STAT0025: Optimisation Algorithms in Operational Research

University College London

2021/22

Contents

1. Linear programming

1.1. Introduction

What is OR? OR stands for Operational Research or Operations Research, and is sometimes called Management Science. To a mathematician, OR means maximising some function subject to a set of constraints. To a manager, OR means building a model for his/her business which:

- allows the efficient or optimal allocation of limited resources, and
- identifies those resources that are limiting the business' output.

Who uses OR? Usually large companies or organisations e.g. oil companies, car manufacturers, the military.

When did it start? Surprisingly recently; during the Second World War when scarce resources had to be assigned for different military purposes. Scientists were asked to do **research** into military **operations**. The simplex algorithm was formulated in 1947.

To do practical OR one needs at least simple computers/calculators.

Main topics in this course

- Linear programming (Chapters 1–3),
- Game theory (Chapter 4),
- Dynamic programming – deterministic and stochastic (Chapters 5 and 6)

1.2. Mathematical programming

Optimisation is the problem of finding the maximum or minimum of a function $f(x)$ and the value of x at which $f(x)$ attains that maximum/minimum.

If x^* is the value that minimises $f(x)$, then x^* is also the value that maximises $-f(x)$, so a minimisation problem can be turned into a maximisation problem, and vice versa.

Example 1.1 (Unconstrained minimisation)**.** Minimise

$$z = f(x) = x_1^2 + x_2^2.$$

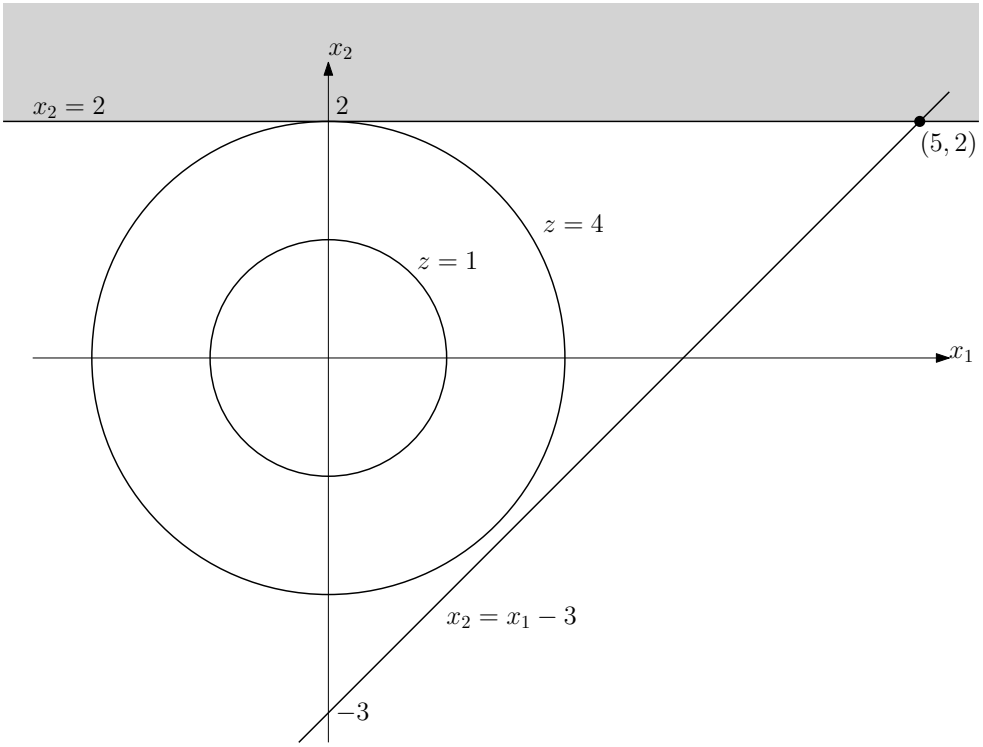
It is easy to see that $x_1^* = 0$ and $x_2^* = 0$ gives the unique minimum, with $z^* = f(x_1^*, x_2^*) = 0$. For a less obvious problem partial derivatives would yield the solution.

Example 1.2 (Constrained minimisation)**.** Minimise

$$z = f(x) = x_1^2 + x_2^2.$$

Subject to the conditions

$$\begin{aligned} g_1(x) = x_1 - x_2 &= 3 \\ g_2(x) = x_2 &\geq 2. \end{aligned}$$



The value of z is the squared distance of the point (x_1, x_2) from the origin (Pythagoras's Theorem). From the figure the point that satisfies the constraints that is also the closest to the origin is $x_1^* = 5, x_2^* = 2$, which is the optimal solution.

A **mathematical program** is an optimisation problem in which the objective and constraints are given as mathematical functions and functional relationships:

$$\begin{array}{ll} \text{Minimise (or maximise) } z = f(x_1, x_2, \ldots, x_n) & \\ \text{subject to:} & \left.\begin{array}{l} g_1(x_1, x_2, \ldots, x_n) \\ g_2(x_1, x_2, \ldots, x_n) \\ \vdots \\ g_m(x_1, x_2, \ldots, x_n) \end{array}\right\} \begin{array}{c} \leq \\ = \\ \geq \end{array} \left\{\begin{array}{l} b_1 \\ b_2 \\ \vdots \\ b_m \end{array}\right. \end{array}$$

The x_j are called the **control variables** (or **decision variables**), f is the **objective function** and b_i $(i = 1, \ldots, m)$ are the **resource values**.

1.3. Linear programming problems (LPPs)

Definition 1.3. A mathematical program is a **linear program** if $f(x_1, x_2, \ldots, x_n)$ and all $g_i(x_1, x_2, \ldots, x_n)$ are linear in the decision variables, that is, if f and all g_i can be written as:

$$f(x_1, x_2, \ldots, x_n) = c_1x_1 + c_2x_2 + \ldots + c_nx_n$$

and

$$g_i(x_1, x_2, \ldots, x_n) = a_{i,1}x_1 + a_{i,2}x_2 + \ldots + a_{i,n}x_n,$$

where c_j and $a_{i,j}$ $(i = 1, 2, \ldots, m$ and $j = 1, \ldots, n)$ are known constants.

In most problems we deal with in this course we also insist that the x_i's are non-negative.

Definitions 1.4. If all x_i's are non-negative variables and **all** the constraints are equalities then the LPP is in **canonical form**.

If all x_i's are non-negative variables and **all** the constraints use '$\leq$' then the LPP is in **standard form**.

The **feasible region** for a LPP is the set of all points $(x_1, \ldots, x_n)$ that satisfy all the constraints (and satisfy $\forall i\ x_i \geq 0$, if x_i's are non-negative variables). Points in the feasible region are called **feasible solutions**.

Example 1.5 (Clock manufacturer)**.** A clock manufacturer produces a number of standard clocks and a number of alarm clocks each day.

Resource constraints:	labour hours per day:		1600
	processing hours per day:		1800
	alarm assemblies available per day:		350
Consumption of resources:	one standard clock uses:	2 hours labour	
		6 hours processing	
	one alarm clock uses:	4 hours labour	
		2 hours processing	
		1 alarm assembly.	
Profit:	one standard clock:	gives £3 profit	
	one alarm clock:	gives £8 profit	

The objective is to maximise profit.

In linear programming notation, let x_1 be the number of standard clocks produced per day and let x_2 be the number of alarm clocks per day.

The objective function is $z = 3x_1 + 8x_2$, which we want to maximise.

The constraints are:	$2x_1 + 4x_2 \leq 1600$	(labour)
	$6x_1 + 2x_2 \leq 1800$	(processing)
	$x_2 \leq 350$	(alarm assemblies).

Also, the number of clocks produced per day cannot be negative, so $x_1 \geq 0$ and $x_2 \geq 0$.

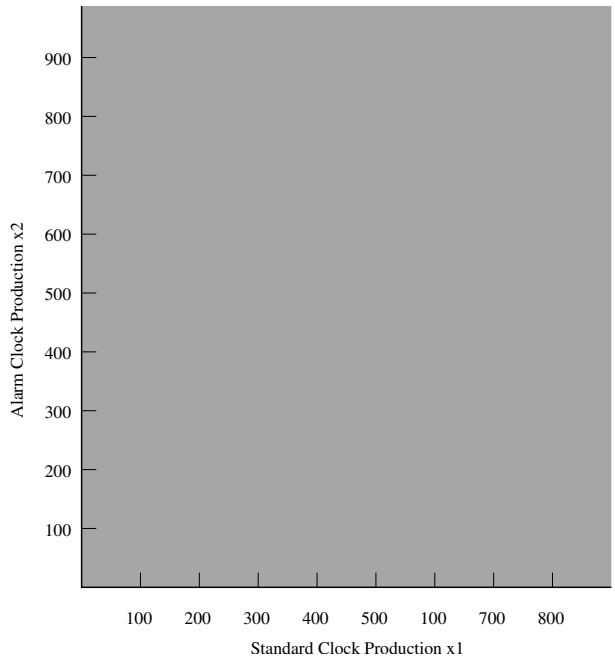
This problem is in standard form.

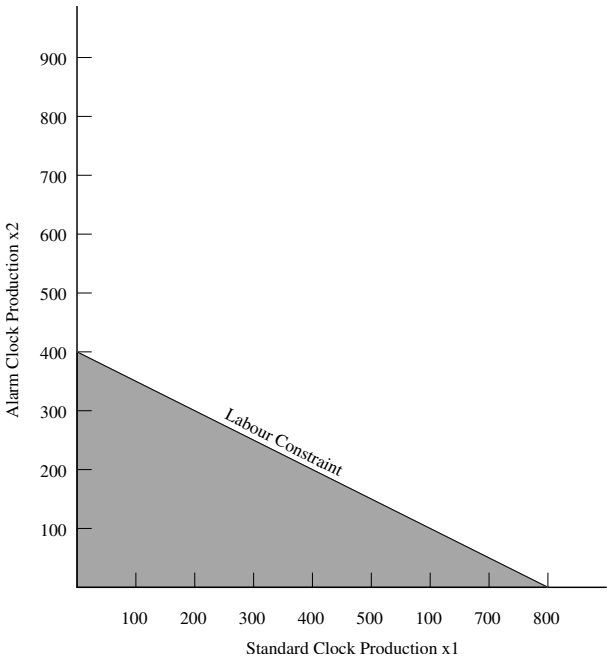
There are several methods of solving this problem: graphical solution methods and numerical methods, including the Simplex algorithm.

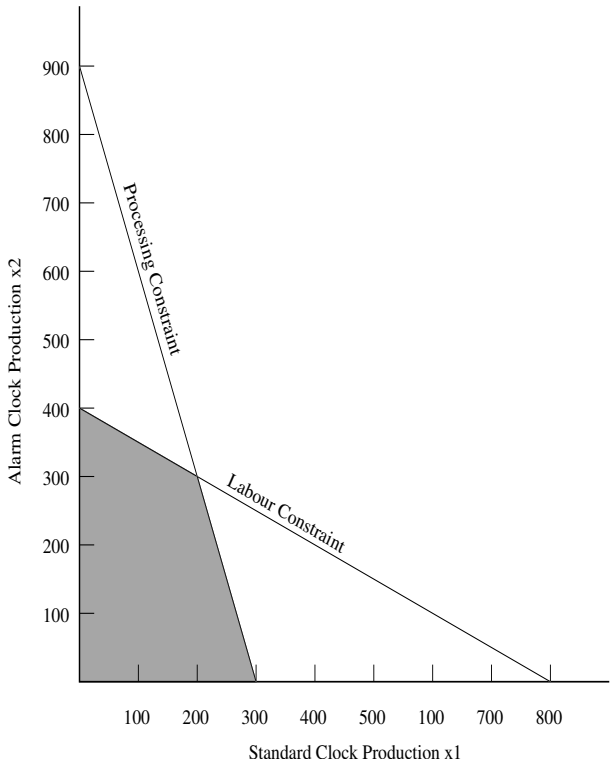
1.4. Graphical solution method for LPPs

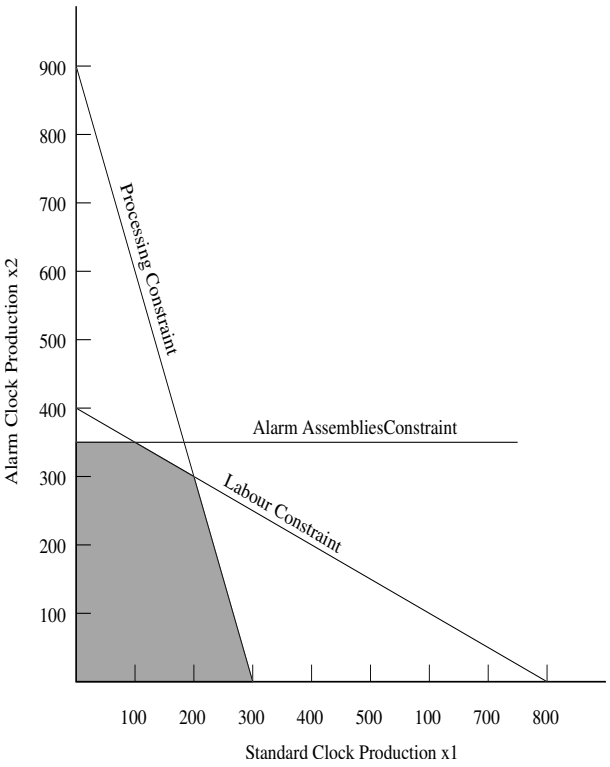
Consider the alarm clock problem above.

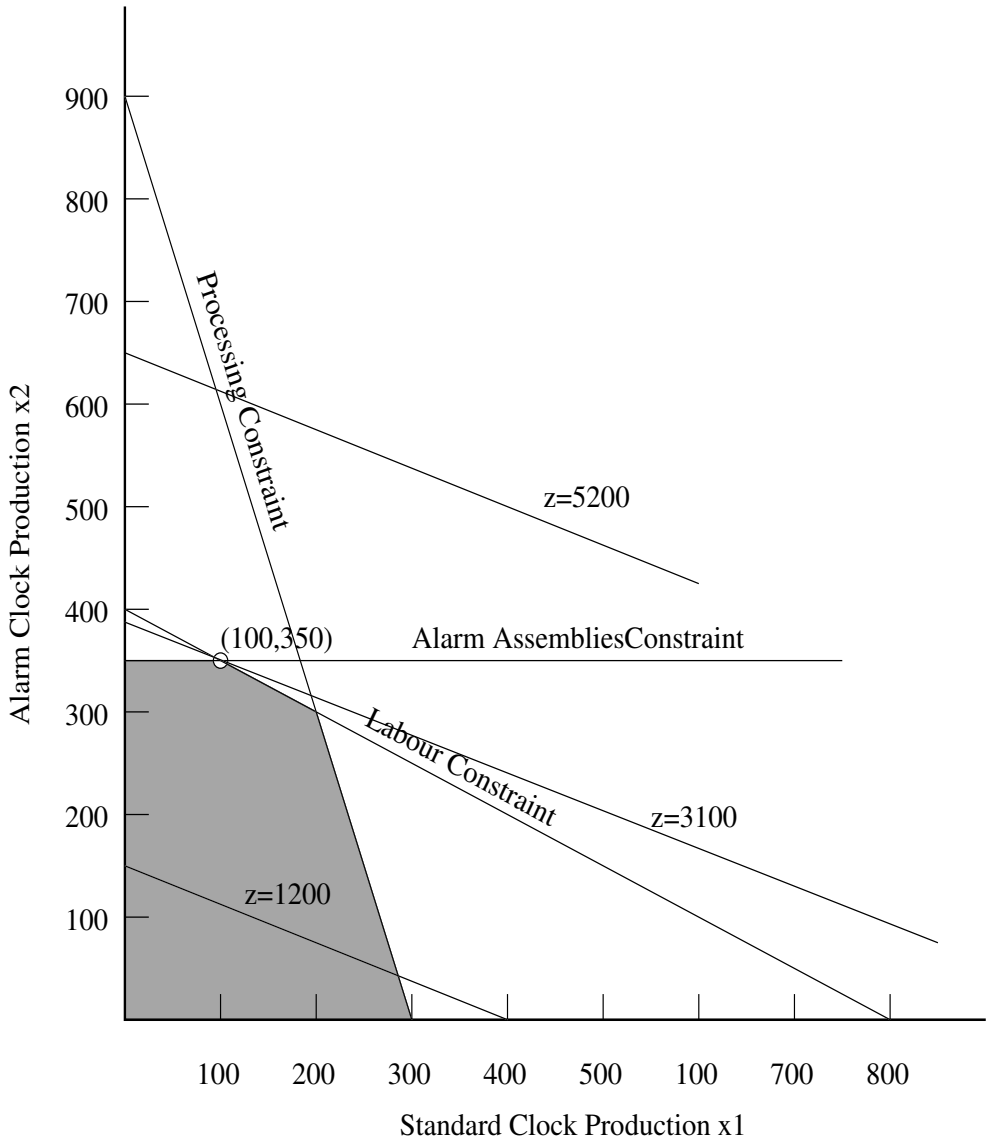
First draw the feasible region, defined by the 3 constraints.


Alarm Clock Production x2
900
800
700
600
500
400
300
200
100
100 200 300 400 500 100 700 800
Standard Clock Production x1


Alarm Clock Production x2
900
800
700
600
500
400
300
200
100
Labour Constraint
100 200 300 400 500 100 700 800
Standard Clock Production x1


Alarm Clock Production x2
900
800
700
600
500
400
300
200
100
Processing Constraint
Labour Constraint
100 200 300 400 500 100 700 800
Standard Clock Production x1


Alarm Clock Production x2
900
800
700
600
500
400
300
200
100
Processing Constraint
Alarm AssembliesConstraint
Labour Constraint
100 200 300 400 500 100 700 800
Standard Clock Production x1



Taking $z = 3x_1 + 8x_2$, we can rearrange this to get $x_2 = -\frac{3}{8}x_1 + \frac{1}{8}z$. So, for any given constant c, all points on this line $x_2 = -\frac{3}{8}x_1 + c$ yield the same value of z, and this value of z is given by $\frac{1}{8}z = c$, i.e. $z = 8c$. Lines $x_2 = -\frac{3}{8}x_1 + c$ that do not intersect with the feasible region are of no interest, as no point on such a line satisfies the constraints. We are looking for the line $x_2 = -\frac{3}{8}x_1 + c$ which both has the greatest intercept, c, and intersects the feasible region. All points on the line yield the same maximum value of z ($z = 8c$), and at least one point on this line is a feasible solution.

By taking the line $x_2 = -\frac{3}{8}x_1 + c$ for some arbitrary value of c, and moving it up and down in a parallel fashion (i.e. increasing/decreasing c whilst keeping the gradient the same) we find the line with the greatest intercept that intersects the feasible region. This gives us both the maximum value for z and the values of x_1 and x_2 that maximise it.

In this example $(x_1^*, x_2^*) = (100, 350)$ gives the optimal value $z^* = 3100$.

If, instead, the objective function were $z = 3x_1 + 2x_2$, meaning that alarm clocks are sold for £2 profit, then the optimal solution would become $(x_1^*, x_2^*) = (200, 300)$ with $z^* = 1200$.

Exercise: Check this.

Note: In both these examples, the optimising values of the control variables are unique, and lie at a vertex of the feasible region.

1.5. Extreme Point Theorem

Definitions 1.6. Let $R \subset \mathbb{R}^n$ be some region. An **extreme point** of R is any vertex on the boundary of R. R is said to be **bounded** if $\sup_{x \in R} \|x\| < \infty$. If R is the feasible region in some optimisation problem, then the extreme points of R are called **feasible basis points**.

Theorem 1.7 (Extreme point theorem)**.** *Consider a linear programming problem.*

- *If the feasible region is nonempty and bounded then an optimal solution exists.*
- *If the optimal solution is unique then it occurs at one of the extreme points of the feasible region.*
- *If a set of points all give the same optimal value of z then these points lie in a linear subspace (e.g., a line or plane) spanned by a set of extreme points.*

Examples The previous example of the alarm clock assembly problem is one that gives a unique solution.

An empty feasible region:

$$\begin{aligned} \text{maximise} \quad & z = x_1 + x_2 \\ \text{s.t.} \quad & x_1 - x_2 \geq 5 \\ & x_1 - x_2 \leq 4 \end{aligned}$$

An unbounded feasible region without an optimal solution:

$$\begin{aligned} \text{maximise} \quad & z = x_1 + x_2 \\ \text{s.t.} \quad & x_1 \geq 2 \\ & x_2 \geq 2 \end{aligned}$$

An unbounded feasible region **with** a (unique) optimal solution:

$$\begin{aligned} \text{minimise} \quad & z = x_1 + x_2 \\ \text{s.t.} \quad & x_1 \geq 2 \\ & x_2 \geq 2 \end{aligned}$$

1.6. Graphical naïve solution method

If the feasible region is bounded, the extreme point theorem tells us that the optimal solution is at an extreme point of the feasible region (a feasible basis point), or in a linear subspace spanned by a set of extreme points. So, to find the optimal solution, one can evaluate the objective function at each feasible basis point. The point that gives the highest (lowest) value for z is the optimal solution.

Example 1.8 (The alarm clock problem)**.**

Feasible basis point coordinates (x_1, x_2)	Value $z = 3x_1 + 8x_2$
$(0, 0)$	0
$(300, 0)$	900
$(0, 350)$	2800
$(200, 300)$	3000
$(100, 350)$	3100

Comments If the feasible region is not bounded, the method may give a suboptimal solution. Moreover, for complex LPPs, especially when the dimension n is very large, finding the feasible basis points is time-consuming.

1.7. Slack Variables and Surplus Variables

In the optimal solution above, $x_1^* = 100$ and $x_1^* = 350$, which implies the following use of resources.

- Labour: $2x_1^* + 4x_2^* = 1600$ hr/day
- Processing: $6x_1^* + 2x_2^* = 1300$ hr/day
- Alarm assemblies: $x_2^* = 350$ assemblies/day.

The processing constraint was $6x_1^* + 2x_2^* \leq 1800$. Since there was spare capacity here, we say that the processing constraint had 500 hr/day **slack**. The labour and alarm assembly constraints were $2x_1^* + 4x_2^* \leq 1600$ and $x_2^* \leq 350$, respectively, so the slack for these was 0. We say these constraints are **tight** for this solution.

Considering further the processing constraint:

$$6x_1 + 2x_2 \leq 1800,$$

by introducing a *slack variable* x_4 we can transform the constraint into

$$6x_1 + 2x_2 + x_4 = 1800,$$

with $x_4 \geq 0$. (If we allowed negative values for x_4 then $6x_1 + 2x_2$ could be > 1800.)

So the inequality has been transformed into an equality. This will be utilised in the next chapter.

Suppose there had been another constraint that was a minimum ($\geq$) not a maximum ($\leq$). E.g. the total number of clocks produced must be at least 100.

$$x_1 + x_2 \geq 100.$$

We now introduce a *surplus variable* x_6 and transform the constraint into

$$x_1 + x_2 - x_6 = 100.$$

with $x_6 \geq 0$.

Note: We always arrange matters so that slack and surplus variables are non-negative.

The full set of constraints, for the clock problem with a minimum of 100 clocks, is:
Labour: $2x_1 + 4x_2 + x_3 = 1600$

Processing: $6x_1 + 2x_2 + x_4 = 1800$
Alarm assemblies: $x_2 + x_5 = 350$
Minimum number of clocks: $x_1 + x_2 - x_6 = 100$
as well as positivity: $x_i \geq 0$ for $i = 1, \ldots, 6$.

1.8. Graphical sensitivity analysis

Sensitivity analysis means determining how the optimal solution changes when the specification of the LPP changes. How sensitive is the optimal solution to the correct model specification?

