Mathematical Modelling
Lecture 8 – Networks

Phil Hasnip
phil.hasnip@york.ac.uk

Overview of Course

- Model construction $\longrightarrow$ dimensional analysis
- Experimental input $\longrightarrow$ fitting
- Finding a 'best' answer $\longrightarrow$ optimisation
- Tools for constructing and manipulating models $\longrightarrow$ networks, differential equations, integration
- Tools for constructing and simulating models $\longrightarrow$ randomness
- Real world difficulties $\longrightarrow$ chaos and fractals

The material in these two lectures is not in the course textbook.

What is a network?

- A network is any system of interconnected locations
- Locations usually less important than the connections
- Not restricted to computer networks!

Using networks we can answer questions like

- What is the shortest route between two points?
- What is the cheapest route between two points?
- Are there any bottlenecks?

What's in a name?

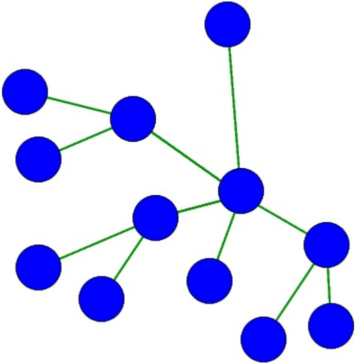
- Locations are called nodes or vertices
- Links are called connections or edges

Types of network

Classified according to the nature of the connections:

- May contain cycles
- May contain directed (one-way) or undirected edges
- May be complete (every node directly connected to every other) or incomplete

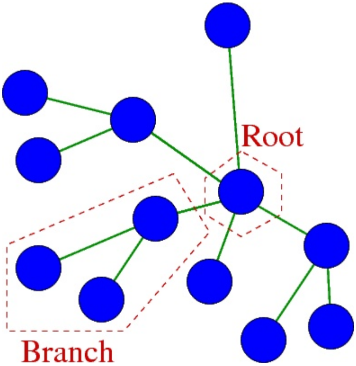
Types of network: Trees



A tree is a special type of network.

- Unique path from any one node to any other
  - Not cyclic
  - Incomplete
- May have directed and/or undirected edges

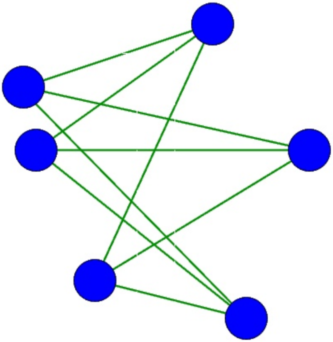
Types of network: Trees



A tree is a special type of network.

- Unique path from any one node to any other
  - Not cyclic
  - Incomplete
- May have directed and/or undirected edges

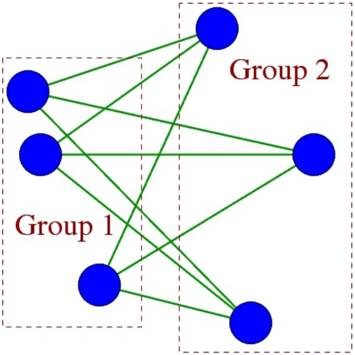
Types of network: Bipartite



A bipartite network has two distinct groups of nodes

- Paths only exist between nodes in different groups
- No paths between nodes in same group
- May have directed and/or undirected edges

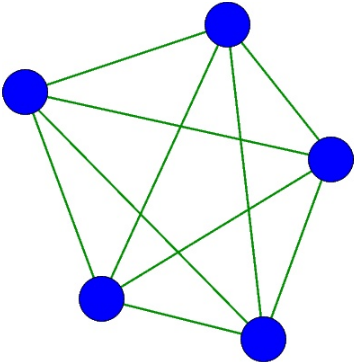
Types of network: Bipartite



A bipartite network has two distinct groups of nodes

- Paths only exist between nodes in different groups
- No paths between nodes in same group
- May have directed and/or undirected edges

Types of network: Complete

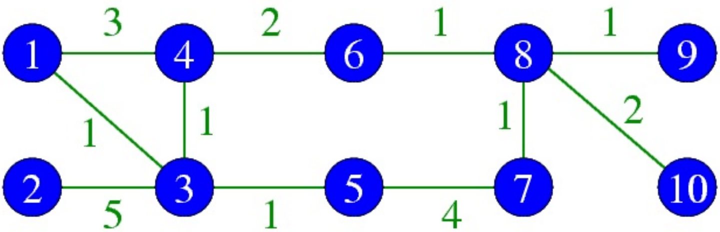


A complete network has:

- Paths from each node to all other nodes
- Lots of cycles

Cheapest path analysis

We are interested in the cheapest path from a particular node, called the root node, to another node.



