Traveling salesman problem

Pramesh Kumar

IIT Delhi

April 7, 2025

Suggested reading

- Goetschalckx Chapter 8
- Larson, R.C. and Odoni, A.R., 1981. Urban operations research Chapter 6

Single vehicle round trip routing

Two classes of problems

- Node covering
- Edge covering

Edge covering: The Chinese Postman Problem (CPP)

Given a graph, find a minimum cost tour that visits each edge at least once.

- Examples:
  - Delivery of mail to residents
  - Cleaning and sweeping of streets
  - Plowing of snow after snowstorm
  - Collection of refuse from houses
  - Distribution of newspaper
- If only a subset of edges need to be visited, then the problem is referred to as Rural Postman Problem (RPP)

History

The problem was first studied by Swiss mathematician Leonhard Euler in the 18th century. Euler wanted to find a way in which a parade possession could cross all seven bridges in Königsberg (now Kalingard on the river Pregel) exactly once. Euler proved in 1736 that there is no solution to the problem.
Remark. The name CPP is derived from an early paper published in Chinese Mathematics journal discussing the problem.

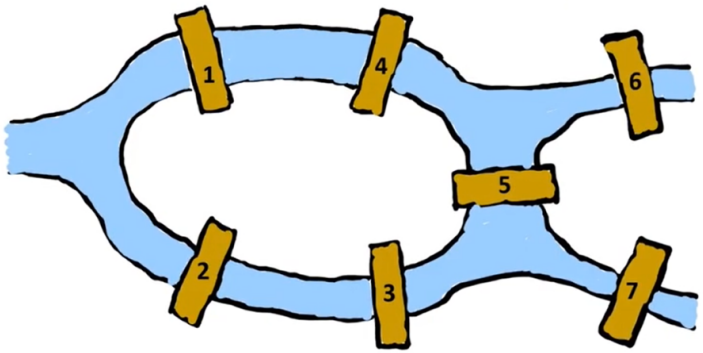


Figure: Königsberg bridge problem [a]

Figure: Leonhard Euler

[a] https://www.lancaster.ac.uk/stor-i-student-sites/harini-jayaraman/konigsberg-bridge-problem-and-the-evolution-of-mathematics/

Definition (CPP). Given an undirected graph $G(N, E)$ with known edge costs $c : E \mapsto \mathbb{R}$, find a circuit that will traverse every edge of the graph at least once and for which $\sum_{(i,j)\in E} n_{ij} c_{ij}$ is minimum, where n_{ij} is the number of times edge $(i, j) \in E$ is traversed.

Definition (Euler tour). Given an undirected graph $G(N, E)$, an Euler tour is a circuit that will traverse every edge of the graph exactly once (begin and end at the same node). An Euler path is a path which traverses every edge of the graph exactly once.

Euler's theorem

Theorem

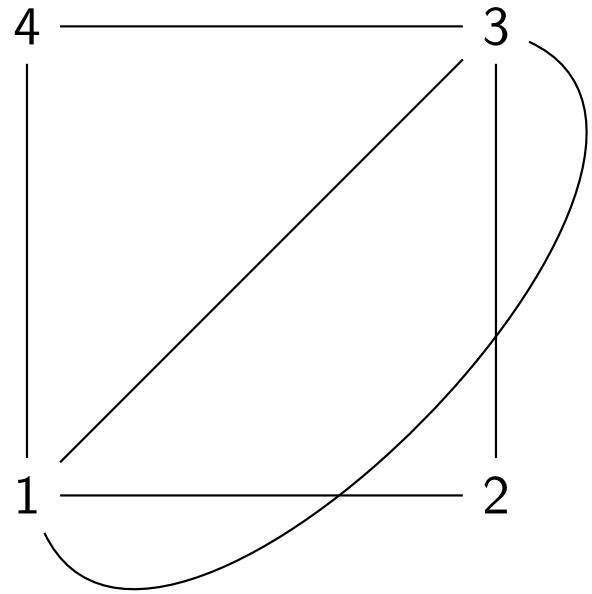
A connected graph G possesses an Euler tour (Euler path) if and only if G contains exactly zero (exactly two) nodes of odd degree.

Remark.

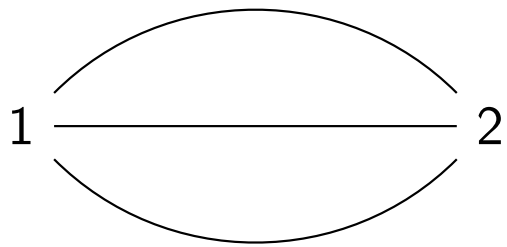
1. If a graph has Euler tour, clearly it is a solution to the CPP.
2. In case of digraphs, the indegree and outdegree of each node must be equal for Euler tour to exist.