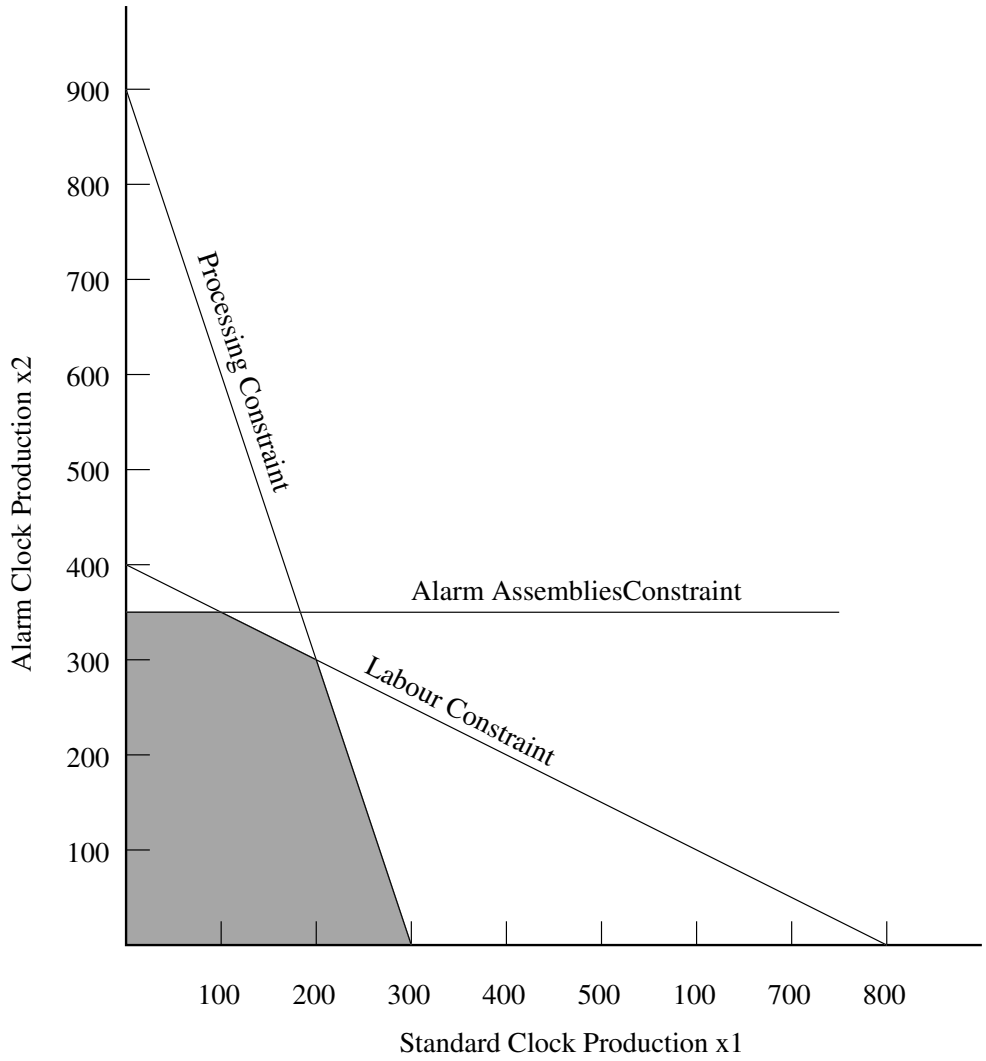
The specification of the LPP may change in three ways:

1. The coefficients of the objective function (c_j) change,
2. The resource constraints (b_i) change, or
3. The coefficients of the constraints ($a_{i,j}$) change.

Below, we look at 1 and 2.

Definition 1.9. The **shadow price** of a constraint is the amount by which the optimal value of the objective function changes when the resource value of that constraint is increased by one unit.

Returning to the original clock problem.



Suppose we increase the available labour resource by 1 unit, to 1601. Then the line representing the labour constraint will change from $x_2 = \frac{1}{4}(1600 - 2x_1)$ to $x_2 = \frac{1}{4}(1601 - 2x_1)$; in other words, the intercept changes but the gradient does not. It is simple to see that thee optimal solution will still be at the "same" vertex as before, in the sense of the intersection of the labour and alarm assemblies lines. Hence, the optimal value of x_2 is still $x_2^* = 350$, thus

$$x_1^* = (1601 - 4x_2^*)/2 = 100.5$$
$$z^* = 3101.5$$

Because the problem is linear, the objective function increases by 1.5 for every unit increase in the labour resource. The shadow price of the labour constraint is £1.5 (per labour unit).

Suppose, instead, the processing resource had increased by one unit. Then the optimal solution would not change at all. This is because the optimal solution does not lie on the line corresponding to the processing constraint. Thus, the shadow price for processing is zero. In general, if the slack/surplus variable for a constraint is non-zero then the shadow price of the constraint will be zero.

Finally, suppose the number of alarm assemblies available increases by one. Then

$$x_1^* = 98$$
$$x_2^* = 351$$
$$z^* = 3102$$

and so the shadow price for assemblies is £2 (per assembly unit).

Interpretation of the shadow price Suppose the clock company considers increasing the number of alarm assemblies available at a cost. The shadow price of £2 tells them that they should do so, provided the cost of buying in each extra alarm assembly is less than £2.

Connection with slack The labour and alarm assembly constraints both have non-zero shadow price, and they are also tight (they have zero slack). This is not a coincidence – we will return to it later.

At what point does the shadow price drop to zero? Look at the previous graph. As the labour resource increases, the optimal solution will increase until some point P. This corresponds to the processing slack being reduced to zero. Increasing the labour resource beyond this point will have no effect on the optimal solution and the labour slack will increase.

Point P is the intersection of the processing and alarm constraints, located at $P = (\frac{550}{3}, 350)$, where $z = 3350$, corresponding to a labour resource of $1766\frac{2}{3}$. It is not worthwhile to increase the available labour to more than $1766\frac{2}{3}$ hours/day.

Changes in the objective function From before,

$$\begin{aligned} z &= 3x_1 + 8x_2 \\ \Rightarrow x_2 &= -\frac{3}{8}x_1 + \frac{z}{8} \end{aligned}$$

So, the gradient of the contours of the objective function is $-\frac{3}{8}$.

In the graph, the gradient of the assembly constraint is zero and the gradient of the labour constraint is $-\frac{1}{2}$. For the optimal solution to be at the assembly-labour vertex, (1) we must be maximising the contour intercept, and (2) the gradient of the objective function must be between $-\frac{1}{2}$ and 0.

What happens if either of these fails? If (2) fails, and the gradient of the objective function is between -3 and $-\frac{1}{2}$, then a greater value of the objective function can be obtained at the labour-processing vertex. If the gradient of the objective function is exactly $-\frac{1}{2}$ then it is parallel to the labour constraint and any point on the labour constraint between the two vertices is optimal. On the other hand, if $z = -3x_1 - 8x_2$, then the gradient of the contours is still $-\frac{3}{8}$, but (1) fails: we are minimising the intercept, so $\mathbf{x}^* = \mathbf{0}$.

Question: If $z = c_1x_1 + c_2x_2$, what values of c_1 and c_2 give the same values of $x_1^* = 100$, $x_2^* = 350$?

Solution: Rearrange the equation to obtain $x_2 = -\frac{c_1}{c_2}x_1 + \frac{1}{c_2}z$. The gradient of this line is $-\frac{c_1}{c_2}$. In order for an optimal solution to be $(x_1^*, x_2^*) = (100, 350)$, we need $c_2 > 0$ to be maximising the intercept, and we also require $-\frac{1}{2} \leq -\frac{c_1}{c_2} \leq 0$, or equivalently $0 \leq \frac{c_1}{c_2} \leq \frac{1}{2}$.

If we change the value of c_1, the optimal solution will be the same if c_1 remains positive and does not exceed $\frac{1}{2}c_2$.

If we change only c_2, the optimal solution will remain the same if c_2 exceeds $2c_1$.

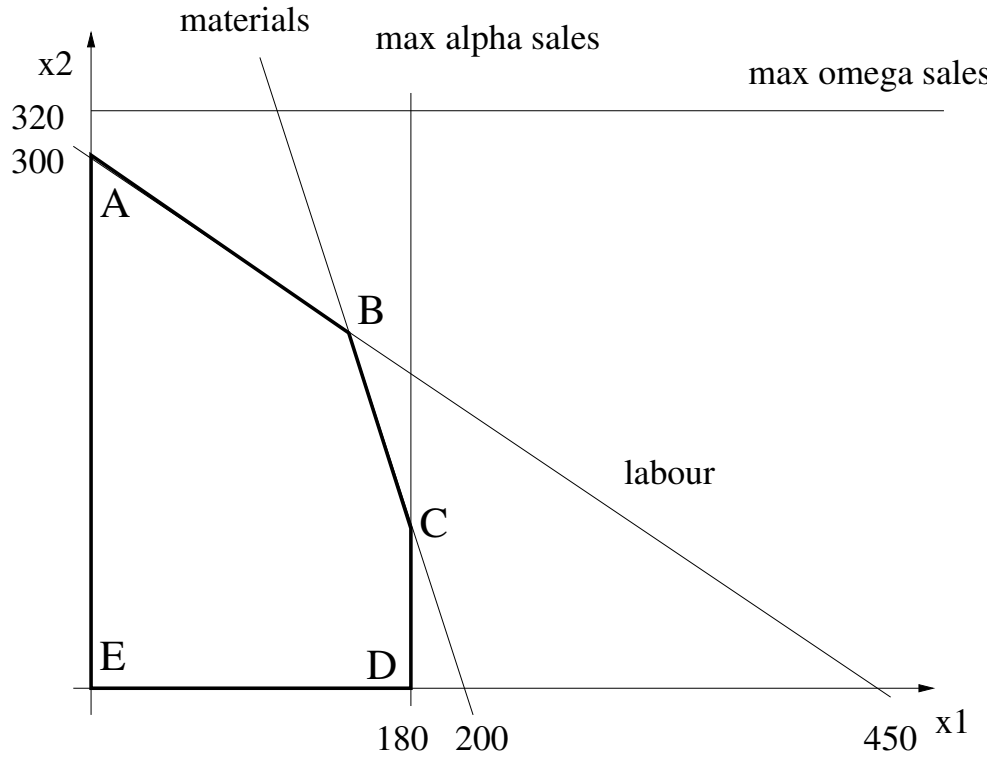
Note: The optimal solution remains the same (i.e., x_1^* and x_2^* remain unchanged), but the optimal value of the objective function will change.

Example 1.10. Suppose a company makes two models of car. These models are called Alpha and Omega. The number of Alpha models to be produced is x_1 and the number of Omega models is x_2.

We have the following linear programming problem:

$$\begin{aligned} \text{maximise} \quad & z = 6x_1 + 5x_2 \quad \text{(profit)} \\ \text{s.t.} \quad & 4x_1 + x_2 \leq 800 \quad \text{(materials)} \\ & 2x_1 + 3x_2 \leq 900 \quad \text{(labour)} \\ & x_1 \leq 180 \quad \text{(max Alpha sales)} \\ & x_2 \leq 320 \quad \text{(max Omega sales)} \\ & x_1, x_2 \geq 0. \end{aligned}$$

Which gives the graph:

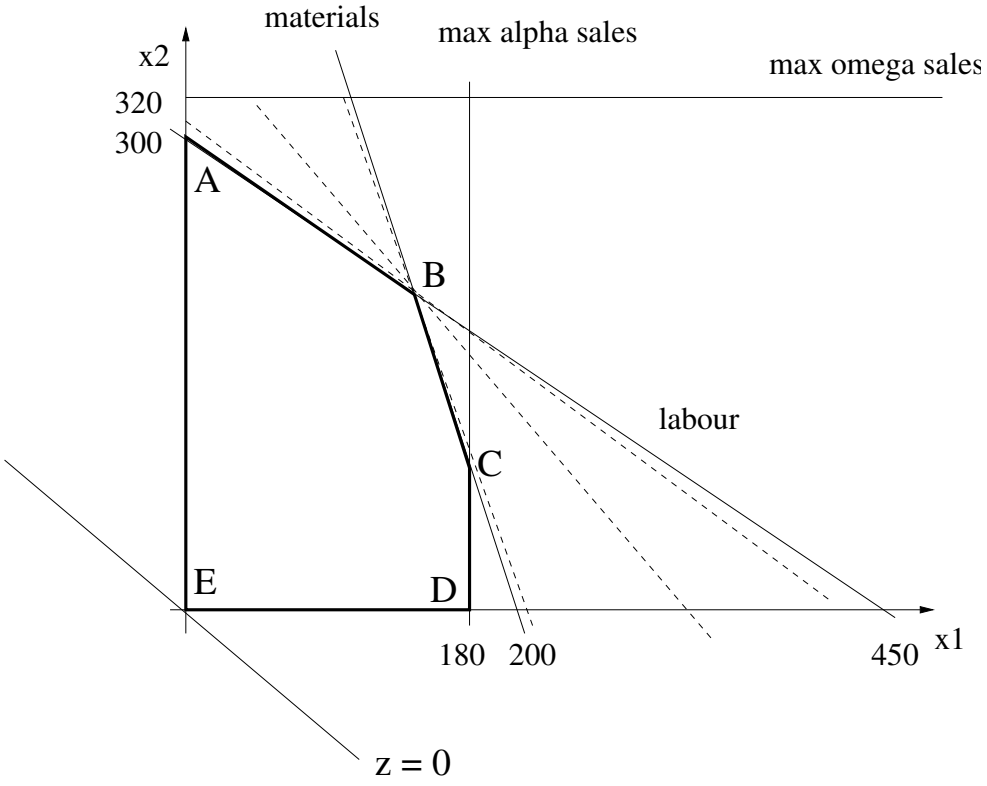


The line representing the contours of the objective function is $x_2 = -\frac{6}{5}x_1 + c$, and this can be used with the graphical method to determine the solution. Equally, we can list the feasible basis points and the value of z at each one:

Feasible basis point coordinate (x_1, x_2)	Value $z = 6x_1 + 5x_2$
$A = (0, 300)$	1500
$B = (150, 200)$	1900
$C = (180, 80)$	1480
$D = (180, 0)$	1080
$E = (0, 0)$	0

The optimal solution is at point B, giving
$x_1^* = 150$, $x_2^* = 200$, $z^* = 1900$.

Again, if the objective function is altered, the location of the optimal solution could change to another vertex. Let $z = c_1x_1 + c_2x_2$, with the original values $c_1 = 6$, $c_2 = 5$. We shall find the values of c_1 and c_2 for which the optimal solution is still located at the same vertex.



In the plot above, a contour $z = 0$ of the original objective function is shown (the solid line through E), along with contours of the objective function through B when the values of c_1 and c_2 are changed (dashed lines). The slope of the contours of z is $-c_1/c_2$.

In order for B to remain the optimal solution, firstly we need that $c_2 > 0$ in order to be maximising the intercept; secondly, the slope of the contours has to lie between the slopes of the materials constraint (-4) and the labour constraint $(-\frac{2}{3})$.

Therefore, for B to be optimal,

$$-4 \leq -\frac{c_1}{c_2} \leq -\frac{2}{3}$$
$$\frac{1}{4}c_1 \leq c_2 \leq \frac{3}{2}c_1$$

In the original problem, $c_1 = 6$, so the **tolerance interval** for c_2 of $[1.5, 9]$. If the profit from the Omega model (c_2) changes next month, then so long as it lies within the tolerance interval, the company does not need to change its production plan (though the overall profit may change.)

Exercise

1. What is the tolerance interval for c_1, given the original value $c_2 = 5$?
2. What are the shadow prices for the four constraints in this problem?

2. The simplex algorithm

In this chapter we shall assume that all linear programming problems (LPPs) are **maximisation** problems and that all control variables, x_i, must be **non-negative**, unless otherwise stated.

General Notation

A LPP may be written using matrix notation.

For example,

$$\begin{aligned} \text{Maximise } z &= \mathbf{c}^T\mathbf{x} && \text{objective function} \\ \text{subject to } \mathbf{A}\mathbf{x} &\leq \mathbf{b} && \text{constraints} \\ \mathbf{x} &\geq \mathbf{0}, \end{aligned} \tag{2.1}$$

where $\mathbf{A} = [a_{i,j}]$ is an $m \times n$ matrix, $\mathbf{b}$ is an m-vector of constraints (resources), $\mathbf{x}$ is an n-vector of variables, and $\mathbf{c}$ is an n-vector of objective function coefficients.

Here, '$\mathbf{Ax} \leq \mathbf{b}$' indicates that the inequality holds elementwise, that is, $(\mathbf{Ax})_i \leq b_i$ for $i = 1, \ldots, m$, where $(\mathbf{Ax})_i$ is the i-th element of the m-vector $\mathbf{Ax}$. Likewise, '$\mathbf{x} \geq \mathbf{0}$' means that $x_i \geq 0$ for all $i = 1, \ldots, n$.

The above LPP is in **standard form**. This is because equation (2.1) has a $\leq$ inequality, and $\mathbf{x}$ are all non-negative.

If instead the problem is

$$\begin{aligned} \text{Maximise} \quad & z = \mathbf{c}_c^T\mathbf{x}_c \\ \text{subject to} \quad & \mathbf{A}_c\mathbf{x}_c = \mathbf{b} \\ & \mathbf{x}_c \geq \mathbf{0}, \end{aligned} \tag{2.2}$$

then the LPP is in *canonical form*. (The subscript 'c' is just to indicate that this is canonical form.)

By using slack/surplus variables, it is always possible to convert any LPP into canonical form. To see this, consider the following example.

Example 2.1 (Clock LPP revisited)**.** The clock LPP is:

$$\begin{aligned} \text{Maximise } z &= 3x_1 + 8x_2 \\ \text{subject to } 2x_1 + 4x_2 &\leq 1600 \\ 6x_1 + 2x_2 &\leq 1800 \\ x_2 &\leq 350 \\ x_1, x_2 &\geq 0. \end{aligned}$$

It can be written in matrix notation in standard form as (2.1), with

$$\mathbf{c} = \begin{pmatrix} 3 \\ 8 \end{pmatrix}, \qquad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix},$$

$$\mathbf{A} = \begin{pmatrix} 2 & 4 \\ 6 & 2 \\ 0 & 1 \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} 1600 \\ 1800 \\ 350 \end{pmatrix}.$$

Add slack variables to transform it into canonical form:

$$\begin{aligned} \text{Maximise } z &= 3x_1 + 8x_2 \\ \text{subject to } 2x_1 + 4x_2 + x_3 &= 1600 \\ 6x_1 + 2x_2 + x_4 &= 1800 \\ x_2 + x_5 &= 350 \\ x_1, x_2, x_3, x_4, x_5 &\geq 0. \end{aligned}$$

In matrix form (2.2), this corresponds to:

$$\mathbf{c}_c = \begin{pmatrix} 3 \\ 8 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad \mathbf{x}_c = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}$$

$$\mathbf{A}_c = \begin{pmatrix} 2 & 4 & 1 & 0 & 0 \\ 6 & 2 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} 1600 \\ 1800 \\ 350 \end{pmatrix}.$$

In general, any problem in standard form (2.1) can be converted to one in canonical form (2.2), by defining the $m \times (n+m)$ matrix

$$\mathbf{A}_c = \left(\begin{array}{c|cccc} & 1 & 0 & \cdots & 0 \\ \mathbf{A} & 0 & 1 & \cdots & 0 \\ & \vdots & & \ddots & \\ & 0 & 0 & \cdots & 1 \end{array} \right)$$

or in other words $\mathbf{A}_c = (\ \mathbf{A} \mid \mathbf{I}_m\ )$, where $\mathbf{I}_m$ is the $m \times m$ identity matrix. We also have $\mathbf{x}_c = (x_1, \ldots, x_{n+m})^T$ and $\mathbf{c}_c = \begin{pmatrix} \mathbf{c} \\ \mathbf{0} \end{pmatrix}$.

2.1. Theory leading to the simplex algorithm

Assume the rows of $\mathbf{A}_c$ are linearly independent. Otherwise, at least one constraint is redundant (i.e., follows from the other constraints) and can be removed without affecting the problem.

Solving $\mathbf{A}_c\mathbf{x}_c = \mathbf{b}$ gives a region $R = \{\mathbf{x}_c \in \mathbb{R}^{n+m} : \mathbf{A}_c\mathbf{x}_c = \mathbf{b}\}$ satisfying the m resource constraints of the canonical problem.

The region $S = \{\mathbf{x}_c \in R : \mathbf{x}_c \geq \mathbf{0}\}$ is the feasible region of the LPP.

Definitions 2.2. If a vector $\mathbf{x}_c$ solves the equation $\mathbf{A}_c\mathbf{x}_c = \mathbf{b}$ and $\mathbf{x}_c$ has at most m non-zero elements, $\mathbf{x}_c$ is called a **basic solution** of the LPP.

If $\mathbf{x}_c$ is a basic solution, and every element of $\mathbf{x}_c$ is also non-negative, then $\mathbf{x}_c$ is called a **feasible basic solution** of the LPP.

Theorem 2.3. *Every extreme point of S is a feasible basic solution of the LPP, and conversely every feasible basic solution of the LPP is an extreme point of S.*

In the last chapter we saw that the optimal solution of a LPP lies at an extreme point of the feasible region (or in a linear space spanned by some set of extreme points.) So, from the theorem above, any optimal solution must be one of the feasible basic solutions (or such a linear space.)

So, if we check all the basic solutions, determine which are feasible, and pick the one which maximises z, we are done.

2.2. Numerical naïve method

How do we find the basic solutions?

The idea is simple: pick m of the control variables, and allow those to be non-zero (which means that the remaining n variables are forced to be zero). Try to solve the resulting equation. This will give one basic solution. Then change which control variables are allowed to be non-zero, and repeat. The idea is probably best understood by example.

The m elements of $\mathbf{x}_c$ which are allowed to be non-zero are called **basic** variables. The remaining n variables set to zero are called **non-basic** variables.

Using this method it is straightforward to find all the basic solutions, check which ones are feasible and then inspect which of these feasible basic solutions gives the largest value of z. This is the optimal solution.

There are $\binom{m+n}{m}$ combinations of m basic variables, and hence (at most) $\binom{m+n}{m}$ basic solutions to be found.

Example 2.4 (Clock LPP revisited)**.** Earlier, we wrote the clock LPP in canonical form using matrix notation:

$$\mathbf{A}_c\mathbf{x}_c = \begin{bmatrix} 2 & 4 & 1 & 0 & 0 \\ 6 & 2 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 1600 \\ 1800 \\ 350 \end{bmatrix} = \mathbf{b}$$

There are $\binom{5}{3} = 10$ ways of choosing $m = 3$ basic variables, and so 10 basic solutions to obtain.

If we choose x_1, x_2, x_3 as the basic (non-zero) variables, we obtain:

$$\begin{bmatrix} 2 & 4 & 1 & 0 & 0 \\ 6 & 2 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 & 4 & 1 \\ 6 & 2 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1600 \\ 1800 \\ 350 \end{bmatrix}$$

This point is very important. Notice that because the non-basic variables are zero, the equation simplifies. When x_1, x_2, x_3 are basic, the equation $\mathbf{A}_c\mathbf{x}_c = \mathbf{b}$ is equivalent to $\mathbf{A}'\mathbf{x}' = \mathbf{b}$, where $\mathbf{A}'$ consists of columns $1, 2, 3$ of $\mathbf{A}_c$, and $\mathbf{x}' = (x_1, x_2, x_3)^T$.

Solving the equation gives $x_1 = 183\frac{1}{3}$, $x_2 = 350$, $x_3 = -166\frac{2}{3}$. This corresponds to $\mathbf{x}_c = (183\frac{1}{3}, 350, -166\frac{2}{3}, 0, 0)^T$, which is a basic solution but not a feasible one, since one of the variables is negative.

Now consider letting x_1, x_2, x_4 be basic. By the reasoning above, this is the same as selecting out columns $1, 2, 4$, to give the equation:

$$\begin{pmatrix} 2 & 4 & 0 \\ 6 & 2 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1600 \\ 1800 \\ 350 \end{pmatrix}$$

corresponding to $\mathbf{x}_c = (100, 350, 0, 500, 0)^T$. This is a feasible basic solution with value $z = 3100$.

Using columns 1, 2 and 5:

$$\begin{pmatrix} 2 & 4 & 0 \\ 6 & 2 & 0 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_5 \end{pmatrix} = \begin{pmatrix} 1600 \\ 1800 \\ 350 \end{pmatrix}$$

corresponding to $\mathbf{x}_c = (200, 300, 0, 0, 50)^T$. This is a feasible basic solution with value $z = 3000$.

Using columns 1, 3 and 4:

$$\begin{pmatrix} 2 & 1 & 0 \\ 6 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1600 \\ 1800 \\ 350 \end{pmatrix}.$$

This equation has no solution.

The following table summarises the results for the other six combinations of three basic variables (three columns):

Columns			x_1	x_2	x_3	x_4	x_5	Feasible?	z
1	3	5	300	0	1000	0	350	Y	900
1	4	5	800	0	0	-3000	350	N	
2	3	4	0	350	200	1100	0	Y	2800
2	3	5	0	900	-2000	0	-550	N	
2	4	5	0	400	0	1000	-50	N	
3	4	5	0	0	1600	1800	350	Y	0

The feasible basic solution maximising the objective function is the second combination we tried (selecting columns 1, 2 and 4) with $x_1^* = 100$, $x_2^* = 350$, $x_3^* = 0$, $x_4^* = 500$, $x_5^* = 0$, giving $z^* = 3100$. This confirms the result found graphically.

Example 2.5 (Another example)**.** The following LPP is in canonical form.

$$\begin{aligned} \text{Maximise} \quad & z = x_1 + x_2 \\ \text{s.t.} \quad & x_1 + 2x_2 + x_3 = 10 \\ & 2x_1 + x_2 + x_4 = 12. \\ & x_1, x_2, x_3, x_4 \geq 0. \end{aligned}$$

This gives

$$\mathbf{A}_c = \begin{pmatrix} 1 & 2 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{pmatrix}$$

with

$$\mathbf{x}_c = (x_1, x_2, x_3, x_4)^T, \qquad \mathbf{b} = (10, 12)^T.$$

Consider the basic solutions. One combination of $m = 2$ columns of $\mathbf{A}_c$ is column 1 and column 3. This gives us the equation

$$\mathbf{A}'\mathbf{x}' = \begin{pmatrix} 1 & 1 \\ 2 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_3 \end{pmatrix} = \begin{pmatrix} 10 \\ 12 \end{pmatrix} = \mathbf{b},$$

yielding $x_1 = 6$, $x_2 = 0$, $x_3 = 4$ and $x_4 = 0$. As $x_1 \geq 0$ and $x_3 \geq 0$, the solution is feasible. Its value is $z = 6$.

