

# REVISION SHEET – DECISION MATHS 2

## DECISION ANALYSIS

The main ideas are covered in

|            |                |            |            |
|------------|----------------|------------|------------|
| <b>AQA</b> | <b>Edexcel</b> | <b>MEI</b> | <b>OCR</b> |
|            |                | <b>D2</b>  |            |

*Before the exam you should know*

- The meaning of the different kinds of node .
- Be able to construct a decision tree and use it to calculate the **EMV**
- Appreciate that decision analysis includes an element of uncertainty
- Understand what is meant by a **utility function** and be able to calculate utility in simple circumstances


### Decision Trees

A decision tree, like a probability tree, is a connected graph which is helpful for organising data and can be used to clarify and facilitate complex decision making.

The decision tree has three types of nodes

**Decision node** 

**Chance node** 

**Payoff node** 

At a decision node you select the best option and reject all the others. At a chance node, the value on the circle (which is the EMV at that point) is calculated from probabilities of the different outcomes.

### EMV – the expected monetary value

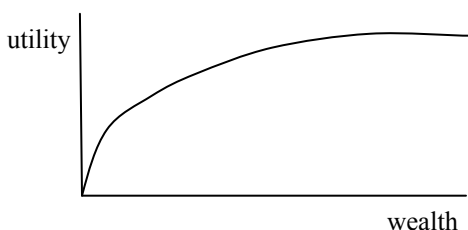
The EMV can be calculated in the same way as expectation in statistics. You do not need to have drawn a decision tree in order to calculate the EMV for a situation.

### Utility Functions

Utility is a function of wealth.

The utility function is specific to a person or situation and, in the exam, it will be given to you.

The Utility increases as wealth increases, but the rate of increase tends to slow down so the shape of a utility function is usually like this:



### Example

**Julia is trying to decide whether to invest £10000 in a small business. A survey indicates that if the business succeeds she will make a 50% profit in the first year, however if it fails she will lose her investment. The probability of failure is estimated to be 25%.**

**(a) What is her EMV if she decides to invest?**

**For £2000, she can hire a consultant to do a business analysis. This would increase the chance of her choosing a successful business to 80%.**

**(b) Assuming that, if she decides to hire the consultant she will pay £2000 in addition to the original £10000, draw a decision tree showing all the options open to Julia. What would you advise her to do?**

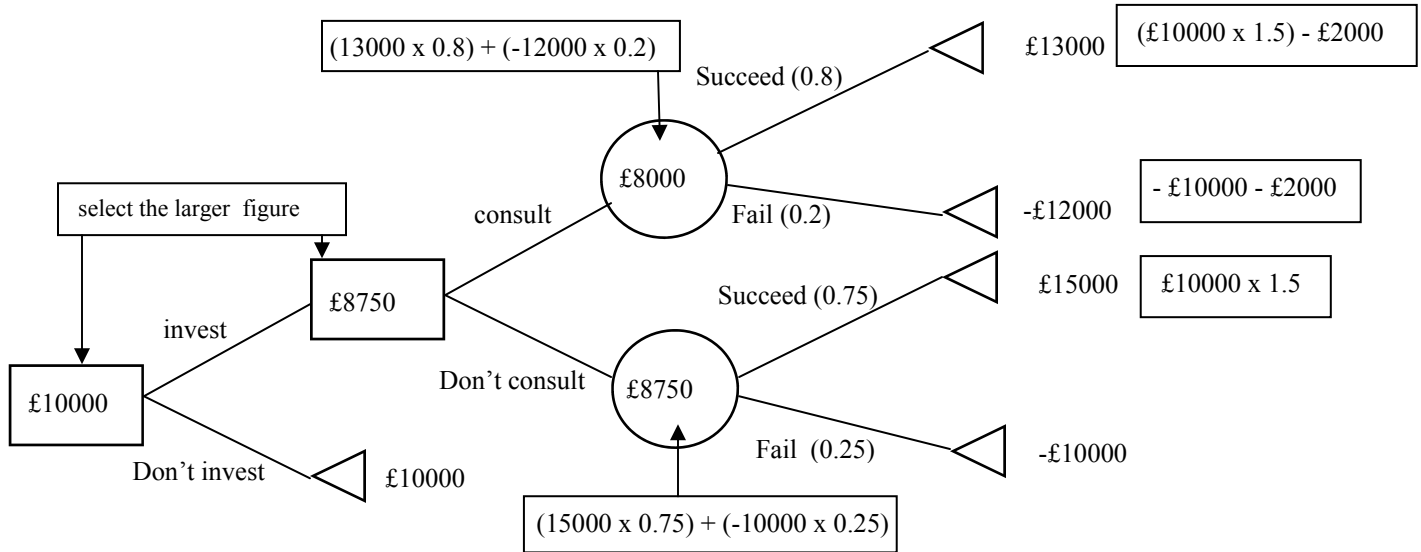
**(c) If Julia is determined to invest in a business, how much should Julia be prepared to pay the consultant in order to make it worth hiring him?**

**(d) If the utility function for this investment is  $\sqrt{(\text{wealth})}$ . Calculate EMV using the utility function and comment on whether this is likely to change the advice in (c).**

**Solution**

(a)  $EMV = (£10000 \times 1.5 \times 0.75) + (-£10000 \times 0.25) = £11250 - £2500 = £8750$

(b)



Advise Julia not to invest.

(c) For it to be worth hiring the consultant, the EMV for “consult” must be greater than £8750.

Let the consultancy fee be C, then

$$\Rightarrow 0.8(10000 \times 1.5 - C) + 0.2(-10000 - C) \geq 8750$$

$$\Rightarrow 12000 - 0.8C - 2000 - 0.2C \geq 8750$$

$$\Rightarrow 10000 - C \geq 8750$$

$$\Rightarrow C \leq £1250$$

It is only worth hiring the consultant if the fees are £1250 or less.

(d) Utility =  $\sqrt{\text{wealth}}$

If she consults then  $EMV = (\sqrt{13000}) \times 0.8 - (\sqrt{12000}) \times 0.2 = 69.3$  (3sf)

If she does not consult then  $EMV = (\sqrt{15000}) \times 0.75 - (\sqrt{10000}) \times 0.25 = 66.9$  (3sf)

Julia is advised to consult.

