The Travelling Salesman Problem

The travelling salesman problem cannot be solved directly without vast amounts of processing power, since every possible Hamiltonian cycle would need to be measured.

For a cycle visiting the largest 8 cities in England, there are 2520 unique cycles to compare, but if we wanted to include every one of the 51 English cities, the number of cycles increases to around 1.5×10^{64}.

The number of Hamiltonian cycles for a complete graph with n nodes is:

$$\frac{(n-1)!}{2}$$

- *Since every unique cycle will contain all vertices, we can choose one arbitrarily as a starting vertex (ABCDA is the same cycle as BCDAB).*
- *There will then be $n-1$ choices for the second vertex, and $n-2$ choices for the third, and so on, since only unconnected vertices can be added until the end.*
- *Finally, we divide by 2 because cycles like ABCDA and ADCBA are actually the same cycle, just reversed. Half of the $(n-1)!$ cycles we are considering are just the reverse of the other half.*

In general, the best we can do is:

- **Generate upper bounds:** Find some potential candidates (valid cycles that are not necessarily the shortest possible)
- **Generate lower bounds:** Combine edges in the most efficient way (shortest collections of edges that are not necessarily valid cycles)

An **upper bound** can be found using the 'Nearest Neighbour' algorithm:

Start at a particular vertex, and travel to the nearest unused vertex.
Continue until all vertices are included, then return to the start vertex.

This will always produce a valid Hamiltonian cycle, and therefore must be at least as long as the minimum cycle. Hence, it is an upper bound. By choosing different starting vertices, a number of different upper bounds can be found.

The lower the upper bound, the better, as it gives us more information.

A **lower bound** can be found by removing a vertex, then finding a minimum spanning tree:

Use Prim's or Kruskal's algorithm to find the length of the minimum spanning tree.
Add to this the lengths of the two shortest edges connected to the missing vertex.

Since any Hamiltonian cycle with a vertex removed will necessarily be a spanning tree of the remaining vertices, it follows that its length will be at least as great as that of the *minimum* spanning tree of those vertices. And the most efficient way to connect this tree to the final vertex is via the two shortest edges. By removing different vertices, a number of different lower bounds can be found.

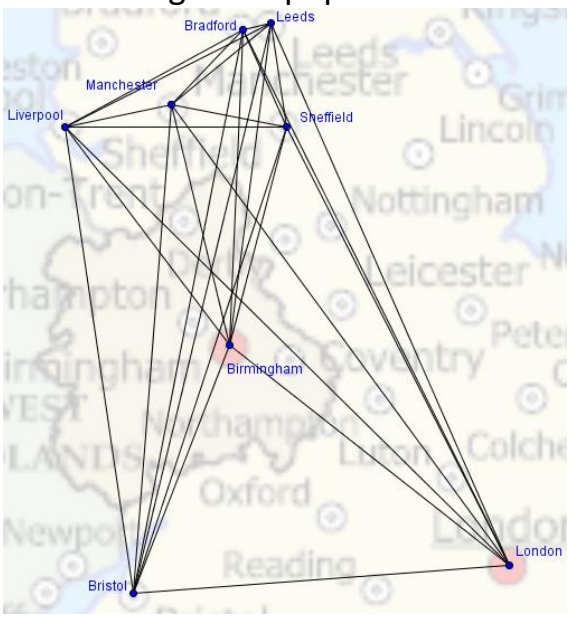
The higher the lower bound, the better, as it gives us more information.

Note: one easy way of generating a lower bound would be to simply choose the shortest n edges, but the minimum spanning tree method usually gives higher (ie better) lower bounds.

While upper and lower bounds won't (usually) categorically identify the optimal route, they allow us to determine the range of possible values it may have. ("I don't know if this is the shortest route, but I do know that it's no more than 10% longer than the shortest.")

Travelling Salesman: Visiting the 8 largest cities in England

A fifth of England's population lives in one of the eight cities listed below:

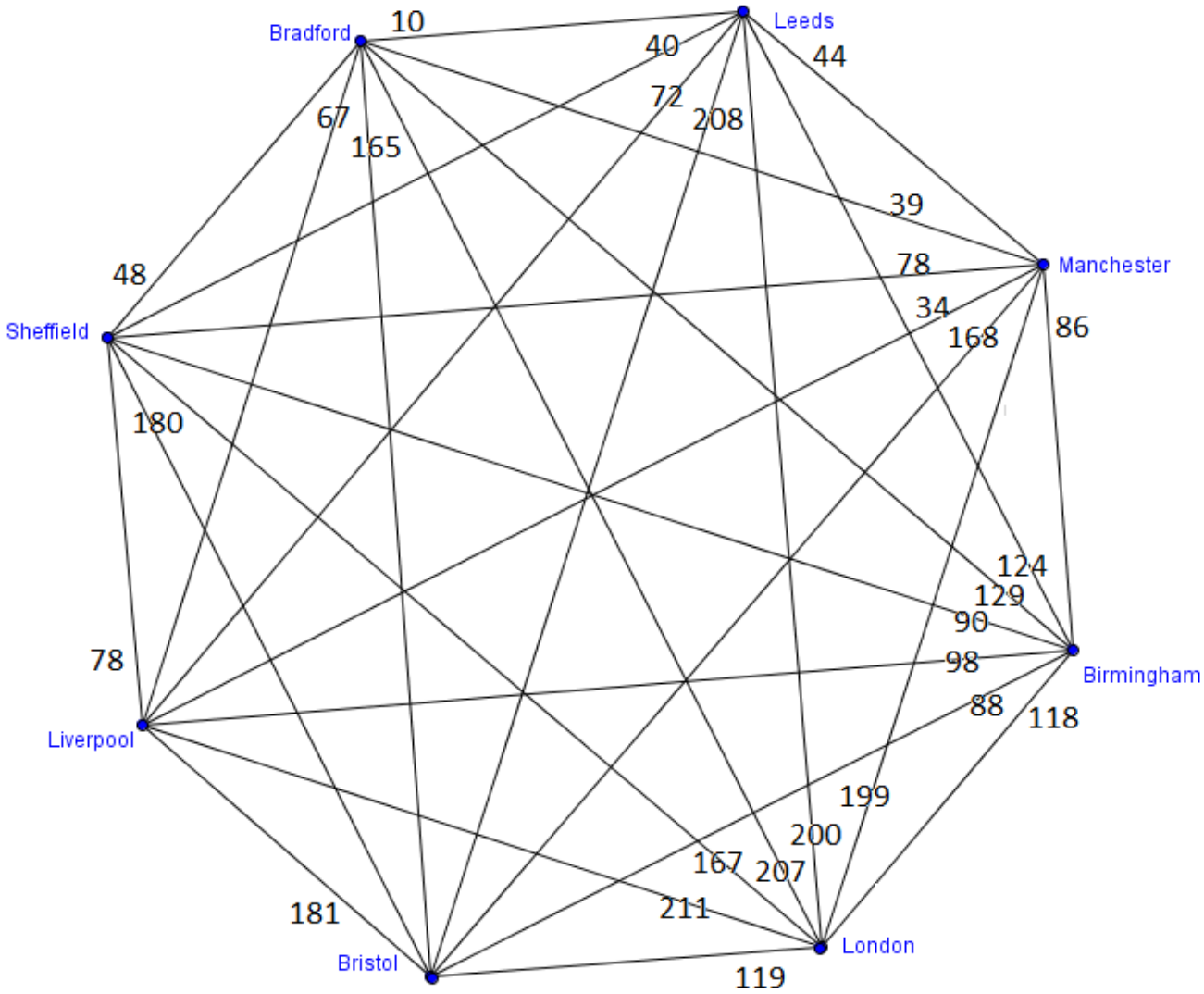


	London	Birmingham	Leeds	Sheffield	Bradford	Liverpool	Manchester	Bristol
London		118	200	167	207	211	199	119
Birmingham	118		124	90	129	98	86	88
Leeds	200	124		40	10	72	44	208
Sheffield	167	90	40		48	78	78	180
Bradford	207	129	10	48		67	39	165
Liverpool	211	98	72	78	67		34	181
Manchester	199	86	44	78	39	34		168
Bristol	119	88	208	180	165	181	168	

Distances between cities are driving distances taken from WolframAlpha, and are correct to the nearest mile.