Cheapest path analysis

Our method is to build up a linked list. We need to define three things:

- P_j is the previous node in the cheapest path found so far from the root node to node j
- K_j is the cost of the cheapest path found so far from the root node to node j
- C_{ij} is the cost of the connection from neighbouring node i to node j

Cheapest path analysis

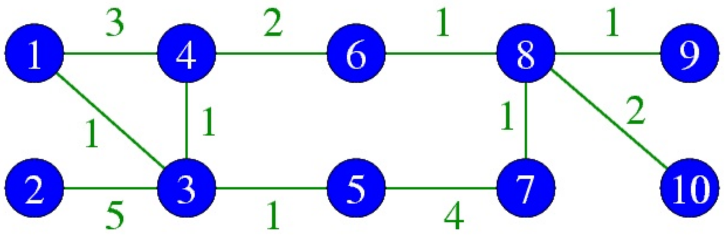
We start by initialising P and K for all the nodes:

- $[P_{root}, K_{root}] = 0$
- $[P_i, K_i] = [0, \infty]$ for non-root nodes
- Label the root node with a 'slash'

Cheapest path analysis

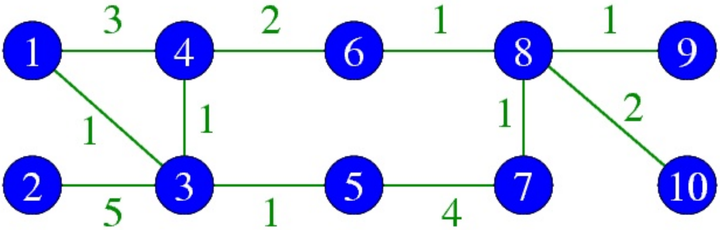
1. From the slashed node i, look at each non-slashed neighbour j
2. Calculate $K_i + C_{ij}$
3. If this is less than the current K_j:
   - $K_j = K_i + C_{ij}$ and $P_j = i$
   - Sketch the new $[P_j, K_j]$ by the node
4. Find the neighbour with the lowest K_j
5. Label this node with a 'slash' and repeat from step 1 until all nodes are 'slashed'

Example 1

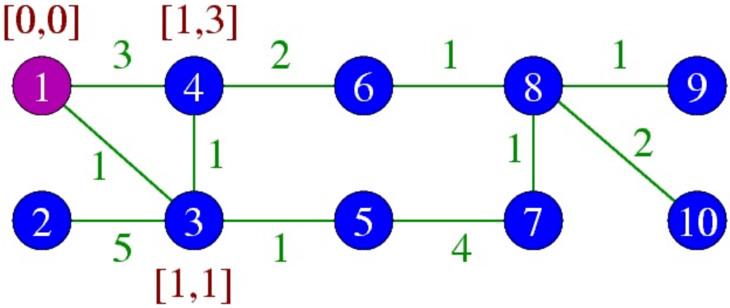


Let's look at the cheapest path from node 1 to nodes 9 and 10.