# REVISION SHEET – DECISION MATHS 2

## NETWORKS: FLOYD’S ALGORITHM

**The main ideas are covered in**

|     |         |     |           |
|-----|---------|-----|-----------|
| AQA | Edexcel | MEI | OCR       |
|     |         |     | <b>D2</b> |

**The main idea in this chapter is:**

**Using Floyd’s algorithm to find all the shortest paths between all pairs of vertices in a network.**

The method successively modifies two matrices:

- A distance matrix which gives the weight of each edge in the network.
- A route matrix.

### The Algorithm

1. Choose the first (or next in the subsequent iterations) row and column, and note the iteration number. Highlight all elements in the chosen row and column and in the corresponding column in the route matrix.
2. Compare each entry that is not highlighted in the distance matrix with the sum of the corresponding entries in the chosen row and column.
3. If the sum is smaller than the element then replace the element by the sum.
4. If the element is replaced then we need to update the corresponding entry of the route matrix. The value to replace it with is found in the same row as the entry in the highlighted column.
5. Repeat steps 1 to 4 until all rows/columns have been chosen.
6. The distance matrix will now contain the shortest distances. The route matrix shows the next vertex “en route” from one to another along a shortest path. It can be used to construct complete shortest paths.

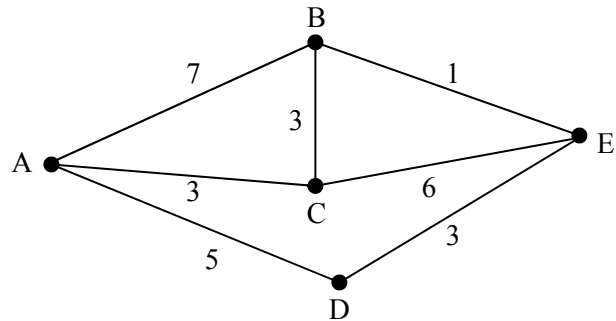
### Before the exam you should know:

- The distance matrix is symmetric unless we are dealing with a digraph.
- The distance from a vertex to itself is shown as 0.
- Where there is no direct connection (i.e. no edge) the distance is shown as  $\infty$ .
- You must perform one iteration for each vertex.
- Sometimes there are no changes on an iteration. This does not mean that the algorithm is complete.
- When you update the route matrix, remember to replace it with the entry that is in the highlighted column and the same row.

### Example:

Use Floyd’s algorithm to find all the shortest routes between vertices in this network.

Give the shortest route from C to D and state it’s length.



### Solution:

Initial Distance Matrix ( $D^0$ )  
destination

|   |          |          |          |          |          |
|---|----------|----------|----------|----------|----------|
|   | A        | B        | C        | D        | E        |
| A | $\infty$ | 7        | 3        | 5        | $\infty$ |
| B | 7        | $\infty$ | 3        | $\infty$ | 1        |
| C | 3        | 3        | $\infty$ | $\infty$ | 6        |
| D | 5        | $\infty$ | $\infty$ | $\infty$ | 3        |
| E | $\infty$ | 1        | 6        | 3        | $\infty$ |

start

Initial Route Matrix ( $R^0$ )  
destination

|   |   |   |   |   |   |
|---|---|---|---|---|---|
|   | A | B | C | D | E |
| A | A | B | C | D | E |
| B | A | B | C | D | E |
| C | A | B | C | D | E |
| D | A | B | C | D | E |
| E | A | B | C | D | E |

These numbers represent distances by single stage routes. The highlighting is ready for the first iteration.

These numbers represent the first vertex on corresponding single stage routes, so the first vertex encountered on the single arc route from 5 to 4 is 4

Via A

|   | A | B  | C | D  | E |
|---|---|----|---|----|---|
| A | ∞ | 7  | 3 | 5  | ∞ |
| B | 7 | 14 | 3 | 12 | 1 |
| C | 3 | 3  | 6 | 8  | 6 |
| D | 5 | 12 | 8 | 10 | 3 |
| E | ∞ | 1  | 6 | 3  | ∞ |

|   | A | B | C | D | E |
|---|---|---|---|---|---|
| A | A | B | C | D | E |
| B | A | A | C | A | E |
| C | A | B | A | A | E |
| D | A | A | A | A | E |
| E | A | B | C | D | E |

Look at the highlighted row and column:  
 $7 + 7 = 14$

The value to replace it with is found in the same row as the entry in the highlighted column

Via B

|   | A  | B  | C | D  | E |
|---|----|----|---|----|---|
| A | 14 | 7  | 3 | 5  | 8 |
| B | 7  | 14 | 3 | 12 | 1 |
| C | 3  | 3  | 6 | 8  | 4 |
| D | 5  | 12 | 8 | 10 | 3 |
| E | 8  | 1  | 4 | 3  | 2 |

|   | A | B | C | D | E |
|---|---|---|---|---|---|
| A | B | B | C | D | B |
| B | A | A | C | A | E |
| C | A | B | A | A | B |
| D | A | A | A | A | E |
| E | B | B | B | D | B |

$1 + 3 = 4; 4 < 6$

Via C

|   | A | B  | C | D  | E |
|---|---|----|---|----|---|
| A | 6 | 6  | 3 | 5  | 7 |
| B | 6 | 6  | 3 | 11 | 1 |
| C | 3 | 3  | 6 | 8  | 4 |
| D | 5 | 11 | 8 | 10 | 3 |
| E | 7 | 1  | 4 | 3  | 2 |

|   | A | B | C | D | E |
|---|---|---|---|---|---|
| A | C | C | C | D | C |
| B | C | C | C | C | E |
| C | A | B | A | A | B |
| D | A | A | A | A | E |
| E | B | B | B | D | B |

This shows that there is a path of length 7 from vertex E to vertex A

The value in the highlighted column is A, so replace with A

The route matrix shows that the first vertex on the route from E to A is vertex B

