z row in optimal tableau:-

	X_1	X_2	S_1	S_2	S_3	b
Z	0	0	$(1-\delta_1)$	$(1/2 + 1/3\delta_1)$	0	$28 + 2\delta_1$

Take current optimal Z row:-

0	0	1	½	0	28

and X_1 row from optimal tableau (product 1):-

1	0	- 1	1/3	0	2

(coefficients of δ_1 for non-basic variables)

Z row entries ≥ 0 for optimality (maximisation) $\rightarrow$

$(1 - \delta_1) \geq 0 \rightarrow \delta_1 \leq 1$

$(1/2 + 1/3\delta_1) \geq 0 \rightarrow \delta_1 \geq -3/2$

$-3/2 \leq \delta_1 \leq 1 \rightarrow$ €2.50 - €5

Similarly for X_2 : $-1 \leq \delta_2 \leq 3 \rightarrow$ €4 - €8

100% Rule - simultaneous parameter changes

Up to now we have considered changes in parameters one at a time. This rule allows us to consider simultaneous changes.

(1) Right-had-side constants:-

Solution feasible iff

$$\sum_i \delta b_i / \Delta b_i \leq 1$$

where δb_i = actual change in b_i

Δb_i = maximum allowable change in b_i

e.g. suppose labour availability increases by 1 hour and materials availability increases by 1 Kg.

$$\sum_i \delta b_i / \Delta b_i = 1/2 + 1/4 = 3/4$$

Suppose labour availability increases by 2 hours and materials availability increases by 2 Kg.

$$\sum_i \delta b_i / \Delta b_i = 2/2 + 2/4 = 3/2$$

(2) **Objective function coefficients**:-

Solution still optimal iff

$$\sum_i \delta c_i / \Delta c_i \leq 1$$

where δc_j = actual change in c_j

Δc_i = maximum allowable change in c_i

e.g. suppose profit per unit for product 1 and product 2 increase by €0.50.

$$\sum_i \delta c_i / \Delta c_i = 0.50 / 1 + 0.50 / 3 = 2/3$$

Suppose profit per unit for product 1 and product 2 increase by €1.

$$\sum_i \delta c_i / \Delta c_i = 1/1 + 1/3 = 4/3$$

Note: 100% rule is not fool-proof! Solution may (rarely) remain unchanged even though rule has been broken.
Note: Our treatment of sensitivity is not complete. We will address it more systematically later in this module, in section 1.5.

1.3 Large scale problems – Tool usage (AMPL)

1.3.1 Overview

Section 1.3 is a brief introduction to the linear programming tool AMPL (Fourer, et al., 2003). There are other such tools available but students are required to use AMPL when carrying out continuous assessments on linear programming for this module. The tool has been installed on the school computer labs but a student's version is freely downloadable from the site http://www.ampl.com/ . In fact, this site is quite informative and students should make use of it.

In lectures, we will just give an indication of how to get started in using the tool, the basic approach of the tool to setting up and solving linear programming problems, and some examples. In the linear programming part of the continuous assessment, there will be a number of problems to be set up and solved using AMPL; students are expected to discover for themselves whatever features of AMPL that may be needed in addressing these problems.

There is a companion text book for AMPL (Fourer, et al., 2003). There is a copy of this in the library, and an electronic copy is available & downloadable on the web-site for free. In this module we will not cover the whole book by any means so it is not strongly recommended to buy it; still, it is a good text to have. Some of the continuous assessment will require independent reading of parts of the book not covered in lectures. In lectures, we will sometimes display and discuss text from the book.

1.3.2 Sketch of AMPL modelling language and approach

A typical sequence of events for a practical linear programming application is summarised in (Fourer, et al., 2003) (page xvi) as follows:

> (a) Formulate a model, the abstract system of variables, objectives, and constraints that represent the general form of the problem to be solved.
>
> (b) Collect data that define a specific problem instance.
>
> (c) Generate a specific objective function and constraint equations from the model and data.
>
> (d) Solve the problem instance by running a program, or *solver*, to apply an algorithm that finds optimal values of the variables.
>
> (e) Analyze the results.
>
> (f) Refine the model and data as necessary, and repeat.

This sequence has a somewhat different emphasis than our examples to date with its separation of general model (a) and specific data (b) & (c).

Regarding (d) we have, up to now, applied the Simplex algorithm in hand calculations; the implication is that there are other possible algorithms and that different solving approaches are possible. *Thus, there are several methods available for solving systems of equations (e.g. "direct" methods such as Gauss or Gauss-Jordan elimination, "iterative" methods such as Gauss-Seidel as well as methods optimised for equation systems of special structure such as sparse system solvers)*. However, investigation of these is more a topic for Numerical Analysis and we will use solvers in AMPL as black boxes.

Step (e) could well include "sensitivity analysis" which we have already introduced. Finally, (f) is a normal step in any modelling or simulation exercise and the use of a tool such as AMPL makes it relatively simple to do whereas re-doing heavy hand calculations would be time-consuming and error-prone.

AMPL is an algebraic modelling language which we introduce and use mainly through examples. Students are expected to familiarise themselves with whatever features they may need in addition to those covered explicitly in class.

Example 1: We start with a simple two-variable example introduced in section 1.1 (Fourer, et al., 2003) (on-line). This problem is formulated in the book and its mathematical statement is

Maximize $25\ X_B + 30\ X_C$

Subject to

$(1/200)\ X_B + (1/140)\ X_C \leq 40$

$0 \leq X_B \leq 6000$

$0 \leq X_C \leq 4000$

The problem has to do with how to allocate next week's time in a steel plant between producing bands (B) and coils (C). X_B and X_C represent, respectively, the number of tons of bands and coils to produce.

The simplest use of AMPL in solving this problem is to create (e.g. you could do it with *Notepad*) a file (usually) of type ".mod" as follows. The actual file used is "prod0.mod" (included in the AMPL download) and its contents are,

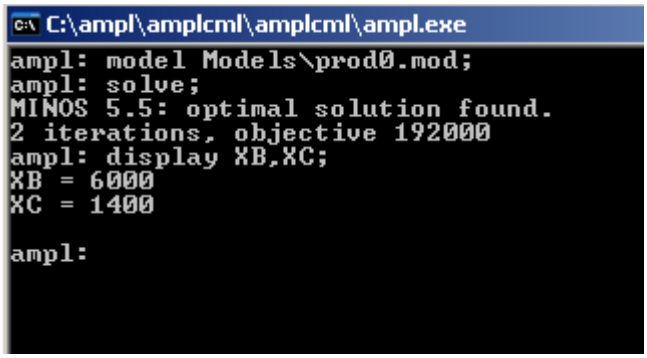
```
var XB;
var XC;
maximize Profit: 25 * XB + 30 * XC;
subject to Time: (1/200) * XB + (1/140) * XC <= 40;
subject to B_limit: 0 <= XB <= 6000;
subject to C_limit: 0 <= XC <= 4000;
```

This is in AMPL syntax but it is a very direct "translation" from the ordinary mathematical statement. The use of ";" and ":" as separators and some key words is clearly apparent. As summarised in (Fourer, et al., 2003) (p5):

"The AMPL linear program that you type into the file parallels the algebraic form in every respect. It specifies the decision variables, defines the objective, and lists the constraints. It differs mainly in being somewhat more formal and regular, to facilitate computer processing. Each variable is named in a **var** statement, and each constraint by a statement that begins with **subject to** and a name like Time or B_limit for the constraint. Multiplication requires an explicit * operator, and the ≤ relation is written <=."

The next step is to run the model (see page 5 of (Fourer, et al., 2003)) which is done using a small number of AMPL commands put in bold italics for emphasis (***model prod0.mod*** (but see below about file paths), ***solve***, ***display XB, XC***, ***quit***). The solver

invoked by ***solve*** is a default one unless otherwise specified. When the AMPL command line is started up (by running the application called ampl) the prompt "ampl:" appears. We have



To finish type ***quit***.

Note: One needs to be clear about setting file paths. In the above, the "ampl" application (ampl.exe) is in folder "amplcml" and file "prod0.mod" is in a sub-folder of this called "Models"; thus, "Models\prod0.mod" is relative to "amplcml".

Example 2: As a second example we take a problem that we previously considered whose formulation is

Maximise $40X_1 + 20X_2 + 30X_3$

subject to

Labour (per day): $7X_1 + 3X_2 + 6X_3 \leq 150$

Materials (per day): $4X_1 + 4X_2 + 5X_3 \leq 200$

And

$X_1 \geq 0, X_2 \geq 0, X_3 \geq 0$

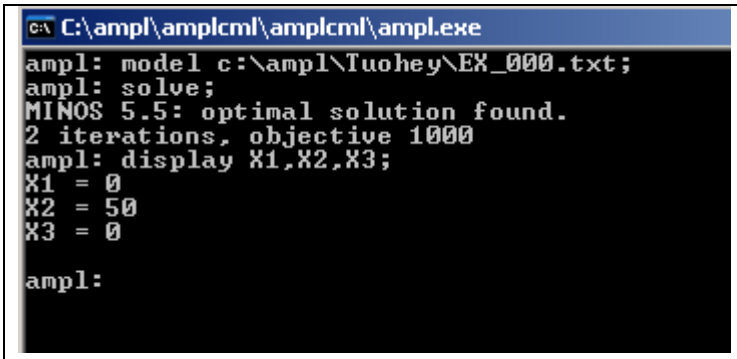
As for Example 1, the simplest use of AMPL in solving this problem is to create a file of type ".mod" as follows. The actual file I have created is "EX_000.txt" which is located on the C drive of my computer as follows:

ampl - Tuohey	[contains "EX_000.txt"]
- amplcml	[contains "sw.exe" & ampl.exe]
- amplcml	
- MODELS	[contains "prod0.mod" of Example 1]
- TABLES	

The contents of "EX_000.txt" are,

```
var X1;
var X2;
var X3;
maximize Profit: 40*X1 + 20*X2 + 30*X3;
subject to Labour   : 7*X1 + 3*X2 + 6*X3 <=150;
subject to Materials: 4*X1 + 4*X2 + 5*X3 <=200;
subject to 1_limit:0<=X1;
subject to 2_limit:0<=X2;
subject to 3_limit:0<=X3;
```

The next step is to run the model. We have



To finish type ***quit***.

Note: the file path used. Students should be able to rationalise their filepaths!

1.3.3 Interfaces; Getting started

AMPL provides various interfaces (see section 1.7 of book) but we confine ourselves here to the basic command line interface. Students may prefer to find out about and use one of the available GUIs.

1.3.4 Ex: Production Models: Maximising profits (Ch1, (Fourer, et al., 2003))

This example is taken from Chapter 1 of (Fourer, et al., 2003). In this section, we just list some key points and will then go to the book chapter for details.

(1) This is Example 1 of section 1.3.2 (above). We had

Math. Formulation	**AMPL** (file prod0.mod)
Maximize 25 X_B + 30 X_C	var XB;
Subject to	var XC;
(1/200) X_B + (1/140) $X_C \leq 40$	maximize Profit: 25 * XB + 30 * XC;
$0 \leq X_B \leq 6000$	subject to Time: (1/200) * XB + (1/140) * XC <= 40;
$0 \leq X_C \leq 4000$	subject to B_limit: 0 <= XB <= 6000;
	subject to C_limit: 0 <= XC <= 4000;

For present purposes we need to explain the context – (Fourer, et al., 2003) page 2.

In brief, X_B = number of tons of steel bands to be produced

X_C = number of tons of steel coils to be produced

One can produce 200 tons of bands per hour and 140 tons of coils so that it will take $X_B/200$ and $X_C/140$ hours to produce X_B and X_C tons,

respectively. There is an overall limit of 40 hours.

(2) The book then (pages 6, 7) generalises the problem to

Given: P, a set of products
a_j = tons per hour of product j, for each $j \in P$
b = hours available at the mill
c_j = profit per ton of product j, for each $j \in P$
u_j = maximum tons of product j, for each $j \in P$

Define variables: X_j = tons of product j to be made, for each $j \in P$

Maximize:

$$\Sigma_{j \in P} c_j X_j$$

Subject to:

$$\Sigma_{j \in P} (1/ a_j) X_j \leq b$$

$$0 \leq X_j \leq u_j, \text{ for each } j \in P$$

The components of this are **sets**, **parameters**, **variables**, **objective** and **constraints**. These appear in the corresponding AMPL file prod.mod ((Fourer, et al., 2003), page 8):

```
set P;
param a {j in P};
param b;
param c {j in P};
param u {j in P};
var X {j in P};
maximize Total_Profit: sum {j in P} c[j] * X[j];
subject to Time: sum {j in P} (1/a[j]) * X[j] <= b;
subject to Limit {j in P}: 0 <= X[j] <= u[j];
```

The various points of AMPL syntax introduced in this are explained in the book (pages 8, 9) including key words set, param, var, maximise, subject to and sum . Also, the **indexing expression** {j in P} meaning "for each j in P".

Another point to note is that the above does not contain any data values (unlike our very first examples). Instead, AMPL allows separation of general model from specific data. The data can be put in a separate file (say prod.dat, page 10 of (Fourer, et al., 2003)) such as

```
set P := bands coils;
param:          a    c    u    :=
bands           200  25   6000
coils      140  30   4000 ;
param b := 40;
```

Refer to book for how to execute this.

In fact, (Fourer, et al., 2003) (pages 10 to 18) goes on to improve and generalise the model in various ways. We discuss these using the text of the book.

1.3.5 Ex: Diet and other input problems (Ch2, (Fourer, et al., 2003))

Refer to book chapter (on Moodle). We discuss various salient points in class.

1.3.6 Ex: Transportation & Assignment Models (Ch3, (Fourer, et al., 2003))

Refer to book chapter (on Moodle) and some aspects are discussed in class. We also provide some additional notes (following).

1.3.6.1 Mathematical formulation of the transportation problem

Let $a_1, a_2, \ldots, a_m$ be the amounts of a product to be shipped from m storage depots or origins. Let $b_1, b_2, \ldots, b_n$ be the amounts of this product to be delivered to n destinations.

Let x_{ij} be the amount of the product to be shipped from origin i to destination j, and let c_{ij} be the cost of shipping 1 unit of the product from i to j.

Then, the total cost is

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \qquad (1)$$

Also, $x_{ij} \geq 0 \ \forall$ i, j

and $\sum_{j=1}^{n} x_{ij} = a_i$ (i = 1, m) & $\sum_{i=1}^{m} x_{ij} = b_j$ (j=1, n) (2)

Assume (for now) that the total shipped equals the total received. Hence,

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j = A(say)$$

The problem is to find x_{ij} to minimise (1) subject to (2).

Example of constraints in a particular Transportation Problem with 3 origins and 7 destinations:

1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	X11		1400
0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	X12		2600
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	X13		2900
1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	X14		900
0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	X15		1200
0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	X16		600
0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	X17		400
0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	X21		1700
0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	X22	=	1100
0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	X23		1000
																				X24		
																				X25		
																				X26		
																				X27		
																				X31		
																				X32		
																				X33		

X34
X35
X36
X37

1.3.6.2 Special properties of the transportation problem

The problem may be solved using the usual Simplex procedure. However, due to particular properties of the problem a special procedure for its solution has been devised (which, however, we will not cover in CA4011). We simply state these special properties and related theorems.

Special property A: The matrix of coefficients in (2) contains only 1s and 0s.

Special property B: Any variable appears only once in the first m equations and only once in the last n equations.

Theorem 1: The transportation problem is feasible.

Theorem 2: An initial basic feasible solution can be constructed.

[One way is via the "North-West Corner technique – not covered in CA4011]

Theorem 3: If a_i, b_j are non-negative integers then every basic feasible solution has integer values.

Theorem 4: A finite minimum feasible solution always exists.

1.3.6.3 Application of Transportation Problem: Assignment Problem

We present this by way of the following example.

Suppose there are N candidates for N different jobs. Let indices i and j refer to candidates and jobs, respectively. The candidates are tested as to their expected job performance and attain scores c_{ij} accordingly, that is c_{ij} is a measure of the (expected) effectiveness of candidate i for job j.

The problem is to assign candidates to jobs in order to maximise effectiveness ***overall***. Thus, the fact that a candidate scores well on a particular job does not mean that he/she will be assigned to that job. Further, the possibility of a person being assigned to different jobs for a fraction of time is not excluded. Let $x_{ij} \geq 0$ be the fraction of the time that candidate i spends in job j. Then we must have

$$\sum_{j=1}^{N} x_{ij} = 1 \quad (i = 1, N) \qquad (1)$$

if the employees are to be fully occupied, and

$$\sum_{i=1}^{N} x_{ij} = 1 \text{ (j=1, N)} \qquad (2)$$

if all jobs are to be filled.

The objective is to maximise

$$\sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij} x_{ij} \qquad (3)$$

This is in the usual "transportation" form (it's simple to transform from a max to a min problem) but in addition as $a_i = b_j = 1$ the x_{ij} cannot exceed one, and by theorem 3, x_{ij} must be integral. Therefore, we must have that $x_{ij} = 0$ or 1. This means that the optimal policy involves assigning one person to one job. This is a special kind of linear programming problem ("zero-one") – see section 2.4.

Example of an assignment problem with 3 candidates & 3 jobs:

1	1	1	0	0	0	0	0	0	X11		1
0	0	0	1	1	1	0	0	0	X12		1
0	0	0	0	0	0	1	1	1	X13		1
1	0	0	1	0	0	1	0	0	X21	=	1
0	1	0	0	1	0	0	1	0	X22		1
0	0	1	0	0	1	0	0	1	X23		1
									X31		
									X32		
									X33		

1.3.6.4 Application of Transportation Problem: Marriage Problem

The problem is to find which social structure (monogamy, polygamy or variants) is optimal.

Consider N men and N women. Let the women rate the men according to desirability and vice versa. Denote these ratings by c_{ji}^{W} and c_{ij}^{M} and define $c_{ij} = -[c_{ji}^{W} + c_{ij}^{M}]/2$.

Denoting by x_{ij} the fraction of time the i^{th} man spends with the j^{th} woman, (1), (2) and (3) of the assignment problem (last section) imply that any x_{ij} must be 1 or 0 for optimum. Hence marriage is best because it maximises overall happiness.

On the other hand, if the sign of c_{ij} were reversed the result that marriage is optimal would still be true but now happiness would be minimised. The pairings would be different, of course!

1.4 Multiple Criteria Decision-Making (MCDM), including Goal Programming

1.4.1 Introduction

A typical traditional mathematical programming problem can be formulated as

Maximise f(**x**)
subject to $\mathbf{x} \in X = \{\mathbf{x} | \mathbf{g}(\mathbf{x}) \leq \mathbf{b}\}$

For example, a linear programming problem with $\mathbf{x} = (x_1, x_2)$, might be
Maximise $f(\mathbf{x}) = 5.2x_1 - 10.3x_2$
subject to $\mathbf{x} \in X = \{(x_1, x_2) | (x_1 - 1.5x_2 \leq 4, 2x_1 + 3x_2 \leq 12, x_1 \geq 0, x_2 \geq 0)\}$

BUT why should we restrict ourselves so that the feasible region is a fixed set X and the objective function is a fixed f(**x**)?
If we have more than one criterion (objective) what could be a good solution concept and how to calculate the solution based on such a solution concept?

These issues arise in the research active area of **Multiple Criteria Decision Making**. This has been applied to such problems as

- **Warehouse layout design** (in order to both increase the effective use of space and to decrease the material handling cost)
- **Planning component recovery from products with component commonality** (determine the number and type of products to be disassembled in order to fulfil a set of demand constraints while satisfying pre-determined profit and recycling goals)

In this section 1.4 we focus on goal programming which has to do with linear programming problems where there are multiple objectives. We also indicate facilities provided in the **AMPL** tool for handling multiple objectives.

1.4.2 Goal Programming

1.4.2.1 Terminology and Definitions

- Similar to Linear Programming (LP) (same limitations).
- Objectives may be incommensurable.
- Solution – 'satisfactory' levels of goal attainment.
- Ranking of goals – lower ranked goals only considered <u>after</u> higher ranked goals.
- All goals assigned target levels – 'optimum' → as close as possible to targets.

<u>'Real' and 'goal' constraints; Deviational variables</u>:
e.g. 'real' constraint:- $X_1 + X_2 = 3$
``'goal' constraint allows: $X_1 + X_2 \geq 3$
or $X_1 + X_2 \leq 3$

→ $X_1 + X_2 + d_1^- - d_1^+ = 3$

d_1^- , d_1^+ are called <u>Deviational Variables</u>

We seek to Minimise $d_1^- + d_1^+$

May assign weights W_1^- , W_1^+ (objective function coefficients)

<u>Pre-emptive Priorities</u>: ("**>>**" means ***"much greater than")***

$$P_1 >> P_2 >> \ldots\ldots P_k >> P_{k+1} >>$$

Minimise $\sum_k P_k \sum_i (W_{ik}^- d_i^- + W_{ik}^+ d_1^+)$

The "pre-emptive priorities" are introduced in order to prioritise between multiple objectives. We do not ever actually have to assign specific numerical values to the "pre-emptive priorities" P_k.

1.4.2.2 Example: Product mix problem (1) – Statement

A company makes three products, and in so doing uses just two resources (labour and materials).

	Labour (hrs/unit)	Materials (Kg/unit)	Profit ($/unit)
Product 1	5	4	3
Product 2	2	6	5
Product 3	4	3	2
Availability (per day)	240	400	

Goals (in order of priority):
1. Avoid under-utilisation of production capacity (no layoffs)
2. Satisfactory profit = $500/day
3. Minimise overtime
4. Minimise purchase of additional materials

1.4.2.3 Example: Product mix problem (2) – Statement & Solution

Company makes two products

- amounts made = (X_1 , X_2) in three departments (A, B, C).
- Product 1 requires 5 hours in A and 1 hour in B
- Product 2 requires 2 hours in A and 1 hour in C
- Available:- A: 750 hours
  B: 100 hours
  C: 100 hours

- **Problem** – how many units of each product to make?

- **Goals**:

1. Use all hours available in department A
2. Use no more than 20 hours of overtime in department B.
3. Use all available hours in B and C. However, use of all hours in C is two-thirds as important as use of all hours in B.
3. Minimise overtime in B and C. However, overtime restriction for B is three-fifths as important as that for C.

- **Formulation:**

Minimise $P_1 d_1^- + P_2 d_4^+ + 3P_3 d_2^- + 2P_3 d_3^- + 3P_4 d_2^+ + 5P_4 d_3^+$

s.t.
$$5X_1 + 2X_2 + d_1^- - d_1^+ = 750$$
$$X_1 + d_2^- - d_2^+ = 100$$
$$X_2 + d_3^- - d_3^+ = 100$$
$$d_2^+ + d_4^- - d_4^+ = 20$$
$$X_j , d_i^- , d_i^+ \geq 0$$

- **Simplex tableau**:- We cannot express the objective function as a single row because of priority levels (not commensurable).

				$-3P_4$	$-5P_4$	$-P_2$	$-P_1$	$-3P_3$	$-2P_3$		
	X_1	X_2	d_1^+	d_2^+	d_3^+	d_4^+	d_1^-	d_2^-	d_3^-	d_4^-	b
P_1											
P_2											
P_3											
P_4											
d_1^-	5	2	- 1				1				750
d_2^-	1			- 1				1			100
d_3^-		1			- 1				1		100
d_4^-				1		- 1				1	20

- **Basic variables:-** d_1^-, d_2^-, d_3^-, d_4^-

- To put into **canonical form**, carry out pivot operations to give zero coefficients for basic variables in Z matrix.

e.g. P_1 row:- add d_1^- row to P_1 row to give new P_1 row as follows:

	X_1	X_2	d_1^+	d_2^+	d_3^+	d_4^+	d_1^-	d_2^-	d_3^-	d_4^-	b
P_1							- 1				
d_1^-	5	2	- 1				1				750
New P_1	5	2	- 1								750

P_2 row and P_4 row remain unchanged
P_3 row requires pivot: P_3 row + $3*d_2^-$ row + $2*d_3^-$ row to give new P_3 row.

Resulting tableau (in canonical form):

	X_1	X_2	d_1^+	d_2^+	d_3^+	d_4^+	d_1^-	d_2^-	d_3^-	d_4^-	b
P_1	5	2	- 1								750
P_2						- 1					0
P_3	3	2		- 3	- 2						500
P_4				- 3	- 5						0
d_1^-	5	2	- 1				1				750
d_2^-	1			- 1				1			100
d_3^-		1			- 1				1		100
d_4^-				1		- 1				1	20

Start with P_1 row (highest priority), searching for largest positive entry (minimisation) i.e. *5 (X_1)*. Therefore X_1 enters the basis.
Minimum ratio test:- Min(750/5 , 100/1 , - , -) = 100/1. Therefore d_2^- leaves the basis. Pivot as usual to give:-

	X_1	X_2	d_1^+	d_2^+	d_3^+	d_4^+	d_1^-	d_2^-	d_3^-	d_4^-	b
P_1		2	- 1	5				- 5			250
P_2						- 1					0
P_3		2			- 2			- 3			200
P_4				- 3	- 5						0
d_1^-		2	- 1	5			1	- 5			250
X_1	1			- 1				1			100
d_3^-		1			- 1				1		100
d_4^-				1		- 1				1	20

Note:- Goal 1 under-achieved by 250, Goal 3 by 200, Goals 2 and 4 are achieved.
Next: d_2^+ enters, d_4^- leaves. This leads to the next tableau:

	X_1	X_2	d_1^+	d_2^+	d_3^+	d_4^+	d_1^-	d_2^-	d_3^-	d_4^-	b
P_1		2	- 1			5		- 5		- 5	150
P_2						- 1				- 1	0
P_3		2			- 2			- 3			200
P_4					- 5	- 3				3	60
d_1^-		2	- 1			5	1	- 5		- 5	150
X_1	1					- 1		1		1	120
d_3^-		1			- 1				1		100
d_2^+				1		- 1				1	20

Next: d_4^+ enters, d_1^- leaves. The next tableau is

	X_1	X_2	d_1^+	d_2^+	d_3^+	d_4^+	d_1^-	d_2^-	d_3^-	d_4^-	b
P_1							- 1				0
P_2		2/5	-1/5				1/5	- 1		- 1	30
P_3		2			- 2			- 3			200
P_4		6/5	-3/5		- 5		3/5	- 3			150
d_4^+		2/5	-1/5			1	1/5	- 1		- 1	30
X_1	1	2/5	-1/5				1/5				150
d_3^-		1			- 1				1		100
d_2^+		2/5	-1/5	1			1/5	- 1			50

Note: P_1 is satisfied. So we next focus on the P_2 row
Next: From P_2 row X_2 enters and d_4^+ leaves. So the next tableau follows as,

	X_1	X_2	d_1^+	d_2^+	d_3^+	d_4^+	d_1^-	d_2^-	d_3^-	d_4^-	b
P_1							- 1				0
P_2						- 1					0
P_3			1		- 2	- 5	- 1	2		5	50
P_4					- 5	- 3				3	60
X_2		1	-1/2			5/2	1/2	-5/2		-5/2	75
X_1	1					- 1		1		1	120
d_3^-			1/2		- 1	-5/2	-1/2	5/2	1	5/2	25
d_2^+				1		- 1				1	20

Note: P_2 is satisfied. So we next focus on the P_3 row
Next: From P_3 row d_4^- enters, d_3^- leaves. So the next tableau follows as,

	X_1	X_2	d_1^+	d_2^+	d_3^+	d_4^+	d_1^-	d_2^-	d_3^-	d_4^-	b
P_1							- 1				0
P_2						- 1					0
P_3								- 3	- 2		0
P_4			-3/5		-19/5		3/5	- 3	- 6/5		30
X_2		1			- 1				1		100
X_1	1		-1/5		2/5		1/5		- 2/5		110
d_4^-			1/5		-2/5	- 1	-1/5	1	2/5	1	10
d_2^+			-1/5	1	2/5		1/5	- 1	- 2/5		10

Note: P_3 is satisfied. So we next focus on the P_4 row. **However**, there is only one positive in Z matrix in P_4 row (d_1^-). But this is superseded by – 1 above in P_1 row. **Therefore current solution is optimal**: $X_1 = 110$, $X_2 = 100$. Goals 1, 2, and 3 are achieved. Goal 4 is sacrificed (***check overtime in B and C***).