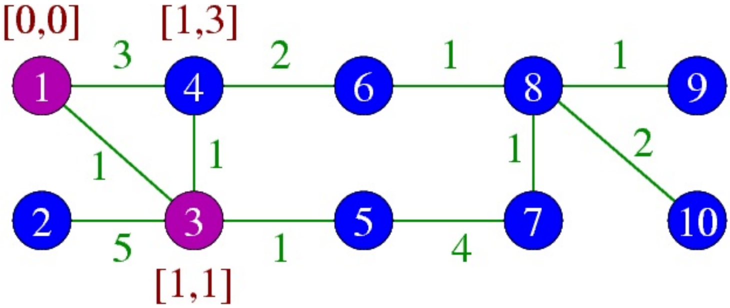
Example 1



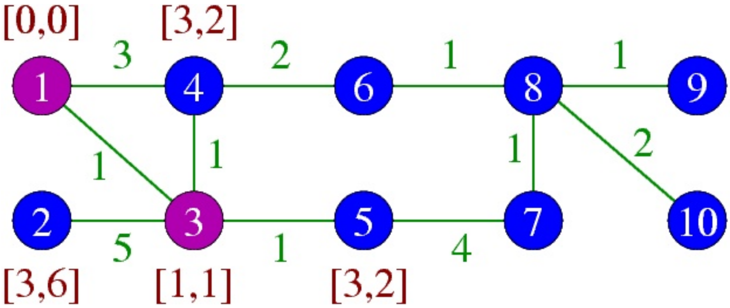
Example 1



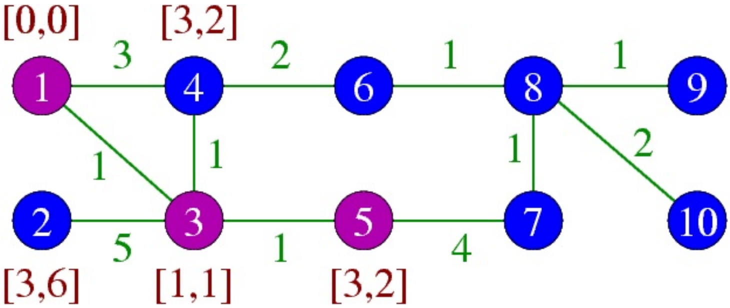
Example 1



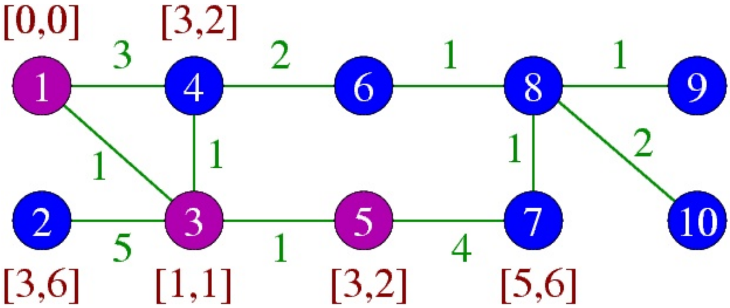
Example 1



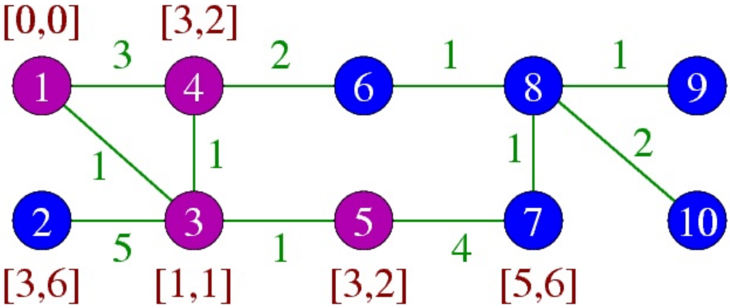
Example 1



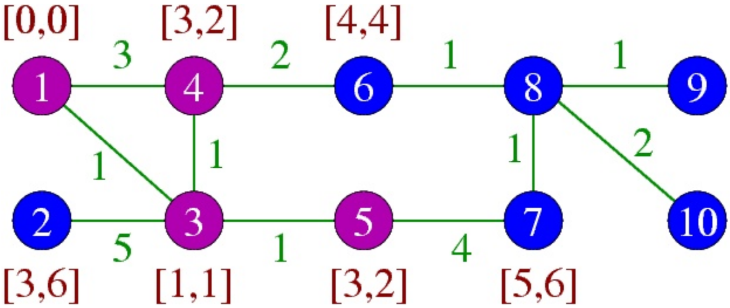
Example 1



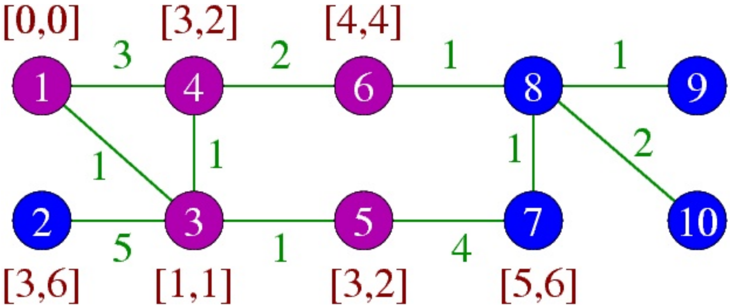
Example 1



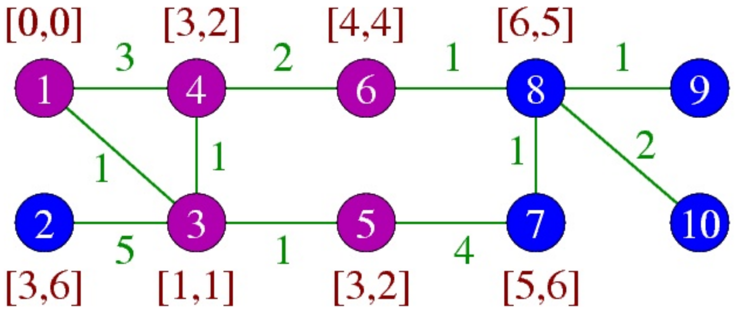
Example 1



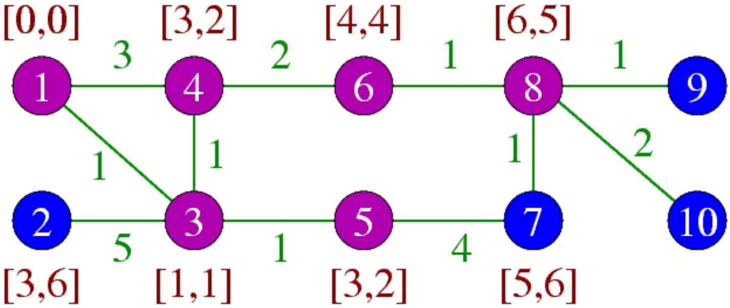
Example 1