Via D

|   | A | B  | C | D  | E |
|---|---|----|---|----|---|
| A | 6 | 6  | 3 | 5  | 7 |
| B | 6 | 6  | 3 | 11 | 1 |
| C | 3 | 3  | 6 | 8  | 4 |
| D | 5 | 11 | 8 | 10 | 3 |
| E | 7 | 1  | 4 | 3  | 2 |

|   | A | B | C | D | E |
|---|---|---|---|---|---|
| A | C | C | C | D | C |
| B | C | C | C | C | E |
| C | A | B | A | A | B |
| D | A | A | A | A | E |
| E | B | B | B | D | B |

There are no changes on this iteration

Via E

|   | A | B | C | D | E |
|---|---|---|---|---|---|
| A | 6 | 6 | 3 | 5 | 7 |
| B | 6 | 2 | 3 | 4 | 1 |
| C | 3 | 3 | 6 | 7 | 4 |
| D | 5 | 4 | 7 | 6 | 3 |
| E | 7 | 1 | 4 | 3 | 2 |

|   | A | B | C | D | E |
|---|---|---|---|---|---|
| A | C | C | C | D | C |
| B | C | E | C | E | E |
| C | A | B | A | B | B |
| D | A | E | E | E | E |
| E | B | B | B | D | B |

The final matrix D<sup>(5)</sup> shows all the shortest distances, so the shortest distance from vertex C to vertex D is 7. The R<sup>(5)</sup> matrix shows that the first vertex on the route is vertex B. Now use the R<sup>(5)</sup> matrix again to find the first vertex on the shortest route from vertex B to vertex D. This is vertex E, so the route is now seen to be C, B, E, .... Finally the first vertex on the shortest route from vertex E to vertex D is vertex D. **So the final route is C,B,E,D of length 7.**

# REVISION SHEET – DECISION MATHS 2

## LOGIC AND BOOLEAN ALGEBRA

| The main ideas are covered in |         |     |     |
|-------------------------------|---------|-----|-----|
| AQA                           | Edexcel | MEI | OCR |
|                               |         | D2  |     |

**Before the exam you should know:**

- The symbols for conjunction (and)  $\wedge$ , disjunction (or)  $\vee$ , condition (implies)  $\Rightarrow$ , double condition (iff)  $\Leftrightarrow$ .
- The truth tables  $\wedge$ ,  $\vee$ ,  $\Rightarrow$ ,  $\Leftarrow$  and  $\Leftrightarrow$ .
- The laws of Boolean algebra.
- The symbols for logic gates.
- How to draw and interpret a simple switching circuit.

### Propositional Calculus

$p$  and  $q$  are propositions (statements) that can be true or false. Compound propositions are made using logical operators (or connectives)

$\sim p$ : not  $p$

$p \wedge q$ : Conjunction;  $p$  and  $q$

$p \vee q$ : disjunction;  $p$  or  $q$

$p \Leftrightarrow q$ : equivalence;  $p$  if and only if  $q$

$p \Rightarrow q$ : implication;  $p$  implies  $q$

$p \Leftarrow q$ : sufficiency;  $q$  is sufficient for  $p$

If the statement is false then it is given the number 0, but if it is true then it is given the number 1. This can be used to construct truth tables.

**Example:** Let  $p$  be the proposition ‘Jim studies Maths’  
 Let  $q$  be the proposition ‘Tim studies History’  
 Let  $r$  be the proposition ‘Tim studies Physics’

(a) use logical operators to write the proposition ‘Tim studies Physics if and only if he studies Maths and History’  
 (b) write in words  $p \wedge \sim(q \vee r)$

**Solution:** (a)  $r \Leftrightarrow (p \wedge q)$   
 (b) Tim studies maths but not history or physics

### Truth tables

#### NOT

| $p$ | $\sim p$ |
|-----|----------|
| 1   | 0        |
| 1   | 0        |
| 0   | 1        |
| 0   | 1        |

#### AND

| $p$ | $q$ | $p \wedge q$ |
|-----|-----|--------------|
| 1   | 1   | 1            |
| 1   | 0   | 0            |
| 0   | 1   | 0            |
| 0   | 0   | 0            |

#### OR

| $p$ | $q$ | $p \vee q$ |
|-----|-----|------------|
| 1   | 1   | 1          |
| 1   | 0   | 1          |
| 0   | 1   | 1          |
| 0   | 0   | 0          |

#### IMPLIES

| $p$ | $q$ | $p \Rightarrow q$ |
|-----|-----|-------------------|
| 1   | 1   | 1                 |
| 1   | 0   | 0                 |
| 0   | 1   | 1                 |
| 0   | 0   | 1                 |

#### IMPLIED BY

| $p$ | $q$ | $p \Leftarrow q$ |
|-----|-----|------------------|
| 1   | 1   | 1                |
| 1   | 0   | 1                |
| 0   | 1   | 0                |
| 0   | 0   | 1                |

#### EQUIVALENCE

| $p$ | $q$ | $p \Leftrightarrow q$ |
|-----|-----|-----------------------|
| 1   | 1   | 1                     |
| 1   | 0   | 0                     |
| 0   | 1   | 0                     |
| 0   | 0   | 1                     |

**Example: Prove**  $a \wedge (b \vee c) \Leftrightarrow (a \wedge b) \vee (a \wedge c)$

LHS

RHS

Alternatively in nested form

| $a$ | $b$ | $c$ | $b \vee c$ | $a \wedge (b \vee c)$ |
|-----|-----|-----|------------|-----------------------|
| 1   | 1   | 1   | 1          | 1                     |
| 1   | 1   | 0   | 1          | 1                     |
| 1   | 0   | 1   | 1          | 1                     |
| 1   | 0   | 0   | 0          | 0                     |
| 0   | 1   | 1   | 1          | 0                     |
| 0   | 1   | 0   | 1          | 0                     |
| 0   | 0   | 1   | 1          | 0                     |
| 0   | 0   | 0   | 0          | 0                     |

