Network models

- **The connectivity structure of networks is the fundamental carrier of their properties:**
- **Topology defines, metrics refine:**
  - We can quantify the distance between two entities in a network by different costs – distance, time, number of litres of petrol, energy needed, …

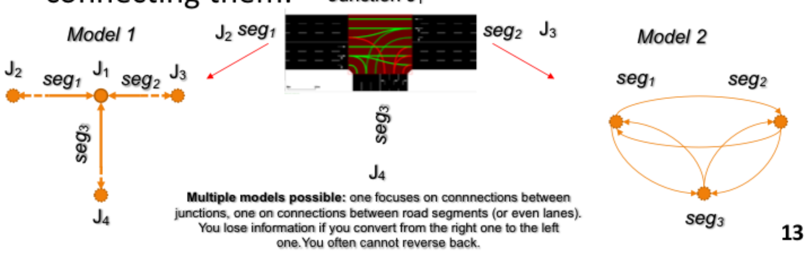
Spatial networks

- **Capture connectivity between entities located in space**
- **The links can be material or immaterial:**
  - Links between two junctions are streets;
  - Links between two streets are junctions;
  - Links between two tram lines are tram interchanges;
  - Links between people (spatial entities!) are their places of work (also spatial);
  - Links between people are their facebook connections (non material links)

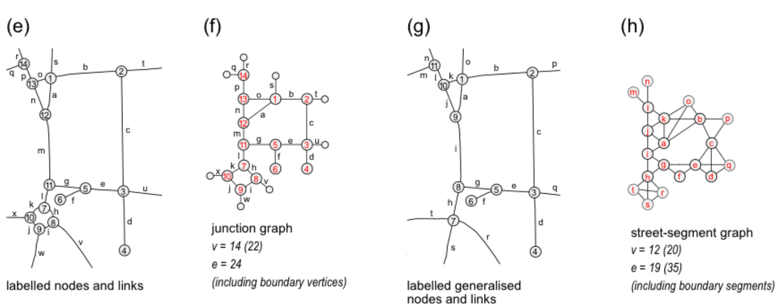
- **Graphs** are mathematical **structures** that **represent** networks, and abstract the **structure** and **properties** of networks **necessary** for **analysis:**
- **Definition: Graph *G(N,E)*, *N* set of Nodes, *E* set of Edges defined by pairs of nodes from *N***
  - Edge e={n1,n2} incident with n1 and n2
  - Edges and Nodes can have properties: Streets have names, length, Stations have opening hours, …

Modelling networks: unit of analysis

- The basic unit is represented as the **node** in a graph;
- The **edges** represent relationships/connections.
- There are multiple ways to model netowrks, depending on the analyst. Consider 3 junctions and segments connecting them:



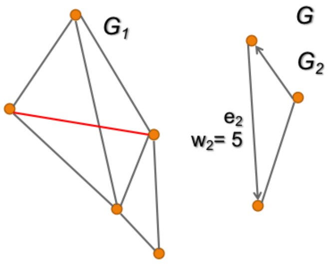
13



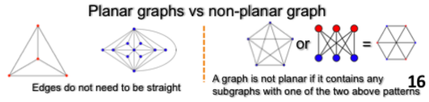
Graph properties

- **Number of nodes (also called vertices, beware)**
- **Number of edges**
- **Degree of node *n*: number of edges incident with *n***
  - Adjacent nodes: if connected by edge
  - In/Out degree (in directed graphs)
  - Max/Average degree/degree distribution

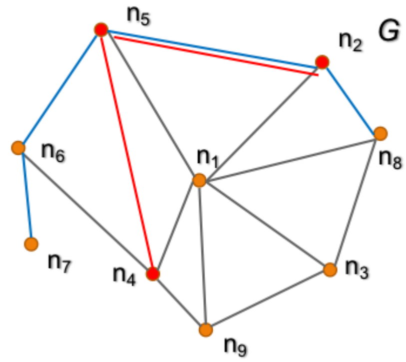
- **Connected Graph – graph where all nodes are connected**
  - Subgraph – subset of *G*
  - Strongly connected (sub) graph: two nodes are reachable
- **Graph types:**
  - Directed graph (digraph)
  - Weighted graph
  - Planar graph
  - Tree…



Planar graphs vs non-planar graph



- **Walk:** sequence of edges and vertices, where each edge's endpoints are the two vertices adjacent to it.
- **Path:** a **walk** in which all vertices are distinct (except possibly the first and last).
- **Reachable nodes: if path exists**
- **Diameter – longest shortest path (blue)**



Degree n_1:	6
Is n_2 adjacent to n_3:	*No*
Is *G* connected?:	yes
Path n_4-n_2:	[n_8,n_5,n_2]

- **Closed walks –** presence of walks that start and end on the same node, without duplicate occurrence of a node in the cycle
- **Tree** – a graph without closed walks
- **Girth –** longest simple cycle in a graph

Graph representation (storage)

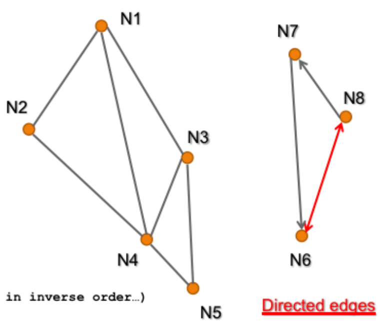
- **Adjacency list (undirected)**

```
G (8,10)
Edges:

N1,N3
N1,N2
N1,N4
N2,N4
N3,N4
N3,N5
N4,N5
N7,N8
N7,N6
N6,N8
```