1.4.2.4 Problem

Use the simplex method to solve the following goal programming problem:

$$\text{Min} \quad P_1 d_1^- + P_2 d_4^+ + 5P_3 d_2^- + 3P_3 d_3^- + P_4 d_1^+$$

$$\text{s.t.} \quad X_1 + X_2 + d_1^- - d_1^+ = 80$$

$$X_1 + d_2^- = 70$$

$$X_2 + d_3^- = 45$$

$$X_1 + X_2 + d_4^- - d_4^+ = 90$$

$$X_j \geq 0, j = 1....2, d_i^-, d_i^+ \geq 0, i = 1.....4$$

Answer:
Optimal solution $X^*_1 = 70$ $X^*_2 = 20$, $d_3^- = 25$, $d_1^+ = 10$
Goal 1 achieved, Goal 2 achieved, Goal out by 75, Goal 4 out by 10.

1.4.2.5 City Park Application – Formulation & Solution

A city parks department has been given a grant of €600million to expand its public recreational facilities. Four different types of facilities have been requested by the city council members: gymnasiums, athletic fields, tennis courts and swimming pools. The total demand by various neighbourhoods has been estimated to be seven gyms, ten athletic fields, eight tennis courts and twelve swimming pools. Each facility costs a certain amount, requires a certain number of hectares and has an expected usage as shown in the following table:

Facility	Cost (€million)	Required hectares	Expected usage (people/week)
Gymnasium	80	4	1500
Athletic field	24	8	3000
Tennis court	15	3	500
Swimming pool	40	5	1000

The parks department has located a total of 50 hectares of land for construction (although more land could be located if necessary).

The council has established the following list of prioritised goals:
1. The parks department must spend the total grant (otherwise the amount not spent will be returned to the government).
2. The facilities should be used by 20,000 or more people weekly.
3. If more land is required, the additional amount should be limited to ten hectares.
4. The parks department would like to meet the demands of the members of the city council. However, this priority should be weighted according to the number of people expected to use each facility.
5. The parks department wants to avoid having to acquire land beyond the 50 hectares presently available.

Formulation:

Let X_1 = number of gymnasiums
X_2 = number of athletic fields
X_3 = number of tennis courts
X_4 = number of swimming pools

Goal 1 Funding (Upper limit = €600m):

$$80X_1 + 24X_2 + 15X_3 + 40X_4 + d_1^- - d_1^+ = 600$$

Here, d_1^- represents the amount of the 600 not used so the first priority is to

Minimise $P_1 d_1^-$

Goal 2 Usage (Usage by 20,000 or more per week):

$$1500X_1 + 3000X_2 + 500X_3 + 1000X_4 + d_2^- - d_2^+ = 20{,}000$$

Here, d_2^- represents the number fewer than 20000 users so the second priority is to

Minimise $P_2 d_2^-$

Goal 3 Land Requirement (50 hectares available):

$$4X_1 + 8X_2 + 3X_3 + 5X_4 + d_3^- - d_3^+ = 50$$

Here, d_3^+ represents the additional amount above 50 hectares. Hence, we introduce the constraint

$$d_3^+ + d_4^- - d_4^+ = 10$$

Then, as d_4^+ represents the amount above 10 additional hectares, the third priority is

Minimise $P_3 d_4^+$

Note: An alternative formulation for Goal 3 is possible, as follows:

$$4X_1 + 8X_2 + 3X_3 + 5X_4 + d_4^- - d_4^+ = 60$$

Minimise $P_3 d_4^+$

However, the first formulation will also accommodate Goal 5 (see below).

Goal 4 Demand:

$$X_1 + d_5^- - d_5^+ = 7$$
$$X_2 + d_6^- - d_6^+ = 10$$
$$X_3 + d_7^- - d_7^+ = 8$$
$$X_4 + d_8^- - d_8^+ = 12$$

The parks department would like to meet demand i.e. try not to under-supply i.e. minimise the d_i^- variables, weighted according to usage.

Minimise $P_4(1.5d_5^- + 3d_6^- + 0.5d_7^- + d_8^-)$

i.e. Minimise $P_4(3d_5^- + 6d_6^- + d_7^- + 2d_8^-)$

Goal 5 Land requirement (Avoid exceeding the 50 hectares):

Minimise $P_5 d_3^+$

Complete problem (summary):

Min $P_1 d_1^- + P_2 d_2^- + P_3 d_4^+ + P_4(3d_5^- + 6d_6^- + d_7^- + 2d_8^-) + P_5 d_3^+$

s.t. $80X_1 + 24X_2 + 15X_3 + 40X_4 + d_1^- - d_1^+ = 600$

$1500X_1 + 3000X_2 + 500X_3 + 1000X_4 + d_2^- - d_2^+ = 20{,}000$

$$4X_1 + 8X_2 + 3X_3 + 5X_4 + d_3^- - d_3^+ = 50$$
$$d_3^+ + d_4^- - d_4^+ = 10$$
$$X_1 + d_5^- - d_5^+ = 7$$
$$X_2 + d_6^- - d_6^+ = 10$$
$$X_3 + d_7^- - d_7^+ = 8$$
$$X_4 + d_8^- - d_8^+ = 12$$

$$X_j \geq 0, j = 1....4, d_i^-, d_i^+ \geq 0, i = 1.....8$$

Illustration of use of tool to solve a Goal Programming problem

Note: Here we illustrate the use of a particular tool (***Mathematica***) to solve this "city parks" example. However, other tools or libraries with suitable "mathematical programming" functionality may be used instead. In fact, we go further than general linear programming in that we use the fact that ***Mathematica*** also supports **integer programming** where some or all of the decision variables are constrained to be whole numbers – in this application it makes sense that the decision variables (at least X_1, X_2, X_3 and X_4) be whole numbers. We will study Integer Programming separately (in a later section) and will see that it poses more difficulties than linear programming.

In *Mathematica* the constraints may be specified as follows: (ordered as X_1 - X_4, d_1^- - d_8^-, d_1^+ - d_8^+) **– *a total of 20 variables***.

Lower bounds on variables :

```
In[56]:= lu = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
```

Following are the coefficients on right-hand side (RHS) of the set of constraints:

Basic set of constraints :

```
In[44]:= C1 = {80, 24, 15, 40, 1, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0};
         C2 = {1500, 3000, 500, 1000, 0, 1, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0};
         C3 = {4, 8, 3, 5, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0};
         C4 = {0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, -1, 0, 0, 0, 0};
         C5 = {1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0};
         C6 = {0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0};
         C7 = {0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0};
         C8 = {0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, -1};
```

(1) RHS of the **first goal** or objective "Minimise $P_1 d_1^-$ " only is "encoded" as

```
In[81]:= Obj1 = {0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
```

The problem is then solved as follows (*the "0" in "{600, 0}" etc indicates an equality*):

```
In[82]:= LinearProgramming[Obj1, {C1, C2, C3, C4, C5, C6, C7, C8},
          {{600, 0}, {20000, 0}, {50, 0}, {10, 0}, {7, 0}, {10, 0}, {8, 0}, {12, 0}}, lu, Integers]

         LinearProgramming::lpip :
           Warning: integer linear programming will use a machine-precision approximation of the inputs. >>

Out[82]= {7, 0, 0, 5, 0, 4500, 0, 7, 0, 10, 7, 0, 160, 0, 3, 0, 0, 0, 2, 0}

In[83]:= LinearProgramming[Obj1, {C1, C2, C3, C4, C5, C6, C7, C8},
          {{600, 0}, {20000, 0}, {50, 0}, {10, 0}, {7, 0}, {10, 0}, {8, 0}, {12, 0}}, lu]

Out[83]= {5148/967, 3030/967, 1096/967, 1980/967, 0, 0, 0, 0, 1621/967, 6640/967, 9624/967, 0, 0, 0, 10, 0, 0, 0, 0, 0}
```

The "integer" solution (where feasible) is the relevant one[2]. We have X1 = 7 (gyms) and X4 = 5 (pools) with no fields or courts. The total cost is 760.

(2) The **second goal** is addressed by making the first goal a constraint i.e. $d_1^- = 0$ (*does this make sense? – see section 1.4.3 (re* (Fourer, et al., 2003)*extract) for a similar "move"*).

We have

```
Make the first objective a constraint :

In[84]:= C9 = {0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

Specify the second objective :

In[85]:= Obj2 = {0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

In[86]:= LinearProgramming[Obj2, {C1, C2, C3, C4, C5, C6, C7, C8, C9}, {{600, 0}, {20000, 0},
           {50, 0}, {10, 0}, {7, 0}, {10, 0}, {8, 0}, {12, 0}, {0, 0}}, lu, Integers]

         LinearProgramming::lpip :
           Warning: integer linear programming will use a machine-precision approximation of the inputs. >>

Out[86]= {7, 3, 1, 0, 0, 0, 0, 5, 0, 7, 12, 0, 47, 0, 5, 0, 0, 0, 0, 0}

In[89]:= LinearProgramming[Obj2, {C1, C2, C3, C4, C5, C6, C7, C8, C9},
          {{600, 0}, {20000, 0}, {50, 0}, {10, 0}, {7, 0}, {10, 0}, {8, 0}, {12, 0}, {0, 0}}, lu]

Out[89]= {5148/967, 3030/967, 1096/967, 1980/967, 0, 0, 0, 0, 1621/967, 6640/967, 9624/967, 0, 0, 0, 10, 0, 0, 0, 0, 0}
```

The "integer" solution (where feasible) is the relevant one. We have X1 = 7 (gyms), X2 = 3 (fields) and X3 = 1 (courts) with no pools. The total cost is 647 and usage is 20000. Land used is 55 [***check***].

(3) The **third goal** is addressed by making the first and second goals constraints (*does this make sense? compare previous step*). We have

Make the second objective a constraint :

```
In[90]:= C10 = {0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
```

Specify the third objective :

```
In[91]:= Obj3 = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0};

In[92]:= LinearProgramming[Obj3, {C1, C2, C3, C4, C5, C6, C7, C8, C9, C10}, {{600, 0}, {20000, 0},
           {50, 0}, {10, 0}, {7, 0}, {10, 0}, {8, 0}, {12, 0}, {0, 0}, {0, 0}}, lu, Integers]
         LinearProgramming::lpip :
           Warning: integer linear programming will use a machine-precision approximation of the inputs. >>
Out[92]= {14, 0, 0, 0, 0, 0, 0, 4, 0, 10, 12, 0, 520, 1000, 6, 0, 7, 0, 2, 0}

In[93]:= LinearProgramming[Obj3, {C1, C2, C3, C4, C5, C6, C7, C8, C9, C10},
          {{600, 0}, {20000, 0}, {50, 0}, {10, 0}, {7, 0}, {10, 0}, {8, 0}, {12, 0}, {0, 0}, {0, 0}}, lu]
Out[93]= {5148/967, 3030/967, 1096/967, 1980/967, 0, 0, 0, 0, 1621/967, 6640/967, 9624/967, 0, 0, 0, 10, 0, 0, 0, 0, 0}
```

The "integer" solution (where feasible) is the relevant one. We have X1 = 14 (gyms) and no fields, courts or pools. The total cost is 1120 and usage is 21000. Land used is 56 [***check***].

(4) The **fourth goal** is addressed by making the first to third goals constraints (*does this make sense? see previous steps*). We have

Make the third objective a constraint :

```
In[94]:= C11 = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0};
```

Specify the fourth objective :

```
In[95]:= Obj4 = {0, 0, 0, 0, 0, 0, 0, 0, 3, 6, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0};

In[96]:= LinearProgramming[Obj4, {C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11}, {{600, 0}, {20000, 0},
           {50, 0}, {10, 0}, {7, 0}, {10, 0}, {8, 0}, {12, 0}, {0, 0}, {0, 0}, {0, 0}}, lu, Integers]
         LinearProgramming::lpip :
           Warning: integer linear programming will use a machine-precision approximation of the inputs. >>
Out[96]= {7, 3, 1, 1, 0, 0, 0, 0, 0, 7, 11, 0, 87, 1000, 10, 0, 0, 0, 0, 0}

In[97]:= LinearProgramming[Obj4, {C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11}, {{600, 0}, {20000, 0},
           {50, 0}, {10, 0}, {7, 0}, {10, 0}, {8, 0}, {12, 0}, {0, 0}, {0, 0}, {0, 0}}, lu]
Out[97]= {7, 38/11, 16/11, 0, 0, 0, 0, 0, 0, 72/11, 12, 0, 712/11, 17500/11, 10, 0, 0, 0, 0, 0}
```

[2] The non-integer (real) solution is being reported also as, for this, ***Mathematica*** reports on feasibility.

The "integer" solution (where feasible) is the relevant one. We have X1 = 7 (gyms), X2 = 3 (fields), X3 = 1 (courts) and X4 = 1 (pool). The total cost is 687 and usage is 21000. Land used is 60. Only the demand for gyms is satisfied.

(5) The **fifth goal** is addressed in the same way by making the first to fourth goals constraints (*does this make sense? – see previous steps*). We obtain

Make the fourth objective a constraint (TBC) :

```
In[98]:= C12 = {0, 0, 0, 0, 0, 0, 0, 0, 3, 6, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0};
```

Specify the fourth objective :

```
In[99]:= Obj5 = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0};
```

```
In[100]:= LinearProgramming[Obj5, {C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11, C12},
           {{600, 0}, {20000, 0}, {50, 0}, {10, 0}, {7, 0}, {10, 0},
            {8, 0}, {12, 0}, {0, 0}, {0, 0}, {0, 0}, {0, 0}}, lu, Integers]

          LinearProgramming::lpip :
            Warning: integer linear programming will use a machine-precision approximation of the inputs. >>

Out[100]= {2, 11, 10, 0, 0, 0, 0, 0, 0, -2, 12, 0, 0, 24037, 84, 74, 0, 0, 0, 0}

In[101]:= LinearProgramming[Obj5, {C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11, C12},
           {{600, 0}, {20000, 0}, {50, 0}, {10, 0}, {7, 0},
            {10, 0}, {8, 0}, {12, 0}, {0, 0}, {0, 0}, {0, 0}, {0, 0}}, lu]

          LinearProgramming::lpsnf : No solution can be found that satisfies the constraints. >>
```

We find that this is an infeasible problem (the linear programming problem with "Real" decision variables allowed). If it is infeasible for Reals then it infeasible for Integers as they are more restrictive.

1.4.3 AMPL facilities for multiple objectives

Example 1 (Diet problem re-visited; not goal programming but suggestive!)

In mathematical notation we start with the problem

Minimise $\textbf{Total\_Cost} = \sum_{j\ in\ FOOD} cost_j Buy_j$

Subject to $n\_min_i \leq \sum_j amt_j Buy_j \leq n_{max_j}$

$f\_min_j \leq Buy_j \leq f\_max_j$

The following figures (Figures 2-1 and 2-2) from (Fourer, et al., 2003) depict the corresponding AMPL model and a typical data set.

```
set NUTR;
set FOOD;

param cost {FOOD} > 0;
param f_min {FOOD} >= 0;
param f_max {j in FOOD} >= f_min[j];

param n_min {NUTR} >= 0;
param n_max {i in NUTR} >= n_min[i];

param amt {NUTR,FOOD} >= 0;

var Buy {j in FOOD} >= f_min[j], <= f_max[j];

minimize Total_Cost:  sum {j in FOOD} cost[j] * Buy[j];

subject to Diet {i in NUTR}:
   n_min[i] <= sum {j in FOOD} amt[i,j] * Buy[j] <= n_max[i];
```

Figure 2-1: Diet model in AMPL (`diet.mod`).

```
set NUTR := A B1 B2 C ;
set FOOD := BEEF CHK FISH HAM MCH MTL SPG TUR ;

param:   cost  f_min  f_max :=
  BEEF   3.19    0     100
  CHK    2.59    0     100
  FISH   2.29    0     100
  HAM    2.89    0     100
  MCH    1.89    0     100
  MTL    1.99    0     100
  SPG    1.99    0     100
  TUR    2.49    0     100 ;

param:   n_min  n_max :=
   A      700   10000
   C      700   10000
   B1     700   10000
   B2     700   10000 ;

param amt (tr):
           A    C   B1   B2 :=
   BEEF   60   20   10   15
   CHK     8    0   20   20
   FISH    8   10   15   10
   HAM    40   40   35   10
   MCH    15   35   15   15
   MTL    70   30   15   15
   SPG    25   50   25   15
   TUR    60   20   15   10 ;
```

Figure 2-2: Data for diet model (`diet.dat`).

Also, the corresponding results from AMPL are presented in (Fourer, et al., 2003) as

```
ampl: model diet.mod;
ampl: data diet.dat;

ampl: solve;
MINOS 5.5: optimal solution found.
6 iterations, objective 88.2
ampl: display Buy;
Buy [*] :=
BEEF    0
 CHK    0
FISH    0
 HAM    0
 MCH  46.6667
 MTL  -1.07823e-16
 SPG  -1.32893e-16
 TUR    0
;
```

This problem is re-visited in Chapter 8 of (Fourer, et al., 2003) (section 8.3). This is the part of particular interest for Goal Programming. We have,

8.3 Objectives

The declaration of an objective function consists of one of the keywords `minimize` or `maximize`, a name, a colon, and a linear expression in previously defined sets, parameters and variables. We have seen examples such as

```
minimize Total_Cost: sum {j in FOOD} cost[j] * Buy[j];
```

and

```
maximize Total_Profit:
   sum {p in PROD, t in 1..T}
      (sum {a in AREA[p]} revenue[p,a,t] * Sell[p,a,t] -
         prodcost[p] * Make[p,t] - invcost[p] * Inv[p,t]);
```

The name of the objective plays no further role in the model, with the exception of certain "columnwise" declarations to be introduced in Chapters 15 and 16. Within AMPL commands, the objective's name refers to its value. Thus for example after solving a feasible instance of the Figure 2-1 diet model we could issue the command

```
ampl: display {j in FOOD} 100 * cost[j] * Buy[j] / Total_Cost;
100*cost[j]*Buy[j]/Total_Cost [*] :=
BEEF  14.4845
 CHK   4.38762
FISH   3.8794
 HAM  24.4792
 MCH  16.0089
 MTL  16.8559
 SPG  15.6862
 TUR   4.21822
;
```

to show the percentage of the total cost spent on each food.

Note particularly:
"*Although a particular linear program must have one objective function, a model may contain more than one objective declaration.*
Moreover, any minimize or maximize declaration may define an indexed collection of objective functions, by including an indexing expression after the objective name.
In these cases, you may issue an ***objective*** *command, before typing* ***solve****, to indicate which objective is to be optimized*."

As an example, recall that when trying to solve the model of Figure 2-1 with the data of Figure 2-2, we found that no solution could satisfy all of the constraints; we subsequently increased the sodium (NA) limit to 50000 to make a feasible solution possible. It is reasonable to ask: How much of an increase in the sodium limit is really necessary to permit a feasible solution? For this purpose we can introduce a new objective equal to the total sodium in the diet:

```
minimize Total_NA: sum {j in FOOD} amt["NA",j] * Buy[j];
```

(We create this objective only for sodium, because we have no reason to minimize most of the other nutrients.) We can solve the linear program for total cost as before, since AMPL chooses the model's first objective by default:

```
ampl: model diet.mod;
ampl: data diet2a.dat;
ampl: display n_max["NA"];
n_max['NA'] = 50000

ampl: minimize Total_NA: sum {j in FOOD} amt["NA",j] * Buy[j];
ampl: solve;
MINOS 5.5: optimal solution found.
13 iterations, objective 118.0594032
Objective = Total_Cost
```

The solver tells us the minimum cost, and we can also use `display` to look at the total sodium, even though it's not currently being minimized:

```
ampl: display Total_NA;
Total_NA = 50000
```

Next we can use the `objective` command to switch the objective to minimization of total sodium. The `solve` command then re-optimizes with this alternative objective, and we display `Total_Cost` to determine the resulting cost:

```
ampl: objective Total_NA;

ampl: solve;
MINOS 5.5: optimal solution found.
1 iterations, objective 48186

ampl: display Total_Cost;
Total_Cost = 123.627
```

We see that sodium can be brought down by about 1800, though the cost is forced up by about $5.50 as a result. (Healthier diets are in general more expensive, because they force the solution away from the one that minimizes costs.)

Example 2 (Room allocation as a particular transportation problem)

It will be seen that Example 1 is not really a Goal Programming example as we did not attempt to solve for a number of goals at the same time. . However, this second example does suggest how to investigate Goal Programming problems with AMPL.

This second example involves the ***transportation*** problem (discussed previously):
Let $a_1, a_2, \ldots, a_m$ be the amounts of a product to be shipped from m storage depots or origins.

Let $b_1, b_2, \ldots, b_n$ be the amounts of this product to be delivered to n destinations.

Let x_{ij} be the amount of the product to be shipped from origin i to destination j.

Let c_{ij} be the cost of shipping 1 unit of the product from i to j.

Then, the total cost is

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \qquad (1)$$

Also, $x_{ij} \geq 0 \ \forall$ i, j

and $\sum_{j=1}^{n} x_{ij} = a_i$ (i = 1, m) & $\sum_{i=1}^{m} x_{ij} = b_j$ (j=1, n) (2)

Assume (for now) that the total shipped equals the total received. Hence,

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j = A(say)$$

The problem is to find x_{ij} to minimise (1) subject to (2).

The corresponding AMPL model appears as Fig 3-1a of (Fourer, et al., 2003):

```
set ORIG;   # origins
set DEST;   # destinations

param supply {ORIG} >= 0;   # amounts available at origins
param demand {DEST} >= 0;   # amounts required at destinations

   check: sum {i in ORIG} supply[i] = sum {j in DEST} demand[j];

param cost {ORIG,DEST} >= 0;   # shipment costs per unit
var Trans {ORIG,DEST} >= 0;    # units to be shipped

minimize Total_Cost:
   sum {i in ORIG, j in DEST} cost[i,j] * Trans[i,j];

subject to Supply {i in ORIG}:
   sum {j in DEST} Trans[i,j] = supply[i];

subject to Demand {j in DEST}:
   sum {i in ORIG} Trans[i,j] = demand[j];
```

Figure 3-1a: Transportation model (`transp.mod`).

In Chapter 3 of (Fourer, et al., 2003) the above model is used to solve a room allocation problem, with the data:

```
set ORIG := Coullard Daskin Hazen Hopp Iravani Linetsky
            Mehrotra Nelson Smilowitz Tamhane White ;

set DEST := C118 C138 C140 C246 C250 C251 D237 D239 D241 M233 M239;

param supply default 1 ;

param demand default 1 ;

param cost:
          C118 C138 C140 C246 C250 C251 D237 D239 D241 M233 M239 :=
 Coullard   6    9    8    7   11   10    4    5    3    2    1
 Daskin    11    8    7    6    9   10    1    5    4    2    3
 Hazen      9   10   11    1    5    6    2    7    8    3    4
 Hopp      11    9    8   10    6    5    1    7    4    2    3
 Iravani    3    2    8    9   10   11    1    5    4    6    7
 Linetsky  11    9   10    5    3    4    6    7    8    1    2
 Mehrotra   6   11   10    9    8    7    1    2    5    4    3
 Nelson    11    5    4    6    7    8    1    9   10    2    3
 Smilowitz 11    9   10    8    6    5    7    3    4    1    2
 Tamhane    5    6    9    8    4    3    7   10   11    2    1
 White     11    9    8    4    6    5    3   10    7    2    1 ;
```

Figure 3-2: Data for assignment problem (`assign.dat`).

The solution of this original problem appears on page 136 of (Fourer, et al., 2003):

```
ampl: model transp.mod; data assign.dat; solve;
CPLEX 8.0.0: optimal solution; objective 28
24 dual simplex iterations (0 in phase I)

ampl: option display_1col 1000, omit_zero_rows 1;
ampl: option display_eps .000001;

ampl: display Total_Cost,
ampl?    {i in ORIG, j in DEST} cost[i,j] * Trans[i,j];
Total_Cost = 28

cost[i,j]*Trans[i,j] :=
Coullard  C118   6
Daskin    D241   4
Hazen     C246   1
Hopp      D237   1
Iravani   C138   2
Linetsky  C250   3
Mehrotra  D239   2
Nelson    C140   4
Smilowitz M233   1
Tamhane   C251   3
White     M239   1
;
```

Note that the objective of the original problem is labelled ***Total_Cost***.