| $a$ | $b$ | $c$ | $a \wedge b$ | $a \wedge c$ | $(a \wedge b) \vee (a \wedge c)$ |
|-----|-----|-----|--------------|--------------|----------------------------------|
| 1   | 1   | 1   | 1            | 1            | 1                                |
| 1   | 1   | 0   | 1            | 0            | 1                                |
| 1   | 0   | 1   | 0            | 1            | 1                                |
| 1   | 0   | 0   | 0            | 0            | 0                                |
| 0   | 1   | 1   | 0            | 0            | 0                                |
| 0   | 1   | 0   | 0            | 0            | 0                                |
| 0   | 0   | 1   | 0            | 0            | 0                                |
| 0   | 0   | 0   | 0            | 0            | 0                                |

| $a$ | $\wedge$ | $(b$ | $\vee$ | $c)$ | $\Leftrightarrow$ | $(a$ | $\wedge$ | $b)$ | $\vee$ | $(a$ | $\wedge$ | $c)$ |
|-----|----------|------|--------|------|-------------------|------|----------|------|--------|------|----------|------|
| 1   | 1        | 1    | 1      | 1    | 1                 | 1    | 1        | 1    | 1      | 1    | 1        | 1    |
| 1   | 1        | 1    | 1      | 0    | 1                 | 1    | 1        | 1    | 1      | 1    | 0        | 0    |
| 1   | 1        | 0    | 1      | 1    | 1                 | 1    | 0        | 0    | 1      | 1    | 1        | 1    |
| 1   | 0        | 0    | 0      | 0    | 1                 | 1    | 0        | 0    | 0      | 1    | 0        | 0    |
| 0   | 0        | 1    | 1      | 1    | 1                 | 0    | 0        | 1    | 0      | 0    | 0        | 1    |
| 0   | 0        | 1    | 1      | 0    | 1                 | 0    | 0        | 1    | 0      | 0    | 0        | 0    |
| 0   | 0        | 0    | 1      | 1    | 1                 | 0    | 0        | 0    | 0      | 0    | 0        | 1    |
| 0   | 0        | 0    | 0      | 0    | 1                 | 0    | 0        | 0    | 0      | 0    | 0        | 0    |

## The Laws of Boolean Algebra

|  |                    |
|--|--------------------|
| $p \vee \sim p = 1$<br>$p \wedge \sim p = 0$   | Complement laws    |
| $\sim(\sim p) = p$   | Double negation    |
| $p \vee 0 = p$ $p \wedge 1 = p$<br>$p \vee 1 = 1$ $p \wedge 0 = 0$   | Identity rules     |
| $p \vee q = q \vee p$<br>$p \wedge q = q \wedge p$   | Commutative rules  |
| $p \vee (q \wedge r) = (p \vee q) \wedge (p \vee r)$<br>$p \wedge (q \vee r) = (p \wedge q) \vee (p \wedge r)$ | Distributive rules |
| $(p \vee q) \vee r = p \vee (q \vee r)$<br>$p \wedge (q \wedge r) = p \wedge (q \wedge r)$                     | Associative rules  |
| $p \vee p = p$ $p \wedge p = p$<br>$p \wedge (p \vee q) = p$<br>$p \vee (p \wedge q) = p$                      | Absorption rules   |
| $\sim(p \vee q) = \sim p \wedge \sim q$<br>$\sim(p \wedge q) = \sim p \vee \sim q$                             | De Morgan's rules  |

### Example: Simplify $p \vee (\sim p \wedge q)$

$$\begin{aligned}
 p \vee (\sim p \wedge q) &= (p \vee \sim p) \wedge (p \vee q) && \text{[distributive rule]} \\
 &= 1 \wedge (p \vee q) && \text{[complement law]} \\
 &= p \vee q && \text{[identity]}
 \end{aligned}$$

### Switching Circuits:

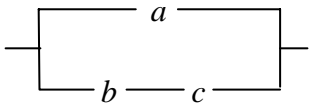
Wiring switches in series gives a conjunction (and).  
Wiring switches in parallel gives a disjunction (or).

### Logic Gates

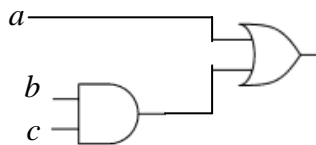
You need to be able to apply your knowledge of truth tables and circuits to simple circuits involving logic gates. Circuits drawn with these symbols are sometimes called **combinatorial circuits**

The diagrams both show the circuit  $a \vee (b \wedge c)$

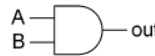
switching circuit



combinatorial circuit



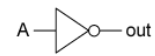
AND



OR



NOT



You may also encounter

NAND (not and)

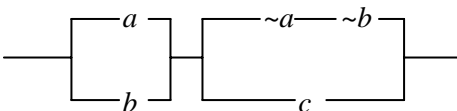


NOR (not or)



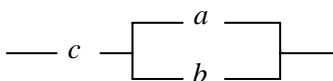
### Example:

- (a) Write a Boolean expression for this switching circuit



- (b) Draw the combinatorial circuit which is equivalent to the circuit shown in (a)

- (c) This diagram shows another circuit.

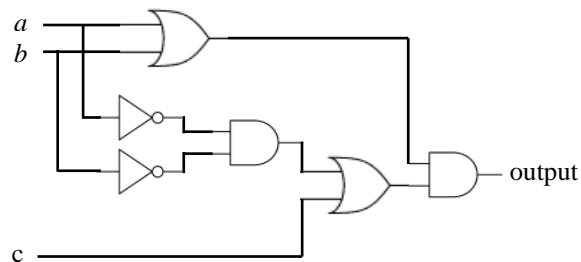


Use Boolean algebra to show analytically that the two circuits are equivalent.