Exercise: repeat this for each combination of two columns of $\mathbf{A}_c$, and find the optimal solution.

Comments

1. The method we have outlined above is essentially a numerical version of the graphical naïve method from section 1.6. The graphical method becomes difficult for $n \geq 3$ as it is harder to visualise the feasible region.

2. The number of possible solutions $\binom{m+n}{m}$ can become large very rapidly. For example, $m = 5$, $n = 5$ gives $\binom{10}{5} = 252$ solutions to be found.

3. The two examples we have looked at are both problems in standard form. When this is not so, the LPP in canonical form will still be $\mathbf{A}_c\mathbf{x}_c = \mathbf{b}$, but $\mathbf{A}_c$ may not have $m+n$ columns. The principle remains the same. Suppose that $\mathbf{A}_c$ is a $m \times l$ matrix, where $l \geq m$. To find basic solutions, we still take $m \times m$ submatrices $\mathbf{A}'$ of $\mathbf{A}_c$ and solve $\mathbf{A}'\mathbf{x}' = \mathbf{b}$. The vector $\mathbf{x}'$ is still of length m and there are still m basic variables, but now there are $l - m$ non-basic variables (instead of exactly n of them.)

2.3. The simplex algorithm

Assume for now that we are looking at a problem in standard form

$$\begin{aligned} \text{Maximise} \quad & z = \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0, \end{aligned}$$

with $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$ and $\mathbf{x}, \mathbf{c} \in \mathbb{R}^n$. Assume moreover that $b_i \geq 0$ for all $i = 1, \ldots, m$. We shall consider later what to do when this is not true.

In the numerical naïve method, we had to solve $\mathbf{A}'\mathbf{x}' = \mathbf{b}$ for each of the $\binom{m+n}{m}$ possible matrices $\mathbf{A}'$. One of these is trivial to solve: the one where $\mathbf{A}'$ consists of the last m columns of $\mathbf{A}_c$. $\mathbf{A}'$ is then the identity matrix and the solution is just $\mathbf{x}' = \mathbf{b}$ (see Example 2.4). Thus, one basic solution is just $x_1 = \cdots = x_n = 0$ (the non-basic variables) and $x_{n+i} = b_i$ for $i = 1, \ldots, m$ (the basic variables). That is, set each slack variable equal to its corresponding resource value and all the original variables to be zero. This basic solution is feasible (because $b_i \geq 0$ for all i) and $z = 0$.

This section will outline an algorithm, the **simplex algorithm**, in which the problem is laid out in a tabular format, starting from this initial solution, and we successively visit other feasible basic solutions in such a way that the objective function always increases. Once z can no longer be increased, we have found an optimal solution.

The advantage of the simplex algorithm is that we do not have to investigate explicitly all of the $\binom{m+n}{m}$ basic solutions, only the interesting ones.

Rearranging the objective function $z = c_1x_1 + \cdots + c_nx_n$, we obtain

$$-c_1x_1 - \cdots - c_nx_n + z = 0.$$

Our m constraints, written in canonical form, are $\mathbf{A}_c\mathbf{x}_c = \mathbf{b}$. So, we can write the entire problem as

$$\begin{bmatrix} & & & & & & 0 \\ & & & \mathbf{A}_c & & & \vdots \\ & & & & & & 0 \\ -c_1 & \cdots & -c_n & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_{n+m} \\ z \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \\ 0 \end{bmatrix} \tag{2.3}$$

From this we have the general format of the initial tableau:

Basic vars.	x_1	x_2	$\cdots$				x_{n+m}	Sol.
x_{n+1}								b_1
x_{n+2}								b_2
$\vdots$				$\mathbf{A}_c$				$\vdots$
x_{n+m}								b_m
z	$-c_1$	$-c_2$	$\cdots$	$-c_n$	0	$\cdots$	0	0

Clock example: initial tableau

	x_1	x_2	x_3	x_4	x_5	Sol.
x_3	2	4	1	0	0	1600
x_4	6	2	0	1	0	1800
x_5	0	1	0	0	1	350
z	-3	-8	0	0	0	0

Our initial basic solution is easy to read off from this tableau: $x_1 = x_2 = 0$ (the non-basic variables) and $x_3 = 1600$, $x_4 = 1800$ and $x_5 = 350$ (the basic variables), with $z = 0$. We say that this tableau corresponds to this basic solution. Note that this basic solution is feasible since all the variables are non-negative.

The simplex algorithm now looks for another feasible basic solution that has a greater value of z. This new basic solution will be 'adjacent' to the old one. That is, $m - 1$ of the basic variables will be same and one will be different.

First, we choose which of the non-basic variables to 'enter' (into the set of basic variables). Second, we choose which basic variable to 'exit' (from the set of basic variables).

1. **Enter.** Find the non-basic variable with the most negative entry in the the last row (the objective function row or z row). This variable will enter.

   In the example, x_2, has z-row entry -8, so x_2 enters.

2. **Exit.** Find the basic variable corresponding to the row with the smallest non-negative 'θ-ratio'. This variable will exit. The θ-ratio for **row i when x_j is entering** is given by

$$\theta_i = \begin{cases} b_i/a_{i,j}, & a_{i,j} \geq 0, \\ -\infty, & a_{i,j} < 0. \end{cases}$$

   The row corresponding to the exiting variable is called the **pivotal row**.

   In this example, the θ-ratios are:

$$\begin{aligned} \theta_1 &= 1600/4 = 400, \\ \theta_2 &= 1800/2 = 900, \\ \theta_3 &= 350/1 = 350. \end{aligned}$$

   The smallest ratio is for row 3, so x_5 is the exiting variable. Row 3 is the pivotal row.

3. **Update.** Use 'elementary row operations' to 'isolate' the entering variable (and keep the other basic variables isolated.) Then relabel the pivotal row with the entering variable.

   A variable is **isolated** when the column corresponding to this variable consists of zero entries except for the entry in its corresponding row, which equals one.

   The elementary row operations are:

   - Multiplying a row by a constant.
   - Adding one row to another.

   (These will be familiar from linear algebra: they are used when solving a system of linear equations.)

   In this example, we want the x_2 column, $\begin{bmatrix} 4 \\ 2 \\ 1 \\ -8 \end{bmatrix}$, to become $\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$.

   Row 3 already has a 1 in the x_2 column. So there is no need to alter row 3, except that we relabel it with x_2, because the old basic variable x_5 is being replaced by x_2. We can write this as $[3] \to [3]$.

   If we add -4 times row 3 to row 1, row 1 becomes

   $$x_3 \mid 2 \quad 0 \quad 1 \quad 0 \quad -4 \mid 200.$$

   The element in the x_2 column is now zero, as required. We can write this as $[1] - 4 \times [3] \to [1]$.

   To isolate x_2, take

$$\begin{aligned} [3] &\to [3] \\ [1] - 4 \times [3] &\to [1] \\ [2] - 2 \times [3] &\to [2] \\ [z] + 8 \times [3] &\to [z] \end{aligned}$$

After performing these steps we have the second tableau:

	x_1	x_2	x_3	x_4	x_5	Sol.
x_3	2	0	1	0	-4	200
x_4	6	0	0	1	-2	1100
x_2	0	1	0	0	1	350
z	-3	0	0	0	8	2800

Variables x_2, x_3 and x_4 are now the basic variables — and so x_1 and x_5 are non-basic. The values of x_2, x_3, x_4 and z can easily be read off the tableau: $x_2 = 350$, $x_3 = 200$, $x_4 = 1100$, $z = 2800$. (Note that $z = 3x_1 + 8x_2$ does indeed equal 2800.)

We have now completed one full iteration in the simplex algorithm.

Notes

- z has increased.
- The values of the basic variables and of z come from:

$$\begin{aligned} 0x_2 + 1x_3 + 0x_4 &= 200 \\ \Rightarrow x_3 &= 200 \\ 0x_2 + 0x_3 + 1x_4 &= 1100 \\ \Rightarrow x_4 &= 1100 \\ 1x_2 + 0x_3 + 0x_4 &= 350 \\ \Rightarrow x_2 &= 350 \\ 0x_2 + 0x_3 + 0x_4 + z &= 2800 \\ \Rightarrow z &= 2800 \end{aligned}$$

  since $x_1 = x_5 = 0$.
- It is easy to read off the values of the basic variables because the submatrix consisting of the columns corresponding to the basic variables is the identity matrix (with columns permuted).

Second iteration We now repeat the same process to see if we can get a solution that is even better.

1. The entering variable is x_1.
2. The θ ratios are 100, $183\frac{1}{3}$ and ∞. The exiting variable is x_3, in row 1 which is the pivotal row.
3. The third tableau is:

	x_1	x_2	x_3	x_4	x_5	Sol.
x_1	1	0	$\frac{1}{2}$	0	-2	100
x_4	0	0	-3	1	10	500
x_2	0	1	0	0	1	350
z	0	0	$\frac{3}{2}$	0	2	3100

Stopping criterion When the last row has non-negative entries for every non-basic variable, then the stopping criterion has been met. This means that the optimal solution has been reached, which can be read off directly from the tableau. (In principle, it is possible for the method to **cycle**, in which case this stopping condition is never reached. This occurs for so-called **degenerate** problems, which will be discussed in the next section.)

In our example, the stopping criterion is satisfied in the third tableau. Therefore, the optimal solution is $x_1^* = 100$, $x_2^* = 350$, $x_3^* = 0$, $x_4^* = 500$, $x_5^* = 0$ and $z^* = 3100$. (Verify that $z^* = 3x_1^* + 8x_2^*$ equals 3100.)

2.4. Why Does the Simplex Algorithm Work?

Full details are available in our references, such as Bertsimas and Tsitsiklis, Hillier and Lieberman, or Gordon and Pressman. Here I provide a brief account.

We have seen that the initial tableau represents the pair of equations

$$\mathbf{A}_c\mathbf{x}_c = \mathbf{b}, \qquad z = c_1x_1 + \cdots + c_nx_n \tag{2.4}$$

as

$$\begin{bmatrix} & & & & & & 0 \\ & & \mathbf{A}_c & & & & \vdots \\ & & & & & & 0 \\ -c_1 & \cdots & -c_n & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_c \\ z \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ 0 \end{bmatrix} \tag{2.5}$$

plus the choice of initial basic variables $x_{n+1}, \ldots, x_{n+m}$. The initial feasible basic solution comes from this tableau:

$$\begin{aligned} x_1 &= \cdots = x_n = 0 \\ x_{n+i} &= b_i \quad (i = 1, \ldots, m) \\ z &= 0 \end{aligned}$$

To obtain the second tableau, and indeed any subsequent tableau, we perform elementary row operations. This amounts to multiplying (2.5) on the left by some invertible matrix. As a result, we obtain equation

$$\begin{bmatrix} & 0 \\ \mathbf{BA}_c & \vdots \\ & 0 \\ -\mathbf{d} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_c \\ z \end{bmatrix} = \begin{bmatrix} \mathbf{Bb} \\ e \end{bmatrix} \tag{2.6}$$

together with a new choice of basic variables. The set of solutions of (2.5) is the same as the set of solutions of (2.6).

The matrix $\mathbf{B}$ is chosen according to our updating rules. This implies that the columns corresponding to the basic variables are a permutation of the identity matrix, and the entries of $-\mathbf{d}$ are zero when they correspond to a basic variable. Because of this, the values of the basic variables and the value of z can be read off from the equation, as before.

In the initial solution, $x_1 = \cdots = x_n = 0$. So, equation (2.4) gives $z = 0$. In the second feasible basic solution, the variable x_i from $\{x_1, \ldots, x_n\}$ with the largest positive c_i value (and hence most negative entry in the z row) has been allowed to be positive, while one of the variables $x_{n+1}, \ldots, x_{n+m}$ has been set to zero. From equation (2.4), we see that setting $x_i > 0$ will increase z, while setting any of $x_{n+1}, \ldots, x_{n+m}$ equal to zero will not change z. Thus, the second feasible basic solution will have a greater z value than the initial solution.

The same reasoning shows that our third feasible basic solution will be better than our second, and so on.

The last row of (2.6) says that

$$z - d_{k_1}x_{k_1} - \cdots - d_{k_n}x_{k_n} = e \tag{2.7}$$

where $x_{k_1}, \ldots, x_{k_n}$ are the n non-basic variables, and $d_{k_1}, \ldots, d_{k_n}$ are the corresponding entries of $\mathbf{d}$. When all entries of the z row are non-negative (so, $d_{k_1}, \ldots, d_{k_n} \leq 0$), it is clear that making any of the non-basic variables into basic variables (i.e. allowing them to be positive) can only decrease z. Therefore, we have found an optimal solution.

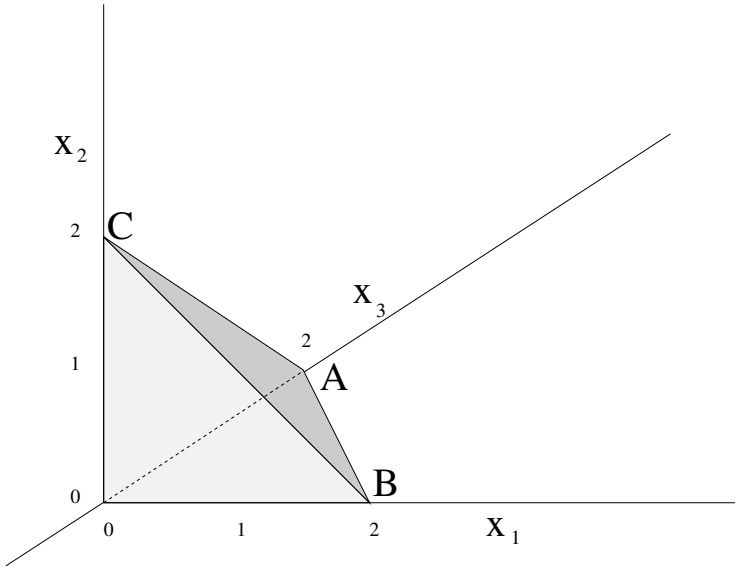
Example 2.6 (A simple example)**.**

$$\begin{aligned} \text{Maximise} \quad & z = x_1 + 2x_2 \\ \text{s.t.} \quad & x_1 + x_2 \leq 2 \\ & x_1, x_2 \geq 0 \end{aligned}$$

The corresponding canonical form is

$$\begin{aligned} \text{Maximise} \quad & z = x_1 + 2x_2 \\ \text{s.t.} \quad & x_1 + x_2 + x_3 = 2 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

As this is in three dimensions, the problem can be plotted.



The area enclosed by triangle ABC is the feasible region. (Notice that if you **project** this region onto the x_1-x_2 plane, you obtain the feasible region of the standard form problem.)

The vertices A, B and C are the three basic solutions (two zero elements and one non-zero) and they are all feasible.

The initial solution is vertex $A = (0, 0, 2)$.
The optimal solution is vertex $C = (0, 2, 0)$.

The simplex algorithm moves, at each iteration, from one feasible basic solution to an adjacent and better solution, stopping when it can no longer find a better solution.

By the way, the triangle ABC is called a two-dimensional simplex (and this isn't a coincidence.)

So, either:

- We move from A to B (i.e., enter x_1 and exit x_3), giving $z = 2$, followed by moving from B to C (i.e., entering x_2 and exiting x_1), giving $z = 4$.
- Or we move directly from A to C (i.e., enter x_2 and exit x_3).

Exercise: Run the simplex algorithm on this problem, and investigate which route to the optimal solution is taken.

2.5. Another Example Of the Simplex Algorithm

Example 2.7.

$$\begin{aligned} \text{Maximise} \quad & z = x_1 + 2x_2 + x_3 + x_4 \\ \text{s.t.} \quad & 2x_1 + x_2 + 3x_3 + x_4 \leq 8 \\ & 2x_1 + 3x_2 + 4x_4 \leq 12 \\ & 3x_1 + x_2 + 2x_3 \leq 18 \\ & x_1, \ldots, x_4 \geq 0 \end{aligned}$$

Introduce the slack variables x_5, x_6 and x_7. These slack variables are the initial basic variables.

Initial Tableau

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	Sol.
x_5	2	1	3	1	1	0	0	8
x_6	2	3	0	4	0	1	0	12
x_7	3	1	2	0	0	0	1	18
z	-1	-2	-1	-1	0	0	0	0

The first iteration of the algorithm:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	Sol.	θ
x_5	2	1	3	1	1	0	0	8	8
x_6	2	[3]	0	4	0	1	0	12	4
x_7	3	1	2	0	0	0	1	18	18
z	-1	-2	-1	-1	0	0	0	0	

$$\begin{aligned} [1] - \frac{1}{3}[2] &\to [1] \\ [3] - \frac{1}{3}[2] &\to [3] \\ \frac{1}{3}[2] &\to [2] \\ [z] + \frac{2}{3}[2] &\to [z] \end{aligned}$$

The second tableau:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	Sol.	θ
x_5	$\frac{4}{3}$	0	[3]	$-\frac{1}{3}$	1	$-\frac{1}{3}$	0	4	$\frac{4}{3}$
x_2	$\frac{2}{3}$	1	0	$\frac{4}{3}$	0	$\frac{1}{3}$	0	4	∞
x_7	$\frac{7}{3}$	0	2	$-\frac{4}{3}$	0	$-\frac{1}{3}$	1	14	7
z	$\frac{1}{3}$	0	-1	$\frac{5}{3}$	0	$\frac{2}{3}$	0	8	

$$\frac{1}{3}[1] \to [1]$$
$$[2] \to [2]$$
$$[3] - \frac{2}{3}[1] \to [3]$$
$$[z] + \frac{1}{3}[1] \to [z]$$

The third tableau:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	Sol.
x_3	$\frac{4}{9}$	0	1	$-\frac{1}{9}$	$\frac{1}{3}$	$-\frac{1}{9}$	0	$\frac{4}{3}$
x_2	$\frac{2}{3}$	1	0	$\frac{4}{3}$	0	$\frac{1}{3}$	0	4
x_7	$1\frac{4}{9}$	0	0	$-\frac{10}{9}$	$-\frac{2}{3}$	$-\frac{1}{9}$	1	$11\frac{1}{3}$
z	$\frac{7}{9}$	0	0	$1\frac{5}{9}$	$\frac{1}{3}$	$\frac{5}{9}$	0	$9\frac{1}{3}$

After the second iteration (giving the third tableau) the stopping criterion has been reached. Thus

$$x_2^* = 4,\ x_3^* = 4/3,\ x_1^* = x_4^* = 0$$

and the slack variables are

$$x_5^* = x_6^* = 0,\ x_7^* = 11\frac{1}{3},$$

and the optimal value is $z^* = 9\frac{1}{3}$. (Check this using $z = x_1 + 2x_2 + x_3 + x_4$.)

2.6. Practical considerations of the simplex algorithm

The choice of entering variable

Different implementations of the simplex algorithm use different methods for choosing the entering variable. The method given in this course, that is "choose the most negative value in the z row", is a good, simple method.

In the situation where two or more entries in the z row have the same largest negative value, just choose one arbitrarily.

Alternative optima

If one (or more) of the non-basic variables has a zero in the z row of the final tableau, there exists an alternative optimal solution. In other words there are two (or more) feasible basic solutions that give the same maximum value of the objective function.

Why is this?

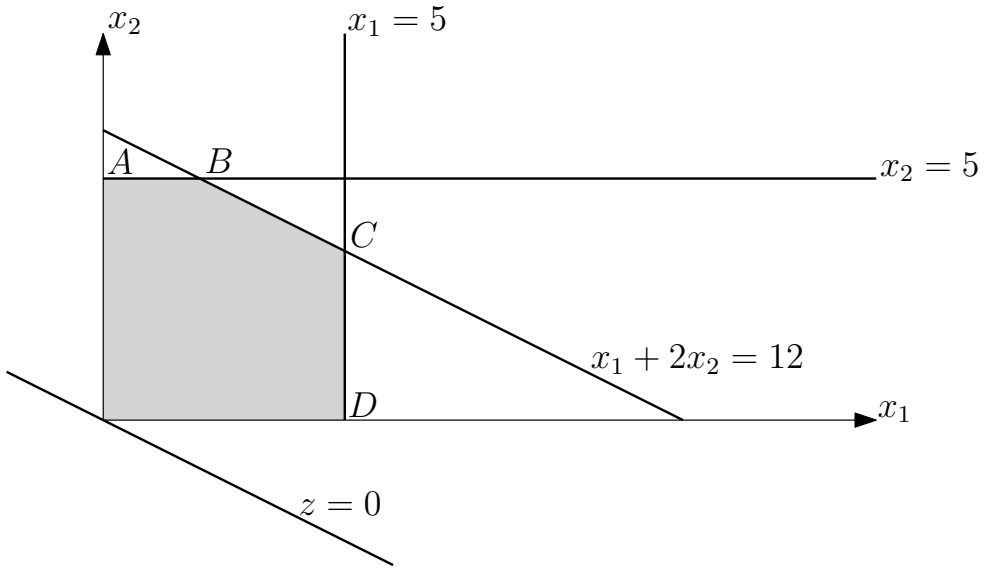
Look at equation (2.7) in Section 2.4. The fact that there is a zero in one of the columns corresponding to a non-basic variable means that one of $d_{k_1}, \ldots, d_{k_n}$ (d_{k_j}, say) equals zero. So, we can enter the non-basic variable x_{k_j} without changing the value of z.

The geometric interpretation of this is that the z-contours are parallel to one (or more) constraints and that that any point on this constraint line also maximises z. So, alternative optimal basic solutions will always be adjacent.

Example 2.8. This is a trivial example to demonstrate the idea. Given the following LPP.

$$\begin{aligned} \text{Maximise} \quad & z = x_1 + 2x_2 \\ \text{s.t.} \quad & x_1 \leq 5 \\ & x_2 \leq 5 \\ & x_1 + 2x_2 \leq 12 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Clearly the objective function is parallel to the third constraint.



The simplex method moves from the origin to point D to point C, where the algorithm stops. Any point on the line BC is optimal, but only B and C are basic solutions.

Exercise: Run the simplex algorithm on this problem and verify that the z row of the final tableau contains a zero in one of the columns corresponding to a non-basic variable.

Unbounded solutions

In the simplex algorithm, if you encounter the situation in which none of the θ-ratios are non-negative, then the feasible region is unbounded. There is no optimal solution, and the simplex algorithm breaks down.

In any practical example this means that the problem has been ill-formulated. It is nonsense to say that an infinite profit (or other objective) is attainable without violating any constraint.

Example 2.9 (Another trivial example)**.**

$$\begin{aligned} \text{Maximise} \quad & z = x_1 + x_2 \\ \text{s.t.} \quad & -2x_1 + x_2 \leq 10 \\ & x_1 - x_2 \leq 5 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Exercise: Check that, in the second tableau for this problem, all of the θ-ratios are negative.

Degenerate problems

If two (or more) rows have the same smallest positive θ-ratio — so that there is a choice of exiting variable — we have a **degenerate problem**. If we are unfortunate in our choice of the exiting variable, then it is possible for the method to cycle, so that after some steps we return to a tableau that was seen before. If that is the case, then the method will never stop, so we will never reach the optimal solution.

What can one do about this?

- Recognise when a choice of exiting variable arises, i.e., when two or more of the θ-ratios are equal.
- If this occurs, continue the simplex algorithm; if the stopping condition is reached, the solution is optimal.
- Be aware that cycling may occur: compare at each step the set of basic variables to the preceding tableaux, and if at any stage you encounter the same set twice, you have a repeated tableau and the simplex method is not going to work.

There are special rules that one can apply to degenerate problems which ensure that cycling will not occur, and good linear programming software can handle these cases. For more details, see Kolman and Beck.

For this course, you only need to know about degeneracy and to be able to identify when it occurs. You will not be expected to find the optimal solution to a degenerate problem where cycling occurs.

2.7. Simplex Algorithm for More General Problems

So far, we have assumed that

- the objective function is to be maximised,
- all constraints are of the form $\leq$,
- all the resource values, b_i, are ≥ 0,
- and all the control variables must be ≥ 0.

We now look at what to do when these are not true.

Minimisation Problems

Minimising

$$z = c_1x_1 + \cdots + c_nx_n$$

is the same as maximising

$$-z = -c_1x_1 - \cdots - c_nx_n.$$

Practically, this simply means that the objective row in the initial tableau reads

$$-z \mid c_1 \; c_2 \; \cdots \; c_n \; 0 \; \cdots \; 0 \mid 0$$

That is, the z and $-c_i$ are replaced with $-z$ and c_i.

Negative Control Variables

So far, we have assumed that all control variables must be non-negative. When a basic-variable (x_1, say) departs the tableau, we set x_1 to zero, because this maximises the increase in z. If x_1

were allowed to be negative, we could increase z even more by putting x_1 equal to some negative number. Therefore, the simplex algorithm as described above will not work.