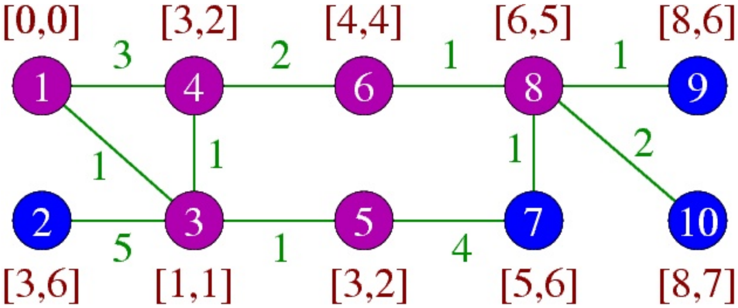
Example 1



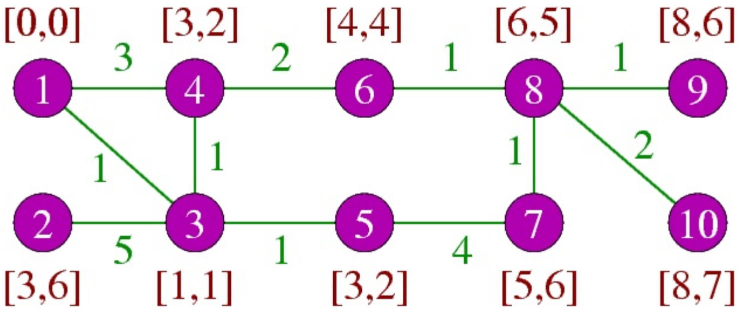
Example 1



Example 1



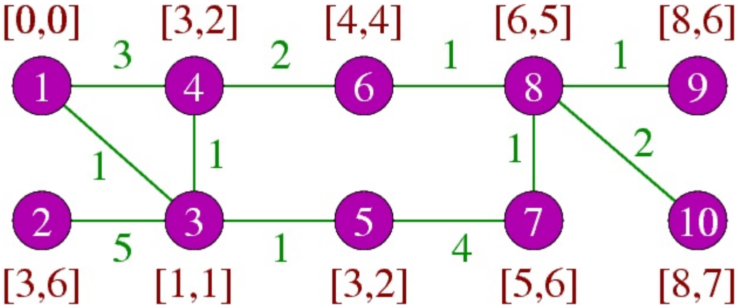
Example 1



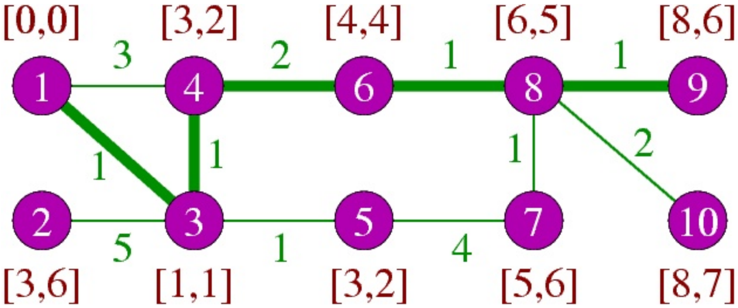
Cheapest path analysis

- At the end, K_{target} is the cost of the cheapest path to the target node
- We can find the cheapest path by working back along the links of previous nodes P_j: from node j move to node P_j and repeat until we reach the root node.