The text of (Fourer, et al., 2003) that we wish to emphasise is

"*To keep the objective value at this optimal level while we experiment, we add a constraint that fixes the expression for the objective equal to the current value, 28*" (and following)

as it shows how one may fix an achieved objective (***so that it stays achieved!***) and then specify another objective. In fact, in this example a whole set of goals ***Pref_of {i in ORIG}*** are defined and then any one of them can be selected as the objective to be solved for.

The full text from (Fourer, et al., 2003) is as follows:

To keep the objective value at this optimal level while we experiment, we add a constraint that fixes the expression for the objective equal to the current value, 28:

```
ampl: subject to Stay_Optimal:
ampl?    sum {i in ORIG, j in DEST}
ampl?       cost[i,j] * Trans[i,j] = 28;
```

Next, recall that `cost[i,j]` is the ranking that person `i` has given to office `j`, while `Trans[i,j]` is set to 1 if it's optimal to put person `i` in office `j`, or 0 otherwise. Thus

```
sum {j in DEST} cost[i,j] * Trans[i,j]
```

always equals the ranking of person `i` for the office to which `i` is assigned. We use this expression to declare a new objective function:

```
ampl: minimize Pref_of {i in ORIG}:
ampl?    sum {j in DEST} cost[i,j] * Trans[i,j];
```

This statement creates, for each person `i`, an objective `Pref_of[i]` that minimizes the ranking of `i` for the room that `i` is assigned. Then we can select any one person and optimize his or her ranking in the assignment:

```
ampl: objective Pref_of["Coullard"];
ampl: solve;
CPLEX 8.0.0: optimal solution; objective 3
3 simplex iterations (0 in phase I)
```

Looking at the new assignment, we see that the original objective is unchanged, and that the selected individual's situation is in fact improved, although of course at the expense of others:

```
ampl: display Total_Cost,
ampl?    {i in ORIG, j in DEST} cost[i,j] * Trans[i,j];
Total_Cost = 28

cost[i,j]*Trans[i,j] :=
Coullard  D241   3
Daskin    D237   1
Hazen     C246   1
Hopp      C251   5
Iravani   C138   2
Linetsky  C250   3
Mehrotra  D239   2
Nelson    C140   4
Smilowitz M233   1
Tamhane   C118   5
White     M239   1
;
```

Note the important text "*We were able to make this change because there are several optimal solutions to the original total-ranking objective. A solver arbitrarily returns one of these, but by use of a second objective we can force it toward others.*"

In Goal Programming, generally, we are "hoping" that there are multiple solutions to achieve the highest priority goal so that we can use this freedom to achieve lower priority goals also.
***Note**: By using the CPLEX solver we can enforce integer constraints if we wish*.

1.5 Duality & Dual Simplex Method

1.5.1 Recap of Dual & Primal problems (1.2.9, 1.2.10)

- Associated with every L.P. there is another L.P. - The Dual

- Simplex method gives the solution to both problems
- Duality is used in sensitivity analysis
- Standard pair of problems (*Transpose denoted by '*):

Primal	**Dual**
Max $Z = \underline{c}\,\underline{x}$	**Min** $W = \underline{b}'\underline{y}$
s.t. $\underline{A}\,\underline{x} \leq \underline{b}$	**s.t.** $\underline{A}'\underline{y} \geq \underline{c}'$
$\underline{x} \geq \underline{0}$	$\underline{y} \geq \underline{0}$

e.g.

Max $Z = 2X_1 + X_2$	**Min** $6Y_1 + 4Y_2 + 3Y_3$
s.t. $X_1 + 2X_2 \leq 6$	**s.t.** $Y_1 + 3Y_2 \geq 2$
$3X_1 - X_2 \leq 4$	$2Y_1 - Y_2 + Y_3 \geq 1$
$X_2 \leq 3$	
$X_1, X_2 \geq 0$	$Y_1, Y_2, Y_3 \geq 0$

Note: The above implies that Y is a column vector. In fact, we will normally regard it as a row vector. (so, for example, W = **Yb**).

Refer to sections 1.9 and 1.10 for a worked example and interpretation of dual variables; also, for an incomplete sensitivity analysis based on considering sensitivity of the objective function to individual changes in **b** and $\mathbf{c_B}$.

1.5.2 Sensitivity Analysis (Generalised wrt 1.2.9 & 1.2.10)

Changes to the original problem may affect:

- optimality
- feasibility

From Section 1.2.10 (*also see under Revised Simplex Method in text books*):

$\underline{x_B} = \mathbf{B^{-1}b}$

where

$\underline{x_B}$ = current values of *basic* variables

$\mathbf{B^{-1}}$ = inverse matrix (found in position of original $\underline{I}$ matrix at any iteration)

b = original right hand side

- For the above example, from initial & final tableaux, one has, **at the optimum**,

$$\underline{B}^{-1} = \begin{vmatrix} 1 & -1/6 & 0 \\ -1 & 1/3 & 0 \\ 4 & -2 & 1 \end{vmatrix} \quad \text{and } \underline{B}^{-1}\underline{b} = \begin{vmatrix} 4 \\ 2 \\ 8 \end{vmatrix}$$

Therefore if **materials availability** were 38 instead of 36, R.H.S. vector at the optimum would be:

$$\underline{B}^{-1}\underline{b} = \begin{vmatrix} 1 & -1/6 & 0 \\ -1 & 1/3 & 0 \\ 4 & -2 & 1 \end{vmatrix} \begin{vmatrix} 10 \\ 38 \\ 40 \end{vmatrix} = \begin{vmatrix} 11/3 \\ 8/3 \\ 4 \end{vmatrix}$$

Suppose **labour availability** were 14. Then, the new R.H.S. is

$$\underline{B}^{-1}\underline{b} = \begin{vmatrix} 1 & -1/6 & 0 \\ -1 & 1/3 & 0 \\ 4 & -2 & 1 \end{vmatrix} \begin{Vmatrix} 14 \\ 36 \\ 40 \end{Vmatrix} = \begin{vmatrix} 8 \\ -2 \\ 24 \end{vmatrix}$$

which is, however, ***infeasible***. (We will see how this can be dealt with in the Dual Simplex method)

Note 1: If we knew (!) $\mathbf{B}^{-1}$ we could calculate the final tableau from the initial one:

$$\left|\begin{array}{ccc} 1 & -1/6 & 0 \\ -1 & 1/3 & 0 \\ 4 & -2 & 1 \end{array}\right| \left|\begin{array}{ccccc|c} 1 & 2 & 1 & 0 & 0 & 10 \\ 6 & 6 & 0 & 1 & 0 & 36 \\ 8 & 4 & 0 & 0 & 1 & 40 \end{array}\right| =$$

$$\left|\begin{array}{ccccc|c} 0 & 1 & 1 & -1/6 & 0 & 4 \\ 1 & 0 & -1 & 1/3 & 0 & 2 \\ 0 & 0 & 4 & -2 & 1 & 8 \end{array}\right|$$

Note 2: From the initial tableau (***notice order of columns***),

$$\underline{B} = \begin{vmatrix} 2 & 1 & 0 \\ 6 & 6 & 0 \\ 4 & 8 & 1 \end{vmatrix}$$

Note 3: The optimal value of the **primal** objective function (Z^*, say) is given by $Z^* = \mathbf{C_B X^*} = \mathbf{C_B}(\mathbf{B}^{-1}\mathbf{b})$, where $\mathbf{C_B}$ denotes the cost coefficients for the basic variables (only).

Thus, for our example, Z* = [5 4 0].[4 2 8] = 28.

Also, for the **dual** problem, the optimal solution is given by $\mathbf{Y^*} = \mathbf{C_B B}^{-1}$

Thus, for our example, $Y^* = [5\ 4\ 0]B^{-1} = [1\ ½\ 0]$.

Then, the optimal value of the dual objective function (W^*, say) is $(\mathbf{C_B B}^{-1})\mathbf{b}$

For our example, ***$W^* = [1\ ½\ 0].[10\ 36\ 40] = 28$***

In general, we have

$Z^* = \mathbf{C_B X^*} = \mathbf{C_B}(\mathbf{B}^{-1}\mathbf{b}) = (\mathbf{C_B B}^{-1})\mathbf{b} = \mathbf{Y^*b} = W^*$.

1.5.3 Dual Simplex Method (to restore feasibility)

In our example, the Simplex tableau arising from a labour availability of 14 (see earlier) is a modification of the final tableau of the original problem:

	X_1	X_2	S_1	S_2	S_3	**b**
Z	0	0	1	1/2	0	32
X_2	0	1	1	-1/6	0	8
X_1	1	0	-1	1/3	0	-2
S_3	0	0	4	-2	1	24

The only thing that has changed with respect to the original final tableau is the **b** column (for which we already saw how to calculate the last three rows using $\mathbf{B}^{-1}\mathbf{b}$). The Z row entry for the **b** column is calculated using $\mathbf{C_B}(\mathbf{B}^{-1}\mathbf{b})$ = [5 4 0].[8 -2 24] = 32.

The **Dual Simplex method** allows us to remove the negative entry in the right hand side (**b**) according to the following **pivot rule**:

- scan **b** column for largest negative entry to determine the leaving variable.
- scan for variable (column) which gives the smallest ratio (in absolute terms) of z-row entry and corresponding equation coefficient. Ignore ratios with positive or zero denominators.
- then pivot as usual.

For our example, this gives X_1 to leave and S_1 to enter and the resulting table is:

	X_1	X_2	S_1	S_2	S_3	**b**
Z	1	0	0	5/6	0	30
X_2	1	1	0	1/6	0	6
S_1	-1	0	1	-1/3	0	2
S_3	4	0	0	-2/3	1	16

As there are no negative entries in the Z row, this is the optimal solution ($X_2 = 6$, $X_1 = 0$ and $Z^* = 30$).

We next examine how to handle various other modifications of the original problem, including use of the Dual Simplex procedure as necessary. We continue with the same example to illustrate.

1.5.4 Adding a constraint

Recall our "running" example is

Max $Z = 4X_1 + 5X_2$

s.t. $X_1 + 2X_2 \leq 10$

$6X_1 + 6X_2 \leq 36$

$8X_1 + 4X_2 \leq 40$

$X_1, X_2 \geq 0$

If the new constraint is non-binding, there is no problem

e.g. $X_1 + X_2 \leq 7$ *(in our example above).*

However, suppose $X_1 + X_2 \leq 5$ is added to the original problem.

i.e. $X_1 + X_2 + S_4 = 5$ (using a new slack variable S_4)

We now obtain a modified Simplex tableau that is a modification of the final tableau of the original problem, in that a new row is added to represent the new constraint and a new column for S_4:

	X_1	X_2	S_1	S_2	S_3	S_4	**b**
Z	0	0	1	1/2	0	0	28
X_2	0	1	1	-1/6	0	0	4
X_1	1	0	-1	1/3	0	0	2
S_3	0	0	4	-2	1	0	8
S_4	1	1	0	0	0	1	5

This tableau is not in canonical form, as we do not have a starting basis (non-zero values have appeared in final row of X_1 and X_2 columns).

Pivot: subtract X_1 row and X_2 row from S_4 row to give new S_4 row:

	X_1	X_2	S_1	S_2	S_3	S_4	**b**
Z	0	0	1	1/2	0	0	28
X_2	0	1	1	-1/6	0	0	4
X_1	1	0	-1	1/3	0	0	2
S_3	0	0	4	-2	1	0	8
S_4	0	0	0	-1/6	0	1	-1

But now we have an infeasibility, indicated by the "-1" on the last row. We can use the dual simplex method to get rid of the infeasibility, resulting that S_4 leaves and S_2 enters.

Check that the solution is:-

$X_1 = 0$ $X_2 = 5$ $S_1 = 0$ $S_2 = 6$ $S_3 = 20$ $S_4 = 0$

$Z = 25$.

1.5.5 Variations in Objective Function Coefficients

1.5.5.1 Variation in Objective Function. Coefficient of a Basic Variable

We have, at any stage of the Simplex method, that the current dual solution is $\mathbf{Y} = \mathbf{c_B B^{-1}}$ where $\mathbf{c_B}$ = **c** vector for basic variables only. (In particular, the optimal dual solution is obtained using $\mathbf{B^{-1}}$ from the final tableau, as we have seen in an earlier slide).

Note also that, in general, Z row entries in a Simplex tableau are given by:

$\mathbf{YP}_j - c_j$ (where $\mathbf{P}_j$ is the corresponding column in the initial tableau).

Example: Suppose profit for product 1 (X_1) were €7 per unit instead of €4 per unit (as in the original problem).

Remember $\mathbf{c_B}$ contains profit/unit for X_2 X_1 S_3 (note order).

First calculate the new **Y**:

$$Y = \left| 5\ 7\ 0 \right| \left\| \begin{matrix} 1 & -1/6 & 0 \\ -1 & 1/3 & 0 \\ 4 & -2 & 1 \end{matrix} \right| = \left| -2\ 3/2\ 0 \right|$$

Next re-calculate the Z row (only need to do this for non-basics S_1, S_2):

For S_1 :

$$\left| -2\ \ 3/2\ \ 0 \right| \left\| \begin{matrix} 1 \\ 0 \\ 0 \end{matrix} \right| - 0 = -2$$

For S_2 :

$$\left| -2\ \ 3/2\ \ 0 \right| \left\| \begin{matrix} 0 \\ 1 \\ 0 \end{matrix} \right| - 0 = 3/2$$

So, the full tableau now is (modifying just the Z row entries of S_1 and S_2 columns) the final tableau of the original problem):

	X_1	X_2	S_1	S_2	S_3	**b**
Z	0	0	-2	3/2	0	28
X_2	0	1	1	-1/6	0	4
X_1	1	0	-1	1/3	0	2
S_3	0	0	4	-2	1	8

There is now a negative entry in the Z row. So, make S_1 a basic variable to restore optimality. Then, by the minimum ratio test, S_3 becomes non-basic.

Check: Complete the calculations to find the new optimal solution.

1.5.5.2 Variation in Objective Function. Coefficient of a non-Basic Variable

There is no need to re-calculate **Y** values:-
Just re-calculate the *Z* row values using

$$Z_j = \mathbf{YP}_j - c_j$$

where **Y** = current dual variable values.
Note: The next item illustrates this procedure.

1.5.6 Adding a new activity (a new decision variable)

Continuing with our example, suppose the company introduces a new product (of which X_3 is the number of units to be made) and for which,

Requirements per unit: 2 hours labour
5 Kgs material
5 m^2 storage

Profit per unit = €5.

The problem becomes:

Max $Z = 4X_1 + 5X_2 + 5X_3$

s.t. $X_1 + 2X_2 + 2X_3 \leq 10$
$6X_1 + 6X_2 + 5X_3 \leq 36$
$8X_1 + 4X_2 + 5X_3 \leq 40$
$X_1, X_2, X_3 \geq 0$

We need to add a new column to the final Simplex tableau, corresponding to X_3.

To calculate the appropriate column (X_3) in optimal tableau:

$$P^* = B^{-1}P = \left|\begin{array}{ccc||c|} 1 & -1/6 & 0 & 2 \\ -1 & 1/3 & 0 & 5 \\ 4 & -2 & 1 & 5 \end{array}\right| = \left|\begin{array}{c} 7/6 \\ -1/3 \\ 3 \end{array}\right|$$

We also need a *Z* row entry (for a non-basic variable – see 1.5.5.2):-

$$\left|\begin{array}{ccc||c|} 1 & 1/2 & 0 & \begin{array}{c}2\\5\\5\end{array} \end{array}\right| - 5 = -1/2$$

So, the full tableau is (modifying the final tableau of the original problem by adding column X_3):

	X_1	X_2	X_3	S_1	S_2	S_3	**b**
Z	0	0	-1/2	1	1/2	0	28
X_2	0	1	7/6	1	-1/6	0	4
X_1	1	0	-1/3	-1	1/3	0	2
S_3	0	0	3	4	-2	1	8

As there is a negative entry in *Z* row the current solution is not optimal.
Applying the usual Simplex Method, we have that X_3 enters basis, replacing S_3 (ratio test).

Exercise: Complete the solution to find the new optimal solution.

1.5.7 Problem(s)

Microbrewers Incorporated makes four products called Light, Dark, Ale, and Premium. These products are made using the resources of water, malt, hops and yeast. Microbrewers has a free supply of water. The technology table (table of resource requirements to make a gallon of each product is as follows:

	Light	Dark	Ale	Premium	Availability
Malt	1	1	0	3	50Kg
Hops	2	1	2	1	150Kg
Yeast	1	1	1	4	80Kg
Profit	€6	€5	€3	€7	

Let the production of light, dark, ale and premium equal X_1, X_2, X_3, and X_4 gallons respectively
The optimal simplex tableau is as follows:

	X_1	X_2	X_3	X_4	S_1	S_2	S_3	b
Z	0	0	0	7	3	1	1	
X_2	0	1	0	7	0	-1	2	10
X_1	1	0	0	-4	1	1	-2	40
X_3	0	0	1	1	-1	0	1	30

What will be the optimal solution if the following changes are made (one at a time):

(a) The availability of malt is reduced by 20 Kg and the availability of hops is reduced by 30 Kg?

(b) A requirement that the amount of light plus the amount of dark produced must be at least twice the amount of ale plus premium produced?

(c) The revenue from a gallon of light is increased by €1?

(d) The revenue from a gallon of Premium is increased to €15?

(e) A new product is introduced which requires (per gallon) 2 Kg of malt, 3 Kg of hops and 1 Kg of yeast and has an associated revenue of €9 per gallon?

2. Integer Programming

2.1 Introduction, including some applications

Statement of the problem:

$$\textbf{Maximise} \quad \sum_{j=1}^{n} c_j \, x_j$$

Subject to

$$\sum_{j=1}^{n} a_{ij} \, x_j \leq b_i \quad \text{for all } i = 1 \ldots\ldots m$$

$$x_j \geq 0 \quad \text{for all } j$$

Types of problem:

- Some x_j integer → Mixed Integer Problem
- All x_j integer → Pure Integer Problem
- $x_j = 0$ or 1 → Zero-one Programming Problem

Solution Methods

(1) Heuristics - rounding, specialised algorithms.
(2) Sequential decomposition – Cutting Plane Methods.
(3) Enumeration/Search – Branch and Bound

- Rounding: solution may be ***sub-optimal*** or ***infeasible***.

Example: **Max** $Z = 100X_1 + 90X_2$ **s.t.** $10X_1 + 7X_2 \leq 70$ $5X_1 + 10X_2 \leq 50$ $X_1, X_2 \geq 0$ **integers** **L.P. solution:** $X_1^* = 5.38,\ X_2^* = 2.31,\ Z^* = 746$ **Rounded solution:** $X_1^* = 5,\ X_2^* = 2,\ Z^* = 680$ **Optimal integer solution:** $X_1^* = 7,\ X_2^* = 0,$ $Z^* = 700$	**Min** $Z = 200X_1 + 400X_2$ **s.t.** $10X_1 + 25X_2 \geq 100$ $3X_1 + 2X_2 \geq 12$ $X_1, X_2 \geq 0$ **integers** **L.P. solution:** $X_1^* = 1.82,\ X_2^* = 3.27,\ Z^* = 1673$ **Rounded solution:** $X_1^* = 2,\ X_2^* = 3,$ **infeasible** **Optimal integer solution:** $X_1^* = 3,\ X_2^* = 3,$ **or** $X_1^* = 5,\ X_2^* = 2,\ Z^* = 1800$

Application of Integer Programming - Fixed Charge Problem**:**

Production cost for product j = $C_j(x_j)$ = $K_j + c_j x_j$ for $x_j > 0$
= 0 for $x_j = 0$

Thus, $\boldsymbol{K_j}$ = **fixed charge** (incurred if any amount of j is produced) & $\boldsymbol{c_j}$ = **variable charge**

The problem to **minimise** $\boldsymbol{C_j(x_j)}$ is **non-linear**.

Define $y_j = 0$ when $x_j = 0$
$= 1$ when $x_j > 0$

Then, $0 \leq x_j \leq M y_j$, where $M > 0$ is an unspecified large positive number.

The problem becomes a mixed-integer one, namely:-

$$\text{Minimise} \sum_{j=1}^{n} (c_j x_j + K_j y_j)$$

subject to $0 \leq x_j \leq M y_j$ for all j
$y_j = 0, 1$

Application of Integer Programming – Minimum activity requirement:
Only activate x_k if above a certain level (e.g. oil well produces flow if at least a minimum level).
Thus, $x_k = 0$ or $x_k \geq L$

Re-write as: $x_k \leq 0$ or $x_k \geq L$

Hence, $x_k \leq 0 + yM$
$x_k \geq L - (1 - y)M$
where y = 0 or 1, and M > 0 is an unspecified large positive number.

$y = 0 \rightarrow$ inactive decision variable (arbitrarily negative)
$y = 1 \rightarrow$ decision variable activated

Application of Integer Programming – Either/Or constraints**:**
Example: $X_1 + 4X_2 \leq 30$ (1)
$2X_1 + 3X_2 \leq 40$ (2)

either (1) or (2) must hold.
Define $y = 0, 1$
$M > 0$, large
$X_1 + 4X_2 \leq 30 + yM$ (1)′
$2X_1 + 3X_2 \leq 40 + (1 - y) M$ (2)′

At solution,
if $y = 0$ (1) is binding, (2) is redundant,
if $y = 1$ (2) is binding, (1) is redundant.

Application of Integer Programming – Dichotomies**:**
At least *k* out of *m* constraints must hold.
m constraints: $g_i (x_1 x_2 \ldots\ldots\ldots x_n) \leq b_i \quad i = 1 \ldots m$

Define $y_i = 0$ if constraint *i* holds
$y_i = 1$ if constraint *i* does not hold
To guarantee *k* out of *m*, *define for M >> 1*,

$$g_i (x_1 x_2 \ldots\ldots\ldots x_n) \leq b_i + My_i \quad i = 1 \ldots m$$

$$\sum_{i=1}^{m} y_j = m - k$$

For example, if m=3 and k=2, then we will get three different possibilities according to which constraint does not hold.

2.2 Cutting Plane Methods

Principles:-

- start with a solution to "relaxed" problem (no integer restrictions).
- add a constraint to reduce the size of the feasible region without excluding integer solutions.
- solve new problem.
- stopping rule:- if solution integer, stop
  else add another constraint.

We focus on a Fractional algorithm, suitable for pure integer problems)

We assume, to start with, that a final solution tableau (below) has been obtained for the relaxed problem, where
$X_1 \ldots\ldots\ldots X_m$ are basic variables (*for convenience shown together*)
$W_1 \ldots\ldots\ldots W_n$ *are* non-basic variables
$X_i = \beta_i$ where β_i is non-integer (***if some are already integer that's a bonus!***) but we are after integer solutions.

	X_1	X_2	…..	X_m	W_1	…	….	W_n	
Z	0	0	…..	0	c_1'	…	…	c_n'	β_0
X_1	1	0	…..	0	α_1^1	…	…	α_1^n	β_1
X_2	0	1	…..	…	….	…	…	…	…
…	…	…	…..	…	….	…	…	…	…
X_m	0	0	…..	1	α_m^1	…	…	α_m^n	β_m

The table implies that

$$X_i + \sum_{j=1}^{n} \alpha_i^j W_j = \beta_i$$

$$\text{or} \quad X_i = \beta_i - \sum_{j=1}^{n} \alpha_i^j W_j \qquad \ldots\ldots\ldots\ldots(1)$$

$$\text{Let } \beta_i = [\beta_i] + f_i$$
$$\alpha_i^j = [\alpha_i^j] + f_{ij}$$

$$N = [a] = \text{largest integer such that } N \leq a$$

$$0 < f_i < 1 \text{ and } 0 < f_{ij} < 1$$

e.g. $3\frac{3}{8} \rightarrow [a] = 3 \quad f = \frac{3}{8}$

$-3\frac{3}{8} \rightarrow [a] = -4 \quad f = \frac{5}{8}$

From(1)

$$X_i = [\beta_i] + f_i - \sum_{j=1}^{n} \{[\alpha_i^j] + f_{ij}\} W_j$$

$$\text{Or} \quad f_i - \sum_{j=1}^{n} f_{ij} W_j = X_i - [\beta_i] + \sum_{j=1}^{n} [\alpha_i^j] W_j$$

Key observation: If all X_i and W_j are to be integer, the right-hand-side of the above equation must be integer and therefore left-hand-side must be integer also.

Given that: $f_{ij} \geq 0$ **and** $W_j \geq 0$

$$\rightarrow \quad \sum_{j=1}^{n} f_{ij} W_j \geq 0$$

$$\rightarrow \quad -\sum_{j=1}^{n} f_{ij} W_j \leq 0$$

and therefore

$$f_i - \sum_{j=1}^{n} f_{ij} W_j \leq f_i$$

i.e. $f_i - \sum_{j=1}^{n} f_{ij} W_j < 1$ **because** $f_i < 1$

{ must be integer }

Therefore we must have

$$f_i - \sum_{j=1}^{n} f_{ij} W_j \leq 0$$

Necessary requirement for integer solution – new constraint – fractional cut

Put into standard form:

$$-\sum_{j=1}^{n} f_{ij} W_j + S_i = -f_i$$

We then add this new constraint to the final tableau of the relaxed problem to obtain

	X_1	X_2	…..	X_m	W_1	…	W_n	S_i	
Z	0	0	…..	0	c_1'	…	c_n'	0	β_0
X_1	1	0	…..	0	α_1^1	…	α_1^n	0	β_1
X_2	0	1	…..	…	….	…	…	0	…
X_m	0	…	…..	1	α_m^1	…	α_m^n	…	β_m
S_i	0	0	…..	0	$-f_{i1}$	…	$-f_{in}$	1	$-f_i$

Next, apply the **dual simplex method**.

If solution is integer, stop.

Else make another fractional cut.

Note: all non-zero coefficients of cut < 1.

Example: Consider the integer programming problem:

$$\textbf{Max } Z = 5X_1 + 8X_2$$

$$\textbf{s.t. } 7X_1 + 8X_2 \leq 28$$

$$X_1 + 2X_2 \leq 6$$

$$X_1, X_2 \geq 0 \text{ integers}$$

The final tableau for the corresponding relaxed problem is as follows (***check***):

	X_1	X_2	S_1	S_2	b
Z	0	0	1/3	8/3	76/3
X_1	1	0	1/3	- 4/3	4/3
X_2	0	1	- 1/6	7/6	7/3

Thus, the optimal solution to the relaxed problem is $X_1^* = 4/3$, $X_2^* = 7/3$, $Z^* = 76/3$.