The solution is straightforward. We replace x_1 by two variables $x_1 = x_1^+$ and x_1^-, and let $x_1 = x_1^+ - x_1^-$ where $x_1^+ \geq 0$ and $x_1^- \geq 0$. This allows the Simplex algorithm to be specified in terms of non-negative control variables while still allowing x_1 to be negative. Although there is no unique solution to $x_1 = x_1^+ - x_1^-$, the simplex algorithm will always reach a final solution in which either $x_1^+ = 0$ or $x_1^- = 0$.

You will not be required to solve problems of this kind in this course.

Artificial Variables

So far, we have assumed that all the constraints have a $\leq$ and all the resource values b_i are non-negative. In this case, we introduce slack variables $x_{n+1}, \ldots, x_{n+m}$ and the initial solution $x_1, \ldots, x_n = 0$ and $x_{n+i} = b_i$ $(i = 1, \ldots, m)$ is feasible.

There are three situations to consider where this is not the case.

1. When the constraint involves $\leq$ and the resource value b_i is negative, simply multiply the constraint by -1. E.g.
$$a_{11}x_1 + \ldots + a_{1n}x_n \leq b_1 < 0$$
becomes
$$-a_{11}x_1 - \ldots - a_{1n}x_n \geq -b_1 > 0$$
But now the constraint is $\geq$.

2. When a constraint involves $\geq$, we introduce a surplus variable. E.g.
$$a_{21}x_1 + \ldots + a_{2n}x_n \geq b_2$$
becomes
$$a_{21}x_1 + \ldots + a_{2n}x_n - x_{n+2} = b_2$$
If $b_2 \leq 0$, we can just set $-x_{n+2} = b_2$. If $b_2 > 0$, we cannot set $-x_{n+2} = b_2$ because then $x_{n+2} < 0$, which is not a feasible solution.

3. When a constraint is an equality, e.g.
$$a_{31}x_1 + \ldots + a_{3n}x_n = b_3$$
we do not introduce a slack or surplus variable and there is no simple way to find an initial basic solution in the original variables.

To solve such LPPs using the simplex algorithm, an **artificial variable** is added to each constraint that has an $=$ sign or is $\geq b_i$ with $b_i > 0$. So, we transform
$$\begin{aligned} a_{21}x_1 + \cdots + a_{2n}x_n \geq b_2 &\rightarrow a_{21}x_1 + \cdots + a_{2n}x_n - x_{n+2} \geq b_2 \\ &\rightarrow a_{21}x_1 + \cdots + a_{2n}x_n - x_{n+2} + y_1 = b_2 \end{aligned}$$
for a '$\geq b_i$' constraint with $b_i > 0$, and
$$a_{31}x_1 + \cdots + a_{3n}x_n = b_3 \;\rightarrow\; a_{31}x_1 + \cdots + a_{3n}x_n + y_2 = b_3$$
for a '$= b_i$' constraint with $b_i > 0$.

Now, in order to find an initial feasible basic solution, we can set $x_1 = \cdots = x_n = x_{n+2} = 0$, $y_1 = b_2$ and $y_2 = b_3$.

The purpose of an artificial variable is simply to allow a feasible initial basic solution; it has no real life meaning, and we **require** that each artificial variable **becomes zero in the final solution**.

There are two adaptations to the simplex algorithm to ensure that artificial variables end up equal to zero: the **big-M method** and the **two-phase method**.

2.8. The big-M Method

Introduce artificial variables, $y_1, \ldots, y_p$, say, using the method described above. Change the objective function to

$$z = c_1x_1 + \cdots + c_nx_n - M(y_1 + \ldots + y_p)$$

where M is a "very large" number. In practice, M is any arbitrary number large enough to ensure that once the artificial variable departs, it will never enter again.

The simplex algorithm is then used to solve the new problem, with one adaptation: the stopping criterion is not met until all the artificial variables are non-basic (i.e., they have all exited the tableau.) This may mean entering a variable whose z row entry is positive.

Example 2.10 (The big-M method)**.** Consider the problem

$$\begin{aligned} \text{Maximise} \quad & z = -x_1 + 5x_2 \\ \text{s.t.} \quad & 3x_1 + 2x_2 \geq 18 \\ & 2x_2 \leq 12 \\ & x_1 \leq 4 \\ & x_1, x_2 \geq 0. \end{aligned}$$

The artificial problem is

$$\begin{aligned} \text{Maximise} \quad & z = -x_1 + 5x_2 - My_1 \\ \text{s.t.} \quad & 3x_1 + 2x_2 - x_3 + y_1 = 18 \\ & 2x_2 + x_4 = 12 \\ & x_1 + x_5 = 4 \\ & x_1, \ldots, x_5, y_1 \geq 0, \end{aligned}$$

where x_4 and x_5 are slack variables, x_3 is a surplus variable, and y_1 is an artificial variable.

The initial feasible basic solution is $x_1 = x_2 = x_3 = 0$, $x_4 = 12$, $x_5 = 4$ and $y_1 = 18$.

Initial tableau

	x_1	x_2	x_3	x_4	x_5	y_1	Sol	θ	
y_1	3	2	-1	0	0	1	18	9	
x_4	0	**2**	0	1	0	0	12	6	$\rightarrow$
x_5	1	0	0	0	1	0	4	∞	
z	1	-5	0	0	0	M	0		
		$\uparrow$							

$\frac{1}{2}[2] \rightarrow [2], [1] - [2] \rightarrow [1], [3] \rightarrow [3], [z] + \frac{5}{2}[2] \rightarrow [z].$

Second tableau

	x_1	x_2	x_3	x_4	x_5	y_1	Sol	θ	
y_1	**3**	0	-1	-1	0	1	6	2	$\rightarrow$
x_2	0	1	0	$\frac{1}{2}$	0	0	6	∞	
x_5	1	0	0	0	1	0	4	4	
z	$+1$	0	0	$\frac{5}{2}$	0	M	30		
	$\uparrow$								

$\frac{1}{3}[1] \rightarrow [1], [2] \rightarrow [2], [3] - \frac{1}{3}[1] \rightarrow [3], [z] - \frac{1}{3}[1] \rightarrow [z].$

Final tableau

	x_1	x_2	x_3	x_4	x_5	y_1	Sol
x_1	1	0	$-\frac{1}{3}$	$-\frac{1}{3}$	0	$\frac{1}{3}$	2
x_2	0	1	0	$\frac{1}{2}$	0	0	6
x_5	0	0	$\frac{1}{3}$	$\frac{1}{3}$	1	$-\frac{1}{3}$	2
z	0	0	$\frac{1}{3}$	$\frac{17}{6}$	0	$M - \frac{1}{3}$	28

The solution is $x_1^* = 2, x_2^* = 6$ with value $z^* = 28$.

Notes:

- If we had not used the big-Ms, then we would have stopped at the second tableau, where all of the z-entries were positive. This wouldn't satisfy the '$\geq$' constraint in the original problem, which requires $y_1 = 0$ to hold. Hence, we continued to the third and final tableau — notice that z decreased when we did this!
- As mentioned, the stopping criterion for the big-M method is slightly different. If all the non-basic variables have positive z row entries, but some of the artificial variables are still basic, then we continue the algorithm, choosing as the entering variable the non-basic variable with the smallest positive z row entry.
- This approach only works for maximisation problems. One can adapt the method to work for minimisation methods, but that is not covered in this course.
- This is a simplified version compared to most books.

2.9. Two-phase Method

This method involves running the simplex algorithm twice: once to eliminate all the artificial variables and so find a feasible basic solution to the original problem; and then again after dropping the artificial variables from the tableau in order to optimise the original problem.

First phase

Define one artificial variable for every constraint that needs one; let us say there are $p \leq m$ such constraints, so we have artificial $y_1, \ldots, y_p$. Define the **auxiliary objective function** as

$$z' = -(y_1 + \cdots + y_p),$$

which we want to maximise.

Suppose that the constraints are ordered such that constraints $1, \ldots, p$ have artificial variables and constraints $p+1, \ldots, m$ do not. Then putting

$$x_i = 0 \text{ for } i = 1, \ldots, n+p$$

$$y_i = b_i \text{ for } i = 1, \ldots, p$$

and

$$x_{n+i} = b_i \text{ for } i = p+1, \ldots, m$$

gives a feasible basic solution to the auxiliary problem.

We solve the auxiliary problem using the Simplex algorithm with the auxiliary objective function. We continue until all the artificial variables have been eliminated. As soon as all the artificial variables have been eliminated, we stop the first phase.

Example 2.11. Consider the same problem we used with the big-M example (with the constraints reordered to fit the convention above):

$$\begin{aligned} \text{Maximise} \quad & z = -x_1 + 5x_2 \\ \text{s.t.} \quad & 3x_1 + 2x_2 \geq 18 \\ & 2x_2 \leq 12 \\ & x_1 \leq 4 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Phase one Add slack and artificial variables to give the auxiliary problem.

$$\begin{aligned} \text{Maximise} \quad & z' = -y_1 \\ \text{s.t.} \quad & 3x_1 + 2x_2 - x_3 + y_1 = 18 \\ & 2x_2 + x_4 = 12 \\ & x_1 + x_5 = 4 \\ & x_1, \ldots, x_5, y_1 \geq 0 \end{aligned}$$

The initial solution is $x_4 = 12$, $x_5 = 4$ and $y_1 = 18$ (and $x_1 = x_2 = x_3 = 0$.) The z' row would normally be

	x_1	x_2	$\ldots$	x_{m+n}	y_1	$\ldots$	y_p	Sol
z'	0	0	$\ldots$	0	1	$\ldots$	1	$-\sum_{i=1}^{p} y_i$

There is a problem here: all the non-basic variables have a zero in the z row. We have to re-write z' as

$$z' = -y_1 = -18 + 3x_1 + 2x_2 - x_3$$

i.e.,

$$z' - 3x_1 - 2x_2 + x_3 = -18.$$

In general, we have

$$z' = -(y_1 + \cdots + y_p)$$

and

$$a_{i,1}x_1 + \ldots a_{i,n+m}x_{n+m} + y_i = b_i$$

giving

$$y_i = b_i - (a_{i,1}x_1 + \ldots a_{i,n+m}x_{n+m}),$$

and hence

$$z' - \sum_{j=1}^{n+m} \left(\sum_{i=1}^{p} a_{i,j} \right) x_j = -\sum_{i=1}^{p} b_i$$

Note: rows $1, \ldots, p$ are those rows which have an artificial variable.

The other requirement is that we "carry through" the original z row, to ensure it is in the correct form at the start of the the second phase.

Phase one: initial tableau (example)

	x_1	x_2	x_3	x_4	x_5	y_1	Sol	θ	
y_1	3	2	−1	0	0	1	18	6	
x_4	0	2	0	1	0	0	12	∞	
x_5	**1**	0	0	0	1	0	4	4	→
z	1	−5	0	0	0	0	0		
z'	−3	−2	1	0	0	0	−18		
	↑								

$$[1] - 3[3] \to [1], [2] \to [2], [3] \to [3], [z] - [3] \to [z], [z'] + 3[3] \to [z'].$$

Second tableau

	x_1	x_2	x_3	x_4	x_5	y_1	Sol	θ	
y_1	0	**2**	−1	0	−3	1	6	3	→
x_4	0	2	0	1	0	0	12	6	
x_1	1	0	0	0	1	0	4	∞	
z	0	−5	0	0	−1	0	−4		
z'	0	−2	1	0	3	0	−6		
		↑							

$$[3] \to [3], [2] - [1] \to [2], \tfrac{1}{2}[1] \to [1], [z] + \tfrac{5}{2}[1] \to [z], [z'] + [1] \to [z'].$$

Final tableau

	x_1	x_2	x_3	x_4	x_5	y_1	Sol
x_2	0	1	$-\frac{1}{2}$	0	$-\frac{3}{2}$	$\frac{1}{2}$	3
x_4	0	0	1	1	3	−1	6
x_1	1	0	0	0	1	0	4
z	0	0	$-\frac{5}{2}$	0	$-\frac{17}{2}$	$\frac{5}{2}$	11
z'	0	0	0	0	0	1	0

Now the artificial variable has exited, and so is zero. The solution for z' is zero. (This must be the case. Why?)

Our feasible solution for the original problem is $x_1 = 4, x_2 = 3, x_4 = 6$ and $x_3 = x_5 = 0$, with value $z = 11$.

Phase two Use the final tableau of phase one as the starting point for phase two. Simply drop the z' row and the y columns.

Phase two (example)

	x_1	x_2	x_3	x_4	x_5	Sol	θ	
x_2	0	1	$-\frac{1}{2}$	0	$-\frac{3}{2}$	3	$-\infty$	
x_4	0	0	1	1	**3**	6	2	$\rightarrow$
x_1	1	0	0	0	1	4	4	
z	0	0	$-\frac{5}{2}$	0	$-\frac{17}{2}$	11		
					$\uparrow$			

$[1] + \frac{1}{2}[2] \rightarrow [1], \frac{1}{3}[2] \rightarrow [2], [3] - \frac{1}{3}[2] \rightarrow [3], [z] + \frac{17}{6}[2] \rightarrow [z].$

	x_1	x_2	x_3	x_4	x_5	Sol
x_2	0	1	0	$\frac{1}{2}$	0	6
x_5	0	0	$\frac{1}{3}$	$\frac{1}{3}$	1	2
x_1	1	0	$-\frac{1}{3}$	$-\frac{1}{3}$	0	2
z	0	0	$\frac{1}{3}$	$\frac{17}{6}$	0	28

So, our final solution is $x_1^* = 2, x_2^* = 6, x_3^* = x_4^* = 0, x_5^* = 2$, with value $z^* = 28$.

Comments The big-M method is often easier for hand calculations than the two-phase method. However, for numerical implementation on a computer, the big-M method is **numerically unstable**, giving (very) inaccurate results. So, most computer algorithms use the two-phase method.

Summary of two-phase method

1. If all constraints are $\leq b_i$ with $b_i \geq 0$ or $\geq b_i$ with $b_i \leq 0$ then use the standard simplex method.
2. Otherwise, introduce artificial variables.
3. Obtain the auxiliary objective function, in terms of x_i.
4. Phase 1: Solve the auxiliary problem using the simplex algorithm on z' and carrying through z.
5. Phase 2: Drop the artificial variables and solve the original problem.

3. Sensitivity analysis and duality

3.1. Sensitivity analysis

Recall that sensitivity analysis means assessing how sensitive the optimal solution is to the specification of the linear program. In Chapter 1, we looked at:

- how much the optimal value changes for a unit change in a resource value: the shadow price of the resource.
- how much a resource value can change before the optimal solution moves to a different extreme point (equivalently, the set of basic variables changes).
- how much the objective function can change before the optimal solution moves to a different extreme point.

We looked at sensitivity analysis using the graphical solution method. Now, we consider an analysis using the simplex algorithm.

Change to a Resource Value

Example 3.1. Consider the following linear programming problem,

$$\begin{aligned} \text{Maximise } z &= 4x_1 + 2x_2 + x_3 \\ \text{subject to } x_1 + x_2 &\leq 4 \\ x_1 + x_3 &\leq 6 \\ x_2 + x_3 &\leq 8 \\ x_1, x_2, x_3 &\geq 0. \end{aligned}$$

In canonical form this is

$$\begin{aligned} \text{Maximise } z &= 4x_1 + 2x_2 + x_3 \\ \text{subject to } x_1 + x_2 + x_4 &= 4 \\ x_1 + x_3 + x_5 &= 6 \\ x_2 + x_3 + x_6 &= 8 \\ x_1, \ldots, x_6 &\geq 0. \end{aligned}$$

This gives the following simplex algorithm tableaux.

Initial tableau

	x_1	x_2	x_3	x_4	x_5	x_6	Sol.	θ	
x_4	**1**	1	0	1	0	0	4	4	$\rightarrow$
x_5	1	0	1	0	1	0	6	6	
x_6	0	1	1	0	0	1	8	∞	
z	-4	-2	-1	0	0	0	0		
	$\uparrow$								

$[1] \rightarrow [1], [2] - [1] \rightarrow [2], [3] \rightarrow [3], [z] + 4[1] \rightarrow [z].$

Second tableau

	x_1	x_2	x_3	x_4	x_5	x_6	Sol.	θ	
x_1	1	1	0	1	0	0	4	∞	
x_5	0	-1	**1**	-1	1	0	2	2	$\rightarrow$
x_6	0	1	1	0	0	1	8	8	
z	0	2	-1	4	0	0	16		
			$\uparrow$						

$[1] \rightarrow [1], [2] \rightarrow [2], [3] - [2] \rightarrow [3], [z] + [2] \rightarrow [z]$.

Final tableau

	x_1	x_2	x_3	x_4	x_5	x_6	Sol.
x_1	1	1	0	1	0	0	4
x_3	0	-1	1	-1	1	0	2
x_6	0	2	0	1	-1	1	6
z	0	1	0	3	1	0	18

So the optimal solution is $x_1^* = 4, x_2^* = 0, x_3^* = 2, x_4^* = x_5^* = 0, x_6^* = 6$, with value $z^* = 18$.

We shall see later that the entries in the z row corresponding to the slack variables are the shadow prices! The shadow prices for the first, second and third constraints are 3, 1 and 0, respectively.

Definition 3.2. If the slack variable corresponding to a particular constraint is zero then it is a *binding constraint*, otherwise it is a *non-binding constraint*.

Equivalently, a constraint is binding if the optimal solution exactly satisfies that constraint (with equality).

Theorem 3.3 (Complementary slackness)**.**

- *Non-binding constraints have zero shadow prices.*
- *Equivalently: constraints with non-zero shadow prices are binding.*

One might ask whether the converse of this theorem is true: do binding constraints always have non-zero shadow prices? No. This might seem odd: if a constraint is binding, we would think that we could obtain a higher return if we had a larger resource value for that constraint, since we are currently bound by it. However, this isn't guaranteed.

In Example 3.1, a greater profit can be obtained by increasing the first two resource values, b_1 and b_2, but not the third resource value, b_3.

How much can we change a resource value?

Suppose the value of b_1, with shadow price 3, changes from 4 to $4 + \Delta$. Then, z increases by 3Δ units. Typically, this will not be true for unlimited values of Δ: there will be a point at which this constraint becomes non-binding because the other resource values prevent further increases in z. Increasing b_1 beyond this point will make no further difference to z. At this point, the set of basic variables will change (the slack variable for the first constraint will become basic). An important question: is for what range of values of b_1 does the constraint remain binding?

We could re-run the whole simplex algorithm again with the value of b_1 replaced by its new value. However, there is a quicker way.

Were we to repeat all the elementary row operations performed in the simplex algorithm, we would find that only the solution column would change. So, instead of repeating all the

elementary row operations, just repeat them for the solution column. If all the entries in the solution column remain non-negative, the solution we obtain is still feasible (since $x_i \geq 0\ \forall i$) and still optimal (since all the entries in the z row are still ≥ 0 — they have not changed).

In the above example, we obtain:

	Solution column		
	Initial tableau	2nd tableau	Final tableau
Row 1	$4+\Delta$	$4+\Delta$	$4+\Delta$
Row 2	6	$2-\Delta$	$2-\Delta$
Row 3	8	8	$6+\Delta$
Row z	0	$16+4\Delta$	$18+3\Delta$

So, for the final basic solution, we have $x_1 = 4+\Delta$, $x_3 = 2-\Delta$ and $x_6 = 6+\Delta$. This is feasible if

$$4+\Delta \geq 0, \quad 2-\Delta \geq 0, \quad 6+\Delta \geq 0,$$

or equivalently if

$$\Delta \geq -4, \quad \Delta \leq 2, \quad \Delta \geq -6.$$

So, the solution is feasible if $-4 \leq \Delta \leq 2$, that is, if $0 \leq b_1 \leq 6$.

From our new solution column, we also see that the new optimal value is $z^* = 18+3\Delta$, which confirms the shadow price.

Notes

1. In fact, it is not necessary to recalculate the solution column for each iteration of the Simplex algorithm. The new final solution column is just the old final solution column plus Δ times the column corresponding to the slack variable for the constraint whose resource value has been changed (in this case, the x_4 column).

   This also confirms that the entries in the z row corresponding to the slack variables are indeed the shadow prices.

2. In the example, if the value of Δ is outside the range $-4 \leq \Delta \leq 2$, then the optimal solution will have a different set of basic variables and the whole simplex algorithm will have to be re-run using the new value of b_1.

3. We have assumed that only one resource value is changed. There are methods which allow simultaneous changes to more than one resource value, but these are not covered in this course.

4. Good linear programming software will usually report, for each constraint i, the range of values of b_i for which the optimal solution still has the same set of basic variables.

Exercise: If b_2 changes from 6 to 8, what is the new optimal solution?

Changes to the objective function

We consider changes to basic and non-basic variables of the optimal solution separately.

Basic variables Suppose we change the coefficient c_3 of x_3 from $c_3 = 1$ to $c_3 = 1 + \delta$, with $\delta \in \mathbb{R}$. That is,

$$z = 4x_1 + 2x_2 + (1 + \delta)x_3$$

The old optimal solution we obtained will always still be feasible, since the constraints are unaltered. However, the value of z will change and the solution may no longer be optimal. In the final tableau the z row will change in the following way:

Take the row corresponding to the changed variable (x_3), multiply by δ and, **for the columns corresponding to the non-basic variables and the solution column only**, add this to the z row.

	x_1	x_2	x_3	x_4	x_5	x_6	Sol.
x_1	$\cdots$						
x_3	0	-1	1	-1	1	0	2
δx_3	0	$-\delta$	δ	$-\delta$	δ	0	2δ
x_6	$\cdots$						
z	0	1	0	3	1	0	18
z_{new}	0	$1-\delta$	0	$3-\delta$	$1+\delta$	0	$18+2\delta$

This solution will still be optimal provided that $1 - \delta \geq 0$, $3 - \delta \geq 0$ and $1 + \delta \geq 0$.

Simplifying this gives $-1 \leq \delta \leq 1$, and so the coefficient for x_3 (i.e., $c_3 = 1 + \delta$) must be within the range $0 \leq c_3 \leq 2$, and the optimal value will be

$$z^* = 18 + 2\delta.$$

Non-basic variables Suppose that the objective function has a change in the coefficient of a non-basic variable (e.g., x_2) by an amount δ. That is, z becomes

$$z = 4x_1 + (2 + \delta)x_2 + x_3$$

The only change to the final tableau is that the z-row entry of the variable whose coefficient has been changed (x_2) changes by an amount $-\delta$. Thus the z row becomes

$$z \mid 0 \quad 1-\delta \quad 0 \quad 3 \quad 1 \quad 0 \mid 18$$

The solution will still be optimal, and the value unchanged, provided $1 - \delta \geq 0$, i.e., provided $\delta \leq 1$. In other words, we require $c_2 \leq 3$.

Again, the above methods only work for a change to one c_i.

3.2. Duality

Each LPP can be formulated in two different ways: the *primal problem* and the *dual problem.*

We have been considering LPPs of the form

$$\begin{aligned}
\text{Maximise:} \quad & z = c_1x_1 + \cdots + c_nx_n \\
\text{subject to:} \quad & a_{11}x_1 + \cdots + a_{1n}x_n \leq b_1 \\
& \qquad\qquad\qquad\quad \vdots \quad \vdots \\
& a_{m1}x_1 + \cdots + a_{mn}x_n \leq b_m \\
& \qquad\qquad x_1, \ldots, x_n \geq 0.
\end{aligned}$$

This is primal problem. The **dual problem** is the LPP:

$$\begin{array}{rrcl} \text{Minimise:} & g = b_1y_1 + \cdots + b_my_m & & \\ \text{subject to:} & a_{11}y_1 + \cdots + a_{m1}y_m & \geq & c_1 \\ & \vdots & \vdots & \\ & a_{1n}y_1 + \cdots + a_{mn}y_m & \geq & c_n \\ & y_1, \ldots, y_m & \geq & 0. \end{array}$$

Notice that the dual of the dual problem is the primal problem.

Example 3.4. The dual for the LPP in Example 3.1 is

$$\begin{aligned} \text{Minimise } g &= 4y_1 + 6y_2 + 8y_3 \\ \text{subject to } y_1 + y_2 &\geq 4 \\ y_1 + y_3 &\geq 2 \\ y_2 + y_3 &\geq 1 \\ y_1, y_2, y_3 &\geq 0 \end{aligned}$$

Exercise: You may want to check that the optimal solution to this dual problem is

$$y_1^* = 3, \quad y_2^* = 1, \quad y_3^* = 0, \quad g^* = 18$$

Notice that $g^* = z^*$ and y_1^*, y_2^* and y_3^* equal the shadow prices for the primal problem.