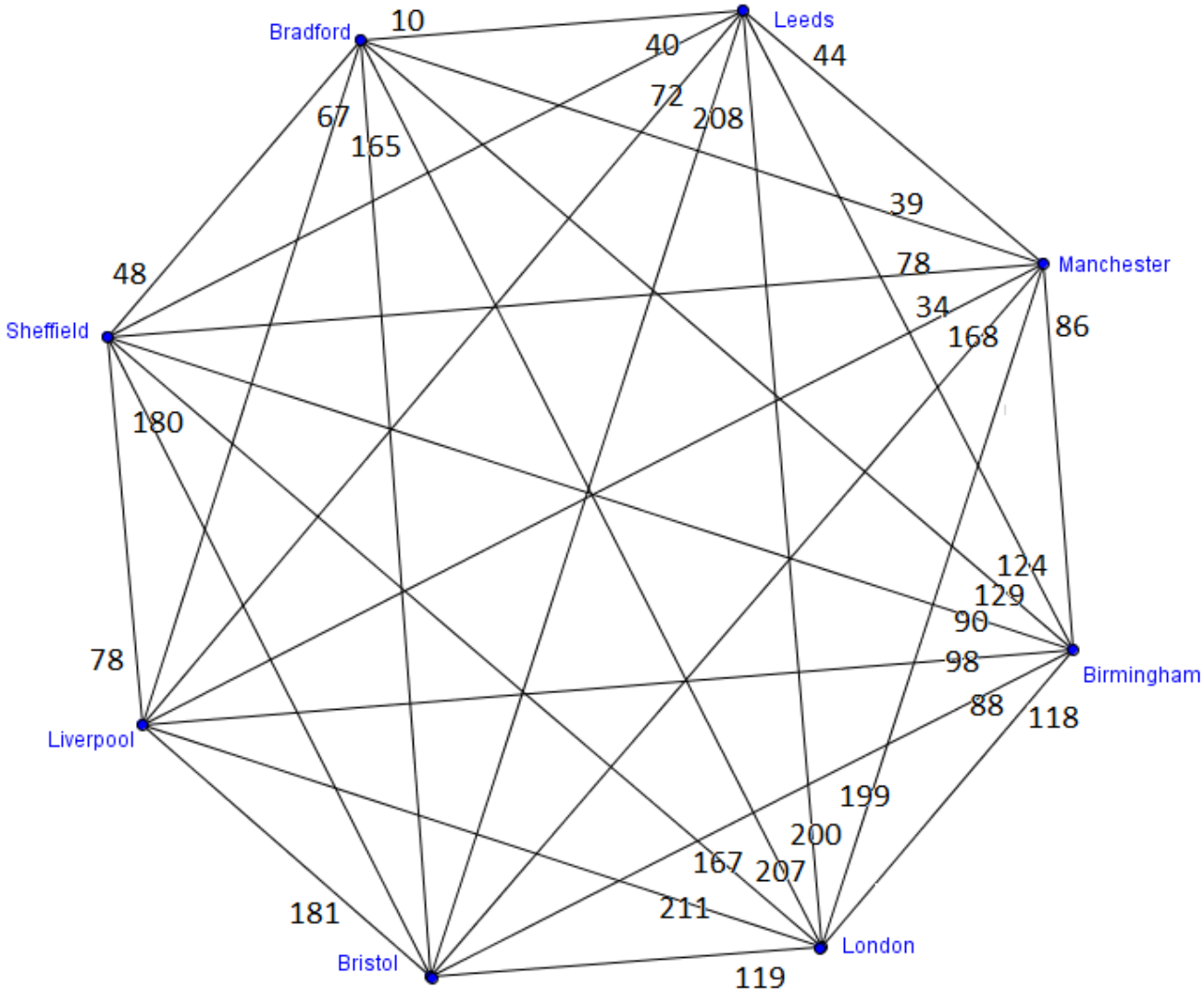
1. The complete graph, shown above, has geographically accurate positions for the cities. Before you begin the process of finding upper and lower bounds, use this graph to make a prediction about the optimal travelling salesman route. Use the table to find its length.

2. Now use the network below (nodes repositioned for ease of use) to find an upper bound and a lower bound on the optimal Hamiltonian cycle. Show your method clearly.



Resource Sheet

	London	Birmingham	Leeds	Sheffield	Bradford	Liverpool	Manchester	Bristol
London		118	200	167	207	211	199	119
Birmingham	118		124	90	129	98	86	88
Leeds	200	124		40	10	72	44	208
Sheffield	167	90	40		48	78	78	180
Bradford	207	129	10	48		67	39	165
Liverpool	211	98	72	78	67		34	181
Manchester	199	86	44	78	39	34		168
Bristol	119	88	208	180	165	181	168	



