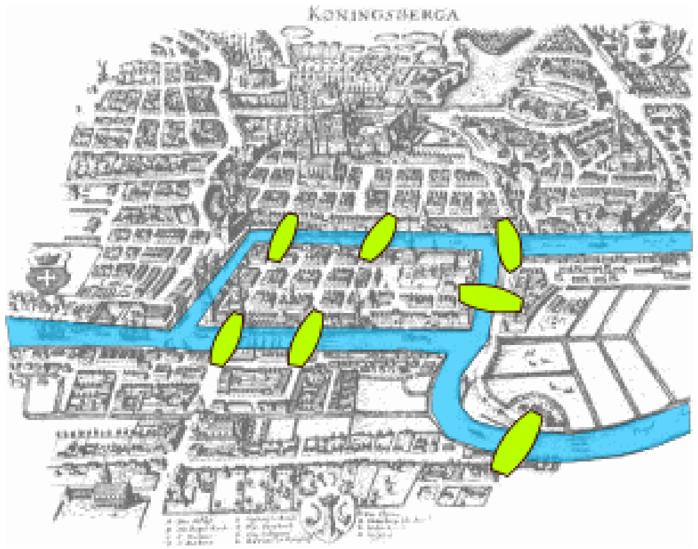
Traveling Salesperson Problem

Outline

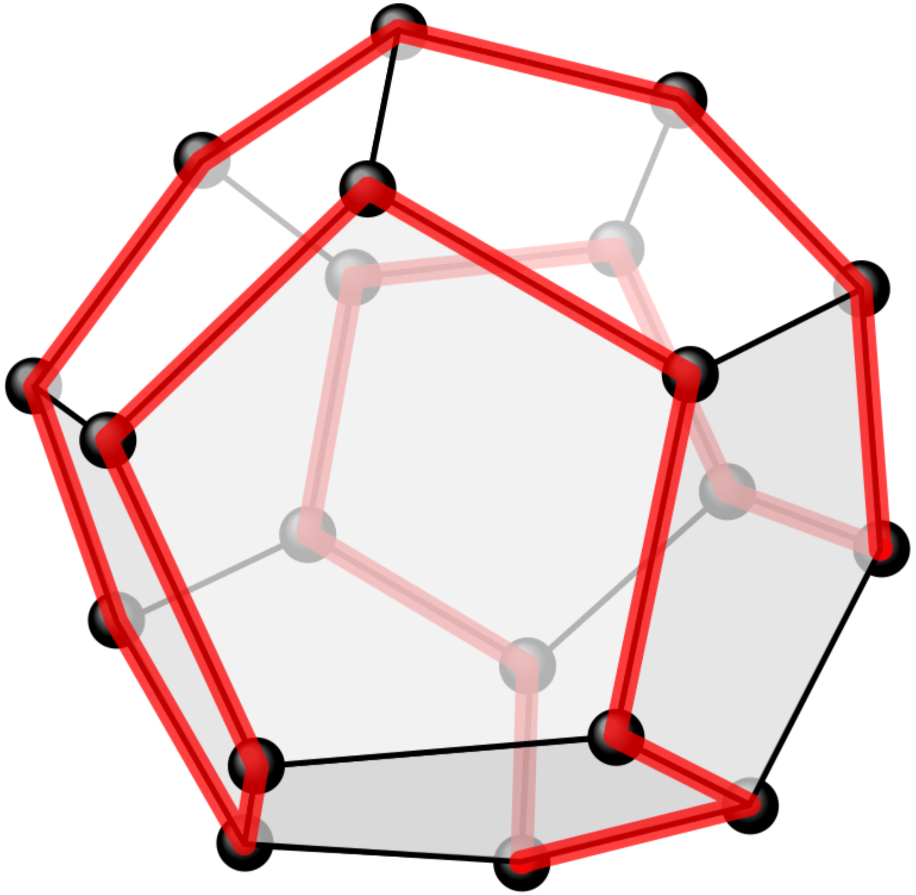
- Problem Background
- Notable Variants
- Popular Heuristics
- Christofides Algorithm

Graph Theory and Eulerian Cycles



- In 1736 Leonhard Euler laid the foundations of graph theory with the Königsberg Bridge problem
- He wanted to take a walk where he would pass over each bridge exactly once
- An Eulerian Cycle (circuit) is a path that goes over each edge once and returns to the origin
- Euler proved that in order to have such a circuit all nodes needed to have an even degree

TSP Origins



- William Rowan Hamilton invented the icosian game which involved finding a path that visits every vertex of a dodecahedron once
- A **Hamiltonian cycle** is the general term in graph theory for a cycle on an undirected graph that visits every node once and then return to the origin (starting node)

Integer Programming Formulation

$$x_{ij} = \begin{cases} 1 & \text{the path goes from city } i \text{ to city } j \\ 0 & \text{otherwise} \end{cases}$$

$$\min \sum_{i=1}^{n} \sum_{j \neq i, j=1}^{n} c_{ij} x_{ij}:$$

$$x_{ij} \in \{0,1\} \qquad i,j = 1,\ldots,n;$$

$$\sum_{i=1, i \neq j}^{n} x_{ij} = 1 \qquad j = 1,\ldots,n;$$

$$\sum_{j=1, j \neq i}^{n} x_{ij} = 1 \qquad i = 1,\ldots,n;$$

$$\sum_{i \in Q} \sum_{j \neq i, j \in Q} x_{ij} \leq |Q| - 1 \qquad \forall Q \subsetneq \{1,\ldots,n\}, |Q| \geq 2$$

- IP solutions are slow compared to heuristic and meta-heuristic approaches
- There is a lot about the specific geometry of the problem that can be leveraged for a better solution

The Cutting-Plane Method

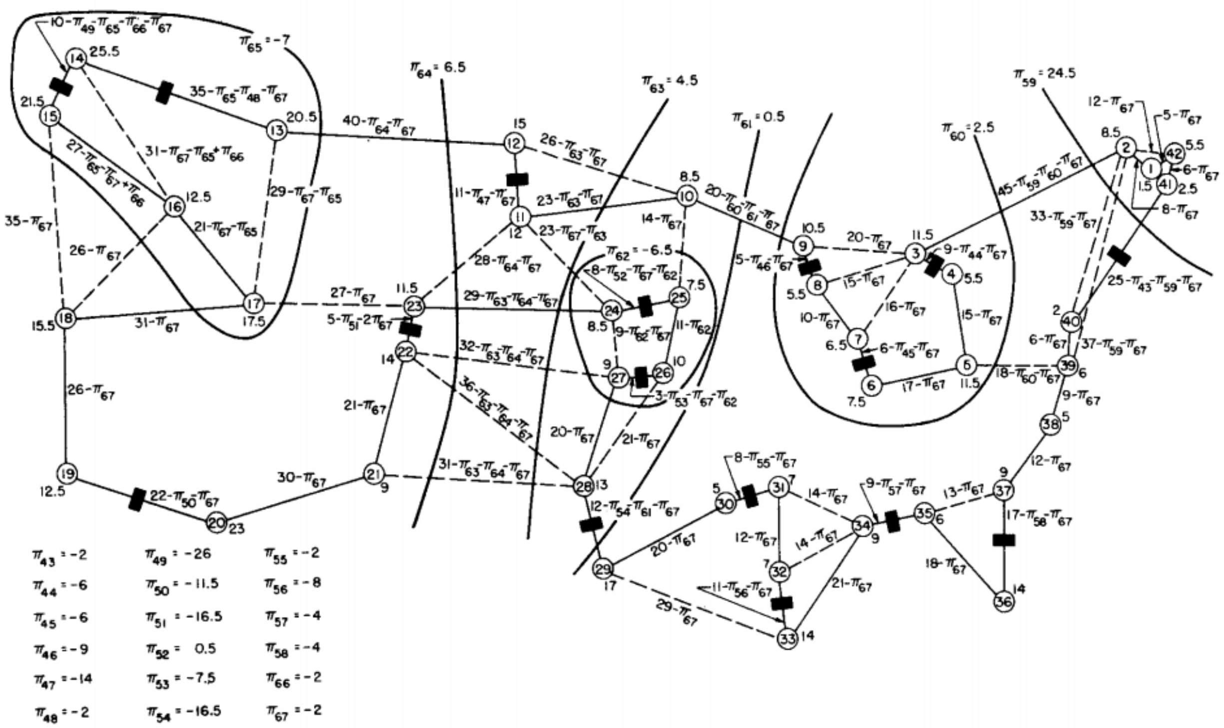


FIG. 17. Only the right-hand side of the equations satisfied by π_I are shown on the map; the left-hand side on line (I,J) is $\pi_I + \pi_J$. Dotted links (I,J) correspond to additional basic variables x_{IJ}.