In matrix form, the primal problem is

$$\begin{array}{rrcl} \text{Maximise:} & z = \mathbf{c}^T\mathbf{x} & & \\ \text{subject to:} & \mathbf{Ax} & \leq & \mathbf{b} \\ & \mathbf{x} & \geq & \mathbf{0} \end{array}$$

and the corresponding dual problem is

$$\begin{array}{rrcl} \text{Minimise:} & g = \mathbf{b}^T\mathbf{y} & & \\ \text{subject to:} & \mathbf{A}^T\mathbf{y} & \geq & \mathbf{c} \\ & \mathbf{y} & \geq & \mathbf{0} \end{array}$$

Theorem 3.5. *The optimal value of the objective function in the primal problem is the same as the optimal value of the objective function in the dual problem. That is*

$$z^* = g^*.$$

Consider the dual problem in Example 3.4. Suppose b_1 is changed to $b_1' = b_1 + 1 = 5$. Assuming that the solution $\mathbf{y}^*$ remains optimal, the new value of g^* is

$$g'^* = 5y_1^* + 6y_2^* + 8y_3^* = g^* + y_1^*.$$

By increasing b_1 by one unit the new optimal value of g has increased by an amount y_1^*. Theorem 3.5 tells us that the optimal value of z^* will have also increased by y_1^*. Hence y_1^* is the shadow price for the first constraint of the primal problem. This gives us:

Theorem 3.6. *The optimal values of the control variables in the dual problem are the shadow prices of the primal problem and vice versa.*

Interpretation of the dual problem

Let us return to the alarm clock problem.

Mr Primal is running a factory. He has 1600 units of labour, 1800 units of processing and 350 alarm assemblies at his disposal. He plans to produce x_1 standard clocks and x_2 alarm clocks. For each standard clock he will get £3 and for each alarm clock £8.

An accident occurs and one unit of one of the resources (labour, processing or alarm assemblies) is damaged and can no longer be used. Fortunately, Mr Primal is insured for such losses by Ms Dual.

Mr Primal and Ms Dual have agreed beforehand how much one unit of each resource is worth, and so how much Ms Dual will compensate Mr Primal to ensure that he does not suffer a financial loss. Let y_i denote the value of one unit of the ith resource ($i = 1, 2, 3$) that they agreed.

Mr Primal explains to Ms Dual that each standard clock uses 2 labour units and 6 processing units and makes £3 profit. Each alarm clock uses 4 labour units, 2 processing units and 1 alarm assembly and makes £8. Thus, for the compensation to be fair, he insists that y_1, y_2 and y_3 must satisfy the constraints

$$\begin{aligned} 2y_1 + 6y_2 &\geq 3 \\ 4y_1 + 2y_2 + y_3 &\geq 8 \end{aligned} \tag{3.1}$$

since, for example, with 2 units of labour and 6 of processing, he can make £3 of profit.

Ms Dual agrees this is fair. However, she want to minimise her liabilities. If the factory burns down and all Mr. Primal's resources are lost, she will have to pay out

$$g = 1600y_1 + 1800y_2 + 350y_3$$

pounds.

So, Ms Dual wants to solve the following LPP:
minimise g subject to the constraints (3.1).
This is the dual of Mr Primal's original optimisation problem.

4. Game theory

4.1. Introduction

In Chapters 1–3, we looked at situations in which **one** individual chooses what to do in order to optimise some quantity (the objective function). In this chapter, we look at strategic games (or games). In a game, **two or more** individuals (the **players**) each try to maximise a different quantity (the **payoff** for that player) which depends not only on what he/she does, but also on what the other player(s) do.

Game Theory is applied in many fields:

- games of fun (Monopoly, card games, noughts and crosses)
- business and economics
- sociology
- politics

It is often necessary to idealise (simplify) the real problem, in order to make it susceptible to analysis.

Games may be classified as:

- simultaneous-move or sequential-move
- zero-sum or non-zero sum (variable-sum)
- two-player or n-player ($n > 2$)

In this chapter, we shall look at two-player, zero-sum, simultaneous-move games. In Section 4.9 we shall briefly look at 'games against nature'.

An important assumption of game theory is that each player seeks to maximise their payoff and choose the best strategy to achieve this. (Economists call this 'rationality'.) Under this assumption, we can try to understand each player's best (optimal) strategy.

4.2. Basic formulation

We have two players (A and B). Each has a number of possible moves and must choose one of these. We are dealing with simultaneous-move games, so players must choose their move before knowing which move the other has chosen. The outcome of the game is a payoff for Player A and a payoff for Player B. These depend on Player A and B's choices of move. As we are dealing with zero-sum games, the payoff to Player B is minus the payoff to Player A. So, we require only one **payoff matrix**:

		Player B			
		B_1	B_2	$\cdots$	B_m
Player A	A_1	a_{11}	a_{12}	$\cdots$	a_{1m}
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	A_n	a_{n1}	a_{n2}	$\cdots$	a_{nm}

If Player A chooses move A_i and Player B chooses move B_j, the payoff to Player A is a_{ij} and

to Player B is $-a_{ij}$. The convention is that the numbers in the payoff matrix are the payoffs for the player to the left of the matrix (Player A in the matrix above).

Example 4.1.

		Player B	
		B_1	B_2
Player A	A_1	0	-4
	A_2	3	2

Definitions 4.2. A *strategy* is a rule for determining which move to play.

A *pure strategy* specifies that the same move always be chosen.

A *mixed strategy* specifies that the move be chosen at random, with each move having some specific probability of being chosen. E.g., Player B chooses B_1 with probability y_1, B_2 with probability y_2, etc. (and $y_1 + y_2 + \cdots + y_m = 1$).

(Notice that a pure strategy is a mixed strategy in which one of the probabilities equals one and the rest are zero.)

Suppose the payoff matrix is

		Player B		
		B_1	$\ldots$	B_m
	A_1	a_{11}	$\ldots$	a_{1m}
Player A	$\vdots$	$\vdots$		$\vdots$
	A_n	a_{n1}	$\ldots$	a_{nm}

Let x_i $(i = 1, \ldots, n)$ be the probability that Player A chooses move A_i and y_j be the probability that Player B chooses move B_j, and let $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_m)$. Then the **expected payoff** for Player A is

$$E(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \sum_{j=1}^{m} x_i a_{ij} y_j.$$

Note the non-standard notation for expectation.

4.3. Nash equilibrium

We seek the optimal strategy for each player. These are defined in terms of the **Nash equilibrium**.

The Nash equilibrium of a game is a pair of strategies, one for each player, such that each player's strategy is best for them, given that the other player is playing their own equilibrium strategy.

Another way of looking at this is that neither player should want to change his strategy if he knew what strategy the other player was using.

In fact, we shall often pretend that the other player knows our strategy in order to work out what our optimal strategy is.

Example 4.3. Consider this payoff matrix:

		Player B	
		B_1	B_2
Player A	A_1	0	-4
	A_2	3	2

The Nash equilibrium is: Player A always chooses A_2 and Player B always chooses B_2. You can check this: if B plays B_2, is it better for A to play A_1 or A_2? What if A mixes strategies? Likewise, if A plays A_2, what is best for B?

The equilibrium consists of pure strategies: $\mathbf{x} = (0, 1)$, $\mathbf{y} = (0, 1)$. The payoff will be 2 (to Player A).

Example 4.4. Consider this payoff matrix:

		Player B	
		B_1	B_2
Player A	A_1	4	3
	A_2	2	8

Do $\mathbf{x} = (\frac{1}{2}, \frac{1}{2})$ and $\mathbf{y} = (\frac{1}{4}, \frac{3}{4})$ constitute a Nash equilibrium?

The expected payoff with the proposed actions is: $E(\mathbf{x}, \mathbf{y}) = 4 \cdot \frac{1}{2} \cdot \frac{1}{4} + 3 \cdot \frac{1}{2} \cdot \frac{3}{4} + 2 \cdot \frac{1}{2} \cdot \frac{1}{4} + 8 \cdot \frac{1}{2} \cdot \frac{3}{4} = 4\frac{7}{8}$. Suppose that B plays $\mathbf{y}$, and A knows that. If A plays A_2, i.e., uses strategy $\mathbf{x}' = (0, 1)$, then the expected payoff to A is $E(\mathbf{x}', \mathbf{y}) = 2 \cdot \frac{1}{4} + 8 \cdot \frac{3}{4} = 6\frac{1}{2} > 4\frac{7}{8}$. The fact that this is greater than the payoff calculated above tells us that $(\mathbf{x}, \mathbf{y})$ is not an equilibrium.

In fact, the equilibrium is $\mathbf{x}^* = \left(\frac{6}{7}, \frac{1}{7}\right)$, $\mathbf{y}^* = \left(\frac{5}{7}, \frac{2}{7}\right)$, with payoff $3\frac{5}{7}$.

Definition 4.5. If a Nash equilibrium $\mathbf{x}^*, \mathbf{y}^*$ exists, then $\mathbf{x}^*$ and $\mathbf{y}^*$ are the optimal strategies, and

$$v = E(\mathbf{x}^*, \mathbf{y}^*)$$

is called the **value** of the game.

Corollary 4.6.

$$\begin{aligned} &\textit{For all } \mathbf{y}, \quad E(\mathbf{x}^*, \mathbf{y}) \geq v. \\ &\textit{For all } \mathbf{x}, \quad E(\mathbf{x}, \mathbf{y}^*) \leq v. \end{aligned}$$

Theorem 4.7. *Every two-player, simultaneous-move, zero-sum game has a Nash equilibrium and hence an optimal strategy for each player and a value.*

The optimal strategies may be pure or mixed.

If a pure strategy is optimal for one player then a pure strategy is also optimal for the other.

How can we **solve** a game? 'Solving' means finding optimal strategies $\mathbf{x}^*$, $\mathbf{y}^*$, and hence the value v.

We will cover the following solution methods (in order of increasing complexity):

- Is there a single dominant strategy?
- Look for saddle point (a pure Nash equilibrium)
- Formulate in terms of maximin/minimax strategies
- Graphical methods
- Formulate as a linear programming problem

The first two methods will find optimal strategies if they are pure; the others are needed when mixed strategies are involved.

4.4. Dominance

Definition 4.8. Let $\mathbf{x}_p$ and $\mathbf{x}'_p$ be pure strategies. Then $\mathbf{x}_p$ *dominates* $\mathbf{x}'_p$ if

$$E(\mathbf{x}_p, \mathbf{y}_q) \geq E(\mathbf{x}'_p, \mathbf{y}_q) \quad \text{for all pure strategies } \mathbf{y}_q$$

Also, $\mathbf{y}_q$ dominates $\mathbf{y}'_q$ if

$$E(\mathbf{x}_p, \mathbf{y}_q) \leq E(\mathbf{x}_p, \mathbf{y}'_q) \quad \text{for all pure strategies } \mathbf{x}_p$$

Dominated strategies are sub-optimal and can be eliminated (for the purpose of solving the game).

Example 4.9.

		Player B				
		B_1	B_2	B_3	B_4	B_5
Player A	A_1	-1	-2	1	2	0
	A_2	-2	1	0	-3	-1
	A_3	-3	-2	0	0	0

We can drop the dominated strategies from the game. The first row dominates the third row. The first column dominates B_3 and B_5. The sub-payoff matrix is now

		Player B		
		B_1	B_2	B_4
Player A	A_1	-1	-2	2
	A_2	-2	1	-3

If, for Player A, pure strategy $\mathbf{x}_p$ dominates all other pure strategies, $\mathbf{x}_p$ is Player A's optimal strategy. Player B's optimal strategy is now the pure strategy that minimises A's payoff when A uses $\mathbf{x}_p$.

Example 4.10.

		Player B		
		B_1	B_2	B_3
Player	A_1	-2	5	-2
A	A_2	0	7	1
	A_3	-3	6	1

Pure strategy A_2 (i.e. $\mathbf{x} = (0, 1, 0)$) dominates the other two pure strategies, A_1 and A_3 (i.e. $\mathbf{x} = (1, 0, 0)$ and $\mathbf{x} = (0, 0, 1)$. Hence, $\mathbf{x}^* = (0, 1, 0)$.

What is B's optimal strategy? B should play B_1; that is, $\mathbf{y}^* = (1, 0, 0)$.

4.5. Saddle Points

Suppose that neither Player A nor Player B has a pure strategy that dominates all his/her other pure strategies. Then we look for **saddle points**.

Definition 4.11. The **pure maximin strategy** for Player A is the strategy A_{i^*} with maximum row-minimum. That is,

$$i^* = \text{argmax}_{i=1,\ldots n} \min_{j=1,\ldots m} a_{ij} = \text{argmax}_{i=1,\ldots n}\big\{\min(a_{i1}, \ldots, a_{im})\big\}$$

The **pure minimax strategy** for Player B is the strategy B_{j^*} with minimum column-maximum. That is,

$$j^* = \text{argmin}_{j=1,\ldots,m} \max_{i=1,\ldots,n} a_{ij} = \text{argmin}_{j=1,\ldots m}\big\{\max(a_{1j}, \ldots, a_{nj})\big\}$$

If these two payoffs are equal, then the game has a **saddle point**.

Theorem 4.12 (Saddle point theorem)**.** *If a two-player, zero-sum, simultaneous-move game has a saddle point, the pure maximin and minimax strategies constitute a Nash equilibrium and are optimal strategies. The value of the game, v, equals the maximin and minimax payoffs.*

Example 4.13. Two companies have rival products of **b**eer, **w**ine and **s**pirits. Each month they choose which product to advertise. The following table shows the gain in A's profits (in £thousands per month) given the combination of advertising actions. We shall assume that B's profits drop by the same amount.

		Company B		
		B_b	B_w	B_s
Company A	A_b	7	8	5
	A_w	-2	12	0
	A_s	9	4	-4

There is a saddle point at (A_b, B_s): A should advertise beer and B should advertise spirits, for a payoff of 5 to A.

Exercise: check that these strategies do form a Nash equilibrium.

Notes

- A game may have multiple saddle points.
- If there is a pure strategy that dominates all other pure strategies, it will also be found by looking for a saddle point.
- Before looking for saddle points, you may want to eliminate dominated strategies (but this is not necessary.)

4.6. Maximin and minimax mixed strategies

If there is no saddle point, the optimal solution is mixed.

When searching for a saddle point, we considered only pure maximin/minimax strategies. We now consider mixed maximin/minimax strategies.

In order for Player A to determine their maximin strategy, they must consider what strategy Player B would adopt if they knew $\mathbf{x} = (x_1, \ldots, x_n)$. Player B would choose the move B_j for which the expected payoff (to A)

$$\sum_{i=1}^{n} x_i a_{i,j}$$

is minimised. So, the expected payoff would be

$$\min\left(\sum_{i=1}^{n} x_i a_{i,1}, \ldots, \sum_{i=1}^{n} x_i a_{i,m}\right)$$

Player A's maximin strategy is the strategy $\mathbf{x}$ that maximises this:

$$\mathbf{x}^* = \underset{(x_1,\ldots x_n)}{\operatorname{argmax}} \left\{ \min\left(\sum_{i=1}^{n} x_i a_{i,1}, \ldots, \sum_{i=1}^{n} x_i a_{i,m}\right)\right\}$$

Similarly, the minimax strategy for B is

$$\mathbf{y}^* = \underset{(y_1,\ldots,y_m)}{\operatorname{argmin}} \left\{ \max\left(\sum_{j=1}^{m} y_j a_{1,j}, \ldots, \sum_{j=1}^{m} y_j a_{n,j}\right)\right\}.$$

Theorem 4.14. *The maximin and minimax strategies constitute a Nash equilibrium, and so are optimal strategies.*

4.7. Graphical solution for an $n \times 2$ game

Assume we have eliminated all dominated strategies and looked for a saddle point. No saddle point has been found, so we consider mixed strategies.

Example 4.15.

		Player B	
		B_1	B_2
Player A	A_1	2	-1
	A_2	-3	4
	A_3	0	2

(Check that there are no dominated strategies and no saddle point.)

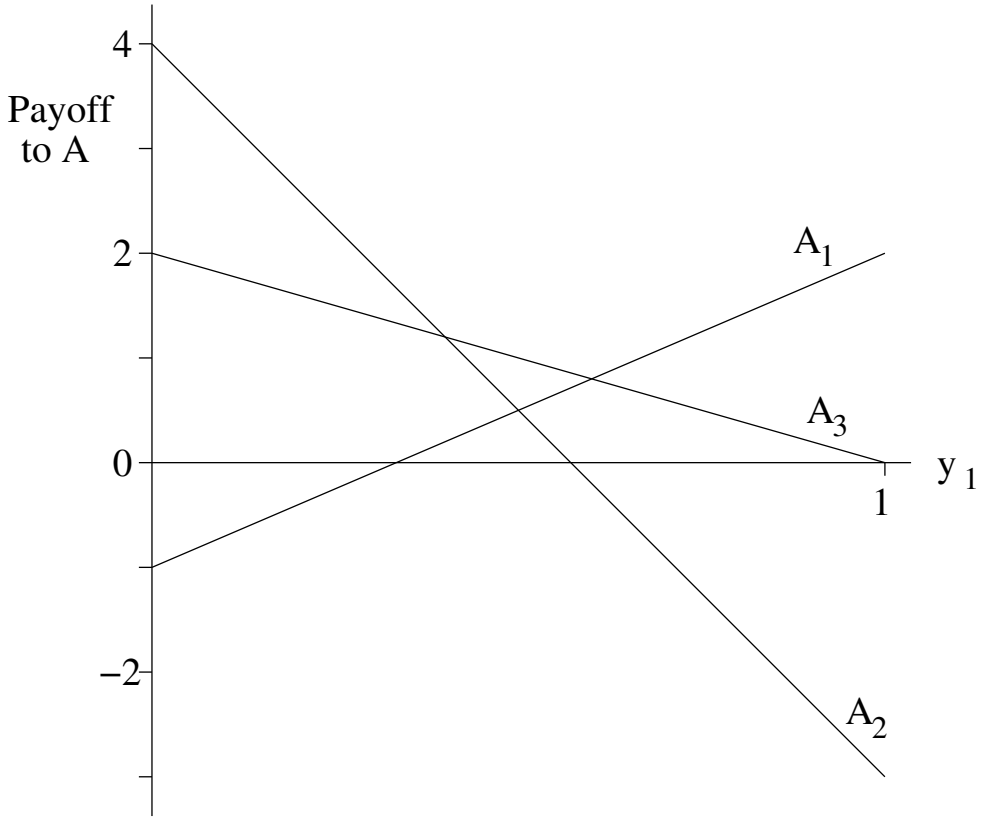
Start by working out B's minimax strategy. Given strategy $\mathbf{y} = (y_1, 1-y_1)$ for B, the expected payoff to A for each of their pure strategies is

$$\begin{aligned} A_1: &\quad 2y_1 - 1(1-y_1) = 3y_1 - 1 \\ A_2: &\quad -3y_1 + 4(1-y_1) = 4 - 7y_1 \\ A_3: &\quad 0y_1 + 2(1-y_1) = 2 - 2y_1. \end{aligned}$$

Player B's minimax strategy is

$$\underset{0\le y_1\le 1}{\operatorname{argmin}} \left\{\max(3y_1 - 1, 4 - 7y_1, 2 - 2y_1)\right\}.$$

We can plot the three components in the above formula as lines with y_1 $(0 \le y_1 \le 1)$ on the horizontal axis.



y_1^*, the optimal choice of y_1, is the place where the maximum of the lines in the plot has its smallest value. We can find it as the intersection of the lines A_1 and A_3.

$$\begin{aligned} 3y_1^* - 1 &= 2 - 2y_1^* \\ y_1^* &= \frac{3}{5} \\ y_2^* &= \frac{2}{5} \end{aligned}$$

So, B's optimal strategy is $\mathbf{y}^* = \left(\frac{3}{5}, \frac{2}{5}\right)$. Now, we need to find A's optimal solution.

Theorem 4.16. *In an $n \times 2$ game, the optimal (maximin) strategy for A requires only two strategies: those that intersect at B's minimax solution.*

If more than two strategies intersect at this point, choose any two that have opposite gradients.

In this example, A_1 and A_3 intersect at the minimax solution. This implies that A_2 is not used in the optimal solution and so $x_2 = 0$.

We need to find the values of x_1 and x_3 that maximise

$$\min\big(2x_1, -x_1 + 2x_3\big),$$

with $x_3 = 1 - x_1$. In other words, we want x_1 that maximises

$$\min(2x_1, 2 - 3x_1).$$

This minimum occurs where the two lines intersect (draw the graph of $2x_1$ and $2-3x_1$ against x_1 to check this). Hence we have

$$\begin{aligned} 2x_1^* &= 2 - 3x_1^* \\ \Rightarrow \quad x_1^* &= \frac{2}{5} \end{aligned}$$

The optimal solution for this game is

$$\begin{aligned} \mathbf{x}^* &= \left(\frac{2}{5}, 0, \frac{3}{5}\right) \\ \mathbf{y}^* &= \left(\frac{3}{5}, \frac{2}{5}\right) \\ v &= 2 \cdot \frac{2}{5} \cdot \frac{3}{5} - 1 \cdot \frac{2}{5} \cdot \frac{2}{5} + 2 \cdot \frac{3}{5} \cdot \frac{2}{5} \\ &= \frac{4}{5}. \end{aligned}$$

To solve a $2 \times m$ game an almost identical approach is used. The maximin strategy for A is found graphically by taking the maximum point of the lowest of the n lines. Then the minimax strategy for B can be found. Always start with the player who has only two possible moves.

4.8. Linear programming formulation of a game

The following method can always be used, even if $n > 2$ and $m > 2$.

Suppose Player B plays the mixed strategy $\mathbf{y}$ and Player A knows what $\mathbf{y}$ is. Then Player A will choose their move A_i to maximise the payoff, i.e., to obtain

$$u = \max_{i=1,\ldots n} \left\{ \sum_{j=1}^{m} a_{i,j} y_j \right\}.$$

Player B's minimax strategy is that which minimises u. From the definition of u we have

$$\sum_{j=1}^{m} a_{1,j} y_j \leq u, \ \sum_{j=1}^{m} a_{2,j} y_j \leq u, \ \ldots, \ \sum_{j=1}^{m} a_{n,j} y_j \leq u \tag{4.1}$$

and

$$y_1 + \cdots + y_m = 1 \tag{4.2}$$

Taking (4.1) and (4.2) and dividing through by u (which for the moment we will assume is positive), we get

$$\begin{aligned} a_{1,1}\frac{y_1}{u} + \cdots + a_{1,m}\frac{y_m}{u} &\leq 1 \\ &\vdots \\ a_{n,1}\frac{y_1}{u} + \cdots + a_{n,m}\frac{y_m}{u} &\leq 1 \\ \frac{y_1}{u} + \cdots + \frac{y_m}{u} &= \frac{1}{u} \end{aligned}$$

Player B wants to choose $y_1, \ldots, y_m$ to minimise u.

The above equations can be expressed as a linear programming problem:

$$\begin{aligned} \text{maximise } Y_1 + \cdots + Y_m &= Z \\ \text{subject to } a_{1,1}Y_1 + \cdots + a_{1,m}Y_m &\leq 1 \\ &\vdots \\ a_{n,1}Y_1 + \cdots + a_{n,m}Y_m &\leq 1 \\ Y_1, \ldots, Y_m &\geq 0, \end{aligned}$$

where $Y_j = y_j/u$ and $Z = 1/u$.

Now, suppose that Player A plays mixed strategy $\mathbf{x}$. Player B will choose their move B_j to minimise the payoff to A, i.e., to obtain

$$w = \min_{j=1,\ldots,m} \left\{ \sum_{i=1}^{n} a_{i,j} x_i \right\}.$$

Player A wants to maximise w. We have

$$\sum_{i=1}^{n} a_{i,1} x_i \geq w, \ldots, \sum_{i=1}^{n} a_{i,m} x_i \geq w$$

and

$$x_1 + \cdots + x_n = 1,$$

which yields (assuming w is positive)

$$\sum_{i=1}^{n} a_{i,1} \frac{x_i}{w} \geq 1, \ \ldots, \ \sum_{i=1}^{n} a_{i,m} \frac{x_i}{w} \geq 1$$

$$\frac{x_1}{w} + \cdots + \frac{x_n}{w} = \frac{1}{w}$$

Let $X_i = x_i/w$ and $V = 1/w$. We have

$$\begin{aligned} \text{minimise } X_1 + \cdots + X_n &= V \\ \text{subject to } a_{1,1}X_1 + \cdots + a_{n,1}X_n &\geq 1 \\ &\vdots \\ a_{1,m}X_1 + \cdots + a_{n,m}X_n &\geq 1 \\ X_1, \ldots, X_n &\geq 0. \end{aligned}$$

Recognise this as the dual of the first LPP. Hence $Z^* = V^*$, i.e. $\min u = \max w$.

Theorem 4.17. *Provided the value of the game is greater than zero ($v > 0$), the value of the game is* $\min u = 1/Z^* = 1/V^*$*, and the optimal strategies are* $x_i^* = X_i^*/V^*$ *and* $y_j^* = Y_j^*/Z^*$.

Solving the first LPP using the simplex algorithm gives $Z^* = 1/u^* = 1/v$ and $Y_j^* = y_j^*/u^*$ $(j = 1, \ldots, m)$.

How should we calculate x_i^* $(i = 1, \ldots, n)$?

To complete the method we must consider what to do when $v \leq 0$. Since the value of the game cannot be less than the smallest value in the payoff matrix, we can simply add a constant to all entries in the payoff matrix, solve the LPP, and then subtract the same constant at the end.