Example: Using inspection to find an upper bound

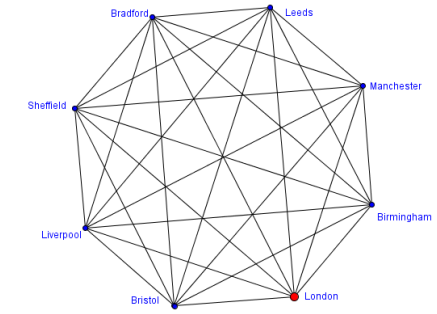
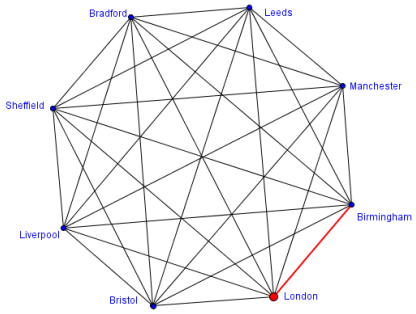
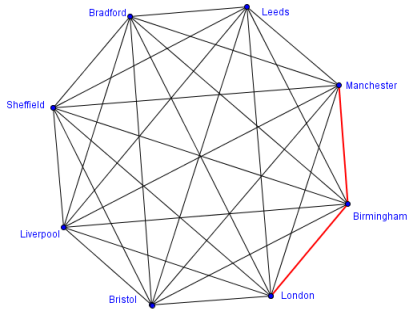
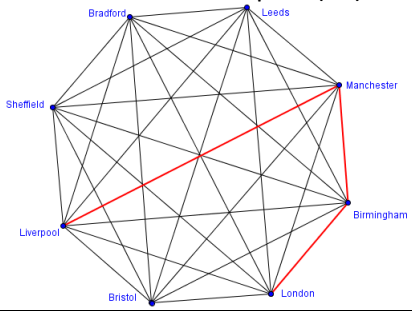
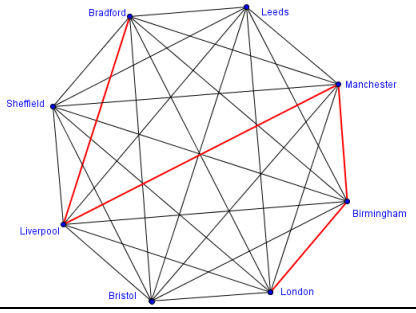
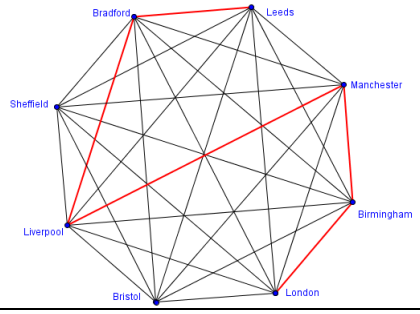
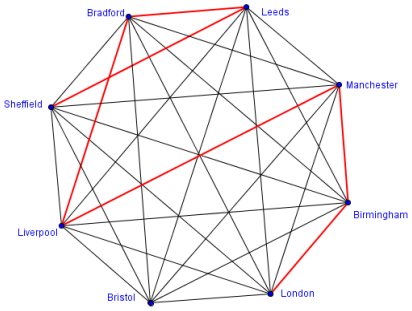
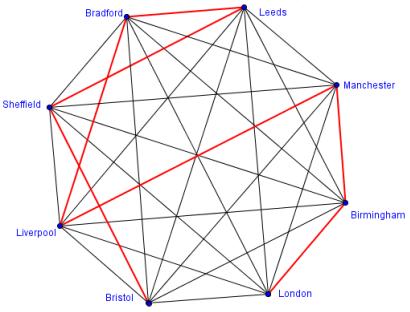
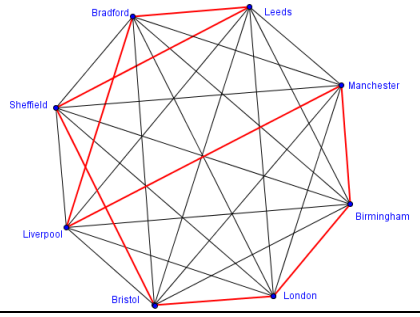
Using the geographical layout to come up with a route 'by eye' might yield:

Leeds – Manchester – London – Bristol – Birmingham – Liverpool – Sheffield – Bradford – Leeds (total: 684)

This, like any valid Hamiltonian cycle we could find, gives an upper bound on the problem:

The minimum cycle must have a length no greater than 684 miles.

Example: Applying the 'Nearest Neighbour' algorithm to find an upper bound, starting from London:

1. Starting vertex: London 	2. London to Birmingham (118) 	3. Birmingham to Manchester (86) 
4. Manchester to Liverpool (34) 	5. Liverpool to Bradford (67) 	6. Bradford to Leeds (10) 
7. Leeds to Sheffield (40) 	8. Sheffield to Bristol (180) 	9. Bristol to London (119) 

Note that equally weighted edges could result in multiple valid alternatives from this algorithm, even starting from the same point. Ideally, you should pursue each route, as each will generate a valid upper bound (the lowest of which will be the best).