Examples

- Euler tour exists in the following graph.



- Euler path exist in the following graph. See exactly two nodes have odd degree.



Examples

- Neither Euler tour nor Euler path exists in the following graph.

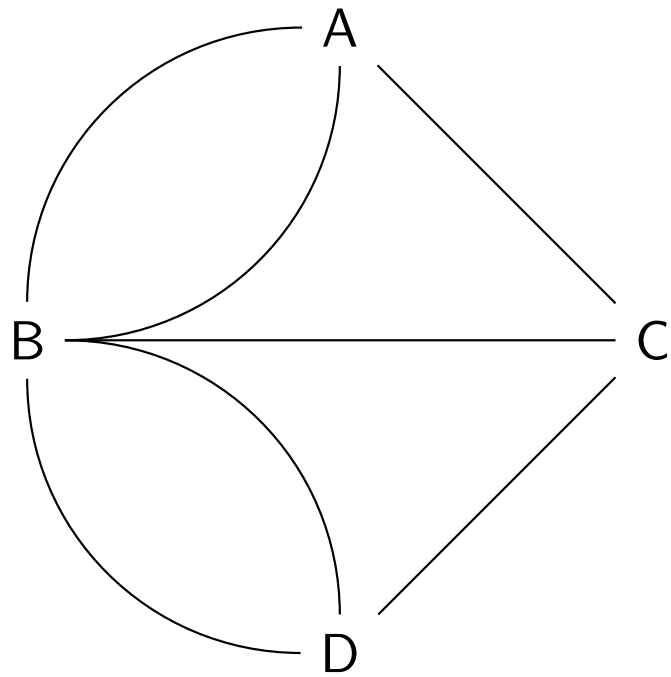
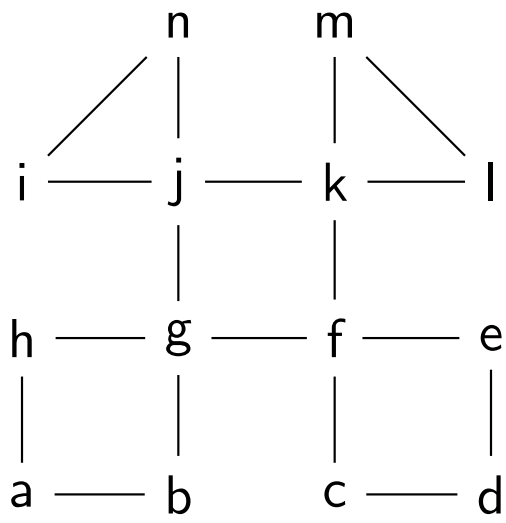


Figure: Königsberg bridge problem

Algorithm for finding the Euler tour

1. Start from any desired node s.
2. Traverse edges successively by keeping track of the route followed and delete traversed edges. Avoid traversing any edge (also called isthmus) whose deletion will disconnect the graph.
3. Continue until all edges are deleted and you come back to s. The traversed route is Euler tour.

Example



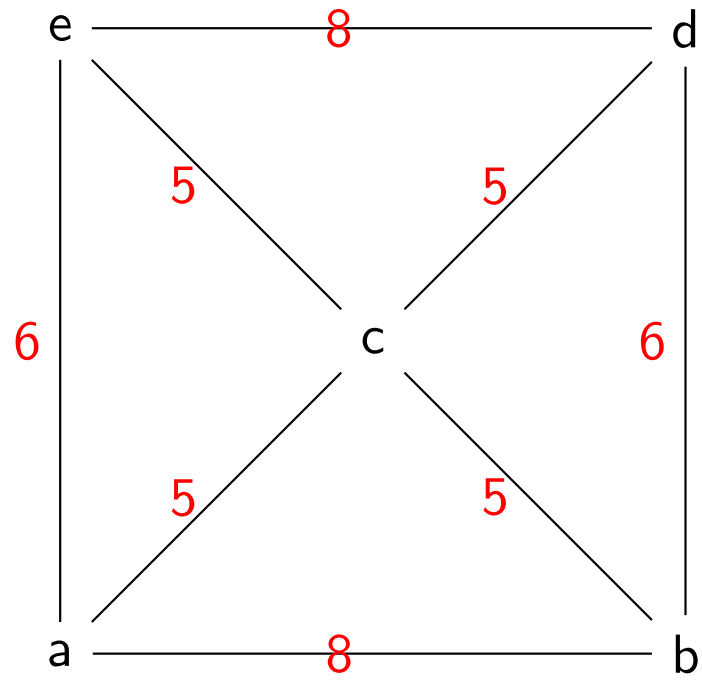
Euler tour: `a-b-g-f-c-d-e-f-k-l-m-k-j-n-i-j-g-h-a`