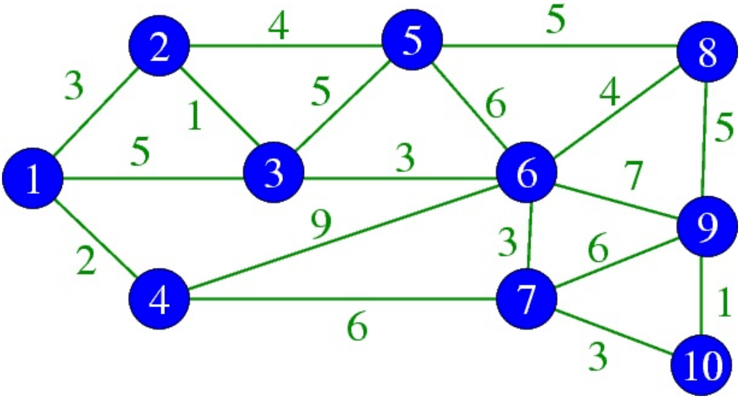
Example 1



Example 1

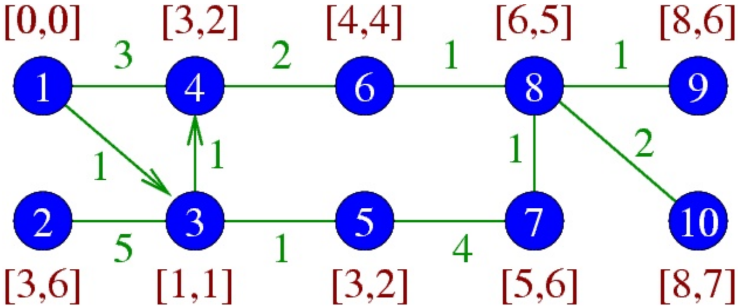


Example 2



Example 3 - Transportation network

Using the network from the first example, but $1 \to 3$ and $3 \to 4$ are directional (one-way). There is a further restriction on $3 \to 4$, that the flow must not exceed 50 units a day.



Example 3 - Transportation network

Using the cheapest routes:

Edge	Flow due to path A	B	C	D	Total flow
$1 \rightarrow 3$	35	20			55
$1 \rightarrow 4$					
$2 \rightarrow 3$			15	30	45
$3 \rightarrow 4$	35	20	15	30	100
$3 \rightarrow 5$					
$4 \rightarrow 6$	35	20	15	30	100
$5 \rightarrow 7$					
$6 \rightarrow 8$	35	20	15	30	100
$7 \rightarrow 8$					
$8 \rightarrow 9$	35		15		50
$8 \rightarrow 10$		20		30	50

Example 3 - Transportation network

Route	Cost	Flow
$1 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 8 \rightarrow 9$	6	x_1
$1 \rightarrow 3 \rightarrow 5 \rightarrow 7 \rightarrow 8 \rightarrow 9$	8	x_2
$1 \rightarrow 4 \rightarrow 6 \rightarrow 8 \rightarrow 9$	7	x_3
$1 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 8 \rightarrow 10$	8	y_1
$1 \rightarrow 3 \rightarrow 5 \rightarrow 7 \rightarrow 8 \rightarrow 10$	10	y_2
$1 \rightarrow 4 \rightarrow 6 \rightarrow 8 \rightarrow 10$	9	y_3
$2 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 8 \rightarrow 9$	10	z_1
$2 \rightarrow 3 \rightarrow 5 \rightarrow 7 \rightarrow 8 \rightarrow 9$	12	z_2
$2 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 8 \rightarrow 10$	11	w_1
$2 \rightarrow 3 \rightarrow 5 \rightarrow 7 \rightarrow 8 \rightarrow 10$	13	w_2

Example 3 - Transportation network

Total cost K:

$$K = 6x_1 + 8x_2 + 7x_3 + 8y_1 + 10y_2 + 9y_3 + 10z_1 + 12z_2 + 11w_1 + 13w_2$$

Subject to:

$$\begin{aligned} x_1 + x_2 + x_3 &= 35 \\ y_1 + y_2 + y_3 &= 20 \\ z_1 + z_2 &= 15 \\ w_1 + w_2 &= 30 \\ x_1 + y_1 + w_1 + z_1 &\leq 50 \end{aligned}$$

i.e. minimise K subject to constraints – a linear programming problem!

Summary

- Networks consist of nodes joined by connections
- Examples include trees, bipartite networks, complete networks
- May have cycles, be directed/undirected, complete/incomplete
- Dijkstra's algorithm computes the cheapest path from a particular node to all other nodes