SOLUTION OF A LARGE-SCALE TRAVELING-SALESMAN PROBLEM*

G. DANTZIG, R. FULKERSON, AND S. JOHNSON

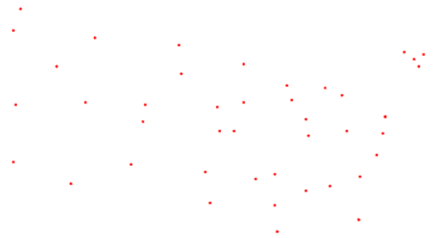
The Rand Corporation, Santa Monica, California

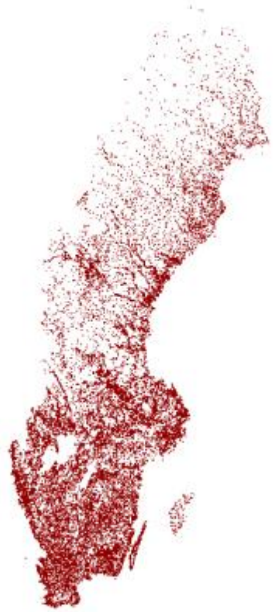
(Received August 9, 1954)

It is shown that a certain tour of 49 cities, one in each of the 48 states and Washington, D. C., has the shortest road distance.

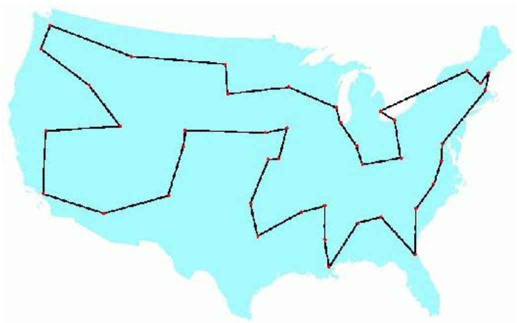
THE TRAVELING-SALESMAN PROBLEM might be described as follows: Find the shortest route (tour) for a salesman starting from a given city, visiting each of a specified group of cities, and then returning to the original point of departure. More generally, given an n by n symmetric matrix $D=(d_{IJ})$, where d_{IJ} represents the 'distance' from I to J, arrange the points in a cyclic order in such a way that the sum of the d_{IJ} between consecutive points is minimal. Since there are only a finite number of possibilities (at most $\frac{1}{2}(n-1)!$) to consider, the problem is to devise a method of picking out the optimal arrangement which is reasonably efficient for fairly large values of n. Although algorithms have been devised for problems of similar nature, e.g., the optimal assignment problem,[3,7,8] little is known about the traveling-salesman problem. We do not claim that this note alters the situation very much; what we shall do is outline a way of approaching the problem that sometimes, at least, enables one to find an optimal path and prove it so. In particular, it will be shown that a certain arrangement of 49 cities, one in each of the 48 states and Washington, D. C., is best, the d_{IJ} used representing road distances as taken from an atlas.

TSP Origins



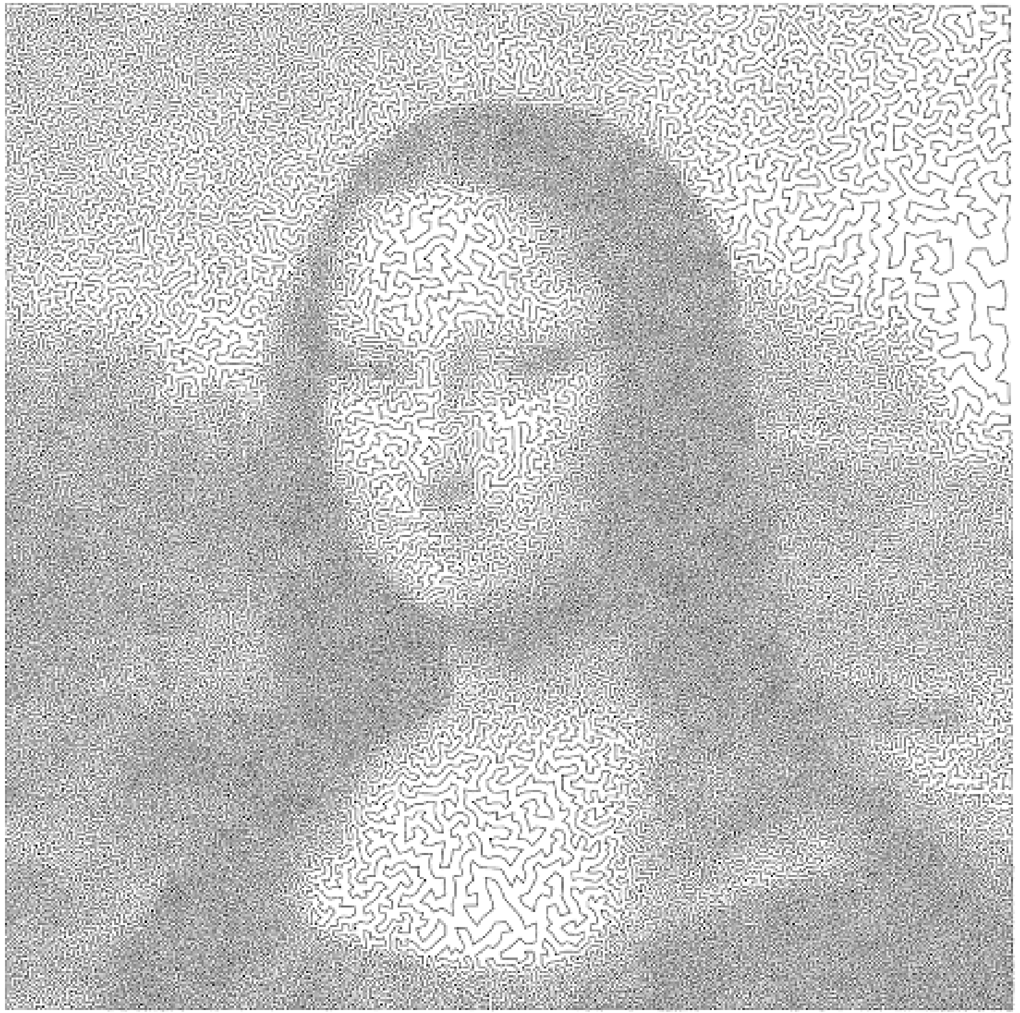


50 yrs



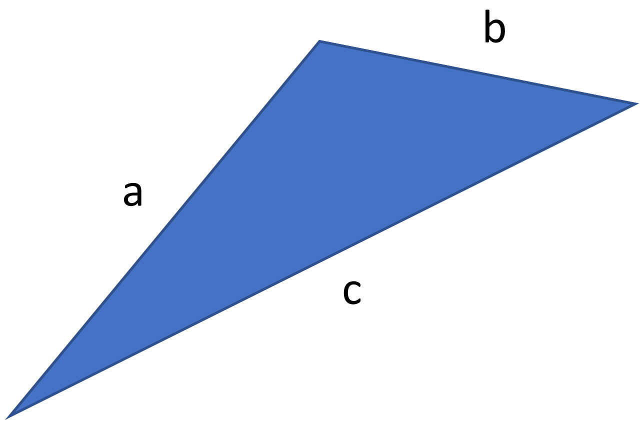
- The general form of the TSP and its mathematical treatment was first worked on in the 1930s
- In 1954 Dantzig (of LP fame) presented a solution for an instance with 49 cities (6,206,957,796,268,036,335,431,144,523,686,687,519,260,743,177,338,880,000,000,000 possible tours)
- In 2004 a paper with a solution for 24,978 cities was published

TSP Today



- Today there exist excellent opensource solvers that can solve small problems to optimality and do well on much larger ones [http://www.math.uwaterloo.ca/tsp/concorde.html]
- People continue to solve larger and larger problems using super computers taking decades worth of compute time on a single processor

