Travelling Salesman Problem (10 pages; 29/6/20)

(1) Terminology

Walk: (a sequence of arcs, where the end of one arc is the start of the next) arcs and nodes can be repeated

Trail: arcs can't be repeated, but nodes can be

Path: arcs and nodes can't be repeated

Tour: a closed walk that visits every node at least once

Hamiltonian (or sometimes Hamilton) cycle: a closed walk that visits every node exactly once (ie a tour for which nodes aren't repeated) [Note that in a Hamiltonian cycle arcs can't be repeated either (as otherwise this would mean repeated nodes), and so a Hamiltonian cycle is a cycle.]

(2) Classical and Practical Problems

The Classical version of the Travelling Salesman Problem (referred to as the 'Classical Problem') is to find the shortest Hamiltonian cycle for a network.

However, a Hamiltonian cycle may not exist (see Figure 1); or, if one does exist, it may not give a good solution (see Figure 2, where the tour ABACA will usually be preferable to the Hamiltonian cycle ABCA; ie the fact that nodes are repeated may be tolerated (or may not be an issue anyway).

If repeated nodes are acceptable (so that the only requirement is to find a tour) then the problem is referred to as the 'Practical Problem'.

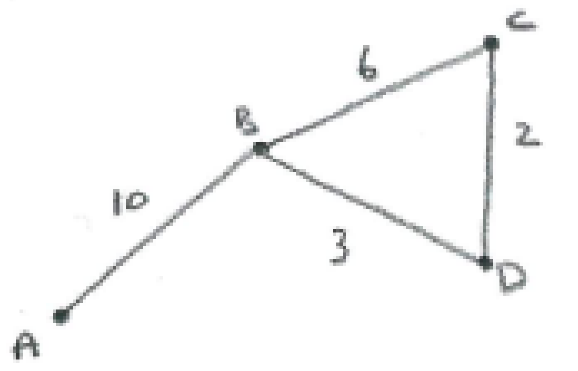

Figure 1

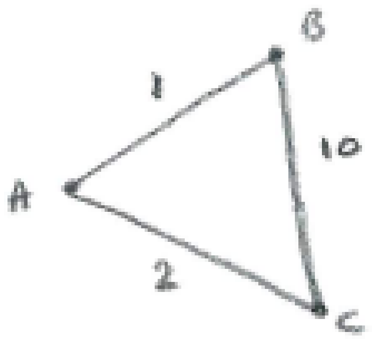

Figure 2

Hamiltonian cycles can always be found for a complete graph.

The Practical problem can always be converted to the Classical Problem by creating a network of shortest distances. Note though that the solution found will need to be converted back to the original form (eg the arc between nodes A and D in the network of shortest distances may represent the path ABCD in the original network).

The networks of shortest distances corresponding to the networks in Figures 1 & 2 are shown in Figures 3 & 4.

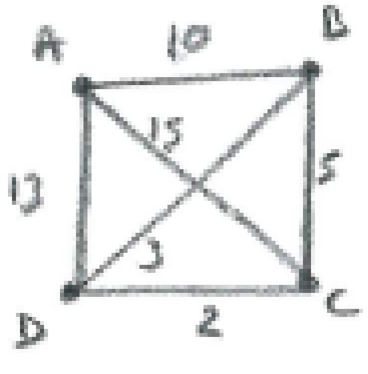

Figure 3

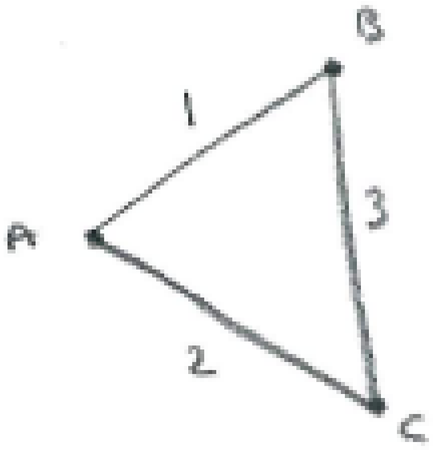

Figure 4

(Thus, in Figure 3, A is now connected directly to C by an arc of weight equal to the shortest distance between A and C in the original network.)

[The so-called 'triangle inequality' is often referred to in connection with the network of shortest distances. The idea is that a complete network can only be a network of shortest distances if the direct arc (AC, say) between two nodes cannot be improved on by travelling via another node (say B). It is sometimes claimed that the network of shortest distances can be created by repeatedly replacing the weight of AC (for example) by the sum of the weights of AB and BC, if this gives a smaller value. However, every possible triangle in the network would have to be examined.]

Some of the algorithms that are available for the Travelling Salesman Problem may not work fully if the network is not complete. This issue can be avoided by using the network of shortest distances. Also, it may be the case that an algorithm gives a better solution if the network of shortest distances is used (even if the original network is complete).

Exam questions should make it clear whether a network of shortest distances is required.