We look for the variable with largest fractional part. In this case there is a tie as both X_1 and X_2 have fractional part 1/3. We choose X_1, arbitrarily:

***X_1* row**: $X_1 + 1/3\ S_1 - 4/3\ S_2 = 4/3$

or

$$X_1 + (0 + 1/3)\ S_1 + (-2 + 2/3)\ S_2 = (1 + 1/3)$$

So, the new constraint is

$$1/3\ S_1 - 2/3\ S_2 \leq -1/3$$

In standard form this is

$$1/3\ S_1 - 2/3\ S_2 + S_3 = -1/3$$

We add this to the final tableau for the relaxed problem to obtain,

	X_1	X_2	S_1	S_2	S_3	b
Z	0	0	1/3	8/3	0	76/3
X_1	1	0	1/3	-4/3	0	4/3
X_2	0	1	-1/6	7/6	0	7/3
S_3	0	0	-1/3	-2/3	1	-1/3

Use dual simplex method to pivot, giving S_3 to leave the basis and finding $\text{Min}\{|(1/3)/(-1/3)|, |(8/3)/(-2/3)|\} = \text{Min}\{1, 4\} \Rightarrow S_1$ enters:

	X_1	X_2	S_1	S_2	S_3	b
Z	0	0	0	2	1	25
X_1	1	0	0	- 2	1	1
X_2	0	1	0	3/2	-1/2	5/2
S_1	0	0	1	2	- 3	1

At this stage, X_2 is the only variable with a fractional value, as $X_1 = 1$, $S_1 = 1$ and $S_2 = S_3 = 0$.

X_2 row:- $X_2 + 3/2\ S_2 - 1/2\ S_3 = 5/2$

or

$$X_2 + (1 + 1/2)\ S_2 + (-1 + 1/2)\ S_3 = (2 + 1/2)$$

So, the new constraint is $-1/2\ S_2 - 1/2\ S_3 \leq -½$

In standard form this is $-1/2\ S_2 - 1/2\ S_3 + S_4 = -½$

We add this to the previous tableau to obtain,

	X_1	X_2	S_1	S_2	S_3	S_4	b
Z	0	0	0	2	1	0	25
X_1	1	0	0	-2	1	0	1
X_2	0	1	0	3/2	-1/2	0	5/2
S_1	0	0	1	2	- 3	0	1
S_4	0	0	0	-1/2	-1/2	1	-1/2

Use dual simplex method to pivot, giving S_4 to leave the basis and finding Min{|2/(-1/2)|, 1/(-1/2)|} = Min{4, 2} => S_3 enters:

	X_1	X_2	S_1	S_2	S_3	S_4	b
Z	0	0	0	1	0	2	24
X_1	1	0	0	-3	0	2	0
X_2	0	1	0	1	0	- 1	3
S_1	0	0	1	5	0	- 6	4
S_3	0	0	0	1	1	- 2	1

Hence, the optimal integer solution is $X_1^* = 0$, $X_2^* = 3$, $Z^* = 24$.

Note: The "rounded" solution would be $X_1 = 1$, $X_2 = 2$, $Z = 21$ – quite a bit poorer.

2.3 Branch & Bound Method

Principles (in the case of a maximisation problem):-

- start with solution to "relaxed" (or continuous) problem (no integer restrictions).
- set of feasible non-integer solutions is branched into subsets eliminating continuous solutions not satisfying integer requirements.
- for each subset,
  Optimal "relaxed" solution = upper bound
  Best integer solution = lower bound
- subsets with upper bounds < current lower bound are eliminated
  – Said to be ***fathomed***.

Steps
- Branching Step
- Bounding Step
- Fathoming Step

Stopping rule - Optimum if there are no unfathomed subsets. Else go to branching step.

Branching rule (for subset selection):
Best bound rule:- select subset with most favourable bound (most promising).

Example (same as previously solved by Cutting Plane method):

Max $Z = 5X_1 + 8X_2$

s.t. $7X_1 + 8X_2 \leq 28$
$X_1 + 2X_2 \leq 6$
$X_1, X_2 \geq 0$ integers

Problem 1
Solution (by Simplex) $X_1 = 4/3,\ X_2 = 7/3,\ Z = 76/3$ = upper bound
$X_1 = 0,\ X_2 = 0,\ Z = 0$ = lower bound.

Branching Step:
$X_1 = 1\frac{1}{3}$
For X_1 = integer, either $X_1 \leq 1$ or $X_1 \geq 2$

[In general $X_j = b_j$ (b_j non-integer)
Either $X_j \leq [\, b_j \,]$
Or $X_j \geq [\, b_j \,] + 1$
where $[\, b_j \,]$ = largest integer $\leq b_j$]

Branch Problem 1 into two problems:-

Problem 2	Problem 3
Original problem plus constraint $X_1 \leq 1$	Original problem plus constraint $X_1 \geq 2$
Solution (Simplex):	Solution (Simplex):
$X_1 = 1$	$X_1 = 2$
$X_2 = 5/2$	$X_2 = 7/4$
$Z = 25$	$Z = 24$
(Upper bound)	*(Upper bound)*

Neither solution is (fully) integer

Branching Step:
Maximisation, so branch Problem 2 (higher upper bound => most promising)
Form 2 new problems:-
- Problem 2 + constraint $X_2 \leq 2$ (Problem 4)
- Problem 2 + constraint ` $X_2 \geq 3$ (Problem 5)

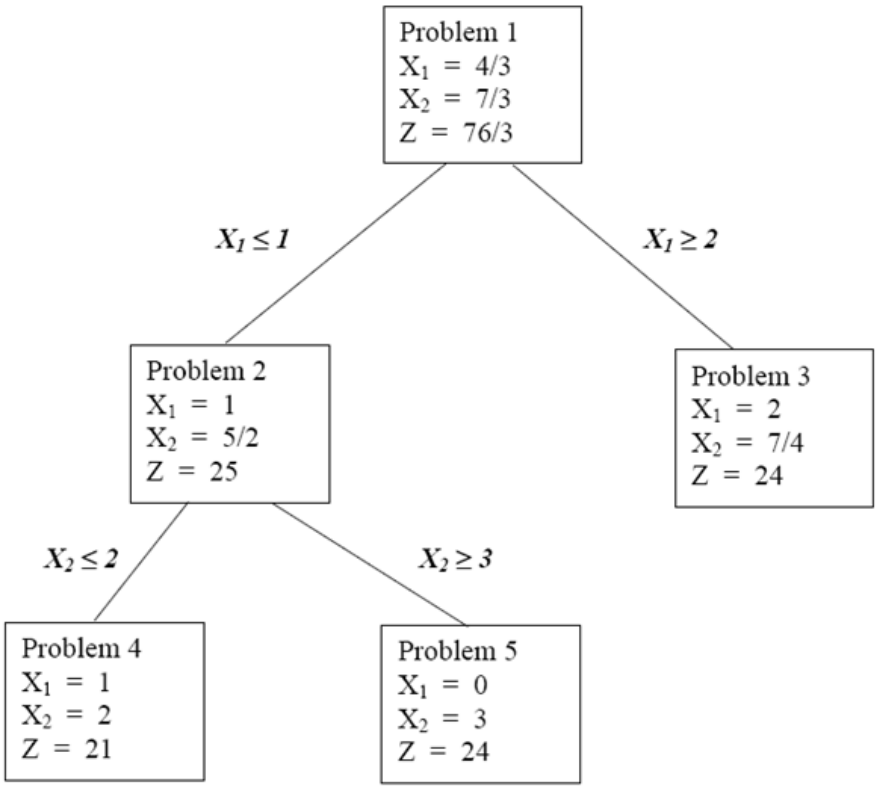
Problem 4 Solution:	Problem 5 Solution:
$X_1 = 1$	$X_1 = 0$
$X_2 = 2$	$X_2 = 3$
$Z = 21$	$Z = 24$

Therefore $Z = 24$ = New Lower Bound.

Note: Could branch Problem 3 but will get $Z < 24$. Therefore Problem 3 is fathomed.

Conclusion: Optimum is solution to Problem 5.

Summary:



2.4 Zero-One programming

Special features of these problems – decision variables = 0 or 1 => makes solution easy.
Could evaluate all solutions ("brute force"), but in practice some are excluded (fathomed).

Problem form:-

$$\textbf{Minimise} \quad \sum_{j=1}^{n} c_j \, x_j$$

Subject to

$$\sum_{j=1}^{n} a_{ij} \, x_j \leq (\text{or} \geq) \quad b_i \quad \text{for all } i = 1 \ldots m$$

$$x_j = 0 \text{ or } 1 \quad \text{for all } j$$

$$c_j \geq 0$$

Example:-
Min $Z = Y_1 + 8Y_2 - 3Y_3 + 7Y_4 - 2Y_5 + 8Y_6$
subject to $Y_j = 0, 1$
$5Y_1 - Y_2 - 2Y_3 - 3Y_4 + 3Y_5 + 2Y_6 \geq -4$
$-Y_1 + 3Y_2 - 2Y_3 + Y_4 - 3Y_6 \geq -3$
$Y_1 + 5Y_2 + Y_3 - 2Y_5 - Y_6 \geq 2$

Note: Before showing our preferred solution method (branch & bound) we illustrate the "brute force" approach (done here in Mathematica)

```
constraint1[y1_, y2_, y3_, y4_, y5_, y6_] := 5 * y1 - y2 - 2 * y3 - 3 * y4 + 3 * y5 + 2 * y6 + 4

constraint2[y1_, y2_, y3_, y4_, y5_, y6_] := -y1 + 3 * y2 - 2 * y3 + y4 - 3 * y6 + 3

constraint3[y1_, y2_, y3_, y4_, y5_, y6_] := y1 + 5 * y2 + y3 - 2 * y5 - y6 - 2

obj[y1_, y2_, y3_, y4_, y5_, y6_] := y1 + 8 * y2 - 3 * y3 + 7 * y4 - 2 * y5 + 8 * y6

feasSet = {};

Do[If[constraint1[y1, y2, y3, y4, y5, y6] ≥ 0 &&
    constraint2[y1, y2, y3, y4, y5, y6] ≥ 0 && constraint3[y1, y2, y3, y4, y5, y6] ≥ 0,
  feasSet = Append[feasSet, {y1, y2, y3, y4, y5, y6, obj[y1, y2, y3, y4, y5, y6]}]],
 {y1, 0, 1}, {y2, 0, 1}, {y3, 0, 1}, {y4, 0, 1}, {y5, 0, 1}, {y6, 0, 1}]

Length[feasSet]

33

feasSet = Sort[feasSet, #1[[7]] < #2[[7]] &];

TableForm[Transpose[feasSet]]
```

ableForm=

1	0	1	1	0	1	0	1	0	1	0	1	0	1	1	0	0	1	1	0	1	0	1	0	1	0	1	0	1	0
0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	1	1	1	1	0
0	0	0	1	0	0	0	0	0	0	1	1	0	0	1	0	1	0	1	0	0	1	1	0	0	1	1	1	1	1
0	1	1	0	0	0	1	1	0	0	1	1	1	1	0	0	1	0	1	1	1	0	0	0	0	1	1	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1	0	1	1	0	0	1	1	1	1	1	1	1
-2	3	4	5	5	6	6	7	8	9	10	11	11	12	13	13	13	14	14	14	15	15	16	16	17	18	19	20	21	21

Thus, the optimal solution (first column) is $Y_1 = 1$, $Y_2 = 0$, $Y_3 = 1$, $Y_4 = Y_5 = Y_6 = 0$ with Z = -2.

Branch & Bound solution method:
First re-order to get $|c_1| \leq |c_2| \leq \ldots\ldots |c_n|$, that is

Min $Z = Y_1 - 2Y_5 - 3Y_3 + 7Y_4 + 8Y_2 + 8Y_6$
subject to $Y_j = 0, 1$
$5Y_1 + 3Y_5 - 2Y_3 - 3Y_4 - Y_2 + 2Y_6 \geq -4$
$-Y_1 - 2Y_3 + Y_4 + 3Y_2 - 3Y_6 \geq -3$
$Y_1 - 2Y_5 + Y_3 + 5Y_2 - Y_6 \geq 2$
Let $Y_1 = X_1$
$Y_5 = (1 - X_2)$
$Y_3 = (1 - X_3)$
$Y_4 = X_4$
$Y_2 = X_5$
$Y_6 = X_6$
Objective function becomes:-

Min $Z(X) = X_1 - 2(1 - X_2) - 3(1 - X_3) + 7X_4 + 8X_5 + 8X_6$

Hence, the complete problem in terms of the new variables is:-

Minimise $Z(X) = X_1 + 2X_2 + 3X_3 + 7X_4 + 8X_5 + 8X_6 \quad (-5)$

Subject to

$$5X_1 - 3X_2 + 2X_3 - 3X_4 - X_5 + 2X_6 \geq -5$$
$$-X_1 + 2X_3 + X_4 + 3X_5 - 3X_6 \geq -1$$
$$X_1 + 2X_2 - X_3 + 5X_5 - X_6 \geq 3$$

$$X_j = 0, 1$$

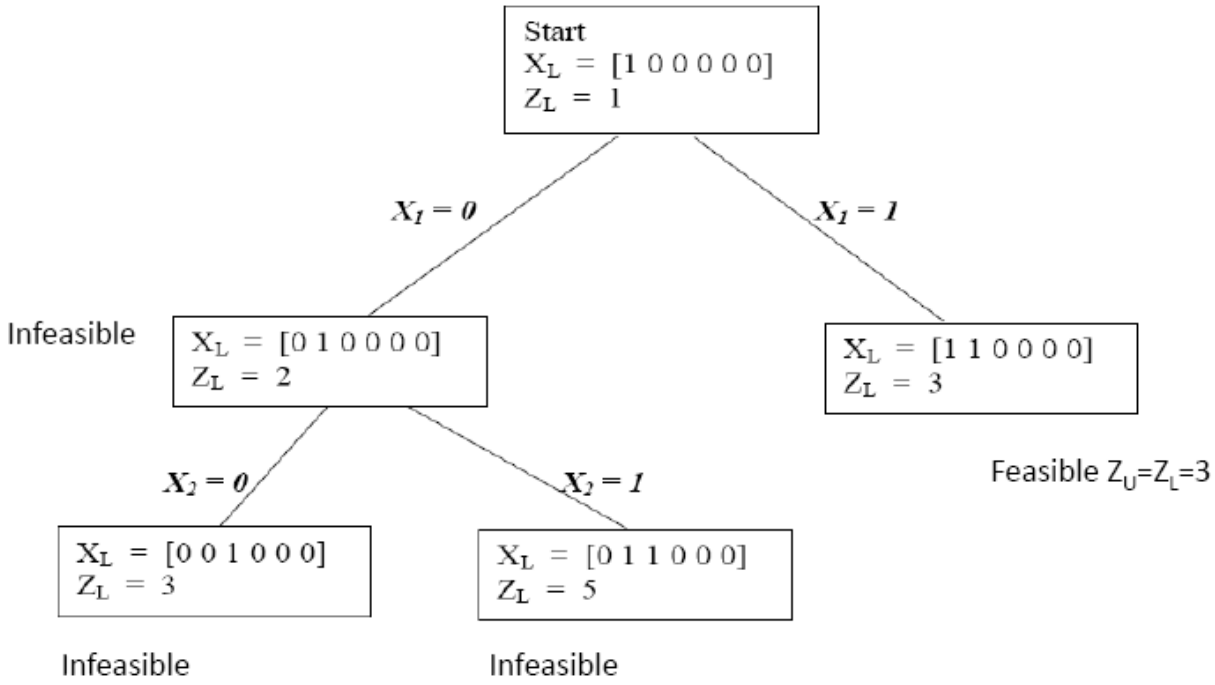
Next, we describe the solution algorithm & apply it to this example.

Zero-one programming Algorithm (minimisation problems)
Step 0 Check is $\underline{x} = \underline{0}$ feasible. If yes, STOP.
If not, set upper bound $Z_U = \sum c_j$ (i.e. $\underline{x}_U = [1\ 1\ \ldots.\ 1]$)
Lower bound $Z_L = c_1$. $\underline{x}_L = [1\ 0\ 0\ 0\ \ldots..\ ]$
If $\underline{x}_L$ feasible, STOP.
Set $k = 1$
Step 1 (**Branch**) Partition into $x_k = 0$ and $x_k = 1$
Step 2 (**Bound**) Set $\underline{x}_L$ with $x_{k+1} = 1$. Use $\underline{x}_L$ to determine new Z_L.
Step 3 (**Fathom**) Fathom if any of following hold
(1) $Z_{\underline{L}} \geq Z_{\underline{U}}$
(2) $\underline{x}_L$ is feasible $\rightarrow \underline{x}_L$ = partial solution - Set $Z_{\underline{U}} = Z_{\underline{L}}$
(3) a constraint cannot be satisfied by any $\underline{x}_L$ in subset
Step 4
If all subsets fathomed, STOP
Else set k = k + 1
Go to Step 1

Application of algorithm to the example:
Step 0
$\underline{x} = [0\ 0\ 0\ 0\ 0\ 0]$ is infeasible. Set $Z_U = 29$
$x_L = [1\ 0\ 0\ 0\ 0\ 0\ ]$. $Z_L = 1$. ($\boldsymbol{X}_L$ *is not feasible so must go to Step 1)*
Set $k = 1$

Step 1: Branch $X_1 = 0$, $X_1 = 1$.

The complete solution process is depicted as follows:

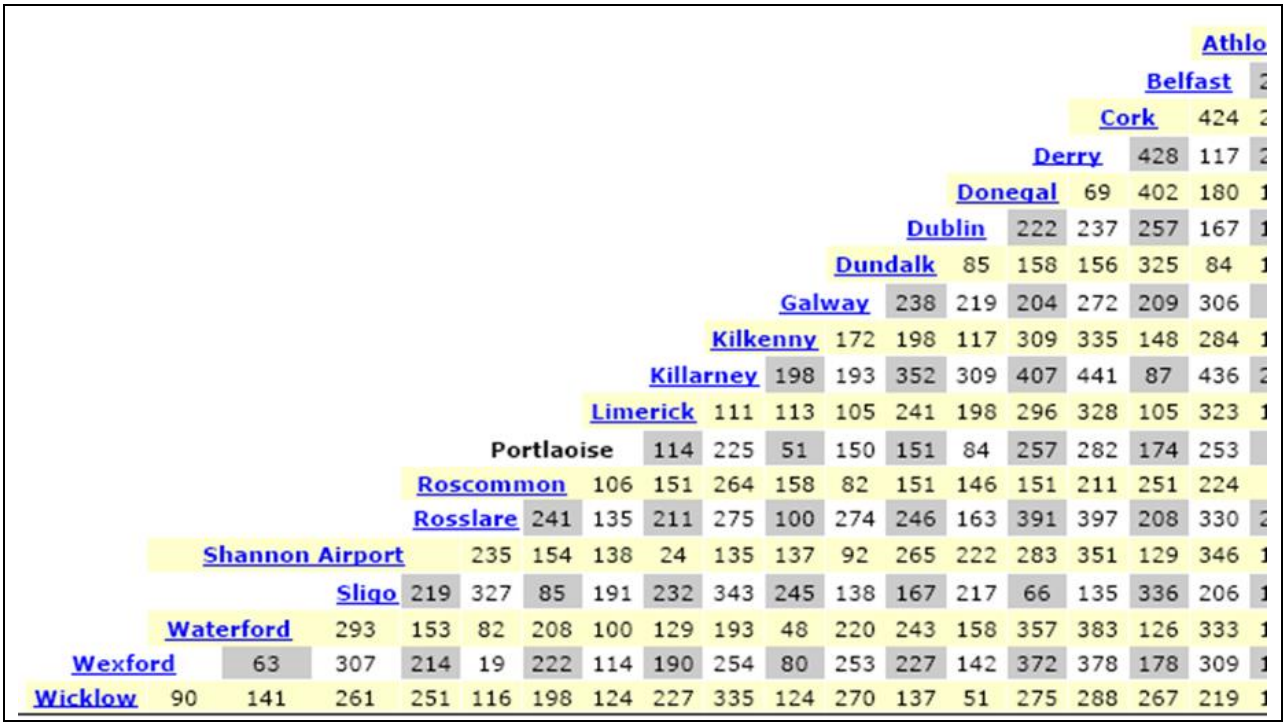


Thus, the optimal solution is : $X^* = [1\ 1\ 0\ 0\ 0\ 0]$
$Z^* = 3$ (= -2 of original)
$Y_1 = 1, Y_5 = 0, Y_3 = 1, Y_4 = 0, Y_2 = 0, Y_6 = 0.$

2.5 Brief look at Traveling Salesman etc

2.5.1 Raw data

These notes give a little background and some indicative calculations on the Traveling Salesman Problem (TSP) (Applegate, et al., 2007) which was one of the motivating problems for Linear Programming and especially Integer Programming.
For our example calculations we make use of part of the following distance chart. We focus on finding a route through 5 towns only Sligo Limerick Dundalk Dublin and Killarney



Athlo																
Belfast																
Cork	424															
Derry	428	117														
Donegal	69	402	180													
Dublin	222	237	257	167												
Dundalk	85	158	156	325	84											
Galway	238	219	204	272	209	306										
Kilkenny	172	198	117	309	335	148	284									
Killarney	198	193	352	309	407	441	87	436								
Limerick	111	113	105	241	198	296	328	105	323							
Portlaoise	114	225	51	150	151	84	257	282	174	253						
Roscommon	106	151	264	158	82	151	146	151	211	251	224					
Rosslare	241	135	211	275	100	274	246	163	391	397	208	330				
Shannon Airport	235	154	138	24	135	137	92	265	222	283	351	129	346			
Sligo	219	327	85	191	232	343	245	138	167	217	66	135	336	206		
Waterford	293	153	82	208	100	129	193	48	220	243	158	357	383	126	333	
Wexford	63	307	214	19	222	114	190	254	80	253	227	142	372	378	178	309
Wicklow	90	141	261	251	116	198	124	227	335	124	270	137	51	275	288	267	219

2.5.2 Establishing lower bounds on route

2.5.2.1 Use of control zones

This approach follows (Applegate, et al., 2007) (pages 24 to 29).

The idea:

- Place a control zone, in the form of a circle or disk, around each city.
- The city is the centre of the circle.
- The circles are required not to overlap.
- Let r_i = radius of control circle for city i. Clearly, $r_i >= 0$
- To visit city i must travel at least $2r_i$ (r_i to arrive and depart)
- In order for the lower bound to be useful, it must be maximised. Hence, we seek to maximise $2\Sigma r_i$.
- Constraints are that the sum of control radii for any 2 cities is at most the distance between these cities (as otherwise the circles overlap). For n cities this means there are ${}^{n}C_{r}$ such constraints.

Using *Mathematica* (***try it with AMPL***) to do the calculations we have:

Lower bounds on variables :

```
In[16]:= lu = {0, 0, 0, 0, 0};
```

Basic set of constraints :

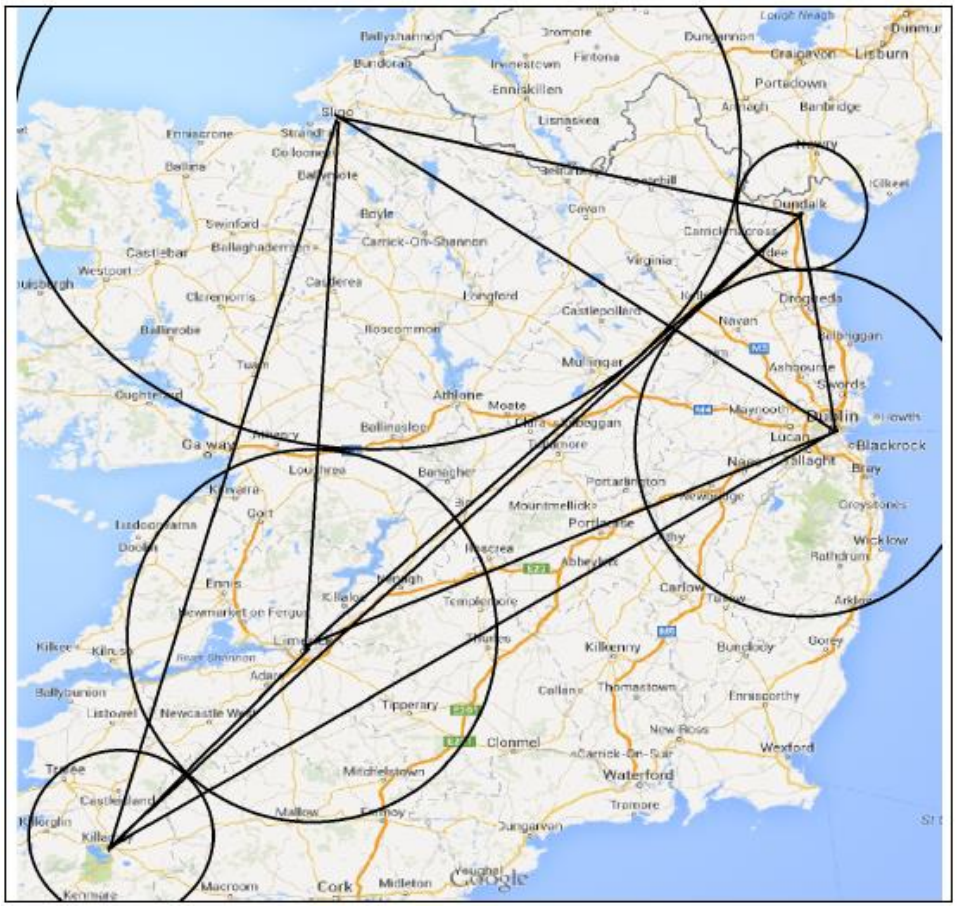
```
In[17]:= C1 = {1, 1, 0, 0, 0};
         C2 = {1, 0, 1, 0, 0};
         C3 = {1, 0, 0, 1, 0};
         C4 = {1, 0, 0, 0, 1};
         C5 = {0, 1, 1, 0, 0};
         C6 = {0, 1, 0, 1, 0};
         C7 = {0, 1, 0, 0, 1};
         C8 = {0, 0, 1, 1, 0};
         C9 = {0, 0, 1, 0, 1};
         C10 = {0, 0, 0, 1, 1};
```

```
Obj = {-2, -2, -2, -2, -2};

radii = LinearProgramming[Obj, {C1, C2, C3, C4, C5, C6, C7, C8, C9, C10},
  {{232, -1}, {343, -1}, {167, -1}, {217, -1}, {111, -1}, {241, -1},
   {198, -1}, {352, -1}, {309, -1}, {85, -1}}, lu]
```

$$\left\{\frac{299}{2}, \frac{165}{2}, \frac{57}{2}, \frac{35}{2}, \frac{135}{2}\right\}$$

```
lowerBound = Apply[Plus, radii] * 2

691
```

So the lower bound found by this method is 691 km. Given the above solution, that is (r_1, r_2, r_3, r_4, r_5) = (299/2, 165/2, 57/2, 35/2, 135/2), we may depict the optimal control zones (roughly) as follows:



Note: It should be noted that this method of control zones may sometimes give very poor lower bounds, depending on the "topology" of the city connections.
For example, do you think that this method would give a good lower bound if the places on the route were to be Sligo, Roscommon, Donegal, Waterford, Rosslare and Wexford?