Chinese postman algorithm

1. Identify all the nodes with odd degree in $G(N, A)$. Let m be the number of odd degree nodes.[1]
2. Pair up all odd degree vertices, considering all possible combinations.
3. Identify the shortest path for all pairs. Select the pairings that result in the shortest total distance for all paths combined.
4. For each pair, create new edges that are on the shortest path. After this step, none of the nodes will have odd degree.
5. Find a Euler tour on the new graph. This Euler tour is an optimal solution to CPP.

[1] Note that no. of odd degree nodes in an undirected graph are always even. Why? because some of the degree of all nodes is even and if we remove the sum of degrees of nodes with even degree, we are left with the sum of degrees of odd nodes which is even.

Example



The nodes a, b, d, e have odd degree. There are three possible pairings of these nodes:

1. a - b and d - e: cost $= 16$
2. a - e and b - d: cost $= 12$
3. b - e and a - d: cost $= 20$

Example

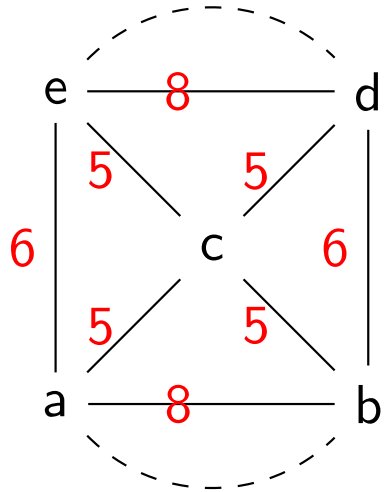


Figure: Pairing 1

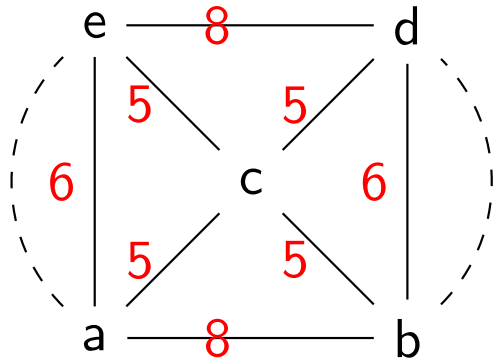


Figure: Pairing 2

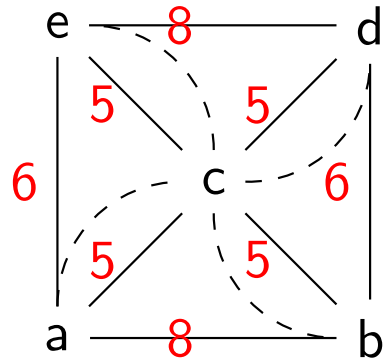


Figure: Pairing 3

The optimal matching is Pairing 2 with cost 12. Now find an Euler tour in the given graph.

`a-b-d-b-c-d-e-a-e-c-a`: Total cost = 60 (48 of which is of original graph and 12 units due to new edges). In other words, the edges a-e and b-d will be traversed twice.

Remark. One can use Edmond's algorithm for optimal pairing.

Node covering: The Traveling Salesman Problem (TSP)

Given a graph, find a minimum cost tour that visits each node at least once.

- Examples:
  - Delivery of packages to houses
  - Printed Circuit Board (PCB)
  - Sequencing DNA fragments to reconstruct the original genome
  - Routing of school bus
- It is possible to ask to visit only a subset of nodes.

Definition (Hamiltonian). A tour that visits all the nodes exactly once.

Definition (Asymmetric TSP). TSP problem defined on a graph in which the cost between any two nodes (cities) i and j may not be the same as cost between j and i.

Remark

TSP has been shown to belong to a class of computationally hard problems for which it is difficult to find the exact solution for problems of even modest size. Naïve solution: $\mathcal{O}(n!)$ and DP solution: $\mathcal{O}(n^2 2^n)$.

Euclidean TSP

Definition (Euclidean TSP). TSP problem defined on a graph whose edge costs satisfy triangle inequality

$$c_{ij} \leq c_{ik} + c_{kj}, \forall i, j, k \in N \tag{1}$$

Proposition (Larson and Odoni (1981))

- *The optimum traveling salesman tour does not intersect itself.*
- *Let m of the $|N|$ points in the Euclidean TSP define the convex hull of the points. Then the order in which these m points appear in the optimum traveling salesman tour must be the same as the order in which these same points appear on the convex hull.*

Asymmetric TSP

Given a set of cities N and cost of traveling from city i to city j denoted as c_{ij}, find a tour that visits all the cities in minimum total travel cost.

