**Traveling Salesperson Problem**

Criteria:

* Salesperson travels between N cities
* Salesperson finishes where he/she started
* Salesperson visits each city exactly once (but the starting/finishing city)
* All cities are connected to each other directly
* Each node (way between cities) has a weight: its difficulty to travel using this node

Goal: Find the easiest/shortest route

**Algorithms:**

**Branch and bound algorithms:** This algorithm consists of a systematic listing of possible solutions. It is thought to form a rooted tree with the full set at the root. The algorithm checks the branches of the tree, which represent the subsets of the solution sets. It then compares the possible solutions against each other to find the optimal answer.

**Heuristics approaches:** This algorithm uses a set of guiding rules to select the next node to visit. But heuristics result in approximations, so there is no guarantee to get the optimal solution. However, high quality admissible heuristics can find a useful solution in a fraction of the time required by a brute force approach like the Branch and Bound Algorithm.

An example of a heuristic for a node would be a summation of how many unvisited nodes are "close by" a connected node. This could encourage the salesman to visit a group of close-by nodes clustered together before moving onto another natural cluster in the graph.

The easiest but most expensive solution is to simply try all possibilities. (for N cities you get (N-1)! possibilities). This means that for only 10 cities there are over 180 thousand combinations to try (since you can define the starting city, there can be variations on the remaining nine). We only count half since each route has an equal route in reverse with the same length or cost.

* (10-1)! /2 = 181440

There are algorithms that can find an approximate solution in polynomial time. The website <http://www.math.uwaterloo.ca/tsp/> gives several applications where such algorithms were applied.