2.5.2.2 Solution approach via Relaxed Problem first

This approach follows (Applegate, et al., 2007) (pages 81 to 86).
Decision variables: We introduce decision variables corresponding to each road between a pair of cities. These variables are special in that they must be either zero or 1.
For example,

X_{Sli_Lim} = **1/0** if the road from Sligo to Limerick **is/is not** on the route

So, if there are **n** cities then there are $^{n}C_{2}$ decision variables.
Constraints: for each city there is only one road in and only one road out that are part of the route. So, for Sligo say, we must have

$$X_{Sli\_Lim} + X_{Sli\_Dun} + X_{Sli\_Dub} + X_{Sli\_Kil} = 2$$

For n cities, there are n such constraints.

The idea here is to first solve the relaxed problem and then, if necessary, add additional constraints to force the required integral solution. The relaxed problem only requires that, for example,

$0 <= X_{Sli\_Lim} <= 1$

Using *Mathematica* (or similar) we have, for the relaxed problem,

```
interpretation = {"Sli-Lim", "Sli-Kil", "Sli-Dun", "Sli-Dub", "Lim-Kil",
   "Lim-Dun", "Lim-Dub", "Kil-Dun", "Kil-Dub", "Dun-Dub"};

ObjMin = {232, 343, 167, 217, 111, 241, 198, 352, 309, 85};

LbUb = {{0, 1}, {0, 1}, {0, 1}, {0, 1}, {0, 1}, {0, 1}, {0, 1}, {0, 1}, {0, 1}, {0, 1}};

con1 = {1, 1, 1, 1, 0, 0, 0, 0, 0, 0};
con2 = {1, 0, 0, 0, 1, 1, 1, 0, 0, 0};
con3 = {0, 1, 0, 0, 1, 0, 0, 1, 1, 0};
con4 = {0, 0, 1, 0, 0, 1, 0, 1, 0, 1};
con5 = {0, 0, 0, 1, 0, 0, 1, 0, 1, 1};

In[54]:= tour = LinearProgramming[ObjMin, {con1, con2, con3, con4, con5},
           {{2, 0}, {2, 0}, {2, 0}, {2, 0}, {2, 0}}, LbUb]

Out[54]= {0, 1, 1, 0, 1, 0, 1, 0, 0, 1}

In[55]:= ObjMin.tour

Out[55]= 904

In[58]:= TableForm[{tour, interpretation}]

Out[58]//TableForm=
0       1       1       0       1       0       1       0       0       1
Sli-Lim Sli-Kil Sli-Dun Sli-Dub Lim-Kil Lim-Dun Lim-Dub Kil-Dun Kil-Dub Dun-Dub
```

We can see that it has turned out that this is, in fact, the optimal integral solution. However, if it were not the optimal integral solution, then the relaxed solution would have provided a lower bound for the route length. For a discussion of how one might add extra constraints to the solution of the relaxed problem see (Applegate, et al., 2007) (page 84 etc).

2.5.2.3 Note on Lin-Kernighan Heuristic – (Applegate, et al., 2007) p103 etc

The main focus in TSP research has been to find provably optimal solutions to the problem. However, another important research direction has been to find tours (routes) that, while not optimal, are nevertheless of low cost. The resulting methods

are "tour improvement methods". Of these, the Lin-Kernighan algorithm has proved one of the most successful.

2.5.2.4 Applying Branch & Bound method to foregoing problem

We need to re-order the variables according to increasing distance:

x	Dun-Dub	Lim-Kil	Sli-Dun	Lim-Dub	Sli-Dub	Sli-Lim	Lim-Dun	Kil-Dub	Sli-Kil	Kil-Dun
y	1	2	3	4	5	6	7	8	9	10

Then the objective is to

Minimize $85y_1+111y_2+167y_3+198y_4+217y_5+232y_6+241y_7+309y_8+343y_9+352y_{10}$

Subject to (the re-ordered constraints):

(C1) $y_3 + y_5 + y_6 + y_9 = 2$
(C2) $y_2 + y_4 + y_6 + y_7 = 2$
(C3) $y_1 + y_3 + y_7 + y_{10} = 2$
(C4) $y_1 + y_4 + y_5 + y_8 = 2$

$\mathbf{y}$ = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0] is infeasible.
Z_U = Sum of all 10 distances!
Hence, set $\mathbf{y}_L$ = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0] – infeasible, $Z_L = 85$

Branch $y_1 = 0$, $y_1 = 1$

y_L = [0, 1, 0, 0, 0, 0, 0, 0, 0, 0] Z_L = 111 Not feasible	y_L = [1, 1, 0, 0, 0, 0, 0, 0, 0, 0] Z_L = 196 Not feasible

Branch again etc.

2.6 Concluding notes

We have presented some methods – through (simple) examples - for solving Integer Programming problems, namely

General Integer Programming Problems:
- Cutting Plane method
- Branch & Bound

Zero-One Problems:
- Branch & Bound

However, it is important to realise that Integer Programming problems are, in general, difficult to solve. The following notes summarise some of the issues, including pointers on strategies that can be helpful in tackling these problems.

1. Continuous (real) linear programming problems are easier to solve than integer programming problems. A key difference is that the optimal solution(s) of continuous problems occur at extreme points (corners) of the feasible region but this is not usually true for integer programming problems.

2. Modern software packages for Integer (as well as continuous) linear programming problems use some kind of **pre-processing** to improve bounds on variables and to eliminate redundant constraints (if there are any) in order to have an easier problem to solve.

3. Integer programming problems may be intractable in the sense that the computational time to solve some instances can be extremely long, and there probably is no algorithm that solves every such problem efficiently. However, there are **certain sub-classes of Integer programming problems** for which efficient methods exist. In particular, special algorithms have been found for integer-valued network flow problems (***check out!***).

4. **Complete enumeration** ("brute force") is the simplest method for Integer programming problems but is only practical for a very small number of variables. Thereafter there is a ***combinatorial explosion***.

5. We have presented some examples of the **Branch and Bound** approach. In general, the idea is to eliminate as many of the possible solutions as we can without actually evaluating them.
A key step is that of "**fathoming a node**" which is what makes this approach better than complete enumeration. By fathoming, we are implicitly exploring a sub-problem and concluding that an integer solution that is better than our current best known integer solution does not exist for this sub-problem. There are various strategies for choosing which nodes to branch on etc.

6. **Cutting plane algorithms** were first introduced by Gomory and were proved by him to converge after a finite number of iterations. However, for a long time, it was believed that in practice a very large number of iterations were often needed. However, it was shown more recently that, if properly used, cutting plane algorithms can be computationally effective.

7, Sometimes, a **Cutting plane** algorithm followed by a **Branch and Bound** algorithm are used together.

8. **Heuristic methods** are also available. These aim to find a good solution efficiently but are not guaranteed to always find the optimal solution(s). Examples include Simulated Annealing and Genetic Algorithms.

2.7 Problems

Problem 1: Use the cutting plane method to solve the following:

$$
\begin{array}{ll}
\text{Max} & X_1 + 2X_2 \\
\text{Subject to} & 2X_1 + 5X_2 \leq 16 \\
 & 6X_1 + 5X_2 \leq 30 \\
 & X_1 \, , \; X_2 \geq 0, \text{ integers}
\end{array}
$$

As a starting point, you are given that the optimal tableau to the relaxed (non-integer) problem is as follows (check):

	X_1	X_2	S_1	S_2	b
Z	0	0	7/20	1/20	71/10
X_2	0	1	3/10	- 1/10	9/5
X_1	1	0	- 1/4	1/4	7/2

Problem 2: Solve Problem 1 using the branch-and-bound method.

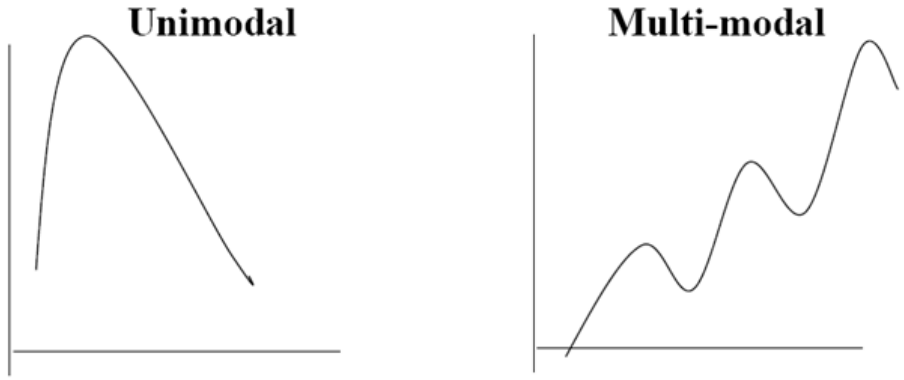
3. Non-Linear Programming

3.1 Overview of Constrained optimization

This introductory section introduce somes of the basic terminology and ideas used in non-linear programming. Subsequently, we will focus on some specific types of problems.

Non-linear problems are characterised by non-linear terms:- e.g. *Sin(X) or* X_1X_2 *or* X^2 etc.

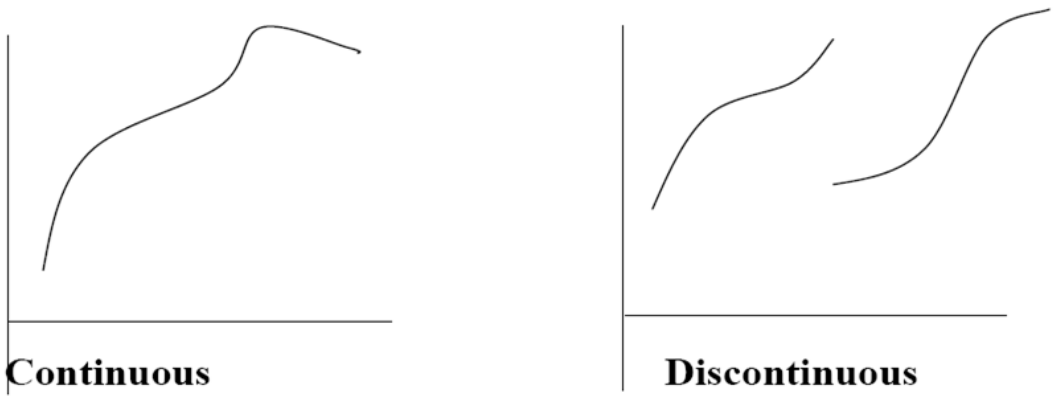
A key distinction from linear problems: The solution may not lie at an extreme point of the solution space.
Examples:



Turning points may be: **Local maxima**, **Local minima**, or **Inflection points**. All are known as Stationary points.

"Overall best" → Global optimum.

Types of function:



For continuous functions stationary points are located using calculus.

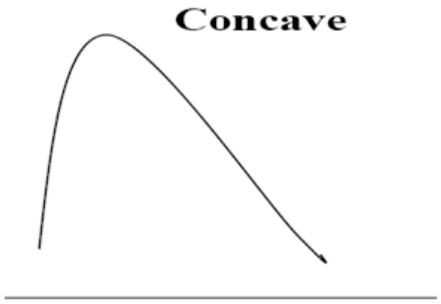
Definition of a Convex Function: Given any two points $\underline{X}_1$ and $\underline{X}_2$, if

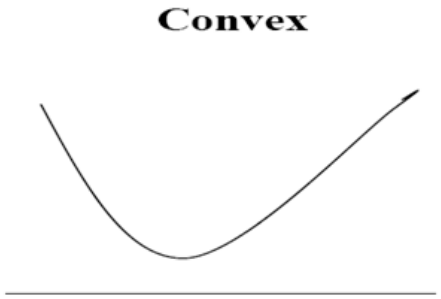
$$f[(1-\theta)\underline{X}_1 + \theta\underline{X}_2] \leq (1-\theta)f(\underline{X}_1) + \theta f(\underline{X}_2), \quad 0 \leq \theta \leq 1$$

then $f(\underline{X})$ is a Convex Function.

Reverse inequality direction → Concave Function.

So, a line drawn between two points on the function is either above (convex) or below (concave) the function.

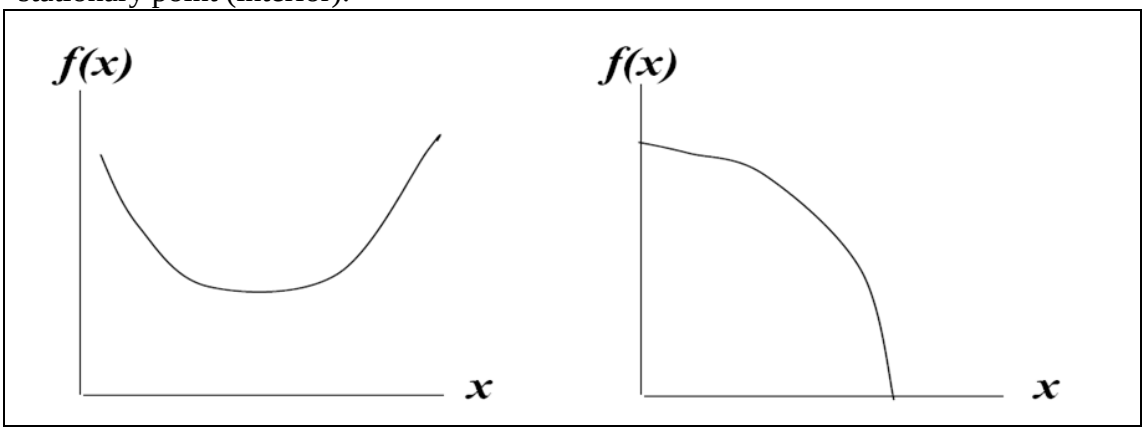




Note: A convex region in 2 or higher dimensions has the property that if a straight line segment is drawn between any 2 points of the region then the segment lies in the region.

The Search for Optimal Solutions:

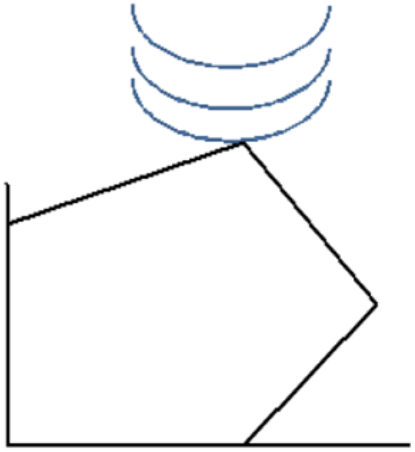
Unconstrained problem: Optimal solution is either at a boundary point or at a stationary point (interior).



Min is at a stationary point (interior) Max is at a boundary point.

Both Min and max are at boundary points

- **Maximisation**:- If objective function is concave and solution space is convex, there is a unique optimum at a stationary point.
- **Minimisation**:- If objective function is convex and solution space is convex, there is also a unique optimum at a stationary point.



Note that in linear programming the objective function is a straight line (& so, both convex and concave) and the solution space is convex.

Note: In Ca4011 this year, we will (probably) look at **constrained** problems only. (***TBC***)

3.2 Use of Lagrange multipliers

Idea: Convert a constrained problem into an unconstrained problem.

(i) Problem with equality constraints:-

Optimise $f(\underline{x})$

s.t. $g_i(\underline{x}) = b_i \quad i = 1 \ldots\ldots M$ (equality constraints)

Note $X_j \geq 0$ is not implicit, that is, variables may be of either sign unless explicitly stated otherwise.

Lagrangian Function:

$$L(\underline{x}, \underline{\lambda}) = f(\underline{x}) + \sum_{i=1}^{m} \lambda_i [g_i(X_1 \ldots\ldots X_n) - b_i]$$

NB: Optimising $L(x, \lambda)$ is equivalent to optimising the original constrained problem.

The quantities λ_i are called Lagrange multipliers.
We sometimes write $\underline{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_m)$.

Method: Differentiate $L(\underline{x}, \underline{\lambda})$ with respect to X's and λ's and apply the Kuhn-Tucker conditions. Then solve the resulting simultaneous equations.

Kuhn-Tucker Conditions (stated without proof): At optimum

$\delta L / \delta X_j = 0 \qquad j = 1 \ldots\ldots n$

And $\delta L / \delta \lambda_i = 0 \qquad i = 1 \ldots\ldots m$ (the original constraints)

Note: Some authors state these conditions in a somewhat different, more mathematically precise way but the above is sufficient for our purposes.

(ii) Problem with inequality constraints:-
In general $g_i(x) \leq b_i$ or $g_i(x) \geq b_i$
These constraints can be converted to equality constraints using slack or surplus variables (see previous linear programming material).

$$g_i(\underline{x}) \text{ +/- } S_i^2 = b_i$$

or $$-b_i + g_i(\underline{x}) \text{ +/- } S_i^2 = 0$$

Lagrangian function becomes:-

$$L(\underline{x}, \underline{S}, \underline{\lambda}) = f(\underline{x}) + \sum_{i=1}^{m} \lambda_i \, (-b_i + g_i(X_1 \ldots\ldots X_n) \text{ +/- } S_i^2)$$

Kuhn-Tucker: $\delta L / \delta X_j = \delta f / \delta X_j + \sum \lambda_i (\delta g_i / \delta X_j) = 0$ (1)
$\delta L / \delta S_i = +/- \, 2 \lambda_i S_i = 0$ (2)
i.e. either λ_i or S_i (or both) = 0
$\delta L / \delta \lambda_i = 0$ (original constraints) (3)

Note: In practice sets of equations (1) - (3) may be difficult to solve.

Interpretation of Lagrange Multipliers: λ_i is a measure of sensitivity of objective function value with respect to b_i (same as dual variables in linear programming).
N.B. Because of non-linearity → λ_i value does not remain constant.

3.3 Quadratic Programming

3.3.1 Definition and example

In this particular case, we have
- objective function is quadratic
- constraints are linear

Hence, it follows that the Kuhn-Tucker conditions lead to linear equations.

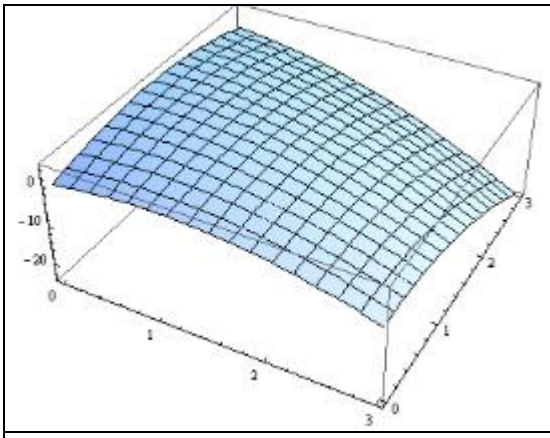
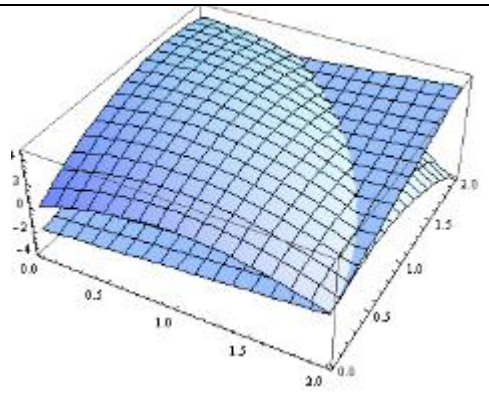
Example:-
Max $f(\underline{x}) = 4X_1 + 6X_2 - 2X_1^2 - 2X_1X_2 - 2X_2^2$

$$= (4 \quad 6)\begin{pmatrix} X_1 \\ X_2 \end{pmatrix} - \tfrac{1}{2}(X_1 \quad X_2)\begin{pmatrix} 4 & 2 \\ 2 & 4 \end{pmatrix}\begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$$

s.t. $X_1 + 2X_2 \leq 2$
$X_1 \geq 0, X_2 \geq 0$

Then, $L(\underline{x}, \underline{S}, \underline{\lambda}) = (4X_1 + 6X_2 - 2X_1^2 - 2X_1X_2 - 2X_2^2) + \lambda_1 (-2 + X_1 + 2X_2 + S_1^2) + \lambda_2 (X_1 - S_2^2) + \lambda_3 (X_2 - S_3^2)$

As this is a two dimensional problem we can depict it graphically to help clarify what is involved:

	
The objective function, X_1 and X_2 in [0, 3]	Objective function and constraint, X_1 and X_2 in [0, 2]

Kuhn-Tucker conditions imply:

$\delta L / \delta X_1 = 4 - 4X_1 - 2X_2 + \lambda_1 + \lambda_2 = 0$

$\delta L / \delta X_2 = 6 - 2X_1 - 4X_2 + 2\lambda_1 + \lambda_3 = 0$

Designating $T_i = S_i^2$

$\delta L / \delta \lambda_1 = X_1 + 2X_2 + T_1 - 2 = 0$

$\delta L / \delta \lambda_2 = X_1 - T_2 = 0$

$\delta L / \delta \lambda_3 = X_2 - T_3 = 0$

Either λ_i or T_i (or both) = 0

One method of solving these equations is to formulate as linear programming problem (outline)

Minimise $R_1 + R_2$ (or maximise $- R_1 - R_2$)

s.t. $4X_1 + 2X_2 - \lambda_1 - \lambda_2 + R_1 = 4$

$2X_1 + 4X_2 - 2\lambda_1 - \lambda_3 + R_2 = 6$

$X_1 + 2X_2 + T_1 = 2$

$X_1 \geq 0, \quad X_2 \geq 0$

At the optimum R_1 and $R_2 = 0$.

Solution via simplex method, but imposing during solution the conditions:-

λ_1 and / or $T_1 = 0$

λ_2 and / or $X_1 = 0$

λ_3 and / or $X_2 = 0$

At each iteration λ_i cannot enter the basis if the appropriate T_i or X_{i-1} is already basic, and vice versa. (called the "***Restricted Basis Entry Rule***")

Solution (in (Taha, 2007))

$X_1^* = 0.33 \qquad X_2^* = 0.83 \qquad f(\underline{x}^*) = 4.16$

$\lambda_1 = -1 \quad \lambda_2 = \lambda_3 = 0$

3.3.2 Outline of two applications of Quadratic Programming

We outline two applications of Quadratic Programming from (Galligani, et al., 2003)

Setting the scene (from (Galligani, et al., 2003)):

Consider the linearly constrained convex quadratic programming (QP) problem:

$$\begin{array}{ll} minimize & f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T G\boldsymbol{x} + \boldsymbol{q}^T\boldsymbol{x} \\ subject\ to & C\boldsymbol{x} = \boldsymbol{d}, \qquad A\boldsymbol{x} \geq \boldsymbol{b}, \end{array} \tag{1.1}$$

where G is a symmetric positive semidefinite matrix of order n, C is an $m_e \times n$ matrix of full row rank ($m_e \leq n$) and A is an $m_i \times n$ matrix. We

In CA4011 we do not need to get into the detailed mathematical properties of the matrices – suffice to say that convexity makes things easier! (Positive definite means that $x^TGx > 0$ if $x \neq 0$).

Application 1: Large QP Problems in Training Support Vector Machines
From Wikipedia (!): "*In machine learning, **support vector machines** (**SVMs**, also **support vector networks**) are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other … . An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.*"

Given a training set of labeled examples

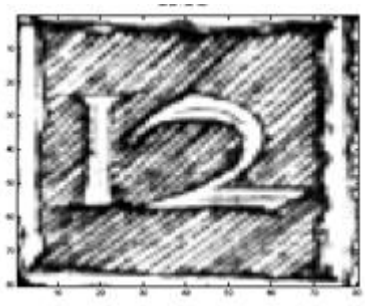
$$D = \{(\boldsymbol{p}_i, y_i),\ i = 1, \ldots, n, \quad \boldsymbol{p}_i \in R^m,\ y_i \in \{-1, 1\}\},$$

the SVM learning technique performs pattern recognition by finding a decision surface, $F : R^m \rightarrow \{-1, 1\}$, obtained by solving a quadratic program of the form:

$$\begin{array}{ll} minimize & \frac{1}{2}\boldsymbol{x}^T G\boldsymbol{x} - \sum_{i=1}^{n} x_i \\ subject\ to & \sum_{i=1}^{n} y_i x_i = 0, \qquad 0 \leq x_j \leq C, \qquad j = 1, \ldots, n, \end{array} \tag{1.7}$$

where G is a symmetric positive semi-definite matrix etc …

Application 2: Numerical Solution of an Image Restoration Problem
The image degradation process can be described as a system which operates on an input image (object) to produce an output image (degraded image). An additive noise term is also included.