### Solution:

- (a) The circuit is  $(a \vee b) \wedge [(\sim a \wedge \sim b) \vee c]$

- (b)



$$\begin{aligned}
 &(c) \quad (a \vee b) \wedge [(\sim a \wedge \sim b) \vee c] \\
 &= [(a \vee b) \wedge (\sim a \wedge \sim b)] \vee [(a \vee b) \wedge c] && \text{[distributive rule]} \\
 &= [(a \wedge \sim a) \wedge (a \wedge \sim b) \vee (b \wedge \sim a) \wedge (b \wedge \sim b)] \vee [(a \vee b) \wedge c] && \text{[distributive rule]} \\
 &= [0 \wedge (a \wedge \sim b)] \vee [(b \wedge \sim a) \wedge 0] \vee [(a \vee b) \wedge c] && \text{[complement rule]} \\
 &= 0 \vee [(a \vee b) \wedge c] && \text{[identity rule]} \\
 &= (a \vee b) \wedge c && \text{[identity rule]}
 \end{aligned}$$

This is the Boolean expression for the circuit shown

## REVISION SHEET – DECISION MATHS 2

### LINEAR PROGRAMMING: THE SIMPLEX ALGORITHM 1

The main ideas are covered in

|            |                |            |            |
|------------|----------------|------------|------------|
| <b>AQA</b> | <b>Edexcel</b> | <b>MEI</b> | <b>OCR</b> |
| <b>D2</b>  | <b>D1</b>      | <b>D2</b>  | <b>D1</b>  |

The main idea in this chapter is:

Using the Simplex algorithm to deal with linear programming problems with more than two variables.

Before the exam you should be able to:

- Formulate a linear programming problem to maximise the objective function, subject to the given constraints
- Use slack variables to convert inequality constraints into equations
- Set up the initial simplex tableau
- Perform the Simplex algorithm for maximising an objective function
- Identify initial, intermediate and final tableaux and know when the solution is optimal
- Interpret the values of the variables and the objective function at any stage in the Simplex method
- Clearly state the solution to the original problem

### Simplex Method for Maximisation Problems

#### Getting started: Formulation

Translating a real life problem into a linear programming problem is called formulating the problem and is an example of mathematical modelling. Each problem must have clearly defined variables, an objective function and is subject to certain constraints

#### Slack Variables

In order to enable problems to be converted into a format that can be dealt with by computer, slack variables are introduced to change the constraint inequalities into equalities. Each vertex of the feasible region would then be defined by the intersection of lines where some of these variables equal zero.

#### The Simplex Method

The Simplex Method starts at one vertex and systematically moves round all the vertices of the feasible region, increasing the objective function as it goes, until it reaches the one with the optimal solution. This is easy to visualise on a 2 dimensional problem, but can be generalised to include more variables. Once there are more than two variables, a graphical approach is no longer appropriate, so we use the simplex tableau, a tabular form of the algorithm which uses row reduction to solve the problem.

#### The Simplex Algorithm

1. Represent the problem in a tableau
2. Use the objective row to find the pivot column
3. Use the ratio test to find the pivot element
4. Divide through the pivot row by the pivot element
5. Add/subtract multiples of the transformed pivot row to/from the other rows to create zeros in the pivot column
6. Repeat until no negatives in objective row
7. Read the solution from the table

#### Note on finding pivot column (step 2)

You can choose any variable in objective row with negative coefficient, but it is usual to pick the most negative. Give priority is to the original rather than slack variables.

#### Note on ratio test (step 3)

Divide each R.H.S. value by the corresponding element in the pivot column, ignore negative ratios and division by zero. Choose row with the smallest ratio as the pivot row.

**Example:**

A manufacturer makes three products  $x$ ,  $y$  and  $z$  which give profits of £160, £120 and £120 per tonne respectively. Production is constrained by availability of staff and storage as summarised in this table:

|              | Staff time<br>(hours /tonne) | Storage<br>(m <sup>3</sup> /tonne) |
|--------------|------------------------------|------------------------------------|
| $x$          | 5                            | 5                                  |
| $y$          | 5                            | 3                                  |
| $z$          | 6                            | 4                                  |
| availability | 30                           | 20                                 |

Formulate a linear programming problem.

**Objective function**

Objective function is maximise  $P = 160x + 120y + 120z$   
 $\Rightarrow P - 160x - 120y - 120z = 0$

**Constraints**

Subject to:  $5x + 5y + 6z + s_1 = 30$  (staff time)  
 $5x + 3y + 4z + s_2 = 20$  (storage)  
 $x \geq 0, y \geq 0, z \geq 0, s_1 \geq 0, s_2 \geq 0$

**Solving the problem**

Considering the problem in the example on the previous page, we must now set up an initial tableau

**Setting up the initial tableau**

| $P$ | $x$  | $y$  | $z$  | $s_1$ | $s_2$ | RHS | <i>Ratio test</i> |
|-----|------|------|------|-------|-------|-----|-------------------|
| 1   | -160 | -120 | -120 | 0     | 0     | 0   |                   |
| 0   | 5    | 5    | 6    | 1     | 0     | 30  | 30/5 = 6          |
| 0   | 5    | 3    | 4    | 0     | 1     | 20  | 20/5 = 4          |

Look for the largest negative value in the objective row. This is in the  $x$  column, so we will choose to increase  $x$  first. Since  $4 < 5$  we choose 5 as the pivot element. The idea now is to make the pivot 1 so divide the pivot row by 5.

**First iteration**

|                     | $P$ | $x$ | $y$ | $z$ | $s_1$ | $s_2$ | RHS | <i>Ratio test</i> |
|---------------------|-----|-----|-----|-----|-------|-------|-----|-------------------|
| row 1 + 160 × row 3 | 1   | 0   | -24 | 8   | 0     | 32    | 640 |                   |
| row 2 - 5 × row 3   | 0   | 0   | 2   | 2   | 1     | -1    | 10  | 10/2 = 5          |
| row 3               | 0   | 1   | 0.6 | 0.8 | 0     | 0.2   | 4   | 4/0.4 = 10        |

Look for the largest negative value in the objective row. This is in the  $y$  column, so we will choose to increase  $y$ . Since  $5 < 10$  we choose 2 as the pivot element. The idea is to make the pivot 1 so divide the pivot row by 2.

**Second iteration**

|                     | $P$ | $x$ | $y$ | $z$ | $s_1$ | $s_2$ | RHS |
|---------------------|-----|-----|-----|-----|-------|-------|-----|
| row 1 + 24 × row 2  | 1   | 0   | 0   | 32  | 12    | 20    | 760 |
| row 2               | 0   | 0   | 1   | 1   | 0.5   | -0.5  | 5   |
| row 3 - 0.6 × row 2 | 0   | 1   | 0   | 0.2 | -0.3  | 0.5   | 1   |

The solution is optimal since there are no negative values in the objective row.