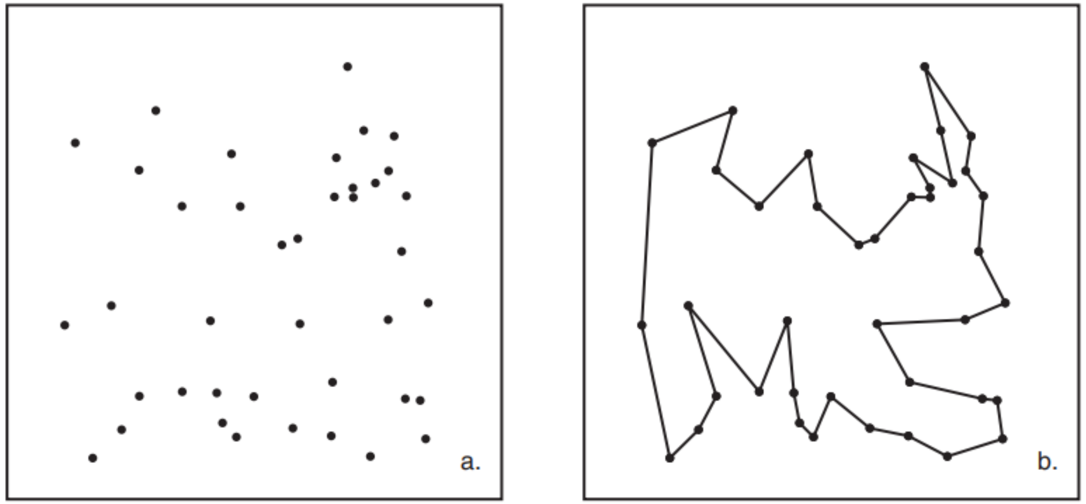
On a Graph vs On a Plane

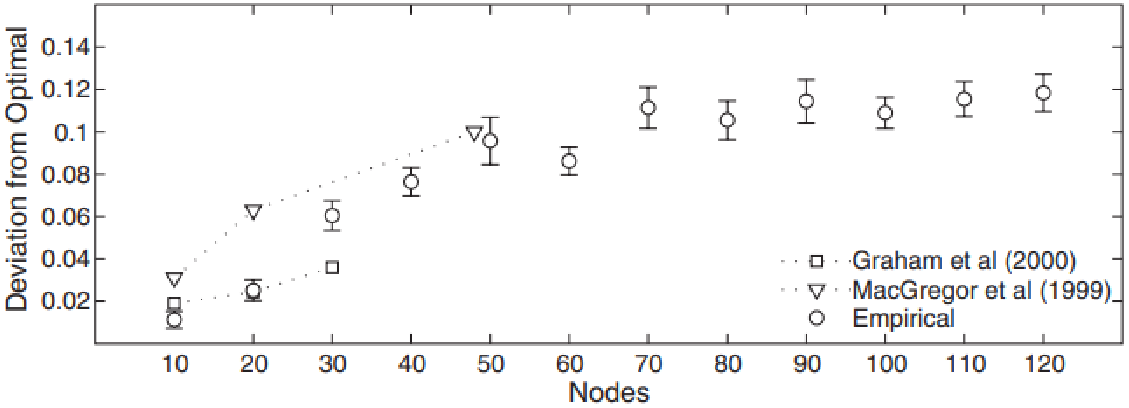


$$a + b \geq c$$

- There is an important distinction between cases that obey a triangle inequality and those that do not
- Performance of some algorithms can be arbitrarily bad on graphs that do not adhere to a triangle inequality

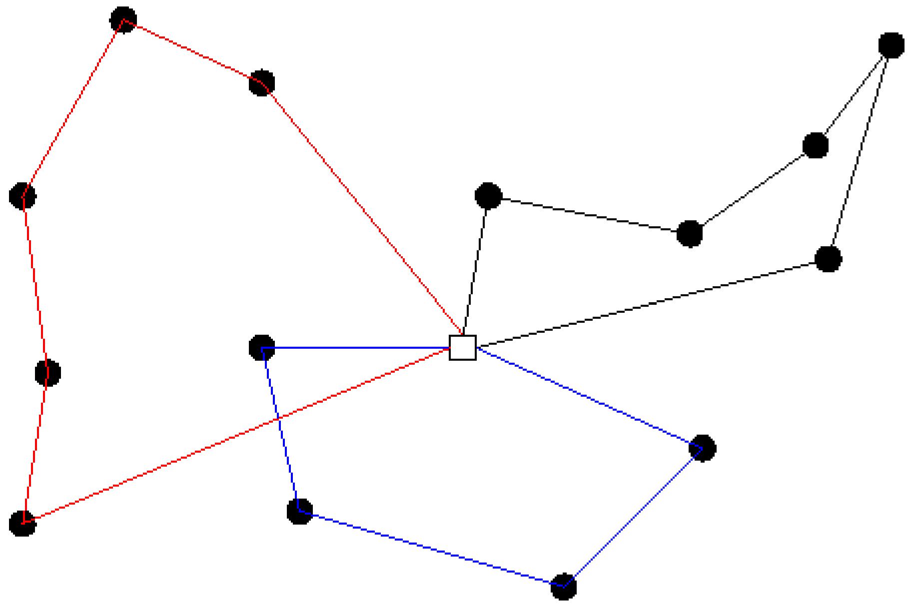
Human Solutions





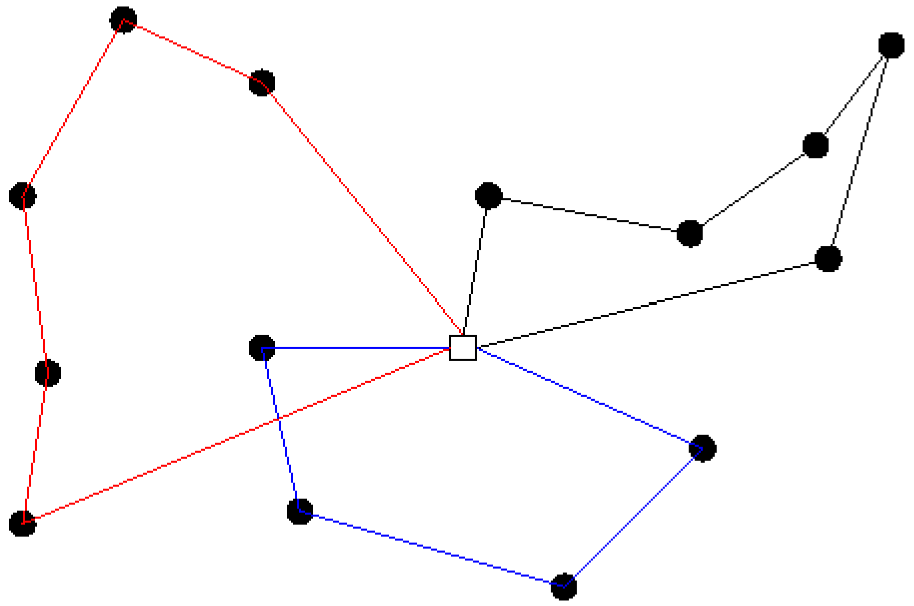
- Human's perform fairly well on the Euclidian TSP finding near optimal solutions for problems with 10-20 nodes
- For problems with more than 70 nodes the error is around 10%
- https://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=069B20F8000AD6C3D4100E5868F9B32F?doi=10.1.1.360.9763&rep=rep1&type=pdf

TSP as a special case of VRP



- The Vehicle Routing Problem is a more generalized version of the TSP that involves minimizing the distance of multiple vehicles traveling between the vertices
- The TSP is just a VRP with a single truck
- VRP can be broken into a clustering (which nodes are visited by the same vehicle) and TSP problem (what order are those nodes visited)

TSP as a special case of VRP



- Transportation accounts for about 10% of the cost of most items
- Small algorithmic improvements can lead to big changes as UPS, USPS, FedEx, etc control thousands of vehicles that use millions of gallons of gasoline
- US uses 390 million gallons of gas per day, a 0.01% improvement in routing equates to over 128,000 tons less CO2 in a year

Many, many VRP Variants

FedEx

UNITED STATES POSTAL SERVICE

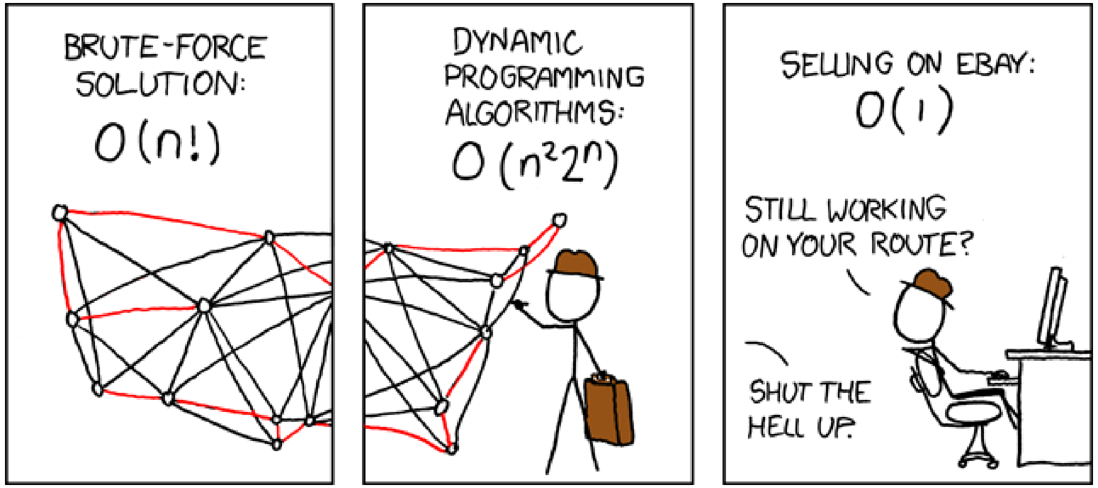
Walmart

amazon

ups

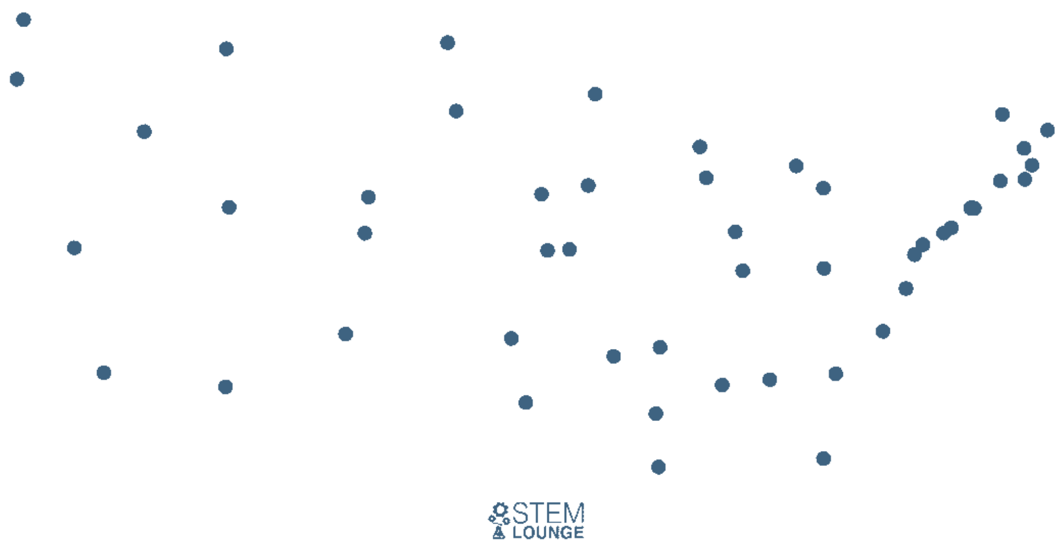
- Multi-Depot
- Capacity constraints
- Route-length constraints
- Time-windows
- Uncertainty in nodes serviced
- Mixed Fleet
- …