The image restoration problem is an ***ill-conditioned*** problem. A well-known method for dealing with such problem is the method of regularization by Phillips and Tikhonov, that consists in solving the following minimization problem (which with a constraint leads to a QP problem):

$$minimize \ \|H\phi - \gamma\|^2 + \lambda\|\Lambda\phi\|^2$$

3.4 Geometric Programming (GP)

Objective and some or all of constraints of the form:-

$$\sum_{j=1}^{m} c_j \prod_{i=1}^{n} x_i^{a_{ij}}$$

$$c_j > 0 \quad , x_i > 0, a_{ij} \text{ unrestricted.}$$

Solution method:-
Transform original problem into one of solving a set of simultaneous equations, and then transform back to original variables.
Define new variables:-

$$Y_j = (c_j \prod_{i=1}^{n} x_i^{a_{ij}}) / z^*$$

Unconstrained problem – simpler than with constraints:-
Optimal values of Y_j are given by solution to:-

$$\sum_{j=1}^{m} a_{ij} Y_j = 0 \qquad i = 1 \ldots n \quad \text{(Orthogonality conditions)}$$

$$\sum_{j=1}^{m} Y_j = 1 \qquad \text{(Normality conditions)}$$

Also, it can be shown that,

$$Z^* = \prod_{j=1}^{m} (c_j / Y_j^*)^{Yj^*}$$

Example of an unconstrained problem: (see (Taha, 2007)):
Minimise
$Z = 7X_1X_2^{-1} + 3X_2X_3^{-2} + 5X_1^{-3}X_2X_3 + X_1X_2X_3$

So, Y1 = $7X_1X_2^{-1}/Z^*$, Y2 = $3X_2X_3^{-2}/Z^*$, Y3 = $5X_1^{-3}X_2X_3/Z^*$, Y4 = $5X_1^{-3}X_2X_3/Z^*$

This leads to solving (3 orthogonality conditions and 1 normality condition):-

$$\begin{aligned} Y_1 &- 3Y_3 + Y_4 = 0 \\ -Y_1 + Y_2 &+ Y_3 + Y_4 = 0 \\ &- 2Y_2 + Y_3 + Y_4 = 0 \\ Y_1 + Y_2 &+ Y_3 + Y_4 = 1 \end{aligned}$$

Solution (check!)
$Y_1^* = ½ \quad Y_2^* = 1/6 \quad Y_3^* = 5/24 \quad Y_4^* = 1/8$

Hence, using

$$Z^* = \prod_{j=1}^{m} (c_j / Y_j^*)^{Yj^*}$$

it follows that
$Z^* = \{ 7 / (1/2) \}^{1/2} \{ 3 / (1/6) \}^{1/6} \{ 5 / (5/24) \}^{5/24} \{ 1 / (1/8) \}^{1/8} = 15.22$
Then, since, by definition,

$$Y_j Z^* = c_j \prod_{i=1}^{n} x_i^{a_{ij}} \qquad j = 1 \ldots n$$

the X_i^* values can be obtained (i.e. we transform back to the original variables). Thus,
$(1/2)(15.22) = 7X_1X_2^{-1}$
$(1/6)(15.22) = 3X_2X_3^{-2}$
$(5/24)(15.22) = 5X_1^{-3}X_2X_3$
$(1/8)(15.22) = X_1X_2X_3$
Solution:-
$X_1^* = 1.315 \quad X_2^* = 1.21 \quad X_3^* = 1.20$

We conclude our brief discussion of Geometric Programming with some extracts from (Boyd, et al., 2007.

2.1 Monomial and posynomial functions

Let $x_1, \ldots, x_n$ denote n real positive variables, and $x = (x_1, \ldots, x_n)$ a vector with components x_i. A real valued function f of x, with the form

$$f(x) = c x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n}, \tag{1}$$

where $c > 0$ and $a_i \in \mathbf{R}$, is called a *monomial function*, or more informally, a *monomial* (of the variables $x_1, \ldots, x_n$). We refer to the constant c as the *coefficient* of the monomial, and we refer to the constants $a_1, \ldots, a_n$ as the *exponents* of the monomial. As an example, $2.3x_1^2x_2^{-0.15}$ is a monomial of the variables x_1 and x_2, with coefficient 2.3 and x_2-exponent -0.15.

A sum of one or more monomials, i.e., a function of the form

$$f(x) = \sum_{k=1}^{K} c_k x_1^{a_{1k}} x_2^{a_{2k}} \cdots x_n^{a_{nk}}, \qquad (2)$$

where $c_k > 0$, is called a *posynomial function* or, more simply, a *posynomial* (with K terms, in the variables $x_1, \ldots, x_n$). The term 'posynomial' is meant to suggest a combination of 'positive' and 'polynomial'.

This paper goes on to explain that "The main trick to solving a GP efficiently is to convert it to a nonlinear but *convex optimization problem*, i.e., a problem with convex objective and inequality constraint functions, and linear equality constraints." However, we will not get into the details.

Note: According to (Hoburg, et al., 2016) "GP has recently undergone a resurgence as researchers have discovered promising applications in statistics , digital circuit design, antenna optimization, communication systems, aircraft design, and other engineering fields."

3.5 Separable Programming

Method: Approximate a non-linear function by a series of linear segments.

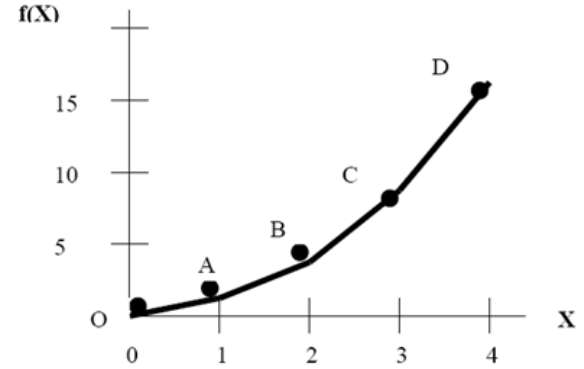
Function must be separable:

i.e. $f(X_1 \ldots\ldots X_n) = f_1(X_1) + f_2(X_2) + \ldots\ldots + f_n(X_n)$

e.g. $X_1^2 + X_2^2 + X_3^2$ is separable

but $4X_1 + 2X_1^2 + 2X_1X_2^2 - 6X_2^2$ is not.

Example $f(X) = X^2 \quad 0 \le X \le 4$



Choose grid points – e.g. X = 0, 1, 2, 3, 4.

$f(X)$ is approximated by O – A – B – C – D

New function - $c(W)$

For X = 2.25 $f(X) = 5.06$ and $c(W) = 5.25$ (see following)

Definition of c (W):

$c(W) = f(X_1)W_1 + f(X_2)W_2 + f(X_3)W_3 + f(X_4)W_4 + f(X_5)W_5$
$= 0\,W_1 + 1\,W_2 + 4\,W_3 + 9\,W_4 + 16\,W_5$

We also impose the conditions:

$W_1 + W_2 + \ldots + W_m = 1$ and

either one W_i or at most two adjacent W_i's are > 0.

Interpretation of W_i's:

$W_i = 1 - \lambda_i$ where λ_i is the proportion of the total distance from point i to point $(i + 1)$ or between point i and point $(i - 1)$.

e.g. At $X = 2.25$, between X_3 and X_4.

Therefore $W_1 = W_2 = W_5 = 0$.

$W_3 = 1 - \lambda_3 = (1 - 0.25) = 0.75$
$W_4 = 1 - \lambda_4 = (1 - 0.75) = 0.25$

λ_3 = distance from X_3 (2.0) to $X = 2.25$
λ_4 = distance from X_4 (3.0) to $X = 2.25$

$c(W) = 0 + 0 + 4(0.75) + 9(0.25) + 0 = 5.25$

Example:-

Maximise $(6X_1 - 3X_1^2) + X_2^2$

s.t. $X_1 + 2X_2 \leq 4$

in the range $0 \leq X_1 \leq 3$

$0 \leq X_2 \leq 2$

Approximate re-formulation in terms of linear segments:

Choose grid points – e.g. $X_1 = 0, 1, 2, 3$

$X_2 = 0, 0.5, 1.5, 2$

Note: The finer the grid the more computation that is involved.

Objective function becomes:-

$6(0U_1 + 1U_2 + 2U_3 + 3U_4) - 3(0U_1 + 1U_2 + 4U_3 + 9U_4)$
$+ 1(0V_1 + 0.25V_2 + 2.25V_3 + 4V_4)$

i.e. Max $3U_2 - 9U_4 + 0.25V_2 + 2.25V_3 + 4V_4$

Constraint ($X_1 + 2X_2 \leq 4$) becomes:-

$U_2 + 2U_3 + 3U_4 + V_2 + 3V_3 + 4V_4 \leq 4$

Also,

and either one, or at most two adjacent U's and V's are > 0.

Solution of re-formulated problem, a linear programming problem, turns out to be:-

$U_1 = U_4 = 0 \quad U_2 = 0.34 \quad U_3 = 0.66$
$V_3 = V_4 = 0 \quad V_1 = 0.12 \quad V_2 = 0.88$

→ $X_1 = 1\ (0.66) + 2\ (0.34) = 1.34$
$X_2 = 0\ (0.88) + 0.5\ (0.12) = 0.06$

Note: Number of decision variables increases as number of non-linear components increases.

N.B. "Fineness" of grid is a key parameter of the solution.

Note: The paper (Lu & Ito, 2003) proposes a particular approach for converting general nonlinear programming problems into separable programming problems. However, the first and second pages are of interest in that they set out the context for separable programming in general.
The idea is:

- (As we have seen) Approximate a separable non-linear programming problem by a linear programming problem that can be solved by Simplex Method (say).
- Previous stage (see above paper) "The assumption of separability of the objective function and the constraints in SP problems is a serious restriction in practical optimization problems. In order to generalize the Simplex method to solve general NLP problems, we need to convert non-separable objective function and/or the constraints into separable forms." ***There are various possibilities for doing this***.

3.6 Problems

Problem 1: Solve the following problem using the method of Lagrange Multipliers:

Minimise $(X_1 - 2)^2 + (X_2 - 1)^2$

subject to $X_1 - 2X_2 = -1$

Problem 2: Solve the following Geometric Programming problem:

Minimise $60X_1^{-3}X_2^{-2} + 50X_1^{3}X_2^{-2} + 20X_1^{-3}X_2^{3}$

Problem 3: Show how the following problem can be solved using Separable Programming (Do not solve the problem!)

Maximise $3X_1 + 2X_2$

subject to $4X_1^2 + X_2^2 \leq 16$

$X_1, X_2 \geq 0$

within the ranges $0 \leq X_1 \leq 2$
$0 \leq X_2 \leq 4$

4. Some Particular Topics

4.1 Introduction

This section includes a number of topics that are traditionally part of Operations Research though not normally included under Mathematical Programming. However, some aspects of these topics may be formulated as mathematical (usually linear) programming problems and so it is convenient to include them at this point.

4.2 Inventory Control or Management

4.2.1 Introduction

INVENTORY MANAGEMENT is concerned with

- When to place orders?
- How much to order?
-

There are three processes involved:

- Replenishment ---------→INVENTORY--------→ Demand (consumption)

There are various specific assumptions or special cases that can be considered:

- DEMAND INDEPENDENT OF INVENTORY LEVEL:-
  - Deterministic demand
  - Probabilistic demand
- DEMAND DEPENDENT ON INVENTORY LEVEL

For CA4011, we will focus on DETERMINISTIC MODELS:

- First Classical Inventory Models
- Second, briefly , variable demand rate models - role of linear programming

Assumptions of Classical Inventory (EOQ) Models

- Repetitive ordering
- Constant demand rate
- Constant lead time
- Continuous ordering

We present a number of models (more exist):

- BASIC EOQ (ECONOMIC ORDER QUANTITY) MODEL
- NON-ZERO LEAD-TIME (L) – What changes?
- QUANTITY DISCOUNTS – How to handle?
- CONTINUOUS RATE EOQ MODEL

WHEN TO USE EOQ MODELS?

- Need to test the assumption of constant demand rate
- If not justified use other methods, e.g. Linear Programming Or Dynamic Programming

4.2.2 Basic EOQ Model

EOQ = ECONOMIC ORDER QUANTITY

Assumptions

- Constant demand = D/year
- Order (set-up) cost = K/order

- Lead time = $L = 0$
- No stock-outs
- Inventory holding cost = h/unit/year
-

Derivation of formula for EOQ:

- Let inventory level = I
- Order when $I = 0$ (to prevent stock-outs)
- Let q = quantity ordered
- Therefore D/q = number of orders per year (no. of order cycles)
  - E.g. if D=1200/year and q = 100 then D/q = 12 orders per year

- Let **total cost/year** = **TC(q)** [*It depends on quantity ordered* ***q***]
- **TC(q)** is made up of three elements, that is,

TC(q) = order cost/year + purchasing cost/year + holding cost/year
where
Order cost/year = (order cost/order)(orders/year) = $(K)(D/q)$

Purchasing cost/year = (price/unit)(units purchased/year) = $(P)(D)$ ***(independent of q)***

Holding cost/year = (Inventory)(holding cost/unit/year) = $(I)(h)$

- Average inventory = ½ maximum I = q/2

Cycles:
Each year contains D/q cycles (as noted above)
Therefore length of a cycle = q/D
Average inventory = $q/2$ each cycle
Holding cost/year = (holding cost/cycle)(cycles/year)
Holding cost/cycle = $h(q/D)(q/2) = (q^2h)/(2D)$

Therefore, **Holding cost/year** = $((q^2h)/(2D))(D/q) = hq/2$

Hence, **TC(q)** = order cost/year + purchasing cost/year + holding cost/year
= (K)(D/q) + (P)(D) + hq/2

To find *q* such that *TC(q)* is minimised, set *TC′(q) = 0*

$T'(q) = -(KD/q^2) + (h/2) = 0$
Therefore $q = \pm (2KD/h)^{½}$
But a negative value of **q** makes no sense so we take the positive value,
$q^* = (2KD/h)^{½}$ (*optimal level of q – we use * to highlight this*)

Check the second derivative to make sure we have a minimum point (rather than a maximum!):

$TC''(q) = (2KD)/q^3 > 0$ for all q
i.e. TC(q) is a convex curve and q* is a <u>minimum</u>

Note:- Ratio of *K* to *h* is critical in determining *q**

If q* is ordered, **holding cost/year = order cost/year** (*i.e. balanced*)
Holding cost/year = $(hq^*/2) = (h/2)(2KD/h)^{½} = (KDh/2)^{½}$
Order cost/year = $(KD/q^*) = (KD)/(2KD/h)^{½} = (KDh/2)^{½}$

Example: An airline uses 500 tail lights per year. It costs €5 to place an order for a batch (of any size) of tail lights, and each light is valued at €0.40. For lights held in inventory, there is a holding cost estimated to be €0.08/light/year. Determine the optimal order batch size (EOQ). How many orders should the airline place per year and what should be the time gap between orders?

Solution: D = 500, K = 5, P = 0.40, h = 0.08
Hence, $\mathbf{q^* = (2KD/h)^{½} = (5000/0.08)^{½} = 250}$.
Hence, no. of orders per year = D/q* = 2
and time between orders = q*/D = ½ year or 6 months.

Note on sensitivity to value of q
Holding cost = hq/2= ½ (0.08 *q*) = 0.04 *q* [***HC(q)***]
Order cost = (K)(D/q) = 5(500/*q*) = 2500/*q* [***OC(q)***]
We calculate these costs, and the total cost (excluding the constant purchasing cost = *(P)(D) = 0.4(500) = 200*) to see how sensitive costs are with respect to the value of q:

q	HC(*q*)	OC(*q*)	Total Cost (*less PD*)
150	6.0	16.67	22.67
200	8.0	12.50	20.50
220	8.8	11.36	20.15
250	***10.0***	***10.00***	***20.00*** *– optimal value q**
280	11.2	8.93	20.13
300	12.0	8.33	20.33
400	16.0	6.25	22.25

We conclude that the results are not that sensitive (***i.e are robust***) to the q value.

4.2.3 Non-zero Lead-time (*L*)

Note: In deriving q* for the basic EOQ model we assumed that the lead time for ordering L = 0. We now relax this assumption.

Definition of Re-order point – Inventory level at which order should be placed.

We have two possibilities:

- (1) If demand in lead-time < EOQ (i.e. *(L)(D) < EOQ)*, re-order when *I = (L)(D).*
- (2) If *(L)(D) > EOQ, I* never reaches *(L)(D).*
-

Example: Suppose EOQ = 250, D = 500

- (1) Suppose L = 0.2 years. Then LD = 100 < 250. Therefore, re-order when I = 100. This ensures that the new order arrives just when inventory reaches zero.
- (2) Suppose L = 0.6 years. Then LD = 300 > 250. Therefore, I never reaches 300. So, the re-ordering policy does not need to be changed.

4.2.4 Quantity discounts

"Quantity Discounts" arise when price depends on quantity ordered.

In general, if q is the quantity ordered, we have

$0 \leq q < b_1$ → **Price =** p_1
$b_1 \leq q < b_2$ → **Price =** p_2
. .
. .
. .
$b_{k-2} \leq q < b_{k-1}$ → **Price =** p_{k-1}
$b_{k-1} \leq q$ → **Price =** p_k

$b_1 \ldots\ldots\ldots\ldots\ldots b_{k-1}$ **are price break points** and $p_1 > p_2 > \ldots > p_k$

We say that

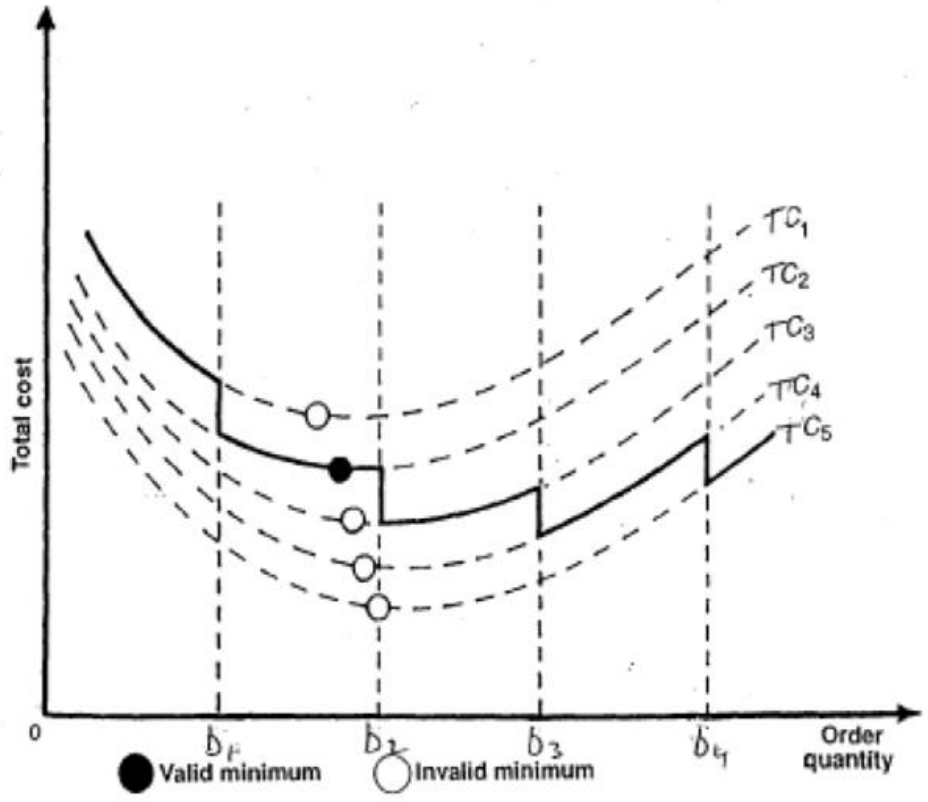
EOQ_i is **admissible** (valid) if $b_{i-1} \leq EOQ_i < b_i$

The value of **q** which minimises total cost (*TC)* is either

a break point
Or an *EOQ* point

Theorem: If EOQ_i is admissible, total cost is NOT minimised at a price higher than p_i .

Method:
-Begin with lowest price.
-Find q_i* which minimises total cost for $b_{i-1} \leq q < b_i$
-Continue until one q_i* is admissible.
-Find q_i* with the smallest total cost.



Example: Buying DVD's

# boxes ordered	**Price per box**
$0 \leq q < 100$	€50.00
$100 \leq q < 300$	€49.00
$300 \leq q$	€48.50

Annual requirement (D) = 1000 boxes
Ordering cost (K) = €100 per order
Holding cost (h) = 20% of purchase price per year.

Solution:
@ €48.50: $q = EOQ_3 = (2KD/h)^{½} = (100000/(0.2(48.50))^{½} = 101.5$
This is <300 and so is inadmissible.
@€49.00: $q = EOQ_2 = (2KD/h)^{½} = (100000/(0.2(49.00))^{½} = 101.0$
This satisfies $100 \leq q < 300$ and so is admissible.

Annual Cost	**q=300**	**q=101.0**	**q=100**
Purchase *(PD)*	*48,500*	*49,000*	*50,000*
Order *(KD/q)*	*333.3*	*990.1*	*1000*
Holding *(hq/2)*	*1455*	*490*	*500*
Total	**50288.3**	**50480.1**	**51500**

Conclude that it is most economical to buy orders of 300 (*CHECK CALCULATIONS!*)

4.2.5 Continuous Rate EOQ Model

Here we assume there is **internal production** (no shortages) at rate r units per year.

- q = number of units produced per run
- K = setup cost per production run
- h = holding cost per unit per year
- D = demand per year
-

Note: must have $r \geq D$ in order to meet demand

Batch of size q is produced at rate r. Therefore time taken to produce a batch = q/r.
For example, if q = 100 and r = 1200 per year, then q/r = 1/12 years or 1 month.

Inventory increases at rate r and depletes at rate D, so inventory increases at net rate $(r – D)$.

- *For example, if r = 1200 per year and D = 1000 per year then net rate of inventory increase is 200 per year.*

Therefore maximum inventory is $(q/r)(r – D)$

- *For example, with r and D as above, if q/r = 1/12 yrs then (q/r)(r–D) = 200/12 = 50/3 units (at the end of a batch production run or cycle).*

As inventory increases and decreases linearly, **average inventory** is just ½ maximum inventory i.e. **$½ (q/r)(r – D)$**
Thus,
holding cost per year = **$½(q/r)(r – D) h$**

Setup cost/year = (Setup cost per cycle).(cycles/year)=***K(D/q)***

Therefore, total cost per year (*TC*) = $hq(r - D)/(2r) + KD/q$

Hence, $TC' = d(TC(q))/dq = h(r - D)/(2r) - KD/q^2$
Set = 0 to minimise TC =>
$KD/q^2 = h(r - D)/(2r)$
So,
$q^* = \{2KDr/(h(r - D))\}^{½}$

In terms of the basic $EOQ = (2KD/h)^{½}$ model,
$q^* = \{2KD/h\}^{½}\{r/(r - D)\}^{½} = \{EOQ\}\{r/(r–D)\}^{½}$

Example: A company is required to provide 10,000 chassis per year. The value of a chassis is estimated to be €2,000. The plant has a capacity to produce at the rate of 25,000 per year, and the cost of setting up each production run is €200, regardless of the run size. Holding cost is calculated as 25% of the product value per annum. Determine the optimal production run size and the number of production runs which should take place each year.
Solution: D=10,000/year, r = 25,000/year, K=200, h=2000/4
Hence, $q^* = \{2(200)(10000)(25000)/(500(25000-10000))\}^{1/2}$
$=200/\sqrt{3} \approx 115.5$.
The optimal number of production runs per year is $D/q = 50\sqrt{3} = 87$ (in whole numbers). (***CHECK CALCULATIONS!***)

4.2.6 When to use EOQ Models?

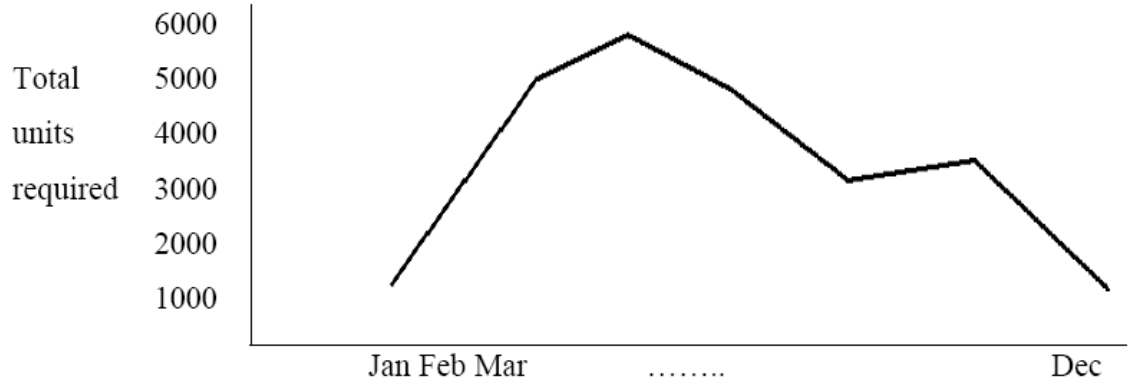
Test the assumption of constant demand:

- Observe demand for *n* periods: $d_1, d_2, \ldots\ldots d_n$
- Determine estimate of average demand per period:
  Mean(d) $= (\Sigma_i\, d_i)/n$
- Determine estimate of variance of demand:
  Var(d) $= (\Sigma_i d_i^2)/n - (\text{Mean(d)})^2$; **Sd(d)** $= \sqrt{\text{Var(d)}}$
- Determine estimate of Variability Coefficient (VC):
  VC = **Sd(d)/Mean(d)**
  - Decision criterion (*approximate*):
    - Use *EOQ* models if $VC < 0.2$
    - If $VC > 0.2$ use other methods (e.g. Dynamic Programming or Linear Programming)

4.2.7 Inventory Control & Parametric Linear Programming

We introduce the idea of **parametric** linear programming and show how ***inventory management & production-scheduling problems*** can be modelled using this approach. (Section 5.6 of (Fourer, et al., 2003) presents a related example).

Problem: The demand for a seasonal item is shown in the following sales-forecast chart:



The manufacturer's problem is to find a production schedule which minimises the costs due to output fluctuations and inventories. In fact, a solution is desired which balances these conflicting aspects. A mathematical model for this problem is set up next.

Model:
Let S_0 be the amount of the product in storage at the beginning of the first month. For a new product, $S_0 = 0$.
Let
X_t = No. of units produced in month t (Production)
R_t = No. of finished units that must be available in month t (Requirement)
S_t = No. of finished units that are not required in month t (Storage)
Clearly, X_t , R_t and $S_t \geq 0$ for all values of t.
Month 1: The production X_1 and the previously stored items S_0 for the 1st month must together be at least equal to the requirement $R_1 \Rightarrow X_1 + S_0 \geq R_1$
Thus, the storage S_1 in the first month satisfies $X_1+S_0-R_1=S_1$ or $X_1+S_0-S_1=R_1$
Similarly, for the second month, $X_2 + S_1 \geq R_2$ or $X_2 + S_1 - S_2 = R_2$
In general, for month **t**, we have

$$X_t + S_{t-1} - S_t = R_t \qquad (1)$$

For the **required smooth production pattern**, the production fluctuations $X_t - X_{t-1}$ between successive months must be minimised. Because any number may be written in the form of two non-negative numbers, we can write

$$X_t - X_{t-1} = Y_t - Z_t \qquad (2)$$

where $Y_t \geq 0$ and $Z_t \geq 0$ represent an increase and a decrease in production, respectively.
In summary, the basic (**constraint**) equations of the model are, for month **t = 1 to n**,

$$X_t + S_{t-1} - S_t = R_t$$
$$X_t - X_{t-1} - Y_t + Z_t = 0$$

with all non-negative variables.