(3) If a graph is complete and has n nodes, then there will be $\frac{1}{2}(n-1)!$ possible tours. (We can choose to start at any node (since the tour is circular), and there will be $n-1$ ways of choosing the next node to proceed to (and so on). We divide by 2, as reversing the order gives the same tour; eg ABCDEA is the same as AEDCBA.

The travelling salesman problem thus has factorial complexity, and computers cannot simply examine all the possibilities if n is even of moderate size:

$n = 5$:   12 tours

$n = 10$:   181 440 tours

$n = 15$: 4.4×10^{10} tours

(4) Strategy

Unfortunately no algorithm exists for the travelling salesman problem which can determine the optimal solution. The best that can be done is to establish upper and lower bounds for the shortest length, and then to try and refine these, so that the optimal value is shown to lie within a small enough interval; ie such that no significant improvement will be obtained from further work.

The algorithms that are used to do this (and which are described below) are examples of 'heuristic' algorithms; ie they usually produce a good solution, but it may not be the best possible one.

Various methods exist for finding tours that have reasonably short lengths. The length for any such tour represents an upper bound for the shortest distance (ie the shortest distance can't be greater than the length found).

An algorithm also exists for establishing a lower bound for the shortest distance (discussed below). This lower bound is not usually attainable. All we can say is that a tour cannot have a shorter length than this.

It is then a matter of continuing to apply the various methods to find tours that are shorter than the current upper bound.

(5) Stage 1: Inspection

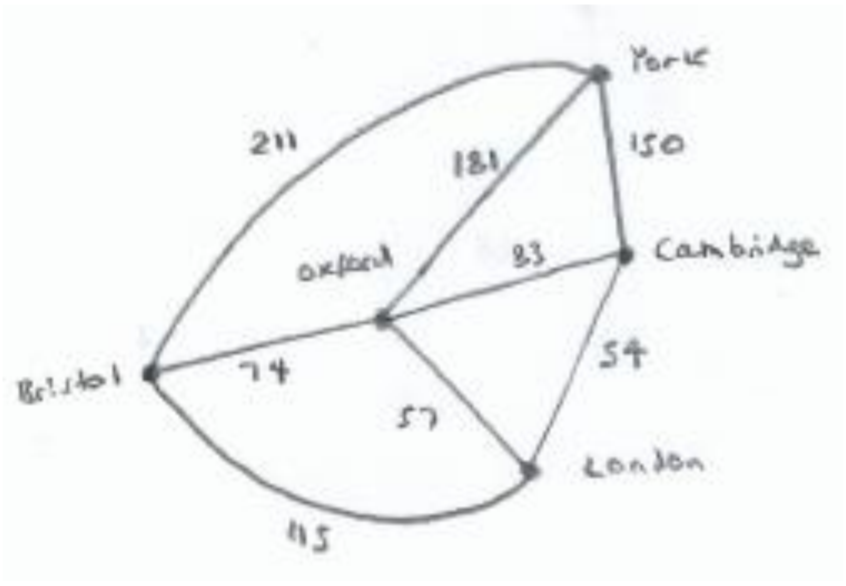

Figure 5

Referring to the cities example in Figure 5, three possible tours (with their total lengths) are:

BYOCLB (211+181+83+54+115=644)

BYCOLB (211+150+83+57+115=616)

BOYCLB (74+181+150+54+115=574)

These lengths have been found by inspection, and so far the upper bound for the shortest distance is 574. The following systematic methods can be applied to tackle the problem (for a bigger network, inspection may not be feasible).

(6) Stage 2: Upper Bound from Minimum Connector

The method is as follows:

(i) Find a minimum connector for the network.

(ii) Duplicate each of the arcs.

This is carried out below:

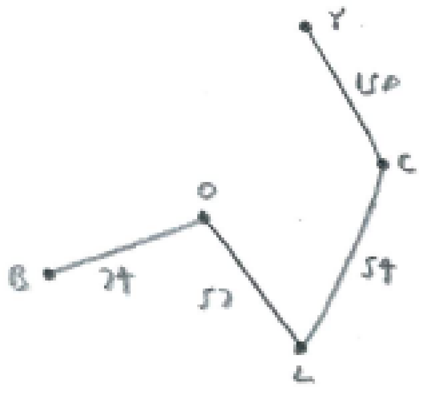

Figure 6

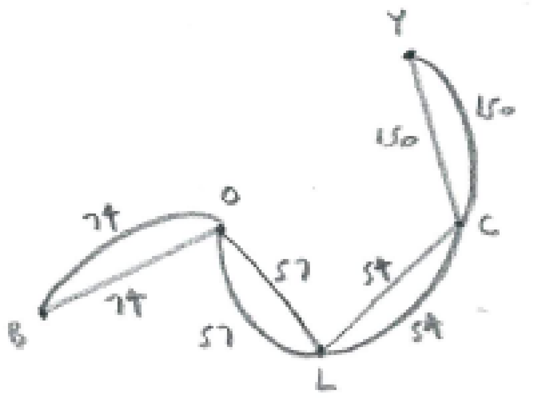

Figure 7

This produces a length of $(74 + 57 + 54 + 150) \times 2 = 670$

This isn't an improvement on the existing upper bound of 574.