Complexity



- The general problem of finding the optimal solution is NP-hard
- The decision problem of, does there exist a route shorter than length x, is NP-complete
- https://xkcd.com/399/

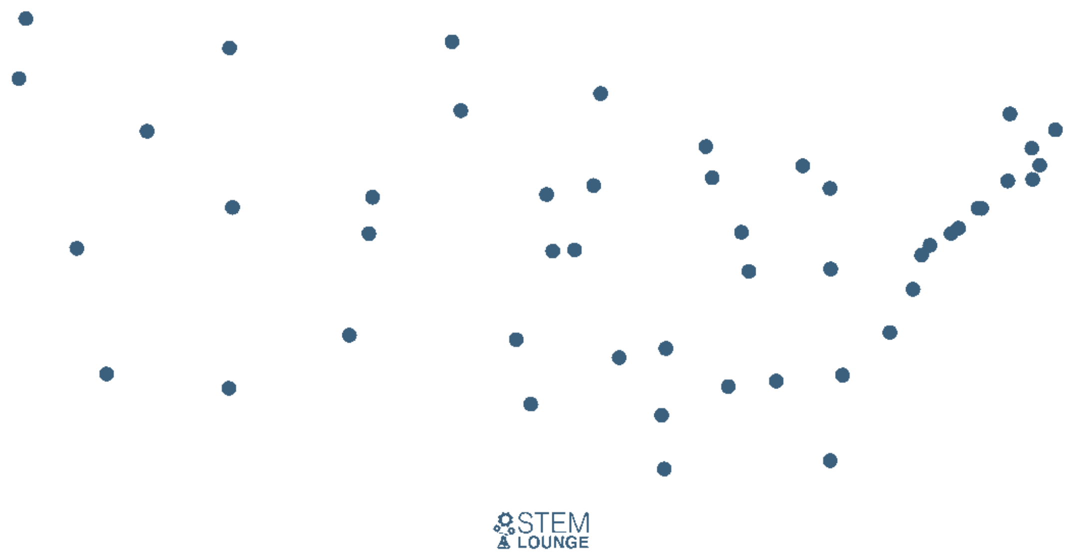
Greedy Algorithm



https://stemlounge.com/animated-algorithms-for-the-traveling-salesman-problem/

- Add the edge x_{ij}, with the minimum length c_{ij} to the solution
- Remove any edges from the candidate set that would create a cycle or have more than two edges attached to a node
- Repeat until no edges left, connect two nodes with only one edge

Nearest Neighbor

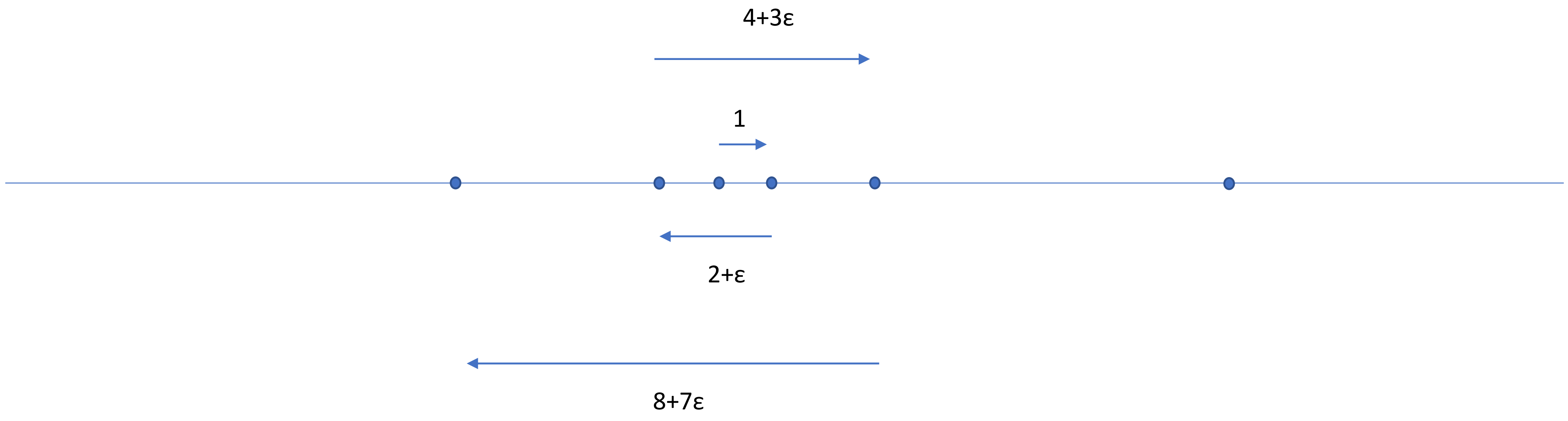


https://stemlounge.com/animated-algorithms-for-the-traveling-salesman-problem/

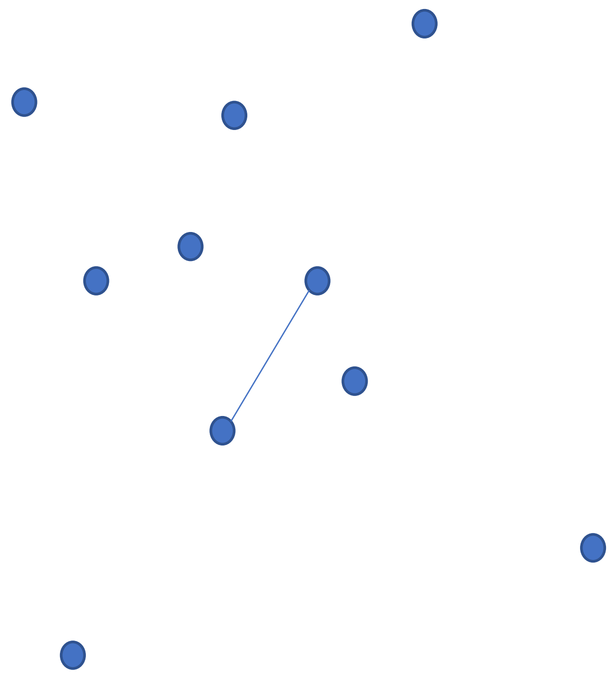
- Step 1: Choose a starting node
- Step 2: Find the next node is the closest node not in the path
- Step 3: Once all nodes have been visited, return to starting node

Worst Case Analysis

Given a bad starting node and the worst spacing of nodes possible, you can generate a worst-case scenario that shows how bad things can go