If a' is the minimum entry in the payoff matrix, and $a' \leq 0$, add $-a' + 1$ to all payoff matrix entries, solve the LPP and then find $y_j^* = Y_j^*/Z^*$ and $v = 1/Z^* + a' - 1$.

Example 4.18. Suppose we have the following payoff matrix.

	B_1	B_2	B_3
A_1	3	-2	-3
A_2	-1	0	-2
A_3	-2	-1	3

The smallest entry is -3, so we add 4 to every entry and get,

	B_1	B_2	B_3
A_1	7	2	1
A_2	3	4	2
A_3	2	3	7

which must have a value greater than zero. The LPP is

$$\begin{aligned} \text{Maximise } Z &= Y_1 + Y_2 + Y_3 \\ \text{subject to } 7Y_1 + 2Y_2 + Y_3 &\leq 1 \\ 3Y_1 + 4Y_2 + 2Y_3 &\leq 1 \\ 2Y_1 + 3Y_2 + 7Y_3 &\leq 1 \\ Y_1, Y_2, Y_3 &\geq 0 \end{aligned}$$

This gives the initial tableau:

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Sol.
Y_4	7	2	1	1	0	0	1
Y_5	3	4	2	0	1	0	1
Y_6	2	3	7	0	0	1	1
Z	-1	-1	-1	0	0	0	0

The final tableau is

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Sol
Y_1	1	0	0	$\frac{2}{11}$	$\frac{1}{11}$	0	$\frac{1}{11}$
Y_2	0	1	0	$-\frac{17}{121}$	$\frac{47}{121}$	$\frac{1}{11}$	$\frac{19}{121}$
Y_3	0	0	0	$\frac{1}{121}$	$-\frac{17}{121}$	$\frac{2}{11}$	$\frac{6}{121}$
Z	0	0	0	$\frac{6}{121}$	$\frac{19}{121}$	$\frac{1}{11}$	$\frac{36}{121}$

The optimal solution is $Z^* = \frac{36}{121}$, $Y_1^* = \frac{1}{11}$, $Y_2^* = \frac{19}{121}$, $Y_3^* = \frac{6}{121}$. This gives the optimal strategy for Player B: $y_1^* = Y_1^*/Z^* = \frac{11}{36}$, $y_2^* = \frac{19}{36}$, $y_3^* = \frac{1}{6}$.

The optimal solution for Player A can be obtained using duality. The shadow prices for the primal problem are the solution to the dual problem. Thus, $X_1^* = \frac{6}{121}$, $X_2^* = \frac{19}{121}$, $X_3^* = \frac{1}{11}$, and Player A's optimal strategy is $x_1^* = X_1^*/Z^* = \frac{1}{6}$, $x_2^* = \frac{19}{36}$, $x_3^* = \frac{11}{36}$.

Finally, $v = 1/Z^* - 4 = -\frac{23}{36}$. The value, v, has turned out to be negative, so we did need to add a constant to the payoff matrix entries.

4.9. Games against nature

In this setting, we only have one 'rational' player, but the payoff also depends on some as yet unknown **state of nature**. Here 'nature' might mean nature (e.g., weather, biology) or some other phenomenon over which the player has no influence (e.g., stock-markets, traffic conditions, political policy).

We adopt a similar framework to the two-player game, assuming that nature is player B. However we no longer assume that player B adopts a strategy to minimise player A's reward. *Nature is not out to get us.* This fact means we must change our optimisation approach.

Example 4.19 (Electronics company)**.** An electronics company is considering launching a new product. They have 3 options:

A1: Launch now with a big advertising campaign

A2: Launch now with a minimal advertising campaign

A3: Make a small pilot study to see if the market is receptive to the product.

Nature is the market reaction, which can be good, moderate or poor.

Strategy	Good	Moderate	Poor
A1 (risky)	High sales High costs	Moderate sales High costs	Poor sales High costs
A2 (neutral)	Would do better with more advertising time	Moderate sales Low costs	Poor sales Low costs
A3 (cautious)	Competitors steal the market	Lower sales Lower costs	Very low costs, can switch to another product

The payoff matrix is given by

	Good	Moderate	Poor
A1 (risky)	110	45	-30
A2 (neutral)	90	55	-10
A3 (cautious)	80	40	10

In the previous sections A's strategy was determined by the **maximin criterion**. The maximin strategy is pure strategy A3, as there is a saddle point at the cautious/poor entry (10). However, unless we assume that the market is trying to minimise this company's costs, this is a worst-case analysis.

To simplify things, we will only consider pure strategies for games against nature.

There is no single optimal strategy. It depends on the player's attitude to risk and/or how likely they think the various possible states of nature are.

4.9.1. Maximax strategy

This is the strategy that can give the maximum payoff (but only if the state of nature happens to be 'right'). Thus the maximax strategy is the one that corresponds to the row with the largest entry in the payoff matrix. In Example 4.19 this is A1.

The maximax strategy is *risk taking*.

4.9.2. Laplacian strategy

The Laplacian strategy is the one that maximises the expected payoff, assuming that each state of nature is equally likely.

The expected payoffs for strategies A_1, A_2 and A_3 are
For A_1: $110/3 + 45/3 - 30/3 = 41\frac{2}{3}$
For A_2: $90/3 + 55/3 - 10/3 = 45$
For A_3: $80/3 + 40/3 + 10/3 = 43\frac{1}{3}$

Thus, the Laplacian strategy is A2.

4.9.3. Hurwicz strategy (not examinable in 2020–21)

Fix $\alpha \in [0, 1]$. For a move A_i the Hurwicz number H_i is

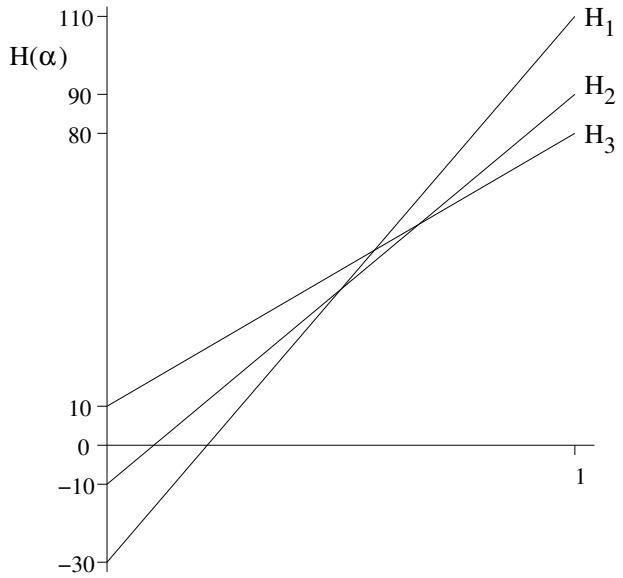
$$H_i = \alpha a_{iu} + (1 - \alpha)a_{il},$$

where a_{iu} is the maximum entry in row i and a_{il} is the minimum entry in row i of the payoff matrix.

The Hurwicz strategy is the strategy that gives the greatest Hurwicz number. It depends on α.

When $\alpha = 0$ we get the maximin solution and when $\alpha = 1$ we get the maximax solution. So, α is a weighting of the best and worse states of nature.

In our example we have:
$H_1 = 110\alpha - 30(1 - \alpha) = 140\alpha - 30$
$H_2 = 90\alpha - 10(1 - \alpha) = 100\alpha - 10$
$H_3 = 80\alpha + 10(1 - \alpha) = 70\alpha + 10$



So, when $\alpha = 0.25$, we have $H_1 = 5$, $H_2 = 15$ and $H_3 = 27.5$, giving A_3 as the optimal strategy.

For what values of α is A_3 the Hurwicz strategy?

A_3 is Hurwicz optimal if $\alpha < \alpha_0$, where α_0 is the value at which the lines of H_1 and H_3 intersect. In other words: $40\alpha_0 - 30 = 70\alpha_0 + 10$, giving $\alpha_0 = \frac{4}{7}$.

In general, the Hurwicz solution is a compromise of the two extreme states of nature. However, the method requires us to choose α and it does not take into account the other states of nature (which may be more likely).

4.9.4. Minimax regret strategy

Definition 4.20. The *regret* for a strategy and state of nature combination is the cost of taking that strategy compared to the best strategy for that state of nature.

For example, if the market turns out to be good, with A_1 we have a payoff of 110 which is the best payoff for that outcome, and we have zero regret. For A_2 we only get 90, when we could have obtained a payoff of 110, which is a regret of 20.

The regret matrix is

	Good	Moderate	Poor
A1 (risky)	0	10	40
A2 (neutral)	20	0	20
A3 (cautious)	30	15	0

The column minimum is always zero.

The minimax regret strategy is that which minimises the maximum regret, which in this case is A_2.

We can also use the minimin, Hurwicz or Laplace criterion on the regret matrix.

4.9.5. Expected Payoff and Variance

Suppose the probabilities of the states of nature are known/estimated, $\mathbf{p} = (p_1, \ldots, p_m)$, with $\sum p_j = 1$, $0 \leq p_j \leq 1$.

The expected payoff for pure strategy A_i is

$$E_{\mathbf{p}}(A_i) = \sum_{j=1}^{m} p_j a_{ij} \quad i = 1, \ldots, n.$$

We can choose our preferred strategy to be the pure strategy A_i that maximises $E_{\mathbf{p}}$. This is the **expected value criterion**. Note that if each $p_j = 1/m$, we have the Laplacian strategy.

Similarly, the variance of the payoff is

$$V_{\mathbf{p}}(A_i) = \sum_{j=1}^{m} p_j a_{ij}^2 - (E_{\mathbf{p}}(A_i))^2$$

4.9.6. Expected value–standard deviation criterion

In many cases, the strategy that maximises the expected payoff is very risky (e.g., investment in a high-return high-risk stock-market portfolio) and a less profitable but less risky option might be more appropriate (e.g., for a pension fund).

The expected value–standard deviation (EVSD) criterion is defined as

$$EVSD_{\mathbf{p}}(A_i, K) = E_{\mathbf{p}}(A_i) - K\sqrt{V_{\mathbf{p}}(A_i)},$$

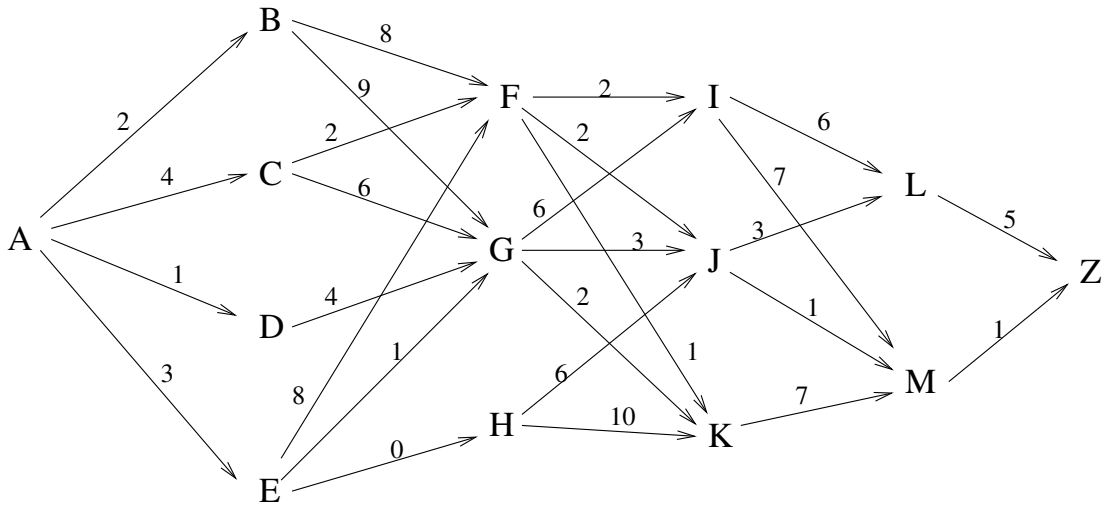
for some K. If K is large, strategies with a highly variable payoff are penalised. The optimal strategy under the EVSD criterion is the strategy A_i that maximises $EVSD_{\mathbf{p}}(A_i, K)$.

In Wilkes (1989) and many other OR books the expected value–variance criterion is considered instead. The EVSD is better from a statistical viewpoint because the standard deviation is measured on the same scale as the expected value, and so the EVSD has the same units as the expected value. In the context of portfolio management, this criterion is connected to so-called 'modern portfolio theory' developed by Markowitz.

5. Dynamic programming

5.1. Introduction

Example 5.1. Look at the following network. We have to get from node A to node Z in the shortest possible time, moving only in the direction of the arrows. The time needed to move from one node to another is indicated above the arrow. E.g. one possible route is ACGJLZ, which takes $4 + 6 + 3 + 3 + 5 = 21$ minutes.



We could inspect all possible routes. However, there are many possible routes, and this number would grow rapidly as the network grew. Dynamic programming is a more efficient method to solve this problem.

Dynamic programming is used when the problem can be separated into stages. An optimisation is performed at each stage in turn, but the optimal decision found at each stage depends on the optimal decision found at the next stage, and so on. It is only when the optimisation has been performed at the 'final' stage that it becomes clear what the optimal decision is at each of the earlier stages.

One application of dynamic programming is to find the route through a network that gives the minimum total 'cost' or maximum total 'reward'. Example 5.1 is an example of such a problem, in which the 'costs' are the times.

Notation

- Stage $n \in \{0, 1, \ldots, N\}$.
- S_n is a state at stage n.
- $\mathcal{Q}_n$ is the state space at stage n, i.e. is the set of all possible states at stage n. (So, $S_n \in \mathcal{Q}_n$.)
- $c_{i,j}$ is the transition cost for moving from state i to state j.

Example 5.2 (Example 5.1 continued)**.** $N = 5$, $\mathcal{Q}_0 = \{A\}$, $\mathcal{Q}_1 = \{B, C, D, E\}$, $\mathcal{Q}_2 = \{F, G, H\}$, ..., $\mathcal{Q}_5 = \{Z\}$, $c_{A,B} = 2$, etc.

- For each state in stage 1, find the quickest route from the source A. How long does each route take?
- For each state in stage 2, find the quickest route from A, using the information from the previous step. How long does each route take?
- ...
- Finally, for the end node Z in stage 5, find the quickest route from A, using the information about stage 4. How long does this route take?

We have separated the problem into stages and 'solved' each stage in turn. This is dynamic programming. The method we have just used is a dynamic programming algorithm called 'forward recursion'.

5.2. Forward recursion

Define $f_n(S_n)$ as minimum cost for moving from (any state in) stage 0 to state S_n in stage n.

Clearly, $f_0(S_0) = 0$ for every $S_0 \in \mathcal{Q}_0$.

The forward recursion equation is given by

$$f_n(S_n) = \min_{S_{n-1} \in \mathcal{Q}_{n-1}} \left\{ f_{n-1}(S_{n-1}) + c_{S_{n-1},S_n} \right\}.$$

(for $n = 1, \ldots, N$).

If the transition from a state S_{n-1} to a state S_n is not possible, treat c_{S_{n-1},S_n} as ∞.

Example 5.3 (Example 5.1 continued)**.** In fact, forward recursion is precisely the algorithm we used above for Example 5.1.

$$f_1(B) = \min_{S_0 \in \{A\}} \{f_0(S_0) + c_{S_0,B}\} = f_0(A) + c_{A,B} = 0 + 2$$

The optimal route to B is $A \to B$. Similarly, for all the other states, C, D and E, in stage 1. Then

$$\begin{aligned} f_2(F) &= \min_{S_1 \in \{B,C,D,E\}} \{f_1(S_1) + c_{S_1,F}\} \\ &= \min\{2 + 8, 4 + 2, 1 + \infty, 3 + 8\} = 6 \end{aligned}$$

So, the optimal route to F passes through C. Since we know that the optimal route to C is $A \to C$, the optimal route to F is $A \to C \to F$. Similarly, for the other states, G and H, in stage 2. We do the same for stages 3 and 4, and find that $f_4(L) = 10$, $f_4(M) = 8$ and the optimal route to M is $A \to E \to G \to J \to M$.

Finally,

$$\begin{aligned} f_5(Z) &= \min_{S_4 \in \{L,M\}} \{f_4(S_4) + c_{S_4,Z}\} \\ &= \min\{10 + 5, 8 + 1\} = 9 \end{aligned}$$

So, the optimal route to Z passes through M. Since we know that the optimal route to M is $A \to E \to G \to J \to M$, the optimal route to Z is $A \to E \to G \to J \to M \to Z$.

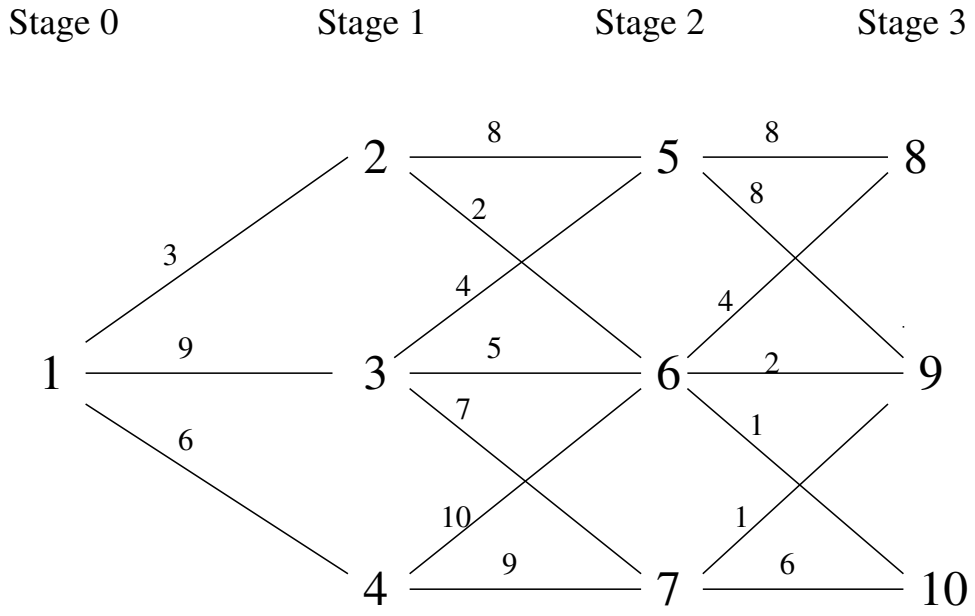
Maximising reward Alternatively, instead of minimising a total cost, we might want to maximise a total reward. Here, instead of $c_{i,j}$ being the transition cost for moving from state i to state j, $r_{i,j}$ is the transition reward.

Define $f_n(S_n)$ as maximum reward for moving from (any state in) stage 0 to state S_n in stage n, and let $f_0(S_0) = 0 \quad \forall S_0 \in \mathcal{Q}_0$ again. The forward recursion equation to maximise a reward is given by

$$f_n(S_n) = \max_{S_{n-1} \in \mathcal{Q}_{n-1}} \left\{ f_{n-1}(S_{n-1}) + r_{S_{n-1},S_n} \right\}.$$

The forward recursion equation implies that it is enough to know the optimal rewards/costs for reaching every state in the previous stage and the transition rewards/costs to get from each of these states to the current state. We **do not** need to consider the entire path to reach the current state.

Example 5.4. The following diagram shows a network with 4 stages (0, 1, 2 and 3) and the transition rewards. Find the route from stage 0 to stage 3 that maximises the total reward. Note: You may end in any of the three states of stage 3.



We find $f_n(S_n)$ for every state at every stage (starting with stage 0) and the state in the previous stage through which we should pass.

Stage 0 $f_0(1) = 0$.

Stage 1 The maximum rewards at the first stage are just the transition rewards:

$f_1(2) = \max\{f_0(1) + r_{1,2}\} = 3$
$f_1(3) = \max\{f_0(1) + r_{1,3}\} = 9$
$f_1(4) = \max\{f_0(1) + r_{1,4}\} = 6$
The optimal routes are obviously $1 \to 2$, $1 \to 3$ and $1 \to 4$.

Stage 2

$f_2(5) = \max\{f_1(2) + r_{2,5}, f_1(3) + r_{3,5}\} = \max\{3 + 8, 9 + 4\} = 13$ (through state 3)
$f_2(6) = \max\{f_1(2) + r_{2,6}, f_1(3) + r_{3,6}, f_1(4) + r_{4,6}\} = \max\{3 + 2, 9 + 5, 6 + 10\} = 16$ (through 4)
$f_2(7) = \max\{f_1(3) + r_{3,7}, f_1(4) + r_{4,7}\} = \max\{9 + 7, 6 + 9\} = 16$ (through 3)

Stage 3

$$f_3(8) = \max\{f_2(5) + r_{5,8}, f_2(6) + r_{6,8}\} = \max\{13 + 8, 16 + 4\} = 21 \text{ (through 5)}$$
$$f_3(9) = \max\{f_2(5) + r_{5,9}, f_2(6) + r_{6,9}, f_2(7) + r_{7,9}\} = \max\{13 + 8, 16 + 2, 16 + 1\} = 21 \text{ (through 5)}$$
$$f_3(10) = \max\{f_2(6) + r_{6,10}, f_2(7) + r_{7,10}\} = \max\{16 + 1, 16 + 6\} = 22 \text{ (through 7)}$$

The optimal route ends at state 10 and passes through 7. The optimal route to 7 passes through 3. So, the optimal route is $1 \to 3 \to 7 \to 10$.

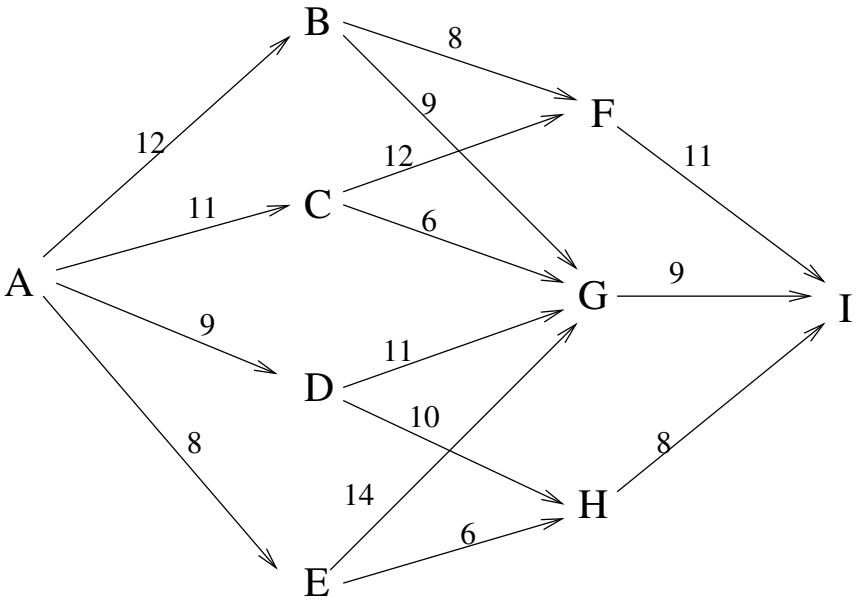
Check that this route gives the correct total reward.

$$r_{1,3} + r_{3,7} + r_{7,10} = 9 + 7 + 6 = 22.$$

Note To find the optimal **route**, we do not actually need to note down the optimal route to every state. We can wait until we get to the end, when we know the optimal **reward**, and then find the optimal route by "rolling back" the solution. You may prefer to find the optimal route this way (it involves less work). In this example we want to finish at state 10, giving a reward of 22. But 22 comes from $f_2(7) + r_{7,10} = 16 + 6$, so if we want the optimal reward we must visit state 7. The optimal reward for getting to state 7 is obtained from $f_1(3) + r_{3,7} = 9 + 7$ so we must visit state 3. Thus the optimal route is to go through states 3,7 and 10, giving a reward of 9+7+6=22.

5.3. Backward recursion

Example 5.5 (Pipeline routeing problem)**.** A company wishes to lay a pipeline from their processing plant to a new local distribution point. The company has identified a network of possible routes with the associated costs of installing each section.



Using the same method as before, the value $f_n(S_n)$ can be written in next to the appropriate state. Note that this is a minimisation problem, as we are dealing with costs.

We can also use **backward recursion**.

Here, we define $g_n(S_n)$ as the maximum reward / minimum cost for moving from state S_n in stage n to (any state in) stage N.

Clearly, $g_N(S_N) = 0 \quad \forall S_N \in \mathcal{Q}_N$.

Now, we work backwards, going first to stage $N-1$, then to $N-2$, etc., and ending in stage 0. For costs,

$$g_n(S_n) = \min_{S_{n+1} \in \mathcal{Q}_{n+1}} \left\{ g_{n+1}(S_{n+1}) + c_{S_n, S_{n+1}} \right\}, \quad (n = 0, 1, \ldots, N-1)$$

(with the obvious changes for a maximising rewards problem).

In forward recursion, $f_n(S_n)$ is the optimum return obtainable by moving from stage 0 to state S_n at stage n.

In backward recursion, $g_n(S_n)$ is the optimum return obtainable by moving from state S_n at stage n to stage N.

