D1

Callum O'Brien

Contents

1 Networks

A **graph** consists of points called **vertices** or **nodes** which are connected by lines called **edges** or **arcs**. If a graph has a number associated with each edge it is a **network**. Vertices/nodes are referred to by letters, whereas edges are referred to are referred to by two letters.

A **subgraph** of G is a graph, all of whose vertices belong to G. The **degree** or **valency** of a vertex is the number of edges incident to it. A **path** is a finite sequence of edges such that the end node of one edge in the sequence is the start node of the next and in which no node appears more than once. A **walk** is a path in which some verticies may appears more than once. A **cycle** or **circuit** is a closed path, where the finial vertex is the same as the initial one. Two nodes are **connected** if there exists some path between them. A graph is connected if all its nodes are connected to every other. A **loop** is an edge which connects a node to itself.

A **simple** graph has no loops. A **directed** edge AB connects A to B but not B to A. A graph with directed edges is a **digraph**. A graph which contains no cycles is a **tree**. If all nodes within a graph share an edge, the graph is **complete**. In a **bipartite** graph, there are two groups of nodes with no direct connections between them.

1.1 Minimum Spanning Trees of Networks

A **minimum spanning tree** of a network is a way to connect all the nodes of the network with a minimum weight. Given a complete network, there are two algorithms to find the minimum spanning tree; Kruskal's algorithm and Prim's algorithm.

1.1.1 Kruskal's Algorithm

1. Sort all the edges into ascending order of weight
2. Select the edge of least weight to start the tree
3. If the edge of next least weight would form a cycle with an already selected edge, reject it; else, add it to the tree
4. If all the nodes are connected, finish; else, goto 3.

1.1.2 Prim's Algorithm

Prim's algorithm provides an advantage when it is preferable to have a central locus for the connections like on the national grid (power stations).

1. Choose any node to start the tree
2. Select an edge of least weight that joins a node that is already in the tree to a node that is not in the tree
3. If all nodes are connected, finish; else, goto 2

1.2 Adjacency Matrices

An adjacency matrix shows which nodes are directly connected to one another.

1.3 Distance Matricies

A distance matrix shows the distances between nodes. Prim's algorithm can be applied to distance matricies;

1.4 Dijkstra's Algorithm

This algorithm is used to find the shortest path from S to T;

1. Label S with the final label 0
2. Record a working value at every node Y that is directly connected to the node X that has just received its final label s.t.;
   - *Working value at* Y = *final value at* X + *weight of arc* XY
   - If there is already a working value at Y it is only replaced if the new value is smaller
   - Once a value has a final label it is not revisited and its working values are no longer considered
3. Look at the working values at all vertices without final labels. Select the smallest working value, This now becomes the final label at that vertex. If two vertices have the same smallest working value, choose either
4. Repeat steps 2 and 3 until the destination node T receives its final label
5. To find the shortest path, trace back from T to S. If B is on the route, include arc AB whenever *final label* B − *final label* A = *weight of arc* AB

1.5 Route Inspection

Non-eulerian graphs have more than two nodes with odd degree; in these, there exists no path that traverses every path once and only once. Semi-eulerian graphs have two nodes with odd degree; there are paths that traverse all edges only once in these graphs, but no cycles. Eulerian graphs have an even N^{o} nodes with odd degree and there are cycles that transverse all edges once.

In an Eulerian graph, the weight of the shortest cycle connecting all nodes is equal to the weight of the network. In a semi-eulerian graph, the weight of the shortest cycle is equal to the weight of the network plus the weight of the shortest path from one of the odd nodes to the other.

For an arbitrary graph, one selects the shortest cycle by reducing it to a eulerian graph;

1. Identify all nodes with odd degree
2. Consider all possible pairings of these vertices
3. select the complete pairing that has the least sum
4. add a repeat of the arc identified by this pairing to the network
5. find your route around this network as in an eulerian graph