Note: For zero surplus at the end of the year (or whatever time horizon) $S_n = 0$. Also, usually $X_0 = 0$ but this could be varied to suit the problem.

The objective is to minimise the cost. Let
a = cost of increasing production by 1 unit from month t-1 to month t

b = cost of storing 1 unit for 1 month
Then, the objective is to minimise

$$\mathbf{b\Sigma S_t + a\Sigma Y_t}$$

Let $\lambda = \mathbf{a/b}$ measure the cost of a unit increase in output relative to that of storing a unit for a month.
Then, we can re-state the objective as

$$\mathbf{Minimise\ \Sigma S_t + \lambda \Sigma Y_t}$$

Note: A minimum solution cannot have both $Y_t > 0$ & $Z_t > 0$ for any month t.

Given the value of λ the problem may be solved using the Simplex method.

- In many instances, however, a value of λ cannot be determined or various schedules for a range of λ values may be required.
- For situations such as these the Simplex method can be altered so that λ need not be given explicitly and production schedules for λ in the range $[0, \infty)$ can be computed.
- Essentially, this is based on the same kind of considerations as used before in sensitivity analysis of linear programming problems.

Summary of a parametric linear programming example (not related to inventory control):

Min $Z = \lambda x - y$

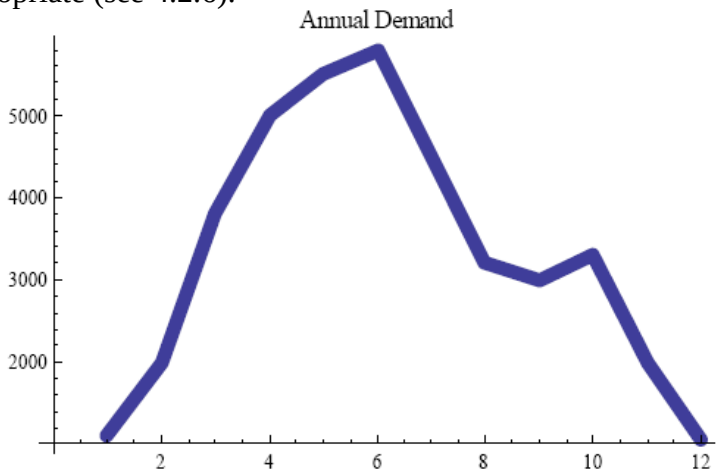
subject to

$3x - y \geq 5$

$2x + y \leq 3$

where $-\infty < \lambda < \infty$ and x and y are unrestricted in sign.
For this problem, the altered (*we did not detail this previously*) Simplex method yields
a) For $-2 \leq \lambda \leq 3$, the solution is $x = 8/5$, $y = -1/5$
& $z = 1/5 + \lambda(8/5)$
b) For $\lambda = 3$, the solution is $x = 0$, $y = -5$ & $z = 5$
In fact, there is a multiple solution at $\lambda = 3$.
Also, it can be shown that no finite solutions exist outside of $-2 \leq \lambda \leq 3$.

Specific example (1): Below, we have an example of fluctuating demand (represented by R) for which VC works out to be 0.48 which means that EOQ method is not appropriate (see 4.2.6).



Instead of EOQ model, we use linear programming instead (via AMPL).

Model:

```
param T > 0; # number of months
param R {1..T} >= 0; # Monthly demand
param lambda default 1;
param S0 default 0;
param X0 default 0;
var S {1..T} >=0;
var X {0..T};
var Y {1..T} >=0;
var Z {1..T} >= 0;
minimize ProdStor: sum {t in 1..T} S[t]+lambda*sum {t in 1..T}Y[t];
subject to Xinit: X[0]=X0;
subject to Req1: X[1]+S0-S[1]=R[1];
subject to Req {t in 2..T}: X[t]+S[t-1]-S[t]=R[t];
subject to Fluc {t in 1..T}: X[t]-X[t-1]=Y[t]-Z[t];
```

Case 1:

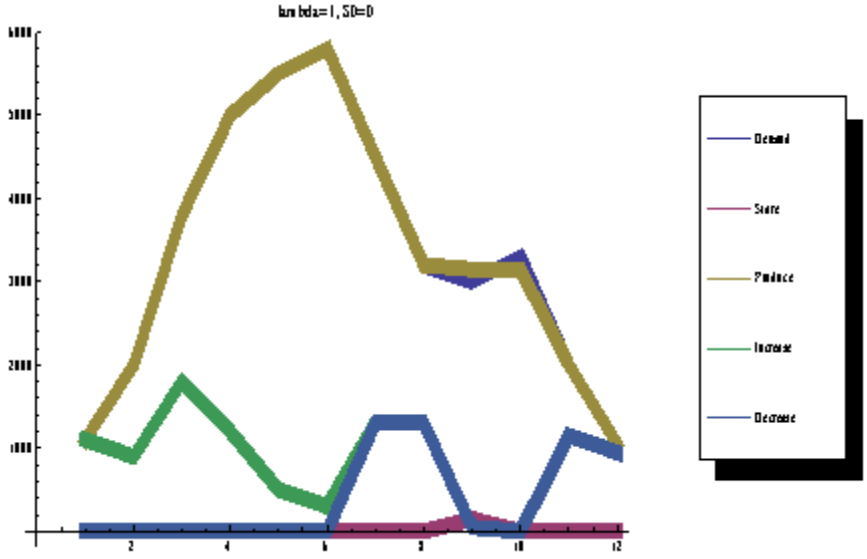
```
param T:=12;
param R :=1 1100 2 2000 3 3800 4 5000 5 5500 6 5800 7 4500 8 3200 9 3000 10
3300 11 2000 12 1050;
```

param lambda:=1;

```
param S0:= 0;
param X0:= 0;
```

```
ampl: model g:\lpinv01.txt;
ampl: data g:\lpinv01d.txt;
ampl: solve;
MINOS 5.51: optimal solution found.
8 iterations, objective 5950
ampl: display S,X,Y,Z;
:     S     X      Y      Z      :=
0     .     0      .      .
1     0   1100   1100     0
2     0   2000    900     0
3     0   3800   1800     0
4     0   5000   1200     0
5     0   5500    500     0
6     0   5800    300     0
7     0   4500      0   1300
8     0   3200      0   1300
9   150   3150      0     50
10    0   3150      0      0
11    0   2000      0   1150
12    0   1050      0    950
;
```

Graphic depiction:



Case 2:

param T:=12;

param R :=1 1100 2 2000 3 3800 4 5000 5 5500 6 5800 7 4500 8 3200 9 3000 10 3300 11 2000 12 1050;

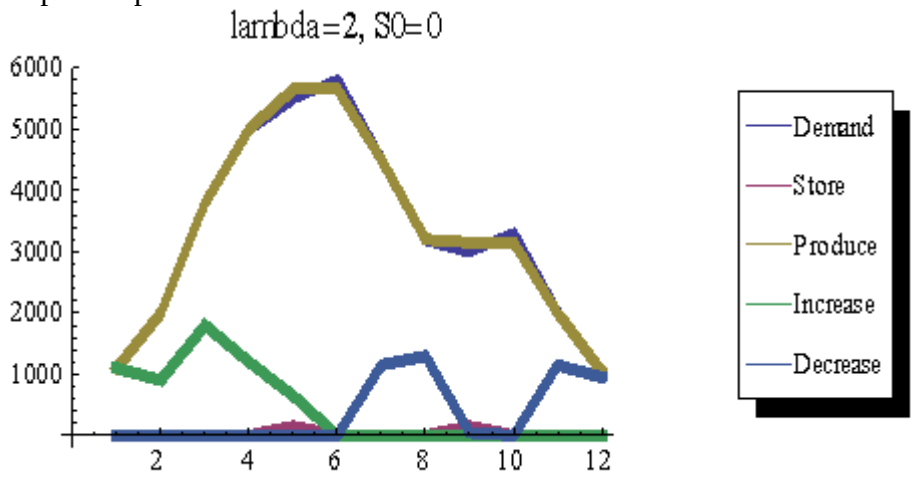
param lambda:=2;

param S0:= 0;

param X0:= 0;

```
ampl: solve;
MINOS 5.51: optimal solution found
9 iterations, objective 11600
ampl: display S,X,Y,Z;
:     S     X     Y     Z     :
0     .     0     .     .
1     0   1100  1100     0
2     0   2000   900     0
3     0   3800  1800     0
4     0   5000  1200     0
5   150   5650   650     0
6     0   5650     0     0
7     0   4500     0  1150
8     0   3200     0  1300
9   150   3150     0    50
10    0   3150     0     0
11    0   2000     0  1150
12    0   1050     0   950
;
```

Graphic depiction:



Case 3:

param T:=12;

param R :=1 1100 2 2000 3 3800 4 5000 5 5500 6 5800 7 4500 8 3200 9 3000 10 3300 11 2000 12 1050;

param lambda:=10;

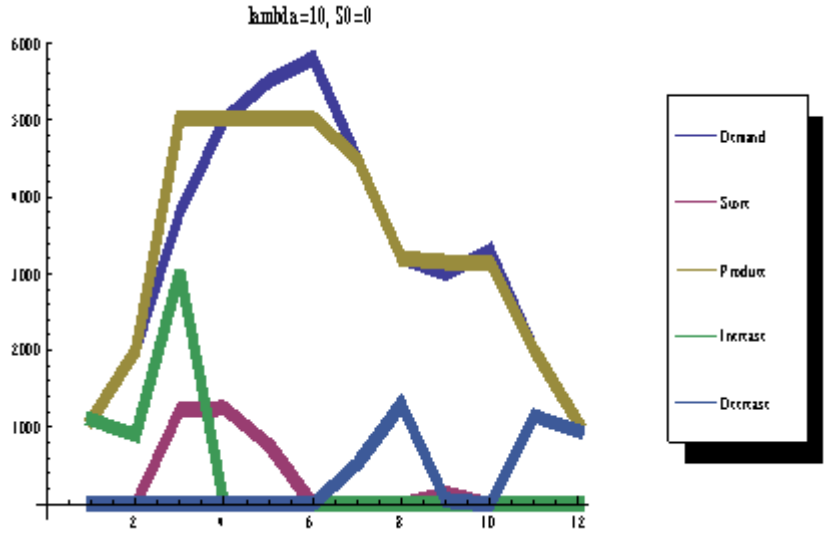
param S0:= 0;

param X0:= 0;

Solution:

```
MINOS 5.51: optimal solution found.
11 iterations, objective 53650
ampl: display S,X,Y,Z;
:     S      X      Y      Z      :=
0     .      0      .      .
1     0   1100   1100      0
2     0   2000    900      0
3  1225   5025   3025      0
4  1250   5025      0      0
5   775   5025      0      0
6     0   5025      0      0
7     0   4500      0    525
8     0   3200      0   1300
9   150   3150      0     50
10    0   3150      0      0
11    0   2000      0   1150
12    0   1050      0    950
```

Graphic depiction:



Case 4:

param T:=12;

param R :=1 1100 2 2000 3 3800 4 5000 5 5500 6 5800 7 4500 8 3200 9 3000 10 3300 11 2000 12 1050;

param lambda:=10;

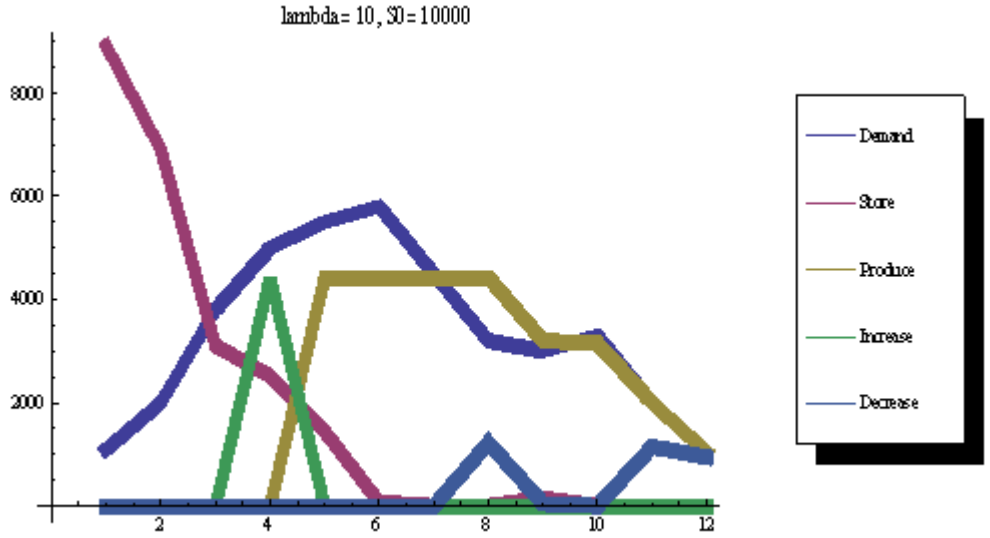
param S0:= 10000;

param X0:= 0;

Solution:

```
ampl: solve;
MINOS 5.51: optimal solution found.
15 iterations, objective 67350
ampl: display S,X,Y,Z;
:     S          X          Y     Z     :=
0       .        0            .     .
1    8900        0            0     0
2    6900     -8.80025e-13    0     0
3    3100     -1.86812e-12    0     0
4    2525     4425           4425   0
5    1450     4425            0     0
6      75     4425            0     0
7       0     4425            0     0
8       0     3200            0  1225
9     150     3150            0    50
10      0     3150            0     0
11      0     2000            0  1150
12      0     1050            0   950
;
```

Graphic depiction:



Summary of cases:

Case	S0	Lambda (λ)	Minimum Cost
1	0	1	5950
2	0	2	11600
3	0	10	53650
4	10000	10	67350

Specific example (2): In this follow-on an inventory problem, where demand varies only slightly from constant over 12 months, is solved using linear programming (***AMPL*** tool).
For the demand data considered below, the coefficient of variability is 0.15 so that the EOQ model could be used.

The **model** is essentially as for **Specific Example (1)** but we have added a non-negativity constraint on X and some comments:

```
param T > 0; # number of months
param R {1..T} >= 0; # Required units per month
param lambda default 1; #(cost of increasing production by 1 unit between months )/(Cost of storing 1 unit for 1 month)
param S0 default 0;
param X0 default 0;
var S {1..T} >=0; #No. of finished units not required in month t
var X {0..T}>=0; # No. of units produced in month t
var Y {1..T} >=0; #Increase in production in month t
var Z {1..T} >= 0; #Decrease in production in month t
minimize ProdStor: sum {t in 1..T} S[t]+lambda*sum {t in 1..T}Y[t];
subject to Xinit: X[0]=X0;
subject to Req1: X[1]+S0-S[1]=R[1];
subject to Req {t in 2..T}: X[t]+S[t-1]-S[t]=R[t];
subject to Fluc {t in 1..T}: X[t]-X[t-1]=Y[t]-Z[t];
```

Case 1:

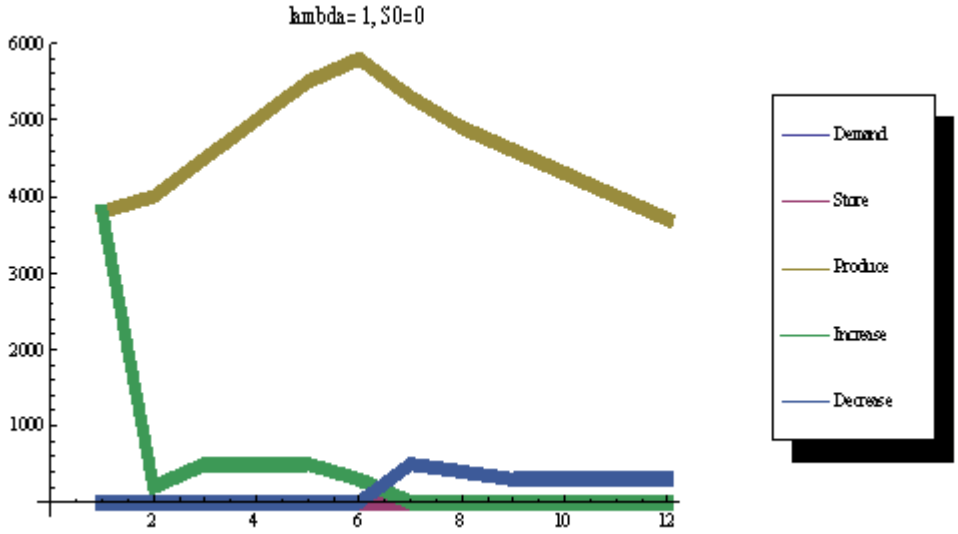
```
param T:=12;
```

param R :=1 3800 2 4000 3 4500 4 5000 5 5500 6 5800 7 5300 8 4900 9 4600 10 4300 11 4000 12 3700;
param **lambda:=1**;
param S0:= 0;
param X0:= 0;
Solution:

Shortcut to ampl

```
ampl: model G:\lpinv01.txt;
ampl: data G:\lpinv01d.txt;
ampl: solve;
MINOS 5.51: optimal solution found.
22 iterations, objective 5800
ampl: display S,X,Y,Z;
:    S     X       Y       Z     :=
0    .       0       .       .
1    0    3800    3800       0
2    0    4000     200       0
3    0    4500     500       0
4    0    5000     500       0
5    0    5500     500       0
6    0    5800     300       0
7    0    5300       0     500
8    0    4900       0     400
9    0    4600       0     300
10   0    4300       0     300
11   0    4000       0     300
12   0    3700       0     300
;
```

Graphic depiction:

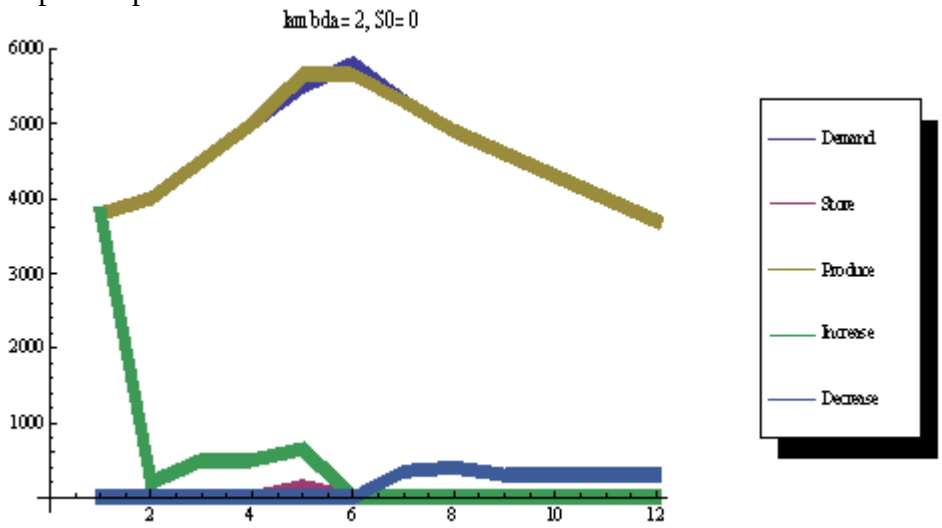


Case 2:
param T:=12;
param R :=1 3800 2 4000 3 4500 4 5000 5 5500 6 5800 7 5300 8 4900 9 4600 10 4300 11 4000 12 3700;
param lambda:=2;
param S0:= 0;
param X0:= 0;

Solution:

```
ampl: model G:\lpinv01.txt;
ampl: data G:\lpinv01d.txt;
ampl: solve;
MINOS 5.51: optimal solution found.
30 iterations, objective 11450
ampl: display S,X,Y,Z;
:     S     X      Y      Z     :=
0     .     0      .      .
1     0   3800   3800     0
2     0   4000    200     0
3     0   4500    500     0
4     0   5000    500     0
5   150   5650    650     0
6     0   5650      0     0
7     0   5300      0   350
8     0   4900      0   400
9     0   4600      0   300
10    0   4300      0   300
11    0   4000      0   300
12    0   3700      0   300
;
```

Graphic depiction:



Case 3:

param T:=12;

param R :=1 3800 2 4000 3 4500 4 5000 5 5500 6 5800 7 5300 8 4900 9 4600 10 4300 11 4000 12 3700;

param lambda:=10;

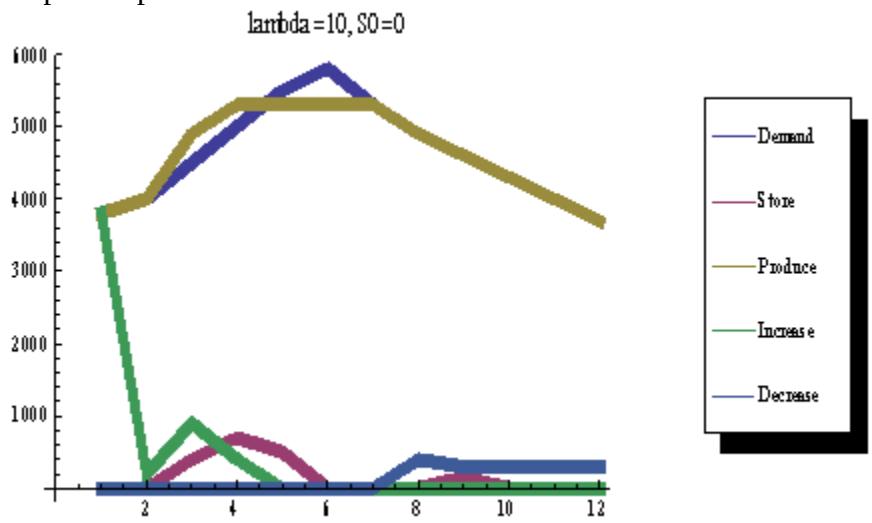
param S0:= 0;

param X0:= 0;

Solution:

```
ampl: model G:\lpinv01.txt;
ampl: data G:\lpinv01d.txt;
ampl: solve;
MINOS 5.51: optimal solution found.
36 iterations, objective 54600
ampl: display S,X,Y,Z;
:     S     X     Y     Z     :=
0     .     0     .     .
1     0   3800  3800     0
2     0   4000   200     0
3   400   4900   900     0
4   700   5300   400     0
5   500   5300     0     0
6     0   5300     0     0
7     0   5300     0     0
8     0   4900     0   400
9     0   4600     0   300
10    0   4300     0   300
11    0   4000     0   300
12    0   3700     0   300
;
```

Graphic depiction:



Summary of cases:

Case	S0	Lambda (λ)	Minimum Cost
1	0	1	5800
2	0	2	11450
3	0	10	54600

4.2.8 Problems

Question 1

(a) A company requires 26,000 cases of diskettes per annum, the demand being essentially constant throughout the year. The cost of placing an order is €130, regardless of the order size and the company pays €4.50 per case. The holding cost is estimated to be €0.10 per unit per month. What size of order should the company place when it is placing an order, and how many orders per year should the company place?

(b) Supposing this company is considering that instead of purchasing the diskettes from a supplier, that it could instead manufacture these diskettes itself. It estimates that its production capacity would be 60,000 cases per annum, the cost of production would be €4.40 per case and the cost of setting up a production run would be €150. Would you recommend that the company should produce the diskettes itself rather than purchase them?

Question 2
A company manufacturing PC's has to purchase components regularly from a supplier who offers the following pricing regime:

Quantity Purchased	Price per Component
< 500	€5.00
500 – 999	€4.90
1000 – 1999	€4.80
≥ 2000	€4.75

The company has an annual demand for 5000 of these components. Furthermore, it costs the company €50 to place up an order (regardless of order size), and there is an inventory holding charge of 20% of the value of the stock per annum. What order quantity will minimise the total annual cost (i.e. purchasing cost + ordering cost + holding cost)?

4.3 Project planning (PERT/CPM)

4.3.1 Introduction

We present two methods that have to do with planning, scheduling, and control of projects.

First method: Critical Path Method (CPM)
Assumptions:
- Activity times are known
- Activity times = *f(resources allocated)*

where an activity is a job or task within a project.
This method evaluates trade-offs between project costs and project completion time.

Second method: Programme Evaluation and Review Technique (PERT)
- Incorporates uncertainty in job times
- Gives probability of completion by specified time
- Identifies bottlenecks

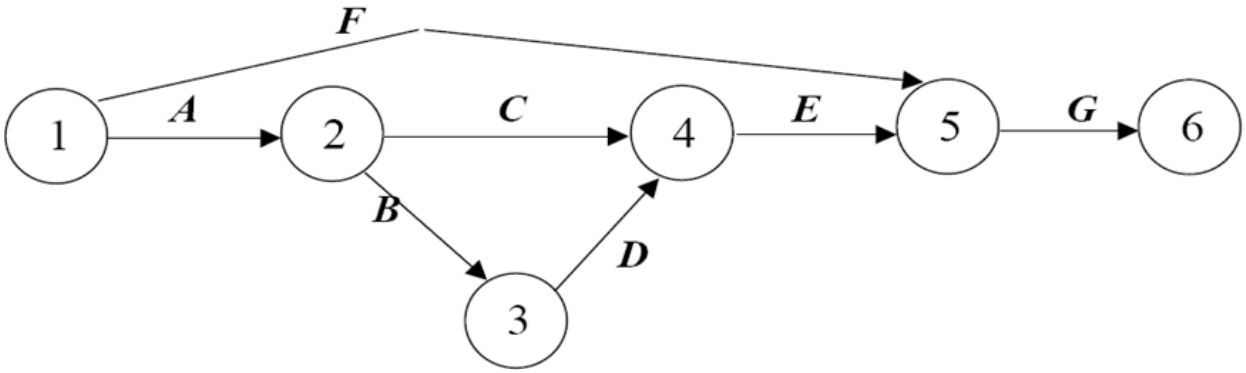
Both methods have application wherever project management is used, including in software project management.