We can write in the values $g_n(S_n)$ next to each state S_n. For example,

$$g_3(I) = 0$$
$$g_1(B) = \min_{S_2 \in \{F,G\}} \{g_2(S_2) + c_{B,S_2}\} = \min\{11+8, 9+9\} = 18.$$

We see that both forward and backward recursion have given the same solution.

Now, we "roll forward" the solution to find the optimal route. This is $A \to E \to H \to I$, with a total cost of 22.

Comments on forward and backward recursion

The computational effort may be different for the two methods.

When a dynamic programming problem has a random component to it, as in the next chapter, only backward recursion can be used.

5.4. Resource allocation problems

This is a common type of problem which once formulated as a dynamic programming problem can be solved in a straightforward manner.

Example 5.6. A company wishes to expand its business by enlarging three of its manufacturing plants. For each type of expansion there is an expense and a future revenue (reward) according to the following table (all entries in £M).

Proposed	Plant 1		Plant 2		Plant 3	
expansion	expense	revenue	expense	revenue	expense	revenue
A	0	0	0	0	0	0
B	1	5	2	8	1	3
C	2	6	3	9	—	—
D	—	—	4	12	—	—

The company has a budget of £5M and it wants to maximise its revenue from the available expansions.

We can consider the problem sequentially.

Stage 0: no expansion plans have been decided.

Stage 1: the expansion plan for plant 1 has been decided.

Stage 2: the expansion plans for plants 1 and 2 have been decided.
Stage 3: the expansion plans for plants 1, 2 and 3 have been decided.

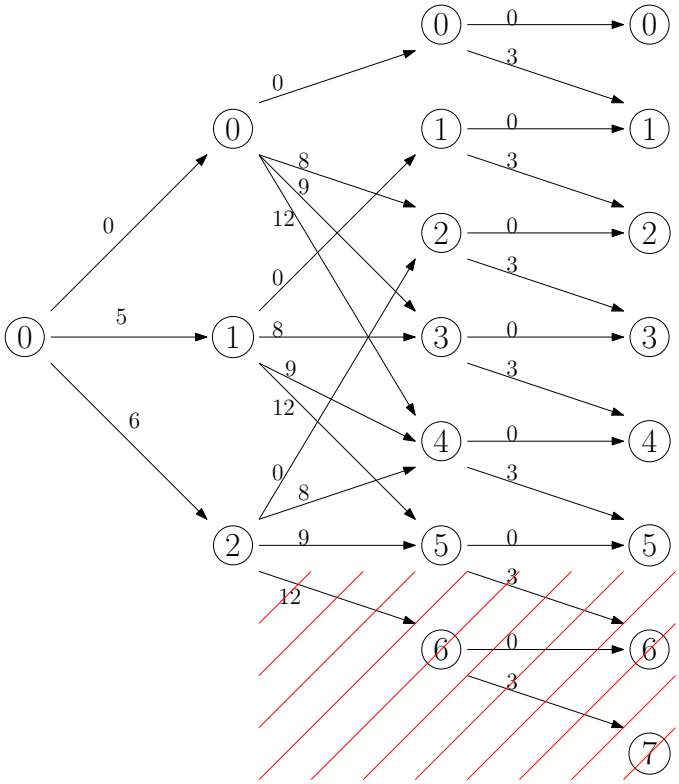
Let S_n, the state at stage n ($n = 0, 1, 2, 3$), be the cumulative expense by stage n.

Let $r_n(i, j)$ be the transition reward of moving from state i at stage n to state j at stage $n+1$. This will be the revenue of the expansion project for plant $n+1$ that has associated expense $j-i$.

We can now draw a network showing the possible states at each stage, the possible transitions and the transition rewards.

This problem can be solved using forward or backward recursion.

$$\begin{aligned} \mathcal{Q}_0 &= \{0\} \\ \mathcal{Q}_1 &= \{0, 1, 2\} \\ \mathcal{Q}_2 &= \{0, 1, 2, 3, 4, 5\} \\ \mathcal{Q}_3 &= \{0, 1, 2, 3, 4, 5\} \end{aligned}$$



Forward recursion Let $f_n(S_n)$ be the maximum revenue available in stages $0, \ldots, n$ when the state at stage n is S_n (i.e. having spent S_n million expanding plants $1, \ldots, n$).

So, the forward recursion equation is

$$f_n(S_n) = \max_{S_{n-1} \in \mathcal{Q}_{n-1}} \{f_{n-1}(S_{n-1}) + r_{n-1}(S_{n-1}, S_n)\}.$$

and $f_0(0) = 0$, as usual.

It is easiest to solve this problem in tabular format.

Stage 1 $f_1(S_1) = \max_{S_0 \in \{0\}} \{f_0(S_0) + r_0(S_0, S_1)\} = r_0(0, S_1)$

S_1	$f_1(S_1)$
0	0
1	5
2	6

Stage 2 $f_2(S_2) = \max_{S_1 \in \{0,1,2\}} \{f_1(S_1) + r_1(S_1, S_2)\}$.

S_2	$S_1 = 0$	$S_1 = 1$	$S_1 = 2$	$f_2(S_2)$	S_1^*
0	0+0	—	—	0	0
1	—	5+0	—	5	1
2	0+8	—	6+0	8	0
3	0+9	5+8	—	13	1
4	0+12	5+9	6+8	14	1 or 2
5	—	5+12	6+9	17	1

Stage 3 $f_3(S_3) = \max_{S_2 \in \{0,1,2,3,4,5\}} \{f_2(S_2) + r_2(S_2, S_3)\}$.

S_3	$S_2 = 0$	$S_2 = 1$	$S_2 = 2$	$S_2 = 3$	$S_2 = 4$	$S_2 = 5$	$f_3(S_3)$	S_2^*
0	0+0	—	—	—	—	—	0	0
1	0+3	5+0	—	—	—	—	5	1
2	—	5+3	8+0	—	—	—	8	1 or 2
3	—	—	8+3	13+0	—	—	13	3
4	—	—	—	13+3	14+0	—	16	3
5	—	—	—	—	14+3	17+0	17	4 or 5

Thus, the maximum revenue is £17M, which is achieved by getting to $S_3 = 5$, i.e. spending a total of £5M.

We now "roll back" the solution to find the optimal investment strategy.

$S_3 = 5$

$S_2 = 4$ or 5

If $S_2 = 4$, then $S_1 = 1$ or 2, and $S_0 = 0$. If $S_2 = 5$, then $S_1 = 1$, and $S_0 = 0$.

So, there is a choice of expansion plans:

Plan	S_0	S_1	S_2	S_3
$B \to C \to B$	0	1	4	5
$B \to D \to A$	0	1	5	5
$C \to B \to B$	0	2	4	5

Backward recursion Let $g_n(S_n)$ be the maximum revenue available over *future* stages, starting in state S_n at stage n (i.e. having spent S_n million expanding plants $1, \ldots, n$).

Clearly, $g_3(S_3) = 0$ for all $S_3 \in \mathcal{Q}_3$. Now,

$$g_n(S_n) = \max_{S_{n+1} \in \mathcal{Q}_{n+1}} \{g_{n+1}(S_{n+1}) + r_n(S_n, S_{n+1})\} \quad (n = 0, 1, 2)$$

Stage 2 $g_2(S_2) = \max_{S_3 \in \{0,1,2,3,4,5\}} \{g_3(S_3) + r_2(S_2, S_3)\}$.

S_2	$S_3=0$	$S_3=1$	$S_3=2$	$S_3=3$	$S_3=4$	$S_3=5$	$g_2(S_2)$	S_3^*
0	0+0	0+3	—	—	—	—	3	1
1	—	0+0	0+3	—	—	—	3	2
2	—	—	0+0	0+3	—	—	3	3
3	—	—	—	0+0	0+3	—	3	4
4	—	—	—	—	0+0	0+3	3	5
5	—	—	—	—	—	0+0	0	5

Stage 1 $g_1(S_1) = \max_{S_2 \in \{0,1,2,3,4,5\}} \{g_2(S_2) + r_1(S_1, S_2)\}$.

S_1	$S_2=0$	$S_2=1$	$S_2=2$	$S_2=3$	$S_2=4$	$S_2=5$	$g_1(S_1)$	S_2^*
0	3+0	—	3+8	3+9	3+12	—	15	4
1	—	3+0	—	3+8	3+9	0+12	12	4 or 5
2	—	—	3+0	—	3+8	0+9	11	4

Stage 0 $g_0(S_0) = \max_{S_1 \in \{0,1,2,3,4,5\}} \{g_1(S_1) + r_0(S_0, S_1)\}$.

S_0	$S_1=0$	$S_1=1$	$S_1=2$	$g_0(S_0)$	S_1^*
0	15+0	12+5	11+6	17	1 or 2

Again, the maximum revenue is £17M. We can now roll forward the solution, which again yields the same optimal routes as we obtained earlier using forward recursion.

5.5. The warehousing problem

This is another example of dynamic programming solving a practical problem.

A company manufactures and supplies a product. Each month they manufacture a certain amount at a production plant, which is then stored at their warehouse. They also distribute some or all of the warehoused stock to the retailers.

Assume that the distributed stock leaves the warehouse early afternoon on the first of each month and the manufactured stock is brought to the warehouse in the evening on the day it is manufactured. On the morning of the first of each month, a manager has to choose how much stock is to be distributed that day, and how much is to be manufactured that month.

Notation

t	month ($t = 1, \ldots, T$)
Q_t	number of items manufactured in month t
V_t	number of items distributed to the retailers during the first day of month t
c_t	manufacturing cost (per item) in month t
p_t	sell price (per item) in month t
I_t	number of items stored in the warehouse at beginning of first day of month t
K	the maximum capacity (no. of items that can be stored) and
π_t	profit available in months $t, t+1, \ldots, T$.

Each of Q_t, V_t and K are ≥ 0, and for any month t the number of items that can be stored cannot exceed the maximum capacity, i.e. $0 \leq I_t \leq K$. The costs, c_t, and prices, p_t, are assumed to be known in advance but may be different from month to month.

The number of items in the warehouse at the start of month $t+1$ is the number of items there at the start of month t (I_t), plus the number of items manufactured in month t (Q_t) minus the number of items distributed in month t (V_t), giving

$$I_{t+1} = I_t - V_t + Q_t$$

We want to choose V_t and Q_t ($t = 1, \ldots, T$) so that the profit π_1 is maximised.

The above is a general problem. In a specific example we have:
The maximum capacity is $K = 1000$.
The initial amount of stock in the warehouse is $I_1 = 300$.
We are considering the problem over a four month period, so $T = 4$,
and at the end of the 4 month period we want an empty warehouse, so $I_5 = 0$.
The values c_t and p_t are

t	c_t	p_t
1	70	90
2	64	82
3	73	70
4	70	85

We need to use backward recursion for this problem.

Month 4 Let $t = 4$ and assume that today is the first day of month 4. Imagine that the number of items in the warehouse at the start of today, I_4, is fixed. (We will write our maximal profit and decisions in terms of I_4.)

The profit for month 4 is $\pi_4 = p_4 V_4 - c_4 Q_4$. Since we are required to have $I_5 = 0$, and we know that $I_5 = I_4 - V_4 + Q_4$, the only way we can satisfy these equations (and still have $Q_4 \geq 0$ and $V_4 \leq I_4$) is if $Q_4 = 0$ and $I_4 = V_4$. So,

$$\pi_4^* = p_4 I_4 + 0 = 85 I_4,$$

with optimal actions $V_4^* = I_4$ and $Q_4^* = 0$.

Month 3 Now set $t = 3$ and assume that I_3 is fixed. We will write everything this month in terms of I_3. The profit (over months 3 and 4) is

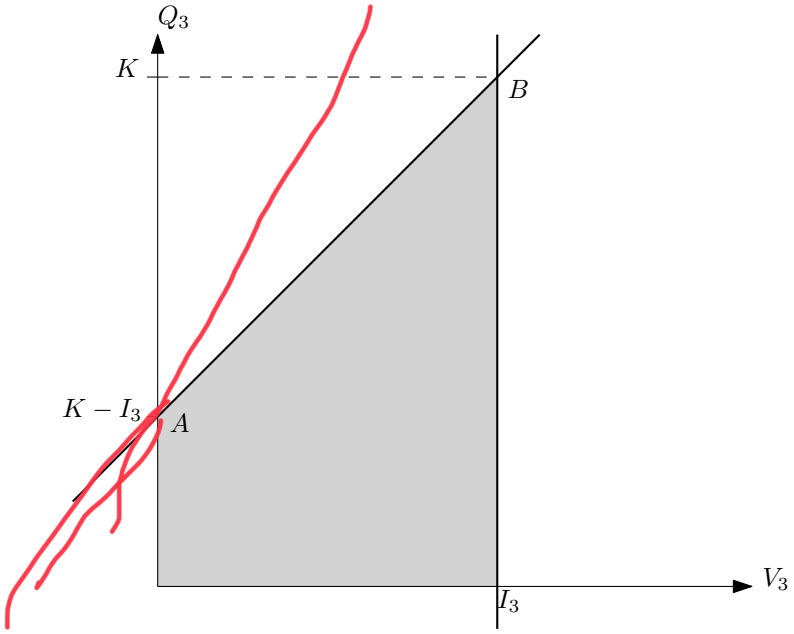
$$\begin{aligned}\pi_3 &= p_3 V_3 - c_3 Q_3 + \pi_4^* \\ &= 70V_3 - 73Q_3 + 85I_4 \\ &= 70V_3 - 73Q_3 + 85(I_3 - V_3 + Q_3) \\ &= -15V_3 + 12Q_3 + 85I_3.\end{aligned}$$

We know that $I_4 = I_3 - V_3 + Q_3 \leq K$, since we cannot exceed the warehouse capacity; and we also know that $V_3 \leq I_3$, since we cannot sell more stock than we have available.

In other words, we are seeking to solve a linear programming problem in the two variables V_3, Q_3 (plus a constant added to the objective function):

$$\begin{aligned}\text{Maximise } \pi_3 &= -15V_3 + 12Q_3 + 85I_3 \\ \text{subject to } V_3 &\leq I_3 \\ Q_3 - V_3 &\leq K - I_3 \\ Q_3, V_3 &\geq 0.\end{aligned}$$

The feasible region is:



We can solve this graphically; a contour of the objective function is

$$Q_3 = \frac{15}{12}V_3 + \frac{1}{12}\pi_3 - \frac{85}{12}I_3,$$

so we should maximise the intercept, and this leads to the optimal solution at point A, i.e.,

$$\begin{aligned}V_3^* &= 0 \\ Q_3^* &= K - I_3 \\ \pi_3^* &= 0 + 12(K - I_3) + 85I_3 = 73I_3 + 12K.\end{aligned}$$

Month 2 We now set $t = 2$ and observe that

$$\begin{aligned}\pi_2 &= p_2V_2 - c_2Q_2 + \pi_3^* \\ &= 82V_2 - 64Q_2 + 73I_3 + 12K \\ &= 82V_2 - 64Q_2 + 73(I_2 - V_2 + Q_2) + 12K \\ &= 9V_2 + 9Q_2 + 73I_2 + 12K.\end{aligned}$$

This leads us to solve the linear programming problem

$$\begin{aligned}\text{Maximise } \pi_2 &= 9V_2 + 8Q_2 + 73I_2 + 12K \\ \text{subject to } V_2 &\leq I_2 \\ Q_2 - V_2 &\leq K - I_2 \\ Q_2, V_2 &\geq 0.\end{aligned}$$

The feasible region is the same shape as in the previous month, but now we are maximising the intercept of a contour with gradient -1, and this leads us to the optimal solution at the point corresponding to B. That is,

$$\begin{aligned}V_2^* &= I_2, \\ Q_2^* &= K, \\ \pi_2^* &= 9I_2 + 9K + 73I_2 + 12K = 82I_2 + 21K.\end{aligned}$$

Month 1 We now set $t = 1$ in order to find π_1^*, which was our goal. We observe

$$\begin{aligned}\pi_1 &= p_1V_1 - c_1Q_1 + \pi_2^* \\ &= 90V_1 - 70Q_1 + 82(I_1 - V_1 + Q_1) + 21K \\ &= 8V_1 + 12Q_1 + 82I_1 + 21K\end{aligned}$$

Solving this again using linear programming, we arrive at

$$\begin{aligned}V_1^* &= I_1, \\ Q_1^* &= K, \\ \pi_1^* &= 8I_1 + 12K + 82I_1 + 21K = 90I_1 + 33K.\end{aligned}$$

We summarise our results, and find numerical values, in the following table. We have used here the information about the initial number of items in the warehouse, $I_1 = 300$, and the numerical value of the capacity, $K = 1000$.

t	I_t	V_t^*	Q_t^*	π_t^*
1	300	$I_1 = 300$	$K = 1000$	$90I_1 + 33K = 60\,000$
2	1000	$I_2 = 1000$	$K = 1000$	$82I_2 + 21K = 103\,000$
3	1000	0	$K - I_3 = 0$	$73I_3 + 12K = 85\,000$
4	1000	$I_4 = 1000$	0	$85I_4 = 85\,000$
5	0			

The maximal profit attainable over the entire period is $\pi_1^* = 60\,000$, and this is obtained via the choices of V_t, Q_t given in the table.

Remark (not examinable). Forward recursion is impractical for this model. Why? Write ϖ_t for the profit attained over periods $1, \ldots, t$. We want to maximise ϖ_6. Let $\varpi_t^*(x)$ be the maximal profit over t periods leading to $I_t = x$; then the forward recursion equation looks like

$$\varpi_t^*(I_t) = \max_{I_{t-1}} \left\{ p_t V_t - c_t Q_t + \varpi_t^*(I_{t-1}) \right\}.$$

The first step of the recursion is to find $\varpi_1^*(x)$ for all $0 \leq x \leq K$, but this value, and the associated strategies V_1, Q_1, depend critically on the value of x. Contrast this with the backward recursion, where each π_t^* depended in a simple way on I_{t+1}.

6. Stochastic Optimisation

6.1. Introduction

In Chapter 5, the rewards or costs of transitions were assumed to be known and fixed, and we were able to choose which transitions to make. In this chapter we deal with problems where these assumptions do not hold. Such problems are solved by **stochastic programming** or **stochastic optimisation** algorithms. We shall concentrate on one aspect of stochastic optimisation known as Markov Dynamic Programming (MDP), sometimes known as Markov Programming or Markov Decision Programming.

A **discrete-time random** (or **stochastic**) **process** is a sequence of random variables indexed by time, $X = \{X_t : t = 0, 1, 2, \dots\}$.

6.2. Markov Chains

Definition 6.1. A discrete-time stochastic process X is called a **Markov chain** if

$$P(X_{n+1} = x_{n+1} \mid X_0 = x_0, \dots, X_n = x_n) = P(X_{n+1} = x_{n+1} \mid X_n = x_n), \tag{6.1}$$

for $n = 0, 1, 2, \dots$. Property (6.1) is called the **Markov property**. If we regard time n as the present, then the Markov property says that 'given the present, the future is independent of the past'.

Example 6.2. Mr Bond is playing roulette. At each turn he places a £1 chip on number 17. He begins with $X_0 = x_0$ chips. Let X_n denote the number of £1 chips he has after n turns.

The number of chips he has after $n + 1$ turns, X_{n+1}, depends on the number of chips he has after n turns, X_n (plus the random factor from the roulette wheel). But given the value of X_n, X_{n+1} is independent of $X_0, \dots, X_{n-1}$. Therefore, $X = \{X_0, X_1, \dots\}$ is a Markov chain.

In most of this chapter we consider **finite horizon** problems, meaning that we only consider times $n = 0, 1, \dots, N$, with N fixed. In Section 6.6 we allow an infinite number of stages.

In a finite horizon Markov dynamic programming problem we have the following:

1. a Markov chain $X = \{X_0, X_1, \dots, X_N\}$ with known transition probabilities,
2. costs or rewards associated with each transition in the Markov chain,
3. terminal costs or rewards associated with the states at the final stage, and
4. (usually) actions that alter the transition probabilities and costs/rewards.

Notation We always use backward recursion for MDP problems, for reasons which will become clear. For this reason, it is convenient to use n to denote the **number of stages to go**, rather than the current stage (as we usually did in Chapter 5.) So, $n = 0$ at the final (terminal) stage and $n = N$ at the initial stage.

- Let $X^{(n)}$ denote the state with n stages to go (i.e., $X^{(n)} = X_{N-n}$.)
- Let $\mathcal{Q}_n$ denote the state space with n stages to go. (Note the difference from Chapter 5.)
- Let $p_{i,j}^{(n)}$ denote the transition probability for moving from state i with n stages to go to state j (with $n-1$ stages to go).

$$p_{i,j}^{(n)} = P(X^{(n-1)} = j \mid X^{(n)} = i) = P(X_{N-n+1} = j \mid X_{N-n} = i).$$

The **transition matrix** $P^{(n)}$ is the matrix whose (i,j)th entry is $p_{i,j}^{(n)}$.
Clearly,

$$p_{i,j}^{(n)} \geq 0, \text{ for all } i,j; \text{ and } \sum_{j \in \mathcal{Q}_{n-1}} p_{i,j}^{(n)} = 1, \text{ for all } i.$$

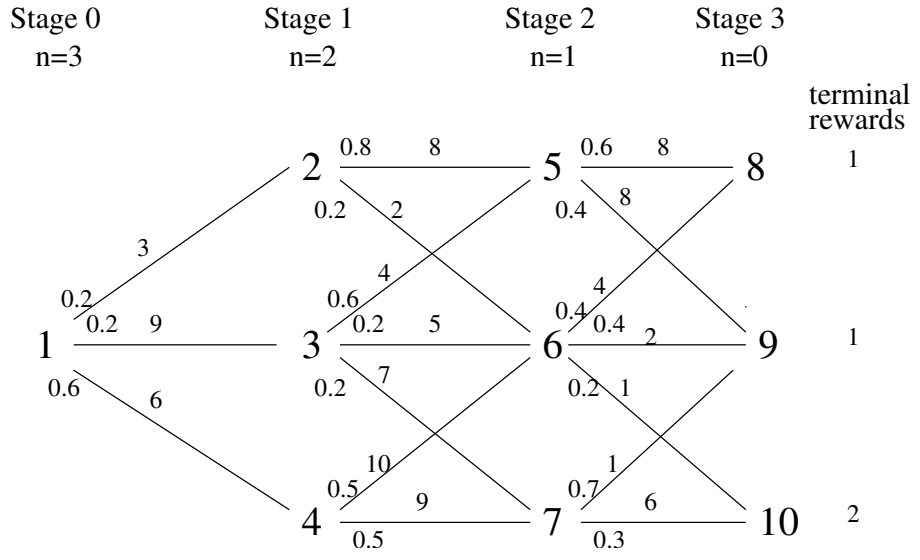
So, each element of $P^{(n)}$ must be between zero and one, and the rows must sum to one.

- Let $r_{i,j}^{(n)}$ or $c_{i,j}^{(n)}$ denote the transition reward/cost for the transition from state i with n stages to go to state j (with $n-1$ stages to go).
- Let $r_i^{(0)}$ or $c_i^{(0)}$ denote the terminal reward/cost for state i, i.e., the reward/cost incurred when we end in state i.

Sometimes the transition probabilities will be the same at every stage, in which case the superscript $\cdot^{(n)}$ in $p_{i,j}^{(n)}$ and $P^{(n)}$ can be dropped.

6.3. Rewards in a Markov setting (with no actions)

Example 6.3. Consider the following network, in which we start at state 1, and move to the right with the probabilities given on the edges, collecting an associated reward for each edge and a terminal reward at the end.



The state space for $X^{(0)}$ is $\mathcal{Q}_0 = \{8, 9, 10\}$, for $X^{(1)}$ is $\mathcal{Q}_1 = \{5, 6, 7\}$, for $X^{(2)}$ is $\mathcal{Q}_2 = \{2, 3, 4\}$ and for $X^{(3)}$ is $\mathcal{Q}_3 = \{1\}$.

The transition probabilities are $P(X^{(0)} = 8 \mid X^{(1)} = 6) = p_{6,8}^{(1)} = 0.4$, etc. The transition rewards are $r_{1,2}^{(3)} = 3$, $r_{2,5}^{(2)} = 8$, etc. Finally, the terminal rewards are $r_8^{(0)} = 1$, $r_9^{(0)} = 1$ and $r_{10}^{(0)} = 2$.

Definitions 6.4. Let $R_n(i)$ be the **one-step expected reward** when in state i with n stages to go ($X^{(n)} = i$), for the next step to $X^{(n-1)}$. The formula for $R_n(i)$ is

$$R_n(i) = \begin{cases} \sum_{j \in \mathcal{Q}_{n-1}} p_{i,j}^{(n)} r_{i,j}^{(n)} & n \geq 1 \\ \\ r_i^{(0)} & n = 0 \end{cases}$$

Let $V_n(i)$ be the total future expected reward when in state i with n stages to go ($X^{(n)} = i$). The function $V_n(i)$ is called the **value function**.

$$\begin{aligned} V_n(i) &= \begin{cases} \sum_{j \in \mathcal{Q}_{n-1}} p_{i,j}^{(n)} \left(r_{ij}^{(n)} + V_{n-1}(j) \right) & n \geq 1 \\ \\ r_i^{(0)} & n = 0 \end{cases} \\ &= \begin{cases} R_n(i) + \sum_{j \in \mathcal{Q}_{n-1}} p_{i,j}^{(n)} V_{n-1}(j) & n \geq 1 \\ \\ R_0(i) = r_i^{(0)} & n = 0. \end{cases} \end{aligned}$$

This is a recursive formula for $V_n(i)$ in terms of V_{n-1}, so we can use a similar methods to those used in the last chapter in order to find the expected reward over the whole problem.