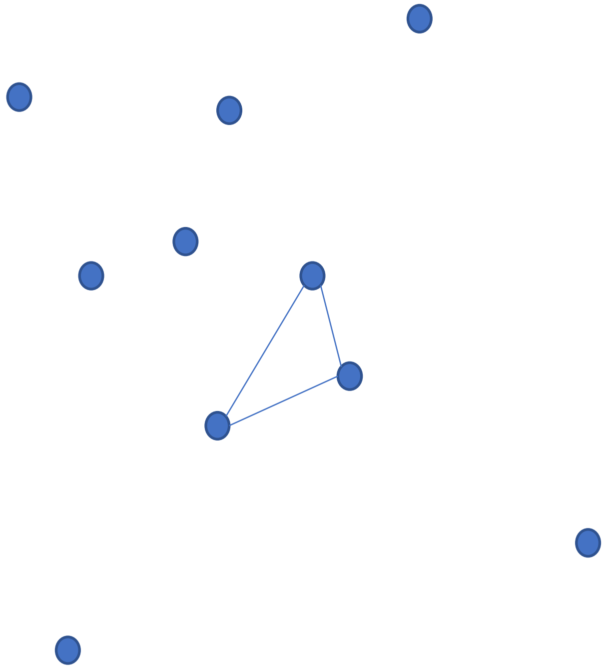


Nearest Insertion



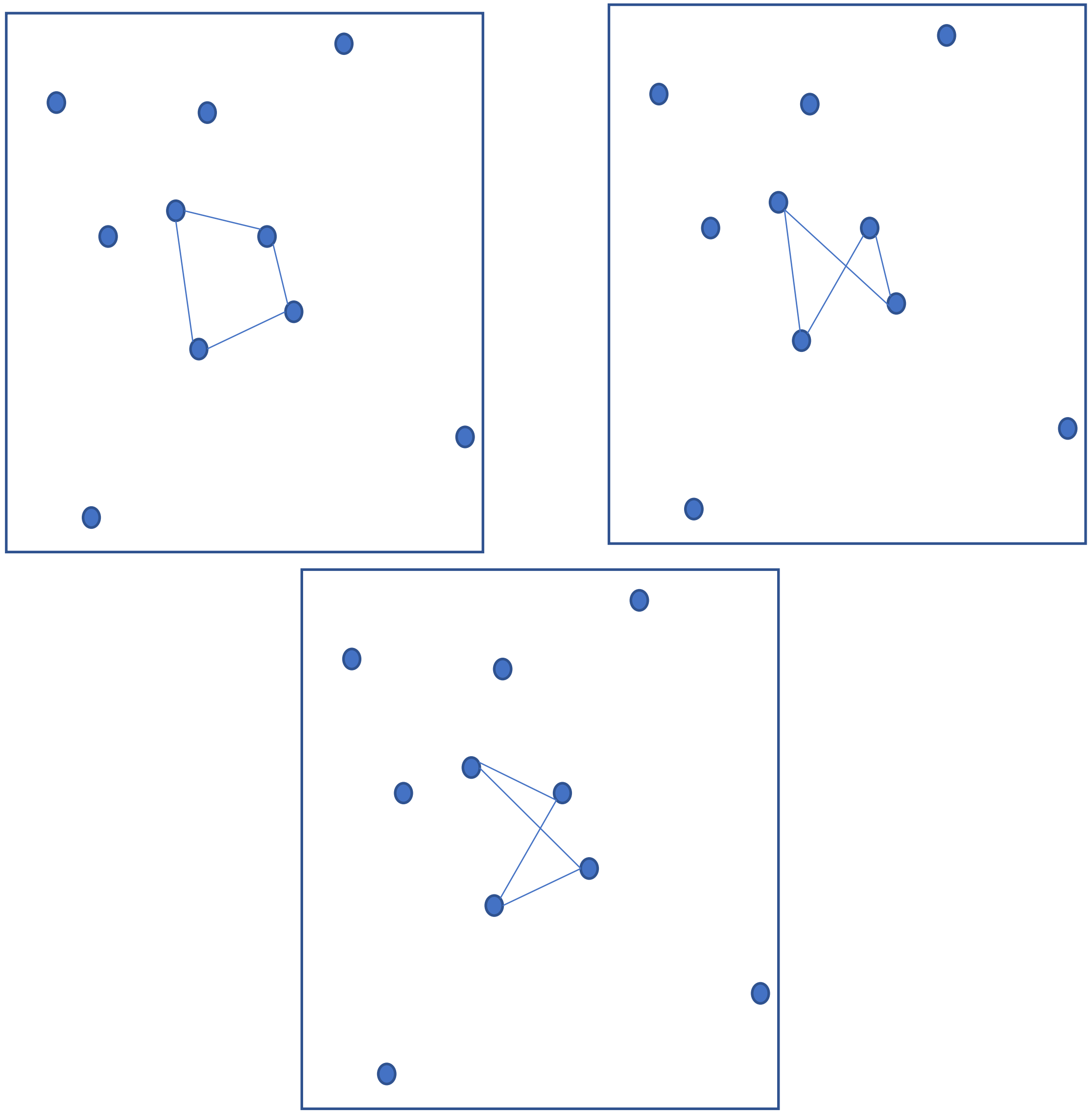
- Step 1: Choose 2 nodes to create a cycle
- Step 2: Add the closest node to the cycle in the way that minimizes the additional length,
  $$c_{ik} + c_{kj} - c_{ij}$$
- Step 3: Repeat Step 2 until all nodes are added

Nearest Insertion



- Step 1: Choose 2 nodes to create a cycle
- Step 2: Add the closest node to the cycle in the way that minimizes the additional length,
$$c_{ik} + c_{kj} - c_{ij}$$
- Step 3: Repeat Step 2 until all nodes are added

Nearest Insertion

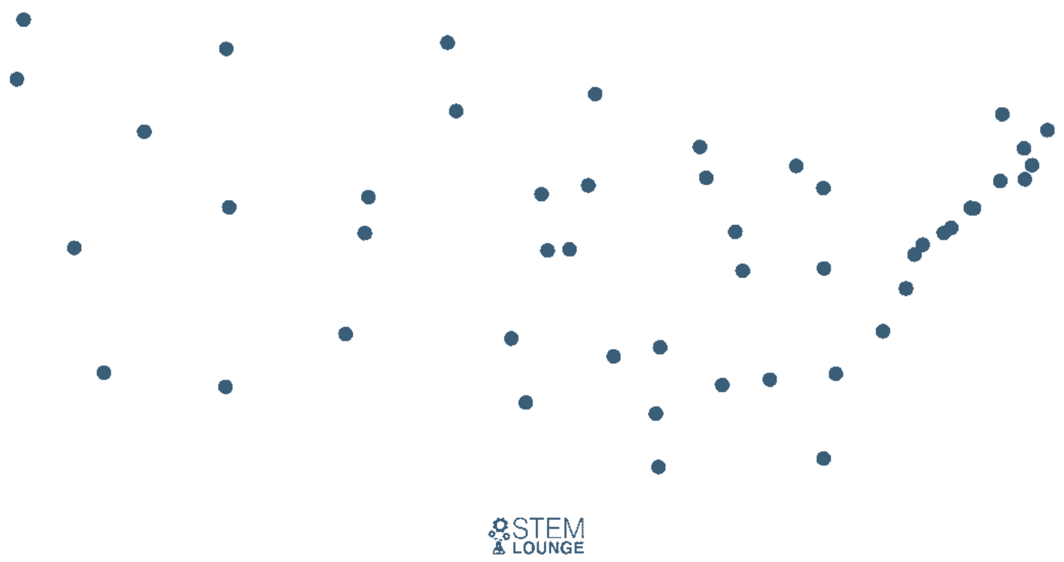


- Step 1: Choose 2 nodes to create a cycle
- Step 2: Add the closest node to the cycle in the way that minimizes the additional length,

$$c_{ik} + c_{kj} - c_{ij}$$

- Step 3: Repeat Step 2 until all nodes are added

Nearest Insertion



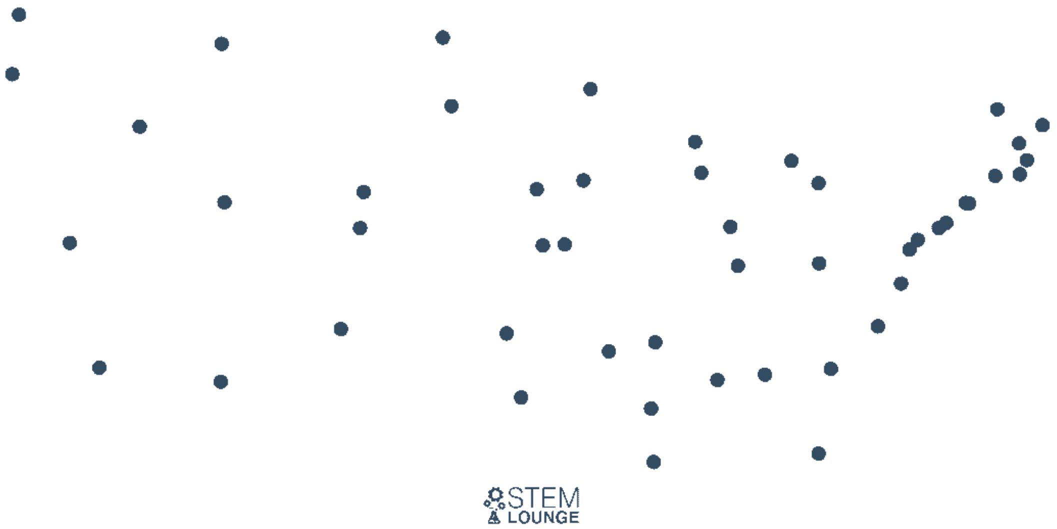
https://stemlounge.com/animated-algorithms-for-the-traveling-salesman-problem/