- **Adjacency list (directed)**

```
G (8,10)
Arcs:
N1,N3
N1,N2
N1,N4
N2,N4
N3,N4
N3,N5
N4,N5
… (all above symmetrical – so repeated again in inverse order…)
N8,N7
N7,N6
N6,N8
N8,N6
```



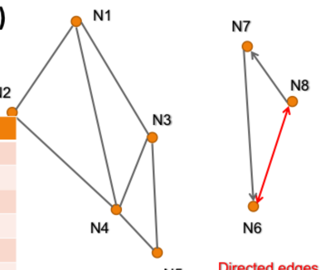
- **Adjacency matrix (undirected)**
  - Symmetrical
  - Sparse (lots of 0s)
  - Easy to compute deg. distribution

- **Adjacency matrix (directed)**
  - A-Symmetrical
  - Easy to compute in/out deg. distribution

	N1	N2	N3	N4	N5	N6	N7	N8
N1	0	1	1	1	0	0	0	0
N2	1	0	0	1	0	0	0	0
N3	1	0	0	1	1	0	0	0
N4	1	1	1	0	1	0	0	0
N5	0	0	1	1	0	0	0	0
N6	0	0	0	0	0	0	$\boxed{0}$	1
N7	0	0	0	0	0	1	0	0
N8	0	0	0	0	0	1	1	0

inDeg> N6: 2

outDeg N6: 1



0 weight on absent combinations

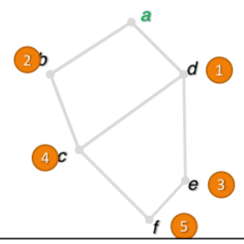
Analyses on graphs

- Influence analysis
  - using Centrality measures (of nodes, of edges)
  - E.g., Degree, Betweenness, Closeness centrality
- Traversal (search) and neighbourhood analysis (k-Nearest Neighbour), k-shortest paths
- Accessibility analysis: paths, travelling salesman problem
- Cost analysis (within cost analysis)
- Set operations: subgraphs, intersections, unions, cliques, motifs

Search in spatial networks

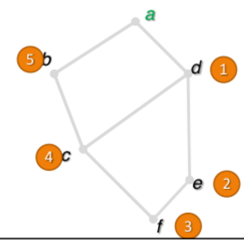
Breadth-first search:

Process all neighbours of start node *a*, then all neighbours of a neighbour of *a*, and so on …
and keep track of already processed nodes.



Depth-first search:

Process a neighbour of starting node *a*, then a neighbour of a neighbour of *a*, and so on …
At each dead end, backtrack to the last unprocessed vertex.



Search for optimal paths

- Types of optimal path problems:
- from a single source to a single destination (SSSD)
  - routing
- from a single source to all other vertices (SSAD)
  - spanning tree
- from all vertices to all other vertices (ASAD)
  - distance matrix

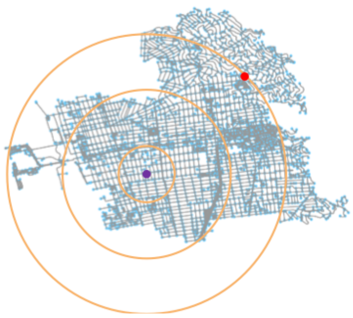
- Related:
- Optimal cycles (travelling salesperson problem)
- Optimal connection between locations
  - network construction; Steiner problem

Dijkstra algorithm

- Dijkstra algorithm
  - Exact solution, faster then breadth first/depth first
  - Requires a connected weighted graph (non-negative costs)
  - Complexity O(V^2)

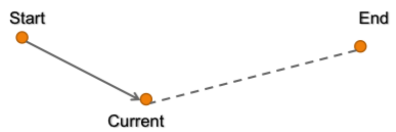
Dijkstra's search strategy: Radial

- Starting from source, searching in all directions for next nearest node



A* algorithm

- A* shortest path algorithm
  - Applies heuristic (~estimate) for distance to travel



$\text{Min}(\text{Cost}_{\text{start-current}} + \text{Estimate\_Cost}_{\text{current-end}})$

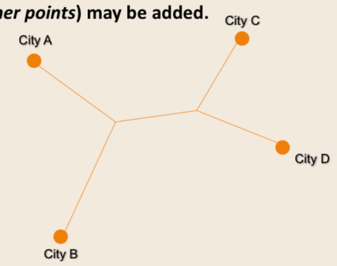
  - Requires a definition of underpinning space to define the distance metric (euclidean, manhattan…)

Alternatives to Dijkstra's algorithm

- Bi-directional Dijkstra
  - Start from source *and* destination
  - Form two trees until they meet

The Steiner problem

- Network *construction*: Given a set vertices *V*, interconnect them by a graph of shortest length. Intermediate vertices (*Steiner points*) may be added.



Other graph problems…Steiner problem

- Given a set of locations, find the shortest possible path that connectes them all.
- Combines shortest path and minimum spanning tree problems
- NP Hard