Problem Description

The **travelling salesman problem** (**TSP**) asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?". Until now, researchers have not found a polynomial time algorithm for the travelling salesman problem. 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 $O(n!)$, of the factorial of the number of cities, so this solution becomes impractical even for only 20 cities. One of the earliest applications of dynamic programming is the Held–Karp algorithm that solves the problem in time $O(n^2.2^n)$. This bound has also been reached by Exclusion-Inclusion in an attempt preceding the dynamic programming approach.

Held-Karp Algorithm

There is an optimization property associated with TSP problem that provides a recursive formulation to the problem given as:

"Every sub-path of a shortest path is itself of shortest distance."

The Held-Karp algorithm suggests a dynamic programming procedure based on the above property, stated as: Compute the solutions of all sub-problems starting with the smallest. Whenever computing a solution requires solutions for smaller problems using the above recursive equations, look up these solutions which are already computed. To compute a minimum distance tour, use the final equation to generate the 1st node, and repeat for the other nodes. For this problem, we cannot know which sub-problems we need to solve, so we solve them all. The exact time complexity, in terms of number of addition and comparison operations, can be shown to be $(n-1)(n-2)2^{n-3}+(n-1)$.

Held-Karp Example

Let $C(a,S,b)$ be the cost of the shortest path from city a, through the cities in set S, to city b

For distance matrix = $\begin{pmatrix} 0 & 3 & 1 & 1 \\ 3 & 0 & 2 & 5 \\ 1 & 2 & 0 & 6 \\ 1 & 5 & 6 & 0 \end{pmatrix}$

Starting with 2 cities one-way paths, setting the cost to the corresponding entry

$C(1,\emptyset,2)=3$

$C(1, \emptyset, 3) = 1$

$C(1, \emptyset, 4) = 1$

3 cities one way paths obtained by adding one by one, all unvisited nodes to each 2 cities-paths

$C(1, \{2\}, 3) = 3 + 2 = 5$

$C(1, \{2\}, 4) = 3 + 5 = 8$

$C(1, \{3\}, 2) = 1 + 2 = 3$

$C(1, \{3\}, 4) = 1 + 6 = 7$

$C(1, \{4\}, 2) = 1 + 5 = 6$

$C(1, \{4\}, 3) = 1 + 6 = 7$

Similar step is repeated to compute 4 cities paths, upon remembering that for the occurrence of cost of the same set S, we take the minimum of the costs.

$C(1, \{2, 4\}, 3) = 8 + 6 = 14$, discarded since $C(1, \{2, 4\}, 3) > C(1, \{4, 2\}, 3)$

$C(1, \{2, 3\}, 4) = 5 + 6 = 11$, discarded since $C(1, \{2, 3\}, 4) > C(1, \{3, 2\}, 4)$

$C(1, \{3, 2\}, 4) = 3 + 5 = 8$

$C(1, \{3, 4\}, 2) = 7 + 5 = 12$, discarded since $C(1, \{3, 4\}, 2) > C(1, \{4, 3\}, 2)$

$C(1, \{4, 2\}, 3) = 6 + 2 = 8$

$C(1, \{4, 3\}, 2) = 7 + 2 = 9$

Computing the cost of each Hamiltonian path resulting from n=4 cities paths

$C(1 \to 3 \to 2 \to 4 \to 1) = 8 + 1 = 9$

$C(1 \to 4 \to 2 \to 3 \to 1) = 8 + 1 = 9$

$C(1 \to 4 \to 3 \to 2 \to 1) = 9 + 3 = 12$

Thus, $1 \to 3 \to 2 \to 4 \to 1$ is the minimum cost Hamiltonian cycle, appeared in clockwise as well as counter-clockwise due to undirected graph.

Note: The above time complexity only counts two types of operations (number addition and comparison) and does not takes into account the list-copy and list-checking operations, which are limitations of the python implementation of the algorithm. Applying them, requires a greater time complexity, formulated as $(n-1)^2 . 2^{n-2} + 3(n-1)$.

A Modification of Held-Karp Algorithm

We present a modified version of the above algorithm with time complexity reduced by 2 times and giving an ideal solution. The algorithm uses "path joining", instead of addition of just one node, in a recursive call. Let $P(a, S, b)$ represent the shortest path from city a, through the cities in set S, to city b. Let $f(A,\ c) = S - A - \{c\}$ be a function where the set $A \subset S$ with city $c \in S$ and $c \notin A$. Then the following recursion defines the problem to sub-problem transition in our algorithm.

$$P(a, S, b) = Min\{\, P(a, A, c) + P(c, f(A, c), b)\ for\ each\ c\ in\ S - A\ and\ for\ all\ sets\ A \subset S\ of\ constant\ cardinality\,\}$$

Where "+" is the joining operation between paths that preserves the order of two paths as in the normal sense. We can see that while the Held-Karp algorithm computes all shortest $i + 1$ length paths by adding one node to each i length path, the new algorithm can jump directly to length $2i - 1$ i.e. when $2|A| + 1 = |S|$. However, the above technique is only applied in the last phase in the algorithm i.e. computing n length one-way paths from $\frac{n-1}{2}$ length one-way paths. The reason for that is if the same technique is applied for each recursive call, one can only solve TSP for $n = 3, 5, 9, 17, \ldots\ i, 2i - 1$ and so on. Even for the last phase, an efficiency of twice execution speed has been observed, reflected in the following chart.

Number of cities, n	Time noted for Held-Karp algorithm in seconds	Time noted for the new algorithm in seconds
16	9.3	4.8
17	20.9	14.3
18	47.7	27.1
19	110.2	72.7
20	278.2	138.0

Example Solved by the New Algorithm

Consider the following undirected distance matrix, of 5 nodes

$$\begin{pmatrix} 0 & 3 & 1 & 1 \\ 3 & 0 & 2 & 5 \\ 1 & 2 & 0 & 6 \\ 1 & 5 & 6 & 0 \end{pmatrix}$$

The computer calculates 3-cities one-way paths as done in the Held-Karp algorithm.

$C(1, \{2\}, 3) = 3 + 2 = 5$

$C(1,\{2\},4) = 3 + 5 = 8$

$C(1,\{3\},2) = 1 + 2 = 3$

$C(1,\{3\},4) = 1 + 6 = 7$

$C(1,\{4\},2) = 1 + 5 = 6$

$C(1,\{4\},3) = 1 + 6 = 7$

Now, we join each pair of 3-cities paths with same end city and disjoint set S, to get $3(2) - 1 = 5$ cities Hamiltonian paths (with 1 at both ends). As an example, consider the following join operation:

$$C(1 \to 2 \to 3) + C(3 \to 4 \to 1) = 5 + 7 = 12$$

Needless to say, we will discard the paths with greater costs. We have the following Hamiltonian paths:

$C(1 \to 2 \to 3 \to 4 \to 1) = 5 + 7 = 12$

$C(1 \to 2 \to 4 \to 3 \to 1) = 8 + 7 = 15$

$C(1 \to 3 \to 2 \to 4 \to 1) = 3 + 6 = 9$

Hence, the optimum solution is taken as $1 \to 3 \to 2 \to 4 \to 1$