**Reading the tableau**

The final tableau represents the following set of equations

Row 1:  $P + 32z + s_1 + s_2 = 760$

Row 2:  $y + z + 0.5 s_1 - 0.5 s_2 = 5$

Row 3:  $x + 0.2z - 0.3 s_1 + 0.2 s_2 = 1$

The most obvious solution to this is obtained by setting the “basic” variables (columns with zeros and a single 1) equal to the RHS and setting the “non-basic” variables (columns with more than one non-zero entry) equal to 0.

This gives the solution  $P = 760, x = 1, y = 5, z = 0, s_1 = 0, s_2 = 0$

**Interpreting the solution**

In order to maximise his profit the manufacturer should make one tonne of product  $x$ , five tones of product  $y$  and no product  $z$ . this would use all the available resources and would generate a profit of £760.00

You can check your solution by substituting the values obtained for  $x$ ,  $y$  and  $z$  into the original objective function to check that the profit is correct:  
 $P = (160 \times 1) + (120 \times 5) + (120 \times 0)$   
 $= 160 + 600 + 0$   
 $= 760$



# REVISION SHEET – DECISION MATHS 2

## LINEAR PROGRAMMING: THE SIMPLEX ALGORITHM 2

The main ideas are covered in

|            |                |            |            |
|------------|----------------|------------|------------|
| <b>AQA</b> | <b>Edexcel</b> | <b>MEI</b> | <b>OCR</b> |
|            |                | <b>D2</b>  |            |

The main ideas in this chapter are:

Understand the use of the two-stage simplex and big M methods with more than two variables.  
 Construct an initial feasible solution to problems involving  $\geq$  constraints, including the possibility that there is no feasible solution.

Before the exam you should know:

- How to formulate a linear programming problem which contains one or more  $\geq$  constraints.
- What a surplus variable is and use surplus and slack variables to convert inequality constraints into equations.
- How to set up the initial simplex tableau.
- Perform the Simplex algorithm for maximising an objective function.
- Be able to identify initial, intermediate and final tableaux and know when the solution is optimal.
- Interpret the values of the variables and the objective function at any stage in the Simplex method.
- Be able to clearly state the solution to the original problem/

### Greater than ( $\geq$ ) inequalities:

These lead to problems which involve either maximisation or minimisation. In 2 dimensions these problems like this can be solved graphically but there are also adaptations of the Simplex algorithm to deal with this type of problem

### Two-Stage Simplex

The simplex algorithm relies on (0,0) being a feasible solution. If it is not, then you add artificial variables  $a_1, a_2$ , etc. to all  $\geq$  constraints which move us from (0,0) into the feasible region. You now need surplus (as opposed to slack) variables which are subtracted from the constraints as they tell us how much greater than the constraint line a point is. Then introduce a new objective function  $Q = a_1 + a_2 + \dots$  which you must minimise, since when  $Q = 0$  you are in the feasible region and you can complete the simplex in the normal way.

If the minimum value of the new objective is always greater than 0, then there is no feasible solution to the original problem.

**Example:** Maximise  $P = x + y$   
 Subject to  $2x + y \leq 16$   
 $2x + 3y \leq 24$   
 $x \geq 1, y \geq 1$

**Solution:** Add the slack variables to the  $\leq$  constraints:  $2x + y + s_1 = 16$   
 $2x + 3y + s_2 = 24$

Subtract the surplus variables from the  $\geq$  constraints and, since clearly  $x = y = 0$  is not a feasible solution add artificial variables:  $x - s_3 + a_1 = 1 \Rightarrow a_1 = 1 - x + s_3$   
 $y - s_4 + a_2 = 1 \Rightarrow a_2 = 1 - y + s_4$

The new objective  $Q = a_1 + a_2$  is written in terms of the other variables and rearranged  $Q + x + y - s_3 - s_4 = 2$ .

We then aim to minimise  $Q$  to ensure that we are in the feasible region before completing the Simplex in the normal way

| Q | P | x  | y  | s <sub>1</sub> | s <sub>2</sub> | s <sub>3</sub> | s <sub>4</sub> | a <sub>1</sub> | a <sub>2</sub> | RHS | Ratio test |
|---|---|----|----|----------------|----------------|----------------|----------------|----------------|----------------|-----|------------|
| 1 | 0 | 1  | 1  | 0              | 0              | -1             | -1             | 0              | 0              | 2   |            |
| 0 | 1 | -1 | -1 | 0              | 0              | 0              | 0              | 0              | 0              | 0   |            |
| 0 | 0 | 2  | 1  | 1              | 0              | 0              | 0              | 0              | 0              | 16  | 16/2=8     |
| 0 | 0 | 2  | 3  | 0              | 1              | 0              | 0              | 0              | 0              | 24  | 24/2=12    |
| 0 | 0 | 1  | 0  | 0              | 0              | -1             | 0              | 1              | 0              | 1   | 1/1=1      |
| 0 | 0 | 0  | 1  | 0              | 0              | 0              | -1             | 0              | 1              | 1   |            |
| 1 | 0 | 0  | 1  | 0              | 0              | 0              | -1             | -1             | 0              | 1   |            |
| 0 | 1 | 0  | -1 | 0              | 0              | -1             | 0              | 1              | 0              | 1   |            |
| 0 | 0 | 0  | 1  | 1              | 0              | 2              | 0              | -2             | 0              | 14  | 14/1=14    |
| 0 | 0 | 0  | 3  | 0              | 1              | 2              | 0              | -2             | 0              | 22  | 22/3=7 1/3 |
| 0 | 0 | 1  | 0  | 0              | 0              | -1             | 0              | 1              | 0              | 1   |            |
| 0 | 0 | 0  | 1  | 0              | 0              | 0              | -1             | 0              | 1              | 1   | 1/1=1      |
| 1 | 0 | 0  | 0  | 0              | 0              | 0              | 0              | -1             | -1             | 0   |            |
| 0 | 1 | 0  | 0  | 0              | 0              | -1             | -1             | 1              | 1              | 2   |            |
| 0 | 0 | 0  | 0  | 1              | 0              | 2              | 1              | -2             | -1             | 13  |            |
| 0 | 0 | 0  | 0  | 0              | 1              | 2              | 3              | -2             | -3             | 19  |            |
| 0 | 0 | 1  | 0  | 0              | 0              | -1             | 0              | 1              | 0              | 1   |            |
| 0 | 0 | 0  | 1  | 0              | 0              | 0              | -1             | 0              | 1              | 1   |            |

Pivot on the x column as it has a positive entry in the Q objective row

Pivot on the y column as it has a positive entry in the Q objective row

Since the artificial variables are now 0, these, and the supplementary objective can be removed and the remaining tableau solved (2 iterations) to give the solution.