For the present example, we want to find $V_3(1)$ by recursion.
So, we start at stage 3, when $n = 0$. We have

$$\begin{aligned} V_0(8) &= r_8^{(0)} = 1 \\ V_0(9) &= r_9^{(0)} = 1 \\ V_0(10) &= r_{10}^{(0)} = 2. \end{aligned}$$

At stage 2 ($n = 1$) we can compute the one-step expected rewards for each state:

$$\begin{aligned} R_1(5) &= \sum_{j=8}^{10} p_{5,j}^{(1)} r_{5,j}^{(1)} = 0.6 \cdot 8 + 0.4 \cdot 8 = 8 \\ R_1(6) &= \sum_{j=8}^{10} p_{6,j}^{(1)} r_{6,j}^{(1)} = 0.4 \cdot 4 + 0.4 \cdot 2 + 0.2 \cdot 1 = 2.6 \\ R_1(7) &= \sum_{j=8}^{10} p_{7,j}^{(1)} r_{7,j}^{(1)} = 0.7 \cdot 1 + 0.3 \cdot 6 = 2.5 \end{aligned}$$

This in turn allows us to compute the value function for each state in stage 2 ($n = 1$):

$$\begin{aligned} V_1(5) &= R_1(5) + \sum_{j=8}^{10} p_{5,j}^{(1)} V_0(j) = 8 + 0.6 \cdot 1 + 0.4 \cdot 1 = 9 \\ V_1(6) &= R_1(6) + \sum_{j=8}^{10} p_{6,j}^{(1)} V_0(j) = 2.6 + 0.4 \cdot 1 + 0.4 \cdot 1 + 0.2 \cdot 2 = 3.8 \\ V_1(7) &= R_1(7) + \sum_{j=8}^{10} p_{7,j}^{(1)} V_0(j) = 2.5 + 1.3 = 3.8 \end{aligned}$$

Repeat this for stage 1 ($n = 2$):

$$R_2(2) = 0.8 \cdot 8 + 0.2 \cdot 2 = 6.8$$
$$R_2(3) = 4.8$$
$$R_2(4) = 9.5$$

hence

$$V_2(2) = R_2(2) + \sum_j p_{2,j}^{(2)} V_1(j) = 6.8 + 0.8 \cdot 9 + 0.2 \cdot 3.8 = 14.76$$
$$V_2(3) = 11.72$$
$$V_2(4) = 13.3.$$

And finally, for stage 0 ($n = 3$):

$$R_3(1) = 0.2 \cdot 3 + 0.2 \cdot 9 + 0.6 \cdot 6 = 6$$
$$V_3(1) = 6 + 0.2 \cdot 14.76 + 0.2 \cdot 11.72 + 0.6 \cdot 13.3 = 19.276$$

The expected reward over the three stages starting in state 1 is 19.276.

Note. The deterministic Example 5.4 had the same network and the same rewards (but no terminal rewards). The maximum total reward was 22 (24 if we add the terminal reward). This was achieved by visiting states 1, 3, 7 and 10. In this example, we cannot decide which route to take (the route is totally random), so it is not surprising that the expected total reward is less than the maximum reward.

6.4. Markov dynamic programming with actions

Suppose the process is in state i with n stages to go ($X^{(n)} = i$), and that we now have K actions available $\left\{a_1^{(n)}, a_2^{(n)}, \ldots, a_K^{(n)}\right\} = \mathcal{A}^{(n)}$. We call $\mathcal{A}^{(n)}$ the *action space*.

We can extend the dynamic programming model to depend on the actions taken:

1. The transition probabilities depend on the action taken at each stage, so

$$p_{i,j}^{(n)}(a^{(n)}) = P(X^{(n-1)} = j \mid X^{(n)} = i, a^{(n)}),$$

   where $a^{(n)} \in \mathcal{A}^{(n)}$ is the particular action chosen with n stages to go. We **also** insist that the probability of a transition to $X^{(n-1)} = j$ depends n $X^{(n)} = i$ and $a^{(n)}$, but does not depend on the previous states, nor the previous actions.

2. The transition rewards/costs depend on the action taken at each stage. So, we have $r_{i,j}^{(n)}(a^{(n)})$ or $c_{i,j}^{(n)}(a^{(n)})$.

3. The terminal rewards/costs **do not depend** on the actions.

Now that we have actions, there is a choice available. We want to find the sequence of actions that will maximise our expected total reward, or minimise our expected total cost. First, we extend the definitions of the one-step expected reward and the value function to take account of the actions chosen.

Definitions 6.5. $R_n(i, a^{(n)})$ is the *one-step expected reward* when the state is i with n stages to go ($X_n = i$) and action $a^{(n)}$ is taken.

$V_n^*(i)$ is the future expected reward when in state i with n stages to go and an optimal action plan is used for these remaining n stages. $V_n^*(i)$ is called the *optimal value function*.

As before, we have an expression for the one-step expected reward:

$$R_n(i, a^{(n)}) = \begin{cases} \sum_{j \in \mathcal{Q}_{n-1}} p_{i,j}^{(n)}(a^{(n)})\, r_{i,j}^{(n)}(a^{(n)}) & n \geq 1 \\ r_i^{(0)} & n = 0 \end{cases}$$

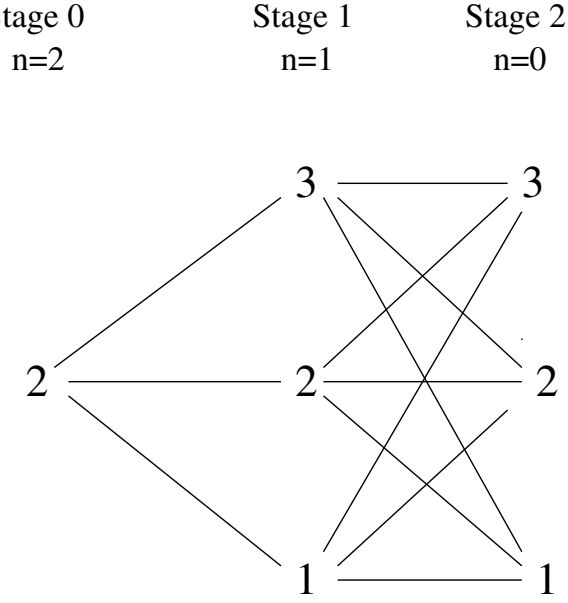
The optimal value function is given by the following recursive equation.

$$\begin{aligned} V_n^*(i) &= \begin{cases} \max_{a^{(n)} \in \mathcal{A}^{(n)}} \left\{ \sum_{j \in \mathcal{Q}_{n-1}} p_{i,j}^{(n)}(a^{(n)}) \left(r_{i,j}^{(n)}(a^{(n)}) + V_{n-1}^*(j) \right) \right\} & n \geq 1 \\ r_i^{(0)} & n = 0 \end{cases} \\ &= \begin{cases} \max_{a^{(n)} \in \mathcal{A}^{(n)}} \left\{ R_n(i, a^{(n)}) + \sum_{j \in \mathcal{Q}_{n-1}} p_{i,j}^{(n)}(a^{(n)})\, V_{n-1}^*(j) \right\} & n \geq 1 \\ r_i^{(0)} & n = 0 \end{cases} \end{aligned}$$

Example 6.6 (Advertising for a concert)**.** A ticket agency is responsible for advertising and selling tickets for a concert which will take place in two weeks. Each week the agency has the choice either to put a half-page advert in the major newspapers and magazines, or just to submit the concert details to the 'what's-on' listings. To simplify the problem, the agency categorises the ticket sales into three states, fast ticket sales, average ticket sales and slow ticket sales. At present, the ticket sales are average.

Let n denote the number of weeks remaining before the concert takes place. Label the states as: 3 for fast sales, 2 for average sales and 1 for slow sales. Label the actions as A for Advertise, and D for Don't advertise. As the actions are the same at each stage we do not have to superscript them with n.

We can represent this as a network:



All numbers are denominated in £1000s.
There are terminal rewards $r_1^{(0)} = 2$, $r_2^{(0)} = 5$ and $r_3^{(0)} = 10$.
The transition rewards are:

$$[r_{i,j}^{(1)}(A)]_{i,j=1,2,3} = \begin{bmatrix} -1 & 0 & 1 \\ 0 & 1 & 2 \\ 1 & 2 & 3 \end{bmatrix} \qquad [r_{i,j}^{(1)}(D)]_{i,j=1,2,3} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{bmatrix}$$

$$P^{(1)}(A) = \begin{bmatrix} .5 & .3 & .2 \\ .4 & .2 & .4 \\ .3 & .3 & .4 \end{bmatrix} \qquad P^{(1)}(D) = \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ .4 & .3 & .3 \\ .4 & .4 & .2 \end{bmatrix}$$

$$[r_{2,j}^{(2)}(A)]_{j=1,2,3} = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \qquad [r_{2,j}^{(2)}(D)]_{j=1,2,3} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

$$P^{(2)}(A) = \begin{bmatrix} .4 & .3 & .3 \end{bmatrix} \qquad P^{(2)}(D) = \begin{bmatrix} .6 & .3 & .1 \end{bmatrix}$$

The way to solve this problem is to set up a table for every state and possible action in each stage. For each action a and state i compute $R_n(i,a)$ and $\sum_{j\in\mathcal{Q}_{n-1}} p_{i,j}^{(n)}(a)\ V_{n-1}^*(j)$. This gives us $R_n(i,a) + \sum_{j\in\mathcal{Q}_{n-1}} p_{i,j}^{(n)}(a)\ V_{n-1}^*(j)$ and we choose the action that maximises this.

n	state i	action a	$R_n(i,a)$	$\sum_{j\in\mathcal{Q}_{n-1}} p_{i,j}^{(n)}(a)V_{n-1}^*(j)$	sum	$V_n^*(i)$	a^*
0	3	—	10	—	10	10	—
0	2	—	5	—	5	5	—
0	1	—	2	—	2	2	—
1	3	A	2.1	6.1	8.2		
1	3	D	3.8	4.8	8.6	8.6	D
1	2	A	1.0	5.8	6.8		
1	2	D	2.9	5.3	8.2	8.2	D
1	1	A	-0.3	4.5	4.2	4.2	A
1	1	D	1	2	3		
2	2	A	-0.1	6.72	6.62		
2	2	D	1.5	5.84	7.34	7.34	D

The last two columns tell us the optimal expected future reward and the action that should be taken in order to attain the optimum. The optimal actions are: don't advertise with two weeks to go; and with one week to go advertise only if the ticket sales are slow. The optimal expected future reward is £7 340.

Notes. To solve the problem, we need to find the optimal action(s) for every time point and every possible state.

In deterministic dynamic programming problems the choice is the path through the network. In Markov dynamic programming, we can not choose the path that is taken. Instead, we choose actions which influence the probabilities of which path is taken.

6.5. Discounting

Suppose a profit of £1 gained now is worth more than a profit £1 gained at the next stage. Reasons for this could be:

- inflation – by the next stage £1 will have less spending power
- investment – £1 can be invested and interest earned by the next stage.
- risk of bankruptcy – if this year's reward will help us stay in business then we should weight it higher than the reward in ten years' time, because without cash this year we might not be in business in ten years.

In order to account for this, we use a discount factor α, which compounds year on year. We shall assume that α is constant and $0 < \alpha < 1$. So an amount of £1 at the next stage is worth £α at "today's" prices.

The recursive equation for the optimal value function is now given by

$$V_n^*(i) = \max_{a^{(n)}} \left\{ R_n(i, a^{(n)}) + \alpha \sum_{j \in \mathcal{Q}_{n-1}} p_{i,j}^{(n)}(a^{(n)})\, V_{n-1}^*(j) \right\}$$

Exercise (from sheet 6 below). Repeat the ticket agency example using a discount factor of

1. $\alpha = 0.9$
2. $\alpha = 0.4$

and compare with the initial answer.

6.6. Infinite horizon problems

When the problem has no (obvious) final stage, we still want to find the optimal value function and the optimal actions.

We shall assume that there is a discount factor α ($0 < \alpha < 1$) and that the state space, action space and transition probabilities and rewards are the same for each stage. So, we can drop the subscript/superscript n.

If we imagine that there are a finite number of stages (and assign terminal values arbitrarily), then the recursion equation in this case becomes

$$V_n^*(i) = \max_{a \in \mathcal{A}} \left\{ R(i, a) + \alpha \sum_{j \in \mathcal{Q}} p_{ij}(a) V_{n-1}^*(j) \right\}.$$

If V_n^* converges, as $n \to \infty$, to an optimal value function V^*, then it must satisfy

$$\begin{aligned} V^*(i) &= \max_{a \in \mathcal{A}} \left\{ R(i, a) + \alpha \sum_{j \in \mathcal{Q}} p_{i,j}(a)\, V^*(j) \right\} \\ &= R\left(i, a^*(i)\right) + \alpha \sum_{j \in \mathcal{Q}} p_{i,j}\left(a^*(i)\right)\, V^*(j). \end{aligned}$$

where $a^*(i)$ denotes the optimal action when in state i.

In general, solving the above equation is difficult.

One approach is to guess a general form of the solution and solve it analytically. Another is to use iterative methods to obtain the optimal value function numerically.

Example 6.7 (Machine replacement)**.** A machine in a manufacturing plant can be in states $\{0, 1, 2, \dots\}$ according to the condition it is in. At the end of each week the machine is inspected, its condition is noted and we make a decision as to whether or not to replace it.

Let Y_t denote the state of the machine at the end of week t. If $Y_t = i$ and the action is "don't replace", we incur a running cost of $c(i) = i$ and

$$Y_{t+1} = \begin{cases} i & \text{with prob. } \frac{1}{2} \\ i+1 & \text{with prob. } \frac{1}{2} \end{cases}$$

If $Y_t = i$ and the action is "replace" we incur a fixed cost of K and

$$Y_{t+1} = \begin{cases} 0 & \text{with prob. } \frac{1}{2} \\ 1 & \text{with prob. } \frac{1}{2} \end{cases}$$

The costs are discounted by a factor α over an infinite horizon.

Thus

- $\mathcal{Q} = \{0, 1, 2, \dots\}$
- $\mathcal{A} = \{1 = \text{"replace"}, 2 = \text{"don't replace"}\}$
- $p_{i,0}(1) = \frac{1}{2}$
- $p_{i,1}(1) = \frac{1}{2}$
- $p_{i,i}(2) = \frac{1}{2}$
- $p_{i,i+1}(2) = \frac{1}{2}$
- $r_{ij}(1) = -K$, for $j = 0, 1$
- $r_{ij}(2) = -i$, for $j = i, i+1$

A **policy** $\pi(i)$ is a decision rule (or action plan) that specifies an action for each state i in the state space $\mathcal{Q}$.

We might guess that the optimal policy is of the form: choose $a = 1$ (replace) if $i \geq M$ and $a = 2$ (don't replace) if $i < M$, for some unknown M.

The above policy is $\pi(i) = 2$ for $0 \leq i \leq M-1$ and $\pi(i) = 1$ for $i \geq M$ and the value function corresponding to this policy, $V_\pi(i)$, is

$$\begin{aligned} V_\pi(i) &= R(i, \pi(i)) + \alpha \sum_j p_{i,j}(\pi(i))\, V_\pi(j) \\ &= \begin{cases} -i + \alpha \left[\frac{1}{2} V_\pi(i) + \frac{1}{2} V_\pi(i+1)\right] & \text{if } 0 \leq i \leq M-1 \\ -K + \alpha \left[\frac{1}{2} V_\pi(0) + \frac{1}{2} V_\pi(1)\right] & \text{if } i \geq M \end{cases} \end{aligned}$$

Analytical methods can be used to show that this policy is optimal and to find the values of $V^*(i)$ and M, but this is not covered in this course. Instead we shall use an iterative method.

For this iterative method, pretend that the number of stages is finite. Let $V_n^*(i)$ be the expected future (discounted) reward when there are n stages to go and we use the optimal policy. Then

$$\begin{aligned} V_0^*(i) &= 0 \quad \forall i \in \mathbb{N} \\ V_n^*(i) &= \max\{U_n(i,1),\, U_n(i,2)\} \quad \forall i \in \mathbb{N} \end{aligned}$$

where

$$\begin{aligned} U_n(i,1) &= -K + \alpha \left[\frac{1}{2} V_{n-1}^*(0) + \frac{1}{2} V_{n-1}^*(1)\right] \\ U_n(i,2) &= -i + \alpha \left[\frac{1}{2} V_{n-1}^*(i) + \frac{1}{2} V_{n-1}^*(i+1)\right]. \end{aligned}$$

As we have said, if $V_n^*(i)$ converges to $V^*(i)$ as $n \to \infty$, then $V^*(i)$ satisfies

$$V^*(i) = \max_{a \in \mathcal{A}} \left\{ R(i,a) + \alpha \sum_{j \in \mathcal{Q}} p_{i,j}(a) V^*(j) \right\}.$$

We check this convergence; the iterative procedure will also give us the optimal policy.

If we now assume specific values for K and α of 10 and 0.75 respectively, and run the above iterative method we get:

State (i)	Iteration Number (n)									
	1		2		3		38		39	
i	a^*	$V_1^*(i)$	a^*	$V_2^*(i)$	a^*	$V_3^*(i)$	a^*	$V_{38}^*(i)$	a^*	$V_{39}^*(i)$
0	2	-0	2	-0.375	2	-0.938	2	-5.106	2	-5.106
1	2	-1	2	-2.125	2	-3.250	2	-8.511	2	-8.511
2	2	-2	2	-3.875	2	-5.562	2	-11.518	2	-11.518
3	2	-3	2	-5.625	2	-7.875	2	-13.864	2	-13.864
4	2	-4	2	-7.375	2	-10.188	1	-15.106	1	-15.106
5	2	-5	2	-9.125	1	-10.938	1	-15.106	1	-15.106
6	2	-6	1	-10.375	1	-10.938	1	-15.106	1	-15.106
7	2	-7	1	-10.375	1	-10.938	1	-15.106	1	-15.106
8	2	-8	1	-10.375	1	-10.938	1	-15.106	1	-15.106
9	2	-9	1	-10.375	1	-10.938	1	-15.106	1	-15.106
10							1	-15.106	1	-15.106
11							1	-15.106	1	-15.106

The function $V_n^*(i)$ converges (but needs 38 iterations to converge to 3 d.p.s). The convergence for such problems in general is often very slow.

Notice that the solution is of the form postulated above: Replace if $i \geq 4$ and continue if $i \leq 3$. The optimal value function is given in the rightmost column (with $V^*(i) = -15.106$ for all $i \geq 4$).

6.7. Optimal stopping problems

An optimal stopping problem, or optimal stopping routine, is a special type of stochastic optimisation problem in which, at every stage, there are two possible actions, **stop** and **continue**. If the action is 'stop' then a reward is obtained, depending on the current state, and that is the end of the routine. If the action is 'continue' then a cost is incurred and the routine proceeds to the next stage, according to a Markov chain.

The routines we shall consider have the following form:

- There are a finite number of stages and no discounting.
- The reward for stopping in state i before the final stage is $r(i)$. This is independent of stage.
- The reward for continuing to the final stage is $r^{(0)}(i)$. The cost of continuing is $c(i)$.
- It depends on the state but not the stage.
- If continuing, the probability that the next state is j given that the current state is i with n stages to go is $p_{ij}^{(n)}$.

The optimal value function is

$$V_n^*(i) = \max\left\{ r(i),\ -c(i) + \sum_{j \in \mathcal{Q}_{n-1}} p_{i,j}^{(n)} V_{n-1}^*(j) \right\} \quad (n \geq 1)$$

and

$$V_0^*(i) = r^{(0)}(i)$$

Example: TV Game Show

A contestant has N boxes each containing a random amount of money from £0 to £1000. All amounts are equally likely, and the amount of money in each box is independent of the amount of money in the others.

The game-show host opens the first box and the contestant has to choose whether to keep the money, in which case the game is over ("stop") or to move on to the next box. Once the contestant has shut a box they can no longer claim the money in that box.

Given that the number of boxes is $N = 4$, what is the optimal strategy that the contestant should follow?

- Stages are the boxes.
- n is the number of stages left, i.e., the number of boxes still to be accepted or rejected.
- For $n \geq 1$, state space $\mathcal{Q}_n$ = all possible values in box. The amounts of money are in pounds, so $\mathcal{Q}_n = \{0, 1, \ldots, 999, 1000\}$.
- The rewards are the amounts in the box (i.e. the states) $r(i) = i$.
- The terminal reward is nothing: $r^{(0)}(i) = 0$.
- The transition costs are zero: $c(i) = 0$.
- For $n \geq 2$, the transition probabilities are $p_{ij}^{(n)} = 1/1001$.
- $\mathcal{Q}_0 = \{0\}$ and $p_{i0}^{(1)} = 1$.

$n = 0$ Once all of the boxes have been rejected:

$$V_0^*(0) = r^{(0)}(0) = 0.$$

$n = 1$ This corresponds to deciding whether to accept or reject the last of the four boxes.

$$\begin{aligned} V_1^*(i) &= \max\{i, V_0^*(0)\} \\ &= \max\{i, 0\} \end{aligned}$$

So the optimal policy is to accept the box no matter how much money is inside.

$n = 2$ When the contestant decides on the third box:

$$V_2^*(i) = \max\left\{ i, \sum_{j=0}^{1000} p_{ij}^{(2)} V_1^*(j) \right\}$$

$$\begin{aligned} \sum_{j=0}^{1000} p_{ij}^{(2)} V_1^*(j) &= \frac{1}{1001} \sum_{j=0}^{1000} j \\ &= \frac{1}{1001}\left(\frac{1000 \times 1001}{2}\right) = 500 \end{aligned}$$

giving $V_2^*(i) = \max\{i, 500\}$.

So the optimal policy is to take the money if and only the amount in the box is at least £500.

$n = 3$ When the contestant opens the second box:

$$V_3^*(i) = \max\left\{ i, \sum_{j=0}^{1000} p_{ij}^{(3)} V_2^*(j) \right\}$$

and

$$\begin{aligned}\sum_{j=0}^{1000} p_{ij}^{(3)} V_2^*(j) &= \frac{1}{1001}\left(\sum_{j=0}^{500} 500 + \sum_{j=501}^{1000} j\right) \\ &= 625.125.\end{aligned}$$

Hence,

$$V_3^*(i) = \max\{i, 625.125\}$$

So the optimal policy is to take the money if and only if the amount in the box is at least £626.

$n = 4$ When the contestant sees the contents of the first box:

$$\begin{aligned} V_4^*(i) &= \max\left\{ i, \sum_{j=0}^{1000} p_{ij}^{(4)} V_3^*(j) \right\} \\ \sum_{j=0}^{1000} p_{ij}^{(4)} V_3^*(j) &= \frac{1}{1001}\left(\sum_{j=0}^{625} 625.125 + \sum_{j=626}^{1000} j\right) \\ &= 695.52 \\ V_4^*(i) &= \max\{i, 695.52\} \end{aligned}$$

So the optimal policy is to take the money if and only if the amount in the box is at least £696.

The complete optimal strategy is for the contestant to accept a box if and only if it contains at least the amount shown

Box number	1	2	3	4
	$n = 4$	$n = 3$	$n = 2$	$n = 1$
Amount	696	626	500	0

A. Reading List

This course covers a range of topics that are not usually covered all in one book. The following list is for further reading, and the ones with asterisks are more relevant than the others.

* F.S. Hillier & G.J. Lieberman *Introduction to Operations Research* (2005, McGraw Hill).

* B.Kolman & R.E.Beck, *Elementary Linear Programming with Applications* (1980, Academic Press).

D. Bertsimas & J. Tsitsiklis *Introduction to Linear Optimization* (1997, Athena Scientific).

G. Gordon & I. Pressman *Quantitative Decision-Making for Business* (1978, Prentice Hall).

Wilkes, M. *Operational Research Analysis and Applications* (1989, McGraw-Hill)

A. K. Dixit & S. Skeath, *Games of Strategy* (1999, Norton).

K.G. Binmore, *Fun and Games* (1992, Houghton Mifflin).

R. Gibbons *A Primer in Game Theory* (1992, Pearson Education).

* W. L. Winston *Operations Research : Applications and Algorithms* (1994, Duxbury Press).

S. M. Ross *Introduction to Stochastic Dynamic Programming* (1983, Academic Press).

* J.Bather, *Decision Theory: An Introduction to Dynamic Programming and Sequential Decisions* (2000, Wiley).

B. Corrections

A list of all major corrections made to this file since it first appeared on Moodle:

- No corrections made yet!