[Note that this method is applicable only to the Practical Problem.]

(7) Stage 3: Improving on the upper bound

The Minimum Connector method can usually be improved on significantly by taking one or more shortcuts.

Referring to Figure 7, we start with BOLCYCLOB:

$(74 + 57 + 54 + 150) \times 2 = 335 \times 2 = 670$

This can be replaced with

BOLCY**CO**B: $335 + 150 + 83 + 74 = 642$

or with BOLC**YO**B: $335 + 181 + 74 = 590$

or with BOLC**YB**: $335 + 211 = 546$

Thus the improved upper bound is 546.

(8) Stage 4: Lower Bound Algorithm (using Minimum Connectors)

Referring to Figure 5, any Hamiltonian cycle will consist of 2 arcs from (eg) B, together with 3 arcs linking O, L, C & Y.

The minimum total length of 2 arcs from B is $74 + 115 = 189$

To find the minimum total length of arcs linking O, L, C & Y, we can find the minimum connector for these nodes.

Applying Prim's Algorithm, for example, starting at O:

OL(57)

OL(57)+LC(54)

OL(57)+LC(54)+CY(150) = 261

Alternatively, applying Kruskal's Algorithm:

LC(54)

LC(54)+LO(57)

LC(54)+LO(57)+CY(150) = 261

Hence the lower bound (so far) is 189+261=450

Further lower bounds can be established by dividing up the nodes differently. Using Kruskal's algorithm gives:

O + BLCY

(57+74)+(54+115+150)=450

L + BOCY

(54+57)+(74+83+150)=418

C + BOLY

(54+83)+(57+74+181)=449

Y + BOLC

(150+181)+(54+57+74)=516

The figure of 516 supersedes the other, lower figures (although it is still true that the shortest tour can't be lower than 418, for example, it is also true that it can't be lower than 516).

[This method is only guaranteed to work if the network is complete, and is applicable only to the Classical Problem.]

(9) Stage 5: Upper Bound - Nearest Neighbour Algorithm

This is the standard algorithm for producing an upper bound.

Referring to Figure 5,

(i) Start at any node (eg B)

(ii) Add the shortest arc leading to a new node: BO

(iii) Repeat the process, to give: BO+OL+LC+CY

(iv) Return to B by the shortest route, to give the cycle:

BOLCYB (74+57+54+150+211=546)

[Note: If only Hamiltonian cycles are acceptable, then we will have to return to B directly - assuming this is possible (in this case, the method is only guaranteed to work if the network is complete).]

(v) Repeat the algorithm, with other starting points:

OLCYBO (57+54+150+211+74=546)

LCOBYL (54+83+74+211+204=626)

CLOBYC (54+57+74+211+150=546)

YCLOBY (150+54+57+74+211=546)

Thus we obtain an upper bound of 546. (In general we would take the lowest of the figures obtained.) However, for this example this is the figure already obtained from the improvement to the Minimum Connector method.

Notes

(i) The Nearest Neighbour algorithm resembles Prim's algorithm. However Prim's algorithm joins the nearest new node to any existing node, whereas the Nearest Neighbour algorithm joins it to the last node obtained. Also, Prim's algorithm is designed to produce a tree and we don't return to the start node.

(ii) It is a 'greedy' algorithm: it doesn't look ahead, and just maximises the short-term gain (by selecting the nearest node).

(iii) Although the solution will usually be much better than for the Minimum Connector method described earlier, the Nearest Neighbour algorithm doesn't usually give the best possible solution.

So far we have 'trapped' the shortest possible tour between 516 and 546. However there is no guarantee that a shorter tour than 546 exists.

(10) Stage 6: Tour improvement algorithm

This can be illustrated by referring to Figure 5 again. The three possibilities that were mentioned earlier were:

BYOCLB (211+181+83+54+115=644)

BYCOLB (211+150+83+57+115=616)

BOYCLB (74+181+150+54+115=574)

Note that BYOCLB is improved by swapping O and C, or by swapping Y and O.

The algorithm consists of examining each sequence of 4 nodes, and seeing if an improvement can be obtained by swapping the middle two nodes.

Thus, for the above example, YCOL gives a shorter distance than YOCL, and BOYC gives a shorter distance than BYOC.

Using computer language, the algorithm could be written as:

```
For i = 1 to n
If $d(N_i, N_{i+2}) + d(N_{i+1}, N_{i+3}) < d(N_i, N_{i+1}) + d(N_{i+2}, N_{i+3})$ then swap $N_{i+1}$& $N_{i+2}$
Next i
```

(where N_i denotes the ith node, and d denotes distance).