Network Representation: This provides a visualization of the tasks making up a project & how they are sequenced.
NB: In CA4011 we use "**activity on arrow**" networks and this is what you are expected to use for this module.
Nodes represent points in time (or events) – when jobs are started/completed

Arcs represent ***individual*** jobs – one job per arc, one arc per job. So, two nodes can be joined directly by ***no more than one arc***.
Direction of arcs defines the sequence of jobs

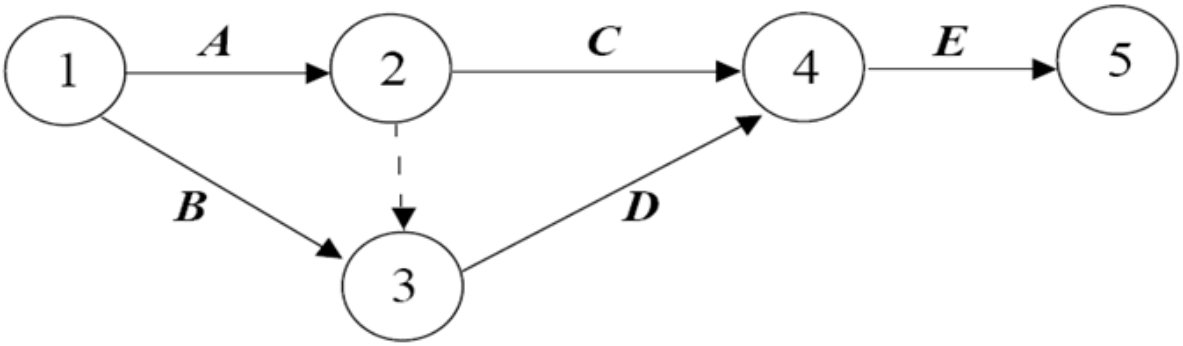
Example: Project with 7 jobs - *A, B, C, ... , G*
Given:
A precedes *B* and *C*
C and *D* precede *E*
B precedes *D*
E and *F* precede *G*

Network representation:



Dummy jobs: These are sometimes needed to show job sequencing. Appear as ***dashed*** arcs.
Example: 5 jobs - *A, B, C, D, E*
A precedes *C* and *D*
B precedes *D*
C and *D* precede *E*

Network representation:



Node 3 marks completion of both jobs *A* and *B.*

Notation:
t_{ij} = time required to complete job represented by arc from node i to node j. It is the **duration** of the job.
In the previous example, we might have (say),
A: t_{12} = 3 days
B: t_{13} = 1 day

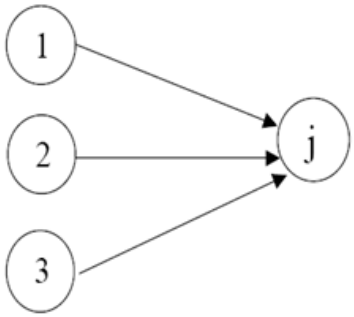
C: t_{24} = 4 days
D: t_{34} = 2 days
E: t_{45} = 5 days
Dummy: t_{23} = 0 days
Note: A dummy activity is always of zero duration.

4.3.2 Finding the Critical Path

Definition of earliest time of a node (= event):

- Earliest time of node $j = U_j$ = earliest time event *j* can occur. It is the time when all jobs (arcs) leading to *j* are completed.

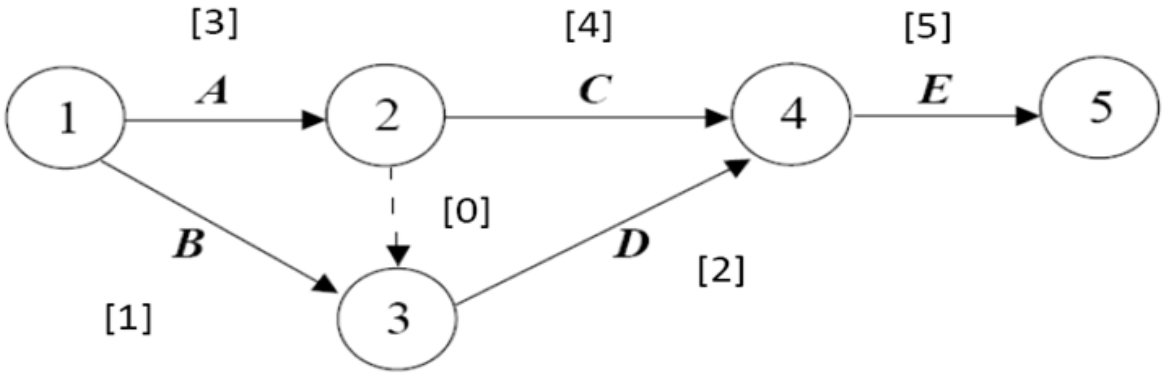
Example:



$U_j = \text{Max}\,(U_1 + t_{1j}\,,\, U_2 + t_{2j}\,,\, U_3 + t_{3j}\,)$

In general: $U_j = \text{Max}_i\,(U_i + t_{ij}\,)$

Example *: We had the following figure – durations in [..]



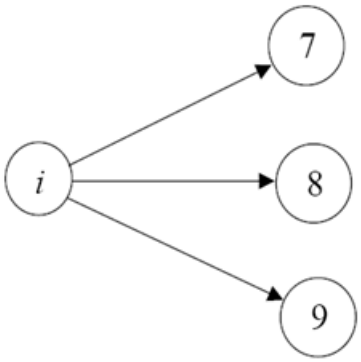
$U_1 = 0$
$U_2 = U_1 + t_{12} = 3$
$U_3 = \text{Max}\,(U_2 + t_{23}\,,\, U_1 + t_{13}\,) = \text{Max}\,(3, 1) = 3$
$U_4 = \text{Max}\,(U_2 + t_{24}\,,\, U_3 + t_{34}\,) = \text{Max}\,(7, 5) = 7$
$U_5 = U_4 + t_{45} = 12$

Note: Earliest time of last node = earliest **project completion time**

Definition of latest time of a node (= event):

- Latest time of node $i = V_i$ = latest time when event *i* can occur without delaying project beyond earliest completion time.

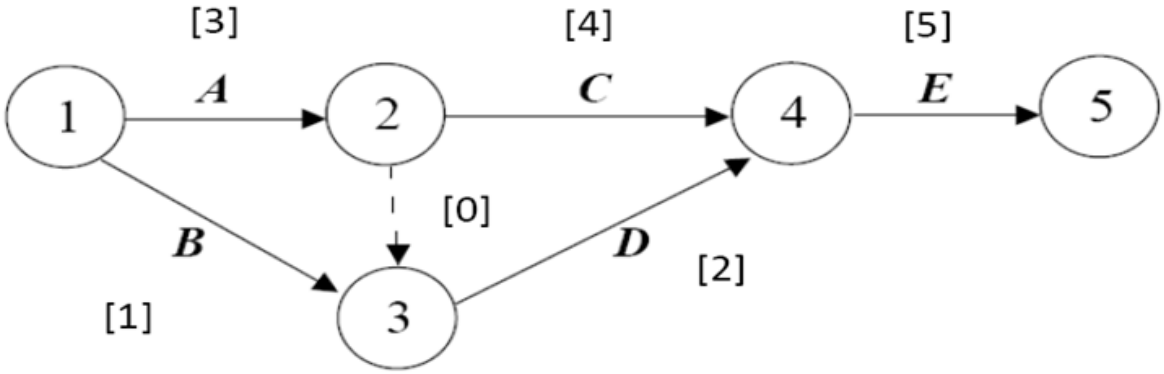
Example:



$V_i = \text{Min}\,(V_7 - t_{i7}\,,\, V_8 - t_{i8}\,,\, V_9 - t_{i9}\,)$

In general: $V_i = \text{Min}_j\,(V_j - t_{ij}\,)$

Example * (continued): We use again the following figure – durations in [..]



$V_5 = U_5 = 12$
$V_4 = V_5 - t_{45} = 7$
$V_3 = V_4 - t_{34} = 5$
$V_2 = \text{Min}\,(V_4 - t_{34}\,,\, V_3 - t_{23}\,) = \text{Min}\,(3, 5) = 3$
$V_1 = \text{Min}\,(V_2 - t_{12}\,,\, V_3 - t_{13}\,) = \text{Min}\,(0, 4) = 0$

Definition: Latest time - Earliest time = Slack time for **event**.
- **Example**: For events 1, 2, 3, 4, 5, slack times are 0, 0, 2, 0, 0.

Definition: Critical Events are events with zero slack.
Definition: Critical Path is a path between critical events:
- **Example**: $1 \rightarrow 2 \rightarrow 4 \rightarrow 5$ (jobs A, C, E)

Definition: Project schedule = start and finish times of all **jobs**.
For any job, from node i to node j, with time requirement of t_{ij}
- earliest start time = U_i
- latest start time = $(V_j - t_{ij}\,)$
- earliest finish time = $(U_i + t_{ij}\,)$
- latest finish time = V_j
- maximum time available = $(V_j - U_i\,)$
- slack time = $(V_j - U_i - t_{ij}\,)$

Schedule for example *:

Job	Expected Duration	Earliest start	Latest start	Earliest finish	Latest finish	Slack	Critical
A	3	0	0	3	3	0	Y
B	1	0	4	1	5	4	N
C	4	3	3	7	7	0	Y
D	2	3	5	5	7	2	N
E	5	7	7	12	12	0	Y

Remember Critical jobs – jobs with zero slack time.

4.3.3 Critical Path Method (enumerative & math. prog. methods)

Crashing: If we assign additional resources to a job, we may reduce the duration – called crashing.
Additional cost = Crashing cost
Each job has specified:-

- Normal completion time (max)
- Crash completion time (min)

There is a cost vs. time relationship (linear) for jobs.
Problem is to minimise total cost of project.
Solution methods:

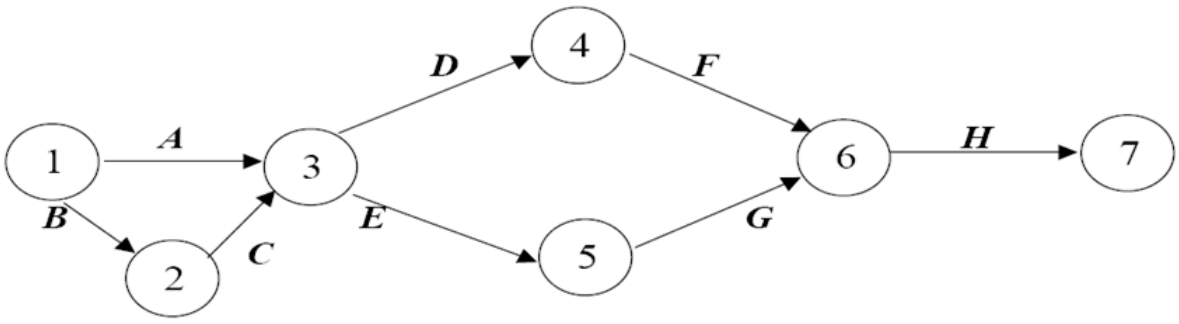
- Enumerative methods (small projects)
- Mathematical programming methods (large projects).

Example: Suppose the following data are specified:

Job	Immediate prededessors	Normal time (days)	Crash time (days)	Crash cost (€ per day)
A	-	10	7	4
B	-	5	4	2
C	B	3	2	2
D	A,C	4	3	3
E	A,C	5	3	3
F	D	6	3	5
G	E	5	2	1
H	F,G	5	4	4

and that there is a **Fixed cost** = €5 per day

Solution: The first (non-trivial) task is to draw the network ***based on the normal job times*** (durations):



Critical Paths: 1→ 3 → 4 → 6 → 7
1→ 3 → 5 → 6 → 7
Between node 3 and node 6 there are parallel critical paths.

Critical activities/jobs:- *A, D, E, F, G, H*
Total cost (normal) = 25 x 5 = €125.

To reduce cost, time (duration) of critical jobs needs to be reduced.
Consider H: Crash by 1 day. Cost = €4. Saving = €5.
- Hence, Total cost = €124 and Project duration = 24 days.

Consider *A*: Can crash by 3 days, but note that *B* and *C* are also being done at this time (taking 8 days in total).
- First crash *A* by 2 days (to 8 days). Cost = €4 x 2 = €8. Saving = €10. So, Total cost = €122 and Project duration = 22 days.
- *B* and *C* now critical. Therefore if crash *A* by one more day, must also crash *B* or *C*. Cost = €4 + 2 = €6. Saving = €5. Therefore not worth it.

Consider D, E, F, G: There are parallel paths & so must be crashed in pairs (see next slide).
Consider D, E, F, G: There are parallel paths & so must be crashed in pairs.

Jobs	Cost	
D + E	3 + 3 = 6	
D + G	3 + 1 = 4*	Economical to crash
F + E	5 + 3 = 8	*D + G* by 1 day (4 < 5)
F + G	5 + 1 = 6	

Total cost = €121. Project duration = 21 days.

Mathematical Programming Methods: Let
- k_{ij} = normal completion time for job *(i , j)*
- l_{ij} = crash time for job *(i , j)*
- c_{ij} = unit cost of shortening job *(i , j)*

Therefore, $l_{ij} \le t_{ij} \le k_{ij}$ and Crashing cost = $c_{ij}(k_{ij} - t_{ij})$

Let t_i = unknown event times $i = 1 \ldots n$ (1 = start, n = finish) and let F = overhead cost per time unit.

Formulation:
Minimise $Z = F(t_n - t_1) + \sum c_{ij}(k_{ij} - t_{ij})$
Subject to
$t_j - t_i \ge t_{ij}$ for all jobs *(i , j)*
$l_{ij} \le t_{ij} \le k_{ij}$ for all jobs *(i , j)*
$t_i \ge 0$ for all events *i*

Example (as before for "enumerative" approach):

Project duration = $(t_7 - t_1)$, so overhead cost = $5(t_7 - t_1)$

Crashing costs: Job A $4(10 - t_{13})$
Job B $2(5 - t_{12})$
…. Etc.

Minimise $5(t_7 - t_1) + 4(10 - t_{13}) + 2(5 - t_{12}) + 2(3 - t_{23}) + 3(4 - t_{34}) + 3(5 - t_{35}) + 5(6 - t_{46}) + 1(5 - t_{56}) + 4(5 - t_{67})$

Subject to

$t_3 - t_1 \ge t_{13}$	$7 \le t_{13} \le 10$
$t_2 - t_1 \ge t_{12}$	$4 \le t_{12} \le 5$
$t_3 - t_2 \ge t_{23}$	$2 \le t_{23} \le 3$
$t_4 - t_3 \ge t_{34}$	$3 \le t_{34} \le 4$
$t_5 - t_3 \ge t_{35}$	$3 \le t_{13} \le 5$
$t_6 - t_4 \ge t_{46}$	$3 \le t_{46} \le 6$
$t_6 - t_5 \ge t_{56}$	$2 \le t_{56} \le 5$
$t_7 - t_6 \ge t_{67}$	$4 \le t_{67} \le 5$

$t_i \geq 0$ for all $i = 1 \ldots 7$

It turns out that solution is:
$t_1 = 0$, $t_2 = 5$, $t_3 = 8$, $t_4 = 11$, $t_5 = 13$, $t_6 = 17$, $t_7 = 21$.
$t_{13} = 8$, $t_{12} = 5$, $t_{23} = 3$, $t_{34} = 3$, $t_{35} = 5$, $t_{46} = 6$, $t_{56} = 4$, $t_{67} = 4$.

Following is a basic **ampl** formulation and solution of the first part of this problem, that is the solution based on normal durations, without crashing.

Note: There are special facilities in **ampl** for network linear programs but these have not been used here:

Model file (cpm01.txt):
```
set I;
var t {i in I};
minimize Path:5*(t[7]-t[1]);
subject to A: t[3]  -  t[1]  >=  10;
subject to B: t[2]  -  t[1]  >=  5;
subject to C: t[3]  -  t[2]  >=  3;
subject to D: t[4]  -  t[3]  >=  4;
subject to E: t[5]  -  t[3]  >=  5;
subject to F: t[6]  -  t[4]  >=  6;
subject to G: t[6]  -  t[5]  >=  5;
subject to H: t[7]  -  t[6]  >=  5;
subject to Limit {i in I}:0<=t[i];
```

Data file (cpm01d.txt):
```
set I:=1 2 3 4 5 6 7;
```

ampl session:
```
ampl: model g:\cpm01.txt;
ampl: data g:\cpm01d.txt;
ampl: solve
ampl? ;
MINOS 5.51: optimal solution found.
2 iterations, objective 125
ampl: display limit,t;
```

```
ampl: display Limit,t;
: Limit    t      :=
1      0     0
2      0     7
3      0    10
4      0    14
5      0    15
6      0    20
7      0    25
;
```

Note: Next step would be to extend the model to include crashing.

4.3.4 Programme Evaluation and Review Technique (PERT)

In the Critical Path Method (CPM), job times (durations) are known.
But in PERT we have:

(1) Most probable job time = m
(2) Optimistic job time = a
(3) Pessimistic job time = b

A Beta distribution of job time is assumed. From this it follows that, μ = mean (i.e. expected) job completion time = ***(a + 4m + b)/6***

We assume that **end values** for single peaked distributions are within 3 standard deviations of the mean (+/-3σ). Therefore the spread = 6σ
Hence, $6\sigma = b - a \Rightarrow$ **$\sigma = (b - a)/6$** or $\sigma^2 = ((b - a)/6)^2$

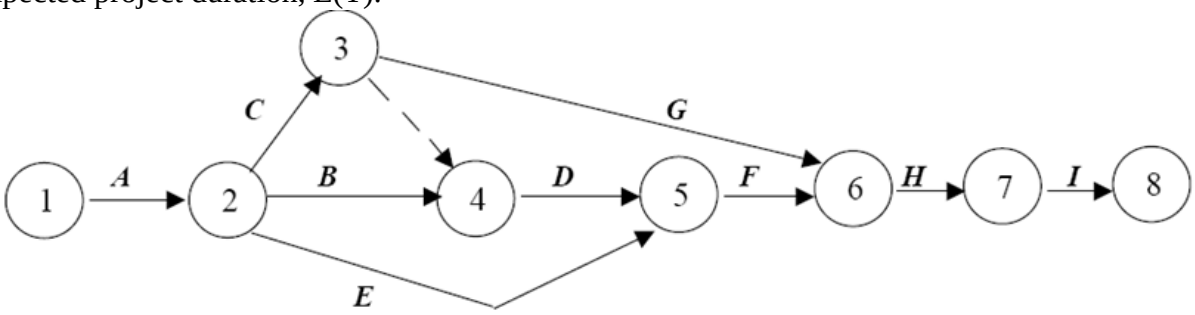
With m, a, and b given for each job, we can calculate the mean (expected) job time (μ) and variance (σ^2)

PERT method:
(1) Calculate the expected job times using (***a+4m+b)/6***).
(2) Find the critical path using ***expected*** job times.
(3) Project duration (T) = $\sum$ job times on critical path.
Job times are random variables, and therefore so is T.
$E(T)$ = $\sum$ expected job times on critical path
$Var(T)$ = $\sum$ variances of expected job times on critical path

Example: ***First*** step is to calculate expected times (3rd col. From right)

Job	Pred-ecessors	Opt. Times(a)	Most prob. Times (m)	Pess. Times (b)	Expect. times (μ)	σ	σ^2
A	-	2	5	8	5	1	1
B	A	6	9	12	9	1	1
C	A	6	7	8	7		
D	B,C	1	4	7	4	1	1
E	A	8	8	8	8		
F	D,E	5	14	17	13	2	4
G	C	3	12	21	12		
H	F,G	3	6	9	6	1	1
I	H	5	8	11		1	8
					-------Calculated------		

Second step is construct network diagram (using the given predecessor information).
Third step is find the critical path based on the ***expected*** durations. This yields the expected project duration, E(T).



Critical jobs:- A, B, D, F, H, I

E (T) = μ_T = 5 + 9 + 4 + 13 + 6 + 8 = 45 days

Fourth step is to calculate **σ = (b – a)/6** for each job on critical path (see table). Hence, Variance (T) = V(T) = 1 + 1 + 1 + 4 + 1 + 1 = 9 => Standard deviation σ(T) = σ_T = √9 =3

Probabilities of Completing Project

Having found μ_T = *E (T) and* σ_T^2 = *V (T)* (for project duration T) one can calculate related probabilities under the following assumption:

Assumption:- job times are independently and identically distributed.

Therefore, using the Central Limit Theorem of Statistics, it follows that T is (approximately) normally distributed, that is

$T \sim N(\mu_T, \sigma_T^2)$

In example $T \sim N(45, 9)$

Therefore: 68% chance that $42 \leq T \leq 48$

99.7% chance that $36 \leq T \leq 54$ and so on.

Probabilities of meeting deadlines:-

P (T ≤ T*) = P (Z ≤ (T* - E(T)) / σ (T))

e.g. T* = 50 days

P (T ≤ 50) = P (Z ≤ (50 - 45) / 3) = P (Z ≤ 1.67) = 0.95

For T* = 41 days

$$P(T \leq 41) = P(Z \leq (41 - 45)/3) = P(Z \leq -1.33) = 0.09$$

Note: If there is more than one critical path, with different variances, use the largest total variance in probability calculations.

4.3.5 Problems

Problem 1: A bank wishes to plan and schedule the development and installation of a new computerised cheque processing system. The changeover in cheque processing procedures requires employment of additional personnel to operate the new system, development of new systems (software), and modification of existing cheque sorting equipment. The activities to complete the project and the precedence relationships among the activities have been determined and are given in the following table:

Activity	Description	Immediate Predecessor	Duration (days)
a	Position recruiting	-	9
b	System development	-	11
c	System training	a	7
d	Equipment training	a	10
e	Manual system test	b,c	1
f	Preliminary system changeover	b,c	5
g	Computer-personnel interface	d,e	6
h	Equipment modification	d,e	3
i	Equipment testing	h	1
j	System debugging & installation	f,g	2
k	Equipment changeover	g,i	8

Construct the network diagram for the project, and find the critical path.

Problem 2: The project described below gives normal and crash times and crashing costs for each activity.

Activity	Immediate Predecessor	Duration (days) Normal	Duration (days) Crash	Crash cost (€/day)
A. Train workers	-	6	1	10
B. Purchase raw materials	-	9	4	20
C. Produce product 1	A,B	14	9	30
D. Produce product 2	A,B	7	2	30
E. Test product 2	D	10	5	40
F. Assemble products 1 and 2	C,E	12	7	50

There is a fixed cost of €40 per day.

Find the activity times which will minimise the total cost of the project.

Problem 3: Given the following time estimates for each of the activities in **Problem 1**, find the probability that the project will be completed within 40 days.

Activity	Time estimates Optimistic	Most likely	Pessimistic
a	5	8	17
b	3	12	15
c	4	7	10
d	5	8	23
e	1	1	1
f	1	4	13
g	3	6	9
h	1	2.5	7
i	1	1	1
j	2	2	2
k	5	8	11

4.4 Decision Theory & Game Theory (TBD)

TBD

Works Cited

Ahumada, O. & Villalobos, J. R., 2009. Application of planning models in the agri-food supply chain: A review. *European Journal of Operational Research,* July, 196(1), pp. 1-20.
Al-Sharrah, G. K., Hankinson, G. & Elkamel, A., 2006. Decision-making for petrochemical planning. *Trans IChemE, Part A, Chemical Engineering Research and Design,* Volume 84(A11), pp. 1019-1030.
Applegate, D. L., Bixby, R. E., Chvatal, V. & Cook, W. J., 2007. *The Traveling Salesman Problem: A Computational Study.* s.l.:Princeton university Press.
Bazaraa, M., Jarvis, J. & Sherali, H., 2010. *Linear Programming and Network Flows.* 4 ed. s.l.:Wiley.
Boyd, S., Kin, S.-J., Vandenberghe, L. & Hassibi, A., 2007. A tutorial on geometric programming. *Optim Eng,* April, Issue 8, pp. 67-127.
Bradley, S. P., Hax, A. C. & Magnanti, T. L., 1977. *Applied Mathematical Programming.* s.l.:Addison-Wesley.
Bueno de Mesquita, B., 2009. *The predictioneer's game: using the logic of brazen self-interest to see and shape the future.* s.l.:Random House.
Chvatal, V., 1983. *Linear programming.* s.l.:W.H Freeman.
Fourer, R., Gay, D. M. & Kernighan, B. W., 2003. *AMPL A Modeling Language for Mathematical Programming.* s.l.:Brooks/Cole CENGAGE Learning.
Galligani, E., Ruggiero, V. & Zanni, L., 2003. *Variable projection methods for large-scale quadratic optimization in data analysis applications.* Dordrecht, Kluwer Academic Publishers, pp. 185-211.
Hastie, T., Tibshirani, R. & Friedman, J., 2008. *The Elements of Statistical Learning.* Second ed. s.l.:Springer.
Hillier, F. S. & Lieberman, G., 1974. *Operations research.* s.l.:Holden-Day.
Hoburg, W., Kirschen, P. & Abbeel, P., 2016. Data fitting with geometric-programming-compatible. *Optim Eng,* August.
Jacovkis, P., Gradowczyk, H., Freisztav, A. & Tabak, E., 1989. A Linear Programming Approach to Water-Resources Optimization. *Methods and Models of Operations Research,* Volume 33, pp. 341-362.
Larson, R. & Odoni, A., 1981. *Urban operations research.* s.l.:Englewood Cliffs, N.J: Prentice-Hall.
Lu, B.-L. & Ito, K., 2003. Converting general nonlinear programming problems into separable programming problems with feedforward neural networks. *Neural Networks,* September, 16(7), pp. 1059-1074.
Luenberger, D. G., 1984. *Linear and nonlinear programming.* 2 ed. s.l.:Addison-Wesley.
Minieka, E., 1978. *Optimization algorithms for networks and graphs.* s.l.:M. Dekker.
Papadimitriou, C. H. & Steiglitz, K., 1982. *Combinatorial optimization: algorithms and complexity.* s.l.:Englewood Cliffs, N.J: Prentice Hall.
Rader, D. J., 2010. *Deterministic Operations Research. Models and methods in Linear Optimization.* s.l.:Wiley.
Schrage, M., 2000. *Serious play: how the world's best companies simulate to innovate.* s.l.:Harvard Business School Press.
Senge, P., 2006. *The fifth discipline: the art and practice of the learning organization.* s.l.:Doubleday/Currency.
Taha, H. A., 2007. *Operations research: an introduction.* 8 ed. s.l.:Pearson Prentice Hall.
Williams, H., 2013. *Model building in mathematical programming.* 5 ed. s.l.:Hoboken, N.J. : Wiley.
Woolsey, R., 2003. *Real world operations research: the Woolsey papers.* s.l.:Lionheart Pub.