### The big M method

The initial formulation is done in the same way as the two-stage simplex but instead of introducing a new objective, the original objective is modified by subtracting  $M(a_1 + a_2 + \dots)$ , where M is an arbitrary large number. Since  $M(a_1 + a_2 + \dots)$  is subtracted from the objective function, in order to maximise the objective you must make  $a_1, a_2, \dots$  zero.

**Example:** Maximise  $P = x + y$

Subject to  $2x + y \leq 16, 2x + 3y \leq 24, x \geq 1, y \geq 1$

**Solution:** Add the slack and surplus variables in the same way as for two stage simplex:

$$2x + y + s_1 = 16$$

$$2x + 3y + s_2 = 24$$

$$x - s_3 + a_1 = 1 \Rightarrow a_1 = 1 - x + s_3$$

$$y - s_4 + a_2 = 1 \Rightarrow a_2 = 1 - y + s_4$$

Write the new objective:  $P = x + y - M(a_1 + a_2)$

Re-writing without the artificial variables:  $P - x - y + M(1 - x + s_3 + 1 - y + s_4) = 0$

$$\Rightarrow P - (1+M)x + (1+M)y + Ms_3 + Ms_4 = -2M$$

|    | P | x      | y      | s <sub>1</sub> | s <sub>2</sub> | s <sub>3</sub> | s <sub>4</sub> | a <sub>1</sub> | a <sub>2</sub> | RHS   | Ratio test |
|----|---|--------|--------|----------------|----------------|----------------|----------------|----------------|----------------|-------|------------|
| R1 | 1 | -(1+M) | -(1+M) | 0              | 0              | M              | M              | 0              | 0              | -2M   |            |
| R2 | 0 | 2      | 1      | 1              | 0              | 0              | 0              | 0              | 0              | 16    | 16/2=8     |
| R3 | 0 | 2      | 3      | 0              | 1              | 0              | 0              | 0              | 0              | 24    | 24/2=12    |
| R4 | 0 | 1      | 0      | 0              | 0              | -1             | 0              | 1              | 0              | 1     | 1/1=1      |
| R5 | 0 | 0      | 1      | 0              | 0              | 0              | -1             | 0              | 1              | 1     |            |
|    |   |        |        |                |                |                |                |                |                |       |            |
| R1 | 1 | 0      | -(1+M) | 0              | 0              | -1             | M              | 1+M            | 0              | 1 - M |            |
| R2 | 0 | 0      | 1      | 1              | 0              | 2              | 0              | -2             | 0              | 14    | 14/1=14    |
| R3 | 0 | 0      | 3      | 0              | 1              | 2              | 0              | -2             | 0              | 22    | 22/3=7½    |
| R4 | 0 | 1      | 0      | 0              | 0              | -1             | 0              | 1              | 0              | 1     |            |
| R5 | 0 | 0      | 1      | 0              | 0              | 0              | -1             | 0              | 1              | 1     | 1/1=1      |
|    |   |        |        |                |                |                |                |                |                |       |            |
| R1 | 1 | 0      | 0      | 0              | 0              | -1             | -1             | 1+M            | 1+M            | 2     |            |
| R2 | 0 | 0      | 0      | 1              | 0              | 2              | 1              | -2             | -1             | 13    |            |
| R3 | 0 | 0      | 0      | 0              | 1              | 2              | 3              | -2             | -3             | 19    |            |
| R4 | 0 | 1      | 0      | 0              | 0              | -1             | 0              | 1              | 0              | 1     |            |
| R5 | 0 | 0      | 1      | 0              | 0              | 0              | -1             | 0              | 1              | 1     |            |

Add (1 + M)R4 to the objective row (R1)

Add (1 + M)R5 to the objective row (R1)

Pivot on the x column as it has a negative entry in the P objective row

Pivot on the y column as it has a negative entry in the P objective row

The M now only appears in the artificial variables columns (the “a” columns). This means that the values of  $a_1$  and  $a_2$  are now 0 and, as the M only appears in those columns, they can be removed and the remaining tableau solved (2 iterations) to give the solution.

## REVISION SHEET – DECISION MATHS 2

## TRAVELLING SALESPERSON AND ROUTE INSPECTION

**The main ideas are covered in**

|            |                |            |            |
|------------|----------------|------------|------------|
| <b>AQA</b> | <b>Edexcel</b> | <b>MEI</b> | <b>OCR</b> |
| <b>D1</b>  | <b>D1</b>      | <b>D2</b>  | <b>D1</b>  |

**The main ideas in this chapter are**

**Finding bounds within which the solution to the Travelling salesperson problem lies.**

**Applying the Nearest Neighbour algorithm to find an upper bound for the solution.**

**Apply the Chinese Postman Algorithm to obtain the closed trail of minimum weight.**

***Before the exam you should know:***

- A Hamiltonian cycle is a tour which contains every vertex (node) precisely once.
- An Euler cycle is a tour which travels along every edge of a network.
- The Nearest Neighbour algorithm is used for finding upper bounds for the TSP.
- The Nearest Neighbour algorithm will always produce a tour. But it may not be the optimal solution.
- Deleting a vertex, finding an MST and adding it back in is used for lower bounds to TSP.
- It may be possible to improve the tour by interchanging the order in which two nodes are visited.
- The meaning of order of a vertex (node), Eulerian (OR transferable) graph.
- Remember that the direct route is not always the shortest.
- When looking for a tour that satisfies the RI problem, note which arcs need to be used twice.