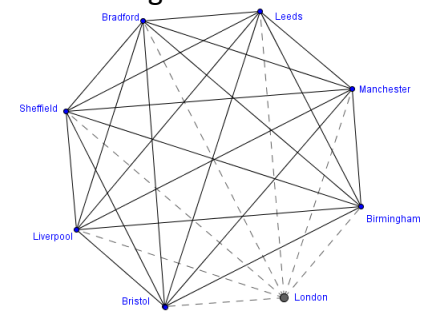
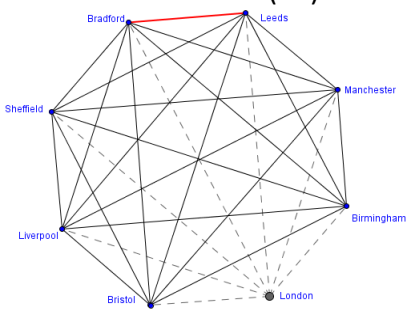
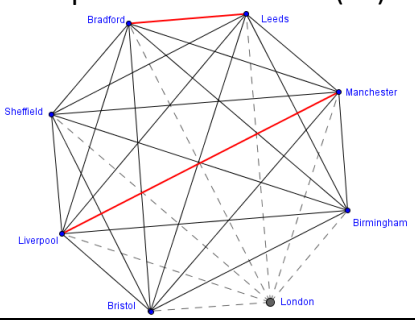
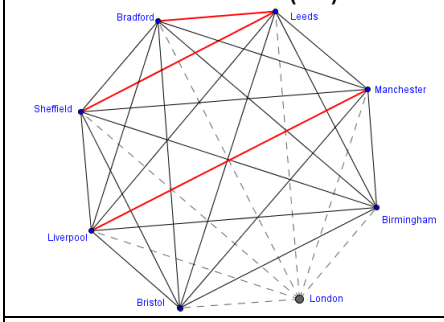
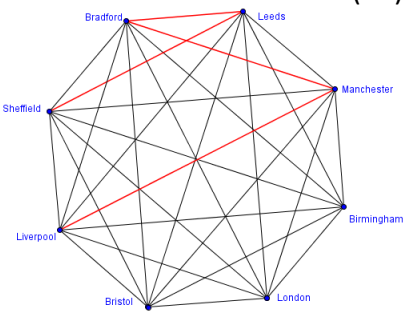
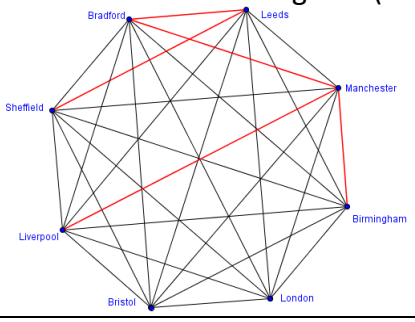
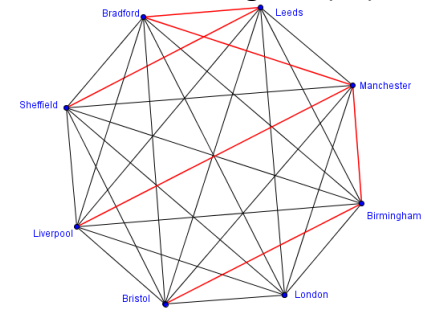
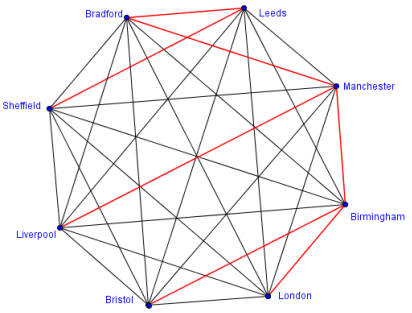
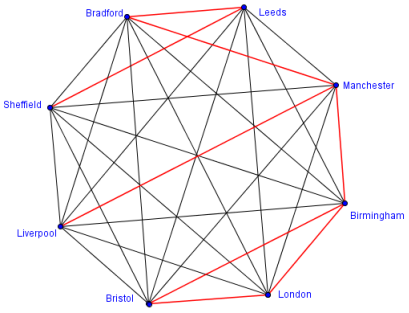
For this application of the 'Nearest Neighbour' algorithm, starting from London, we get:

London – Birmingham – Manchester – Liverpool – Bradford – Leeds – Sheffield – Bristol – London (654)

This gives us an upper bound on our Travelling Salesman route:

The minimum cycle must have a length no greater than 654 miles.

