The Traveling Salesman Problem (TSP)
Discrete Mathematics

Weighted Graphs

Graphs that have a number assigned to each edge are called **weighted graphs**.

A more mathematical definition: A **weighted graph** is a graph $G = (V, E, w)$, where V is a set of vertices, E is a set of edges between vertices of V, and w is a function from $V \times V$ into $\mathbb{R}$. The function w gives to each pair of vertices a weight in $\mathbb{R}$.

Note 1: Usually $w(u, v)$ are positives for all pairs of vertices.
Note 2: Usually $w(u, v) = \infty$ if the pair of vertices (u, v) is not an edge of E.

Length of a Path

Let the path p, form vertices a to z, be given by the sequence of edges $e_1, e_2, \ldots, e_n$ of the graph, such that $f(e_1) = (a, x_1)$, $f(e_2) = (x_1, x_2)$, ..., $f(e_n) = (x_{n-1}, z)$.

The **length** $L(p)$ **of a path** p, in a weighted graph, is the sum of the weights of the edges of this path, i.e.
$L(p) = w(a, x_1) + w(x_1, x_2) + \cdots + w(x_{n-1}, z)$.

A common problem is to find the path, with minimal length, between two given vertices of a weighted graph.

Traveling Salesman Problem

The **Traveling Salesman Problem** (TSP) is a problem in combinatorial optimization studied in operations research and theoretical computer science. Given a list of cities and their pairwise distances, the task is to find a shortest possible tour that visits each city exactly once.
The problem was first formulated as a mathematical problem in 1930 and is one of the most intensively studied problems in optimization. It is used as a benchmark for many optimization methods. Even though the problem is computationally difficult, a large number of heuristics and exact methods are known, so that some instances with tens of thousands of cities can be solved.

Traveling Salesman Problem

The TSP has several applications even in its purest formulation, such as planning, logistics, and the manufacture of microchips. Slightly modified, it appears as a sub-problem in many areas, such as DNA sequencing. In these applications, the concept city represents, for example, customers, soldering points, or DNA fragments, and the concept distance represents traveling times or cost, or a similarity measure between DNA fragments. In many applications, additional constraints such as limited resources or time windows make the problem considerably harder.
In the theory of computational complexity, the decision version of TSP belongs to the class of NP-complete problems. Thus, it is assumed that there is no efficient algorithm for solving TSPs. In other words, it is likely that the worst case running time for any algorithm for TSP increases exponentially with the number of cities, so even some instances with only hundreds of cities will take many CPU years to solve exactly.

As a graph problem

The TSP can be modelled as a weighted undirected simple graph, such that cities are the graph's vertices, paths are the graph's edges, and a path's distance is the edge's length. A TSP tour becomes a Hamiltonian cycle, and the optimal TSP tour is the shortest Hamiltonian cycle. Often, the model is a complete graph (i.e. an edge connects each pair of vertices). If no path exists between two cities, adding an arbitrarily long edge will complete the graph without affecting the optimal tour.

Exact Algorithm

The most direct solution would be to try all permutations (ordered combinations) and see which one is cheapest (using brute force search). The running time for this approach lies within a polynomial factor of $O(n!)$, the factorial of the number of cities, so this solution becomes impractical even for only 20 cities.
The exact number of possible tours is $\frac{1}{2}(n-1)!$ where n is the number of cities. To compute the length of a tour, there are $n-1$ numbers to add. The total cost is then about $\frac{1}{2}(n-1)(n-1)!$ floating point operations. Suppose that the computer use one nanosecond $= 10^{-9}$ s to do one operation, which is 1 000 000 000 operations per second. Then for 20 cities, there are $\frac{1}{2}(20-1)(20-1)! = 1.15562 \times 10^{18}$ possible tours. At a rate of 1 000 000 000 operations per second, the computer will need 1.1556×10^{9} seconds, 19260474 minutes, 321007 hours, 13375 days and 36.6 years. For 40 cities, the computer will need 1.26×10^{31} years.