**The Traveling Salesperson Problem**

A **Hamiltonian cycle** is defined as a tour which contains every vertex precisely once. In a simple case it is easy to list all the Hamiltonian cycles but as the number of nodes increases, the number of Hamiltonian cycles tends to increase very rapidly. All methods discovered to date for solving the **travelling salesperson** problem have exponential order and so take far too long! Attention has been focused on finding not the optimal solution, but simply a reasonably good solution by establishing upper and lower bounds

**Lower Bounds**

1. Delete a vertex and the edges incident on it to form a reduced matrix
2. Find a minimum spanning tree for the remaining network
3. Reconnect the deleted vertex by the two shortest edges

**Repeat for all vertices. Greatest lower bound is the best lower bound**

**Upper bounds: Nearest Neighbour Algorithm**

Before you can apply the nearest neighbour algorithm you need to make a complete matrix of all shortest distance between pairs of vertices.

1. Choose any starting node
2. Consider the edges which join the previously chosen vertex to not-yet-chosen vertex and choose the one with minimum weight.
3. Repeat Step 2 until all nodes have been chosen.
4. Then add the arc that joins the last-chosen node to the first-chosen node

**For a full solution, NNA should be repeated starting at each vertex in turn. The shortest tour will be the least upper bound.**

Note the similarity between the nearest neighbour method and Prim's algorithm. Do not confuse the two: with Prim, choose the least weight arc from *all the nodes* in the tree. In the nearest neighbour, choose the least weight arc from the *current node only*.

**The Route Inspection Problem**

The problem is to find a route of minimum length which goes along each edge in the network once and returns to the starting point. This type of problem arises in contexts such as a rail safety expert needing to inspect every piece of track in a railway system, or a postman needing to walk along every street to deliver mail in the most efficient way possible, hence it is often called the **Chinese Postman problem**.

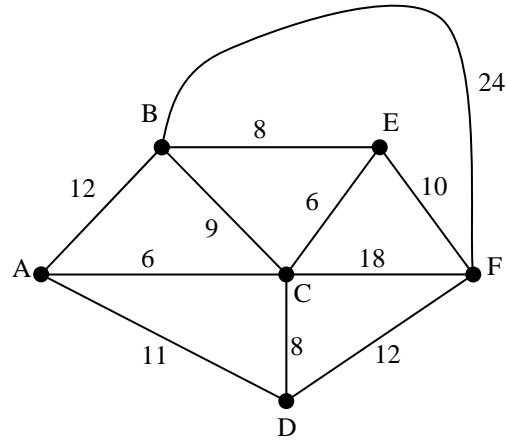
For a network to be traversable it must be Eulerian (no odd nodes) or semi-Eulerian (two odd nodes). A network will always have an even number of odd nodes (handshaking theorem). If the network is Eulerian (every vertex is of even order) there are many equal optimum solutions.

**The Algorithm** can be stated as follows

1. Identify the odd vertices in the network
2. Consider all the routes joining pairs of odd vertices and select the one with the least weight.
3. Find the sum of the weights on all the edges
4. Shortest distance is the sum of the weights plus the extra that must be travelled
5. Find a tour which repeats the edges found in step 2.

**Example:** For the network shown

- (a) Find the length of the shortest closed trail that covers every edge on the network below and write down a suitable route
- (b) Find an upper bound for the traveling salesperson problem, starting at vertex A.
- (c) Find a lower bound for the traveling salesperson by deleting vertex A.
- (d) Give a suitable route.



**Solution:**

- (a) Odd vertices are A, C, D and E. Consider all the possible pairings of odd vertices:  
 AC = 6 and DE = 14 total = 20                      AD = 11 and CE = 6 total = 17  
 AE = 12 and CD = 8 total = 20

The pairing of least weight is AD and CE = 17. The sum of the weights in the network is 124.

Repeating AD and CE gives a total weight = 124 + 17 = 141.

A suitable route is A – B – E – F – D – A – C – B – F – C – E – C – D – A.

(b) Nearest Neighbour algorithm

Upper bound has length

$$6 + 17 + 6 + 12 + 8 + 22 = 71$$

Route: A – C – E – B – D – F – A

Interpreted this is:

A – C – E – B – C – D – F – E – C – A

|   | 1  | 4  | 2  | 5  | 3  | 6  |
|---|----|----|----|----|----|----|
|   | A  | B  | C  | D  | E  | F  |
| A | -  | 12 | 6  | 11 | 12 | 22 |
| B | 12 | -  | 9  | 17 | 8  | 18 |
| C | 6  | 9  | -  | 8  | 6  | 16 |
| D | 11 | 17 | 8  | -  | 14 | 12 |
| E | 12 | 8  | 6  | 14 | -  | 10 |
| F | 22 | 18 | 16 | 12 | 10 | -  |

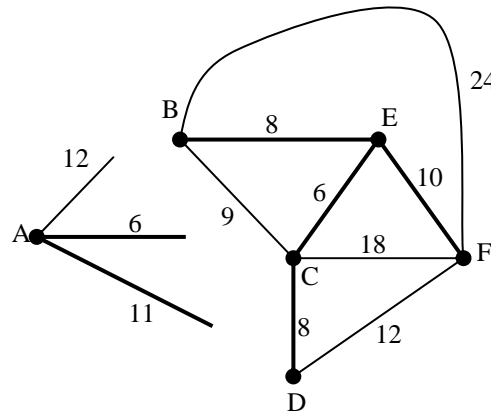
(c) delete vertex A

Minimum connector for the remaining network:

$$6 + 8 + 8 + 10 = 32$$

$$\text{Minimum distance to reconnect A: } 6 + 11 = 17$$

$$\text{Lower bound} = 32 + 17 = 49$$



(d)  $49 \leq \text{the length of the route} \leq 71$

A suitable route would be A – B – E – F – D – C – A

$$\text{Length } 12 + 8 + 10 + 12 + 8 + 6 = 56$$