Example: Using a Minimum Spanning Tree (with Kruskal's) to find a lower bound, removing London:

1. Removing London 	2. Bradford to Leeds (10) 	3. Liverpool to Manchester (34) 
4. Sheffield to Leeds (40) 	5. Bradford to Manchester (39) 	6. Manchester to Birmingham (86) 
7. Bristol to Birmingham (88) 	8. London to Birmingham (118) 	9. London to Bristol (119) 

Note that if there had been any ambiguity in the generation of a minimum spanning tree, although the edges chosen may have differed, the length would still be the same, since Kruskal's and Prim's algorithms both produce a *minimum* spanning tree. In the example above, Kruskal's algorithm is used: adding edges in order of size except where an edge would cause a cycle. (Prim's is often quicker if a matrix is available.) The last two steps involve adding the two shortest edges from the removed vertex back into the graph.

The final collection of edges has a total weighting of: $10 + 34 + 40 + 39 + 86 + 88 + 118 + 119 = 534$

This gives us a lower bound on our Travelling Salesman route:

The minimum cycle must have a length no lower than 534 miles.

SOLUTIONS: Upper and Lower Bounds from each city

City	*London*	*Birmingham*	*Leeds*	*Sheffield*	*Bradford*	*Liverpool*	*Manchester*	*Bristol*
Upper Bound	654	611	658	**595**	662 (or 674)	631	647	611
Lower Bound	534	**581**	463	463	459	482	486	534

Note: Following the 'Nearest Neighbour' algorithm starting from Bradford yields two distinct routes (with different total lengths) due to Sheffield – Liverpool and Sheffield – Manchester both being 78 miles. The most useful of the two is the one linking Sheffield and Liverpool, yielding an upper bound of 662.

The **best upper bound** is the **lowest** one: 595 miles. The **best lower bound** is the **highest** one: 581 miles.

Therefore the length of the minimum route must lie between these two:

$$\mathbf{581} \leq \boldsymbol{x} \leq \mathbf{595}$$

Extra Information: A complete enumeration of the 2520 Hamiltonian cycles (with help from a computer)

Note: this section is included for information only – for most problems of this sort, the brute-force method applied here would be at best prohibitively time-consuming and expensive and at worst, impossible with current technology.

The *least* efficient cycle possible is a route of length 1204 miles: London – Manchester – Sheffield – Liverpool – Bristol – Leeds – Birmingham – Bradford – London		**The *most* efficient cycle possible is a route of length 595 miles**: London – Sheffield – Leeds – Bradford – Manchester – Liverpool – Birmingham – Bristol – London	

Note: the best route in this case is the one found using the Nearest Neighbour algorithm from Sheffield.

The median length of route was 924 miles, and all of our upper bounds are in the best 1% of routes.

Further Reading: Variations on the Travelling Salesman Problem

An alternative algorithm to the 'Nearest Neighbour' is the '**Cheapest Link**'. Another greedy algorithm, it starts with the lowest weighted edge, and adds edges in order of size provided adding the edge does not produce any cycles (unless the complete Hamiltonian cycle) and does not increase the order of any vertex beyond 2. For the 8 cities problem, for instance, 'Cheapest Link' generates a cycle of length 631 miles (at least as good as most of our 'Nearest Neighbour' upper bounds). In general, since 'Nearest Neighbour' can produce more upper bounds, it is better for finding a best upper bound.

The **asymmetric Travelling Salesman Problem** allows different weightings for, say, Leeds to Bradford and Bradford to Leeds. With a complete directed graph, similar algorithms to those we have used can generate upper and lower bounds in the same way. There will be twice as many Hamiltonian cycles for a given number of vertices because a cycle and its reverse could be different for a digraph.

The **Bottleneck Travelling Salesman Problem** modifies the requirements to minimise the length of the longest edge to be included in the route. It has the same computational difficulty as the standard TSP.

The TSP is a specific case of a more general problem, the **Travelling Purchaser Problem**. In this scenario, each edge weighting represents the cost of travel as before, but additionally each vertex represents a marketplace where certain goods are for sale at different prices. The challenge is to find the optimal route that minimises expenditure on both travelling and purchases for a given list of goods.