- Step 1: Choose 2 nodes to create a cycle
- Step 2: Add the closest node to the cycle in the way that minimizes the additional length,

$$c_{ik} + c_{kj} - c_{ij}$$

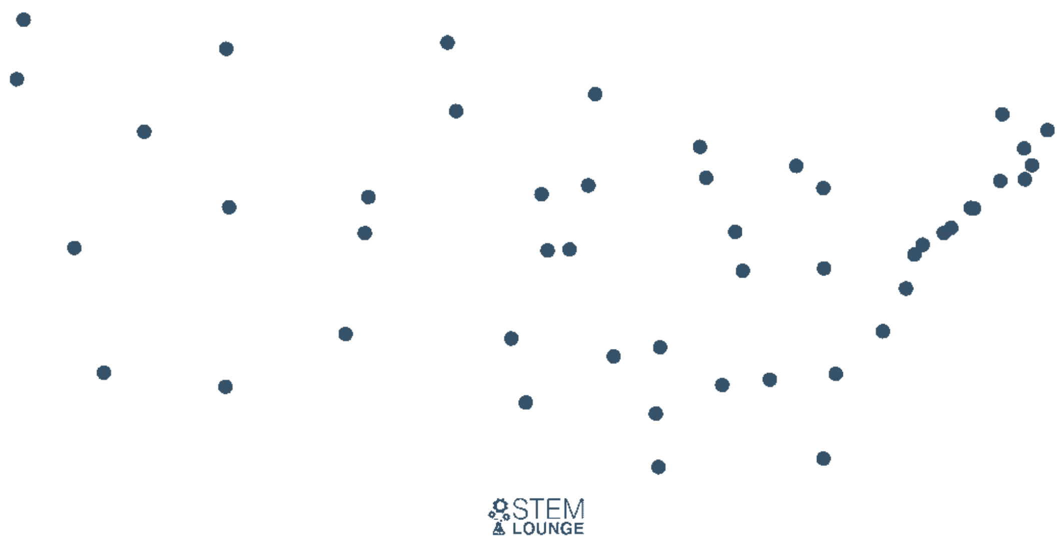
- Step 3: Repeat Step 2 until all nodes are added

Farthest Insertion



- Step 1: Choose a node to start
- Step 2: Add the farthest node to the cycle in the way that minimizes the additional length,
$$c_{ik} + c_{kj} - c_{ij}$$
- Step 3: Repeat Step 2 until all nodes are added

Random Insertion

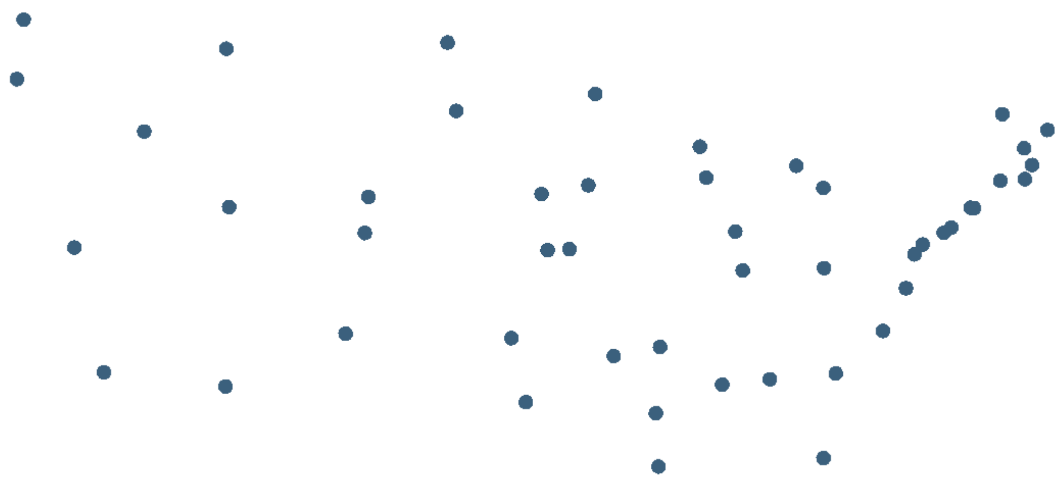


- Step 1: Choose a node to start
- Step 2: Add a random node to the cycle in the way that minimizes the additional length,

$$c_{ik} + c_{kj} - c_{ij}$$

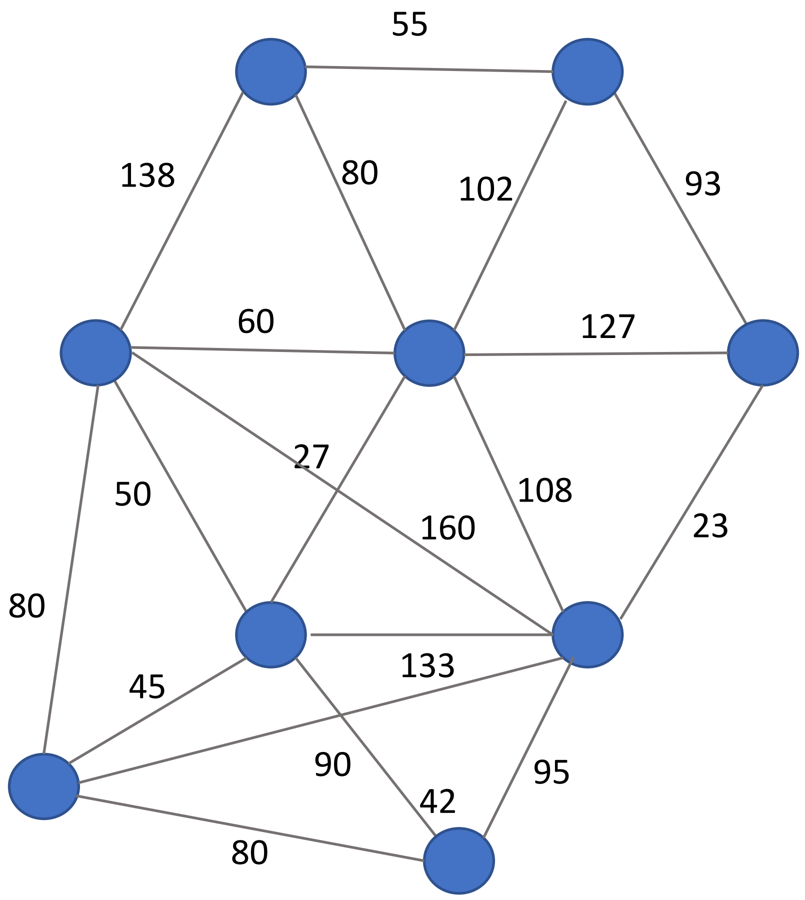
- Step 3: Repeat Step 2 until all nodes are added

Cheapest Insertion



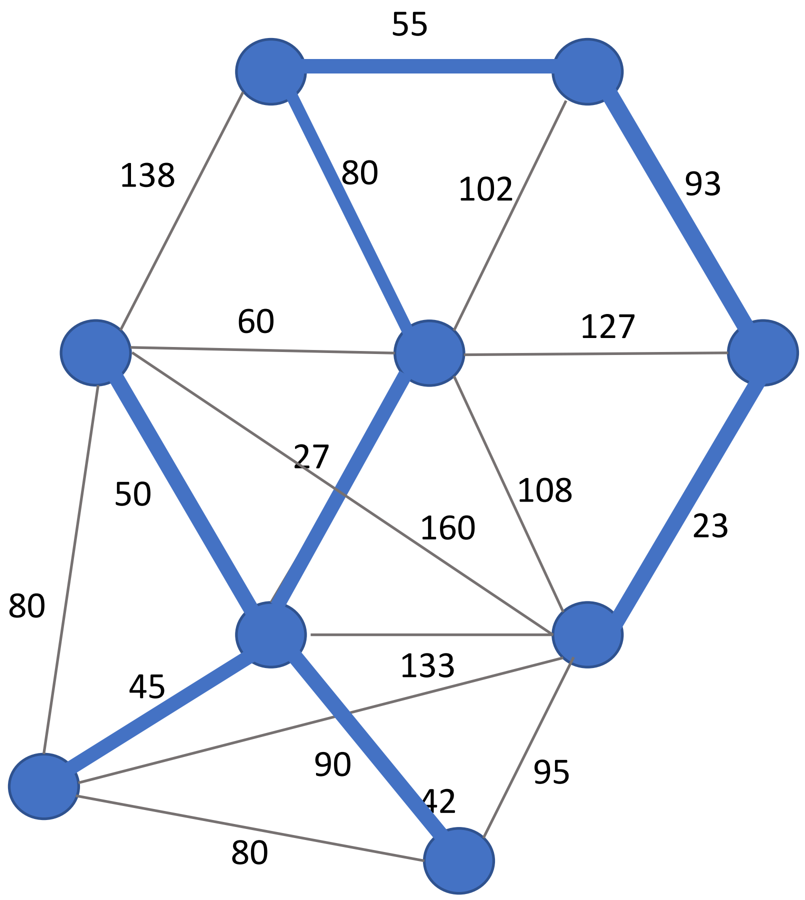
- Step 1: Start with a subgraph of one node, i
- Step 2: Find node such that c_{ik} is minimized and add to tour
- Step 3: Find (i,j) in subtour and k not, such that $c_{ik} + c_{kj} - c_{ij}$is minimized
- Step 4: Repeat until finished

Christofides' Algorithm



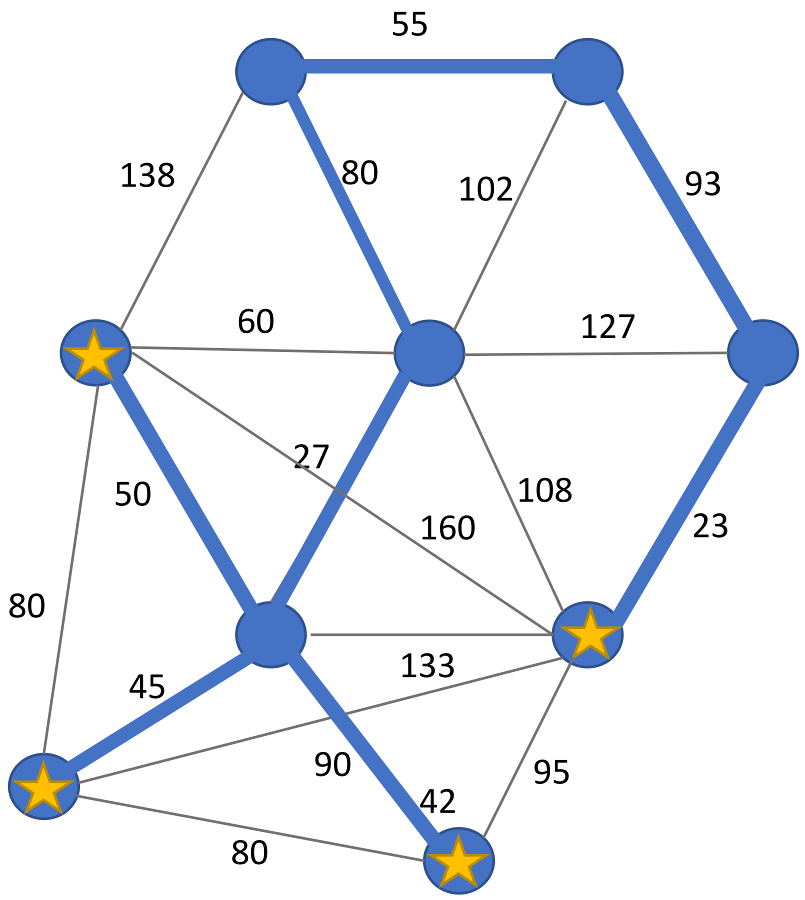
- Step 1: Find minimal spanning tree
- Step 2: Identify all odd degree nodes and optimally match adding links to the graph
- Step 3: Draw an Eulerian circuit on the resulting graph, removing repeated vertices to result in a Hamiltonian circuit

Christofides' Algorithm



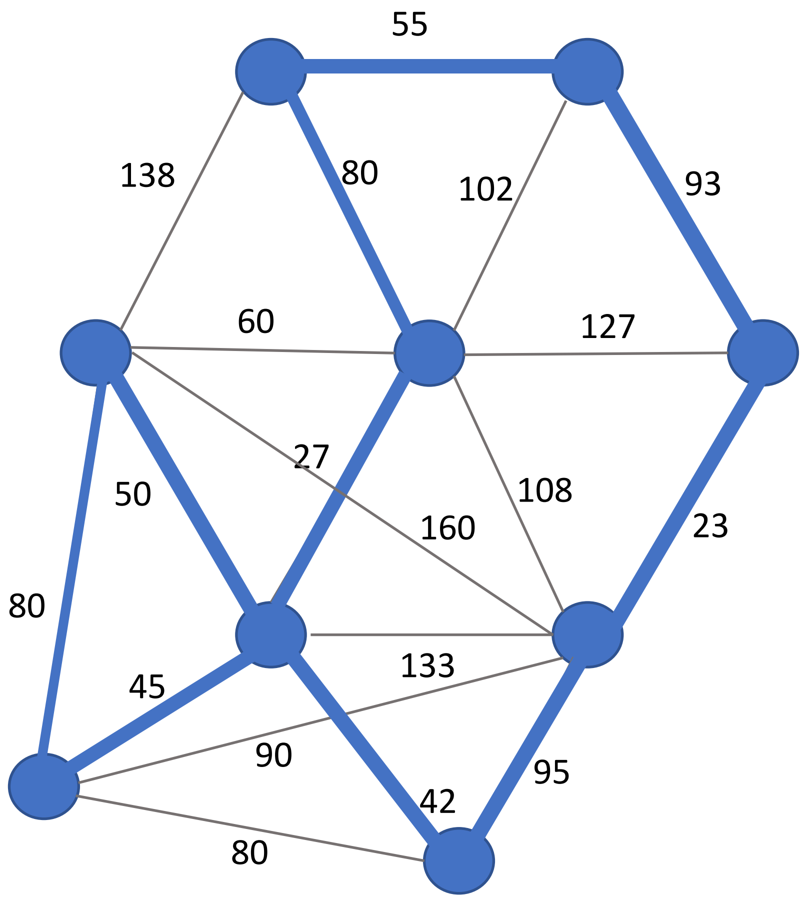
- Step 1: Find minimal spanning tree
- Step 2: Identify all odd degree nodes and optimally match adding links to the graph
- Step 3: Draw an Eulerian circuit on the resulting graph, removing repeated vertices to result in a Hamiltonian circuit

Christofides' Algorithm



- Step 1: Find minimal spanning tree
- Step 2: Identify all odd degree nodes and optimally match adding links to the graph
- Step 3: Draw an Eulerian circuit on the resulting graph, removing repeated vertices to result in a Hamiltonian circuit

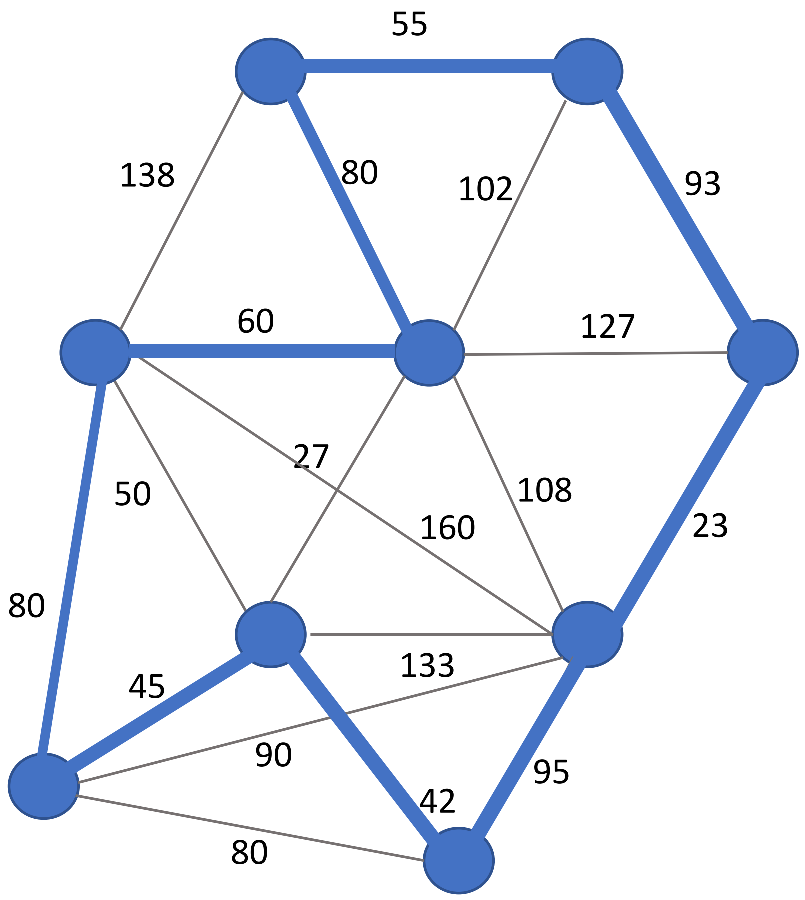
Christofides' Algorithm



95+80 =175
160+80=240

- Step 1: Find minimal spanning tree
- Step 2: Identify all odd degree nodes and optimally match adding links to the graph
- Step 3: Draw an Eulerian circuit on the resulting graph, removing repeated vertices to result in a Hamiltonian circuit

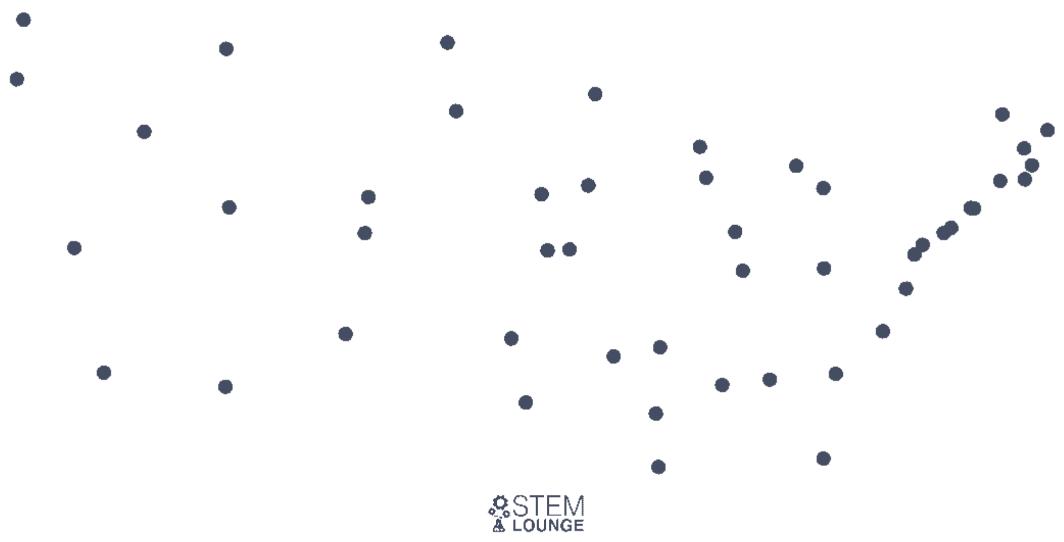
Christofides' Algorithm



80-45-42 =-7
60-50-27 = -17

- Step 1: Find minimal spanning tree
- Step 2: Identify all odd degree nodes and optimally match adding links to the graph
- Step 3: Draw an Eulerian circuit on the resulting graph, removing repeated vertices to result in a Hamiltonian circuit

Christofides' Algorithm



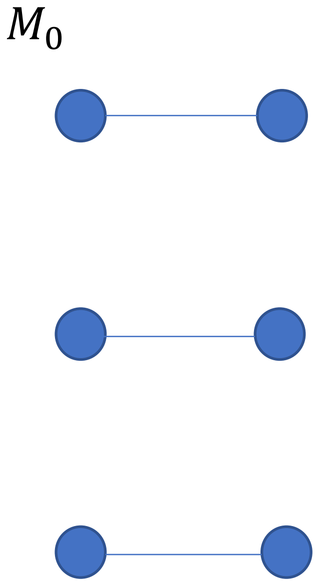
- Step 1: Find minimal spanning tree
- Step 2: Identify all odd degree nodes and optimally match adding links to the graph
- Step 3: Draw an Eulerian circuit on the resulting graph, removing repeated vertices to result in a Hamiltonian circuit

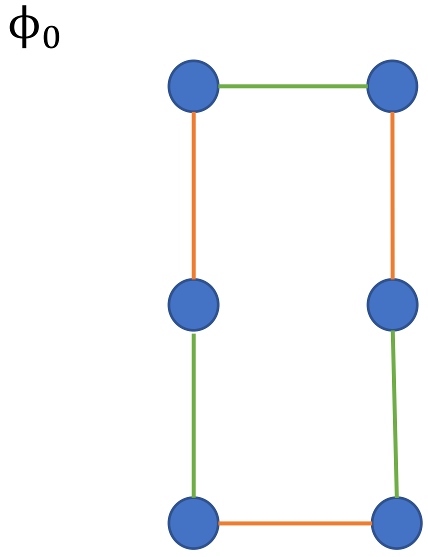
Christofides' Algorithm

- Theorem: The Length of the Christofides' Solution, $\mathrm{l}(\phi_c)$, divided by the optimal solution, $\mathrm{l}(\phi_*)$, is strictly less than 1.5

- Let T be the minimal spanning tree of graph G with length $\mathrm{l}(T)$
- $\mathrm{l}(T)<\mathrm{l}(\phi_*)$
- There are an even number of odd nodes in the spanning tree, and let $\mathrm{l}(M_0)$ be the minimum matching of the odd degree nodes

Christofides' Algorithm





Note that ϕ_0 contains two sets of connections between the nodes of M_0 and since M_0 is the minimal connection we know $l(M_0) \leq l(\phi_0)/2$

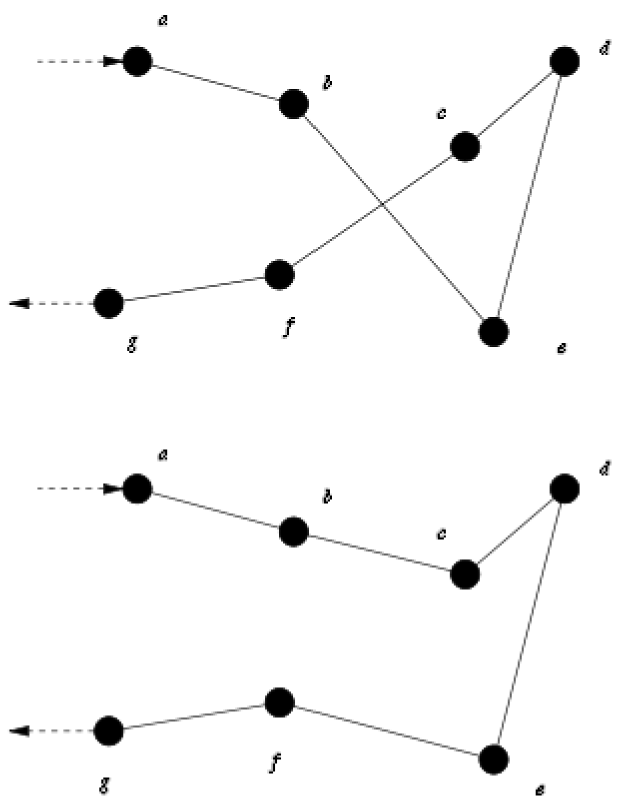
- $l(M_0) \leq l(\phi_0)/2$ (where ϕ_0 is the optimal TSP tour over the nodes in M_0)
- And since M_0 is a subset of points we know that $l(M_0) \leq l(\phi_0)/2 \leq l(\phi_*)/2$
- Thus worst case the optimal solution

$$l(\phi_C) \leq l(T) + l(M_0) \leq l(\phi_*) + l(\phi_*)/2$$

Christofides’ Algorithm

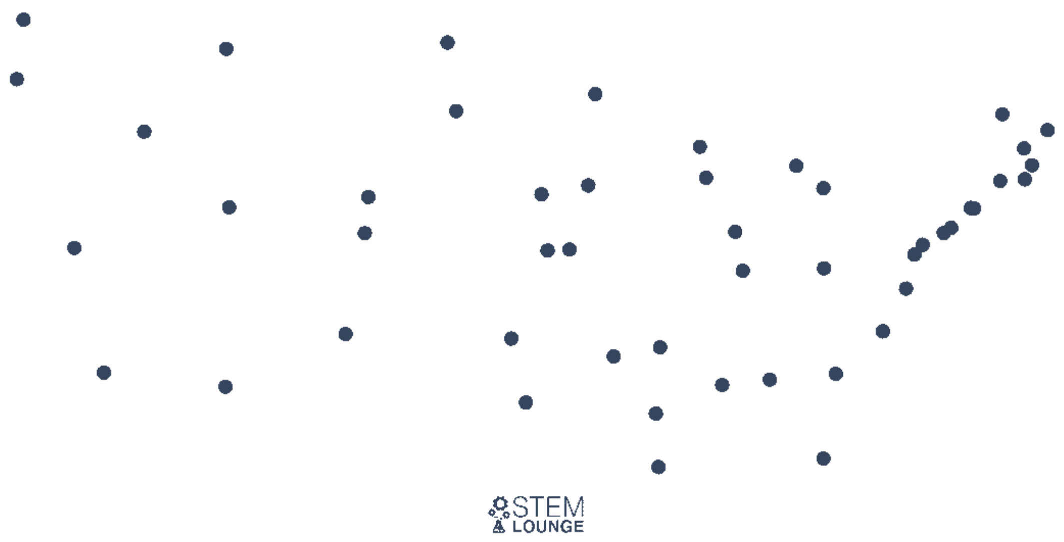
- For a long time Christofides’ had the best worst case analysis with the optimal solution being guaranteed to be within 50% of the actual optimal solution
- In 2011, a Stanford-McGill team proved an algorithm that surpassed Christofides’ with a best worst-case scenario of 49.99999999999999999999999999999999999999999999999996 % longer than the true answer

2-opt, 3-opt, …



- Is a local search algorithm for improving a solution to the TSP
- Takes the old route order:
  - [a,b,e,d,c,f,g] and cuts a section
  - [a,b, (e,d,c), f,g] reversing it
  - [a,b,c,d,e,f,g] and adding it back
- Only improvements are kept

2-opt, 3-opt, …



https://stemlounge.com/animated-algorithms-for-the-traveling-salesman-problem/

- Is a local search algorithm for improving a solution to the TSP
- Takes the old route order:
  - [a,b,e,d,c,f,g] and cuts a section
  - [a,b, (e,d,c), f,g] reversing it
  - [a,b,c,d,e,f,g] and adding it back
- Only improvements are kept

Complexity of Various Heuristics

Heuristic	Complexity
Nearest Neighbor	$O(n^2)$
Nearest Insertion	$O(n^2)$
Cheapest Insertion	$O(n^2 \log_2 n)$
Christofides’	$O(n^4)$
Random Insertion	$O(n^2)$
Farthest Insertion	$O(n^2)$
2-opt	$O(n^2)$
3-opt	$O(n^3)$

Complexity ≠ Better Performance

Heuristic	Avg % above Optimality
Nearest Neighbor	24.2
Nearest Insertion	20.0
Cheapest Insertion	16.8
Christofides’	19.5
Random Insertion	11.1
Farthest Insertion	9.9
2-opt	8.3
3-opt	3.8

- An article comparing the performance of 30 Euclidian TSPs from the literature with optimal solutions (ranging in size from 105 to 2392)

Iterated Lin-Kernighan

- A generalized form of the k-Opt procedure Lin-Kernighan is the most computationally complex heuristic in general use
- By iterating the process, it tends to result in solutions <1% from optimal (0.6% on average)