Dantzig-Fulkerson-Johnson (DFJ) formulation

$$x_{ij} = \begin{cases} 1, & \text{if they go directly from } i \text{ to } j \\ 0, & \text{otherwise} \end{cases}$$

$$\underset{x}{\text{minimize}} \quad \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij}$$

$$\text{subject to} \quad \sum_{j \in N : j \neq i} x_{ij} = 1, \forall i \in N$$

$$\sum_{i \in N : i \neq j} x_{ij} = 1, \forall j \in N$$

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1, \forall S \subset N, S \neq \phi *$$

$$x_{ij} = \{0, 1\}, \forall i \in N, \forall j \in N$$

Miller-Tucker-Zemlin (MTZ) formulation

$$x_{ij} = \begin{cases} 1, & \text{if they go directly from } i \text{ to } j \\ 0, & \text{otherwise} \end{cases}$$

$u_i =$ order in which city i is visited in the tour.

$$\underset{x,u}{\text{minimize}} \quad \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij}$$

$$\text{subject to} \quad \sum_{j \in N : j \neq i} x_{ij} = 1, \forall i \in N$$

$$\sum_{i \in N : i \neq j} x_{ij} = 1, \forall j \in N$$

$$u_1 = 1$$

$$2 \leq u_i \leq n, \forall i \neq 1$$

$$u_i - u_j + 1 \leq (|N| - 1)(1 - x_{ij}) \quad \forall i \neq 1, \forall j \neq 1$$

$$x_{ij} = \{0, 1\}, \forall i \in N, \forall j \in N$$

* One can replace these with $\sum_{i \in S, j \in S} x_{ij} \leq |S| - 1, \forall S \subset N, |S| > 1$

Subtour elimination constraints

Used for removing subtours

- Total number of such constraints $= 2^{|N|} - |N| - 2$

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1, \forall S \subset N, S \neq \phi$$

$$OR \qquad \sum_{i \in S, j \in S} x_{ij} \leq |S| - 1, \forall S \subset N, |S| > 1$$

- Total number of new variables $|N|$ and total number of constraints $= (|N| - 1) + (|N| - 1)^2$

$$2 \leq u_i \leq n, \forall i \neq 1$$

$$u_i - u_j + 1 \leq (|N| - 1)(1 - x_{ij})$$

$$\forall i \neq 1, \forall j \neq 1$$

Formulation with alternative subtour elimination constraints

- Based on item inventory.
- Salesman is assumed to start with $|N|$ items and he leaves exactly one item as he visits a city.
- Let q_{ij} is the item inventory carried out on edge (i, j)
- It creates $|N|(|N|-1)$ additional continuous variables and $(|N|-1)+2+|N|(|N|-1) = \mathcal{O}(|N|^2)$

Formulation with alternative subtour elimination

$$\underset{x}{\text{minimize}} \quad \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij}$$

subject to

$$\sum_{j \in N : j \neq i} x_{ij} = 1, \forall i \in N$$

$$\sum_{i \in N : i \neq j} x_{ij} = 1, \forall j \in N$$

$$\sum_{k \in N} q_{1k} = |N|$$

$$\sum_{k \in N} q_{k1} = 1$$

$$\sum_{i \in N} q_{ij} - \sum_{k \in N} q_{jk} = 1, \forall j > 1$$

$$q_{ij} \leq |N| x_{ij}, \forall i \in N, \forall j \in N : j \neq i$$

$$x_{ij} = \{0, 1\}, q_{ij} \geq 0, \forall i \in N, \forall j \in N$$

Symmetric TSP

$$x_{ij} = \begin{cases} 1, & \text{if they go directly from } i \text{ to } j \\ 0, & \text{otherwise} \end{cases}$$

Let $n = |N|$

$$\underset{\mathbf{x},\mathbf{u}}{\text{minimize}} \quad \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} x_{ij}$$

$$\text{subject to} \quad \sum_{i=1}^{j-1} x_{ij} + \sum_{k=j+1}^{n} x_{jk} = 2, \forall j \in N$$

$$\sum_{i \in S} \sum_{j \in S: j > i} x_{ij} \leq |S| - 1, \forall S \subset N, |S| \geq 2$$

$$x_{ij} = \{0, 1\}, \forall i \in N, \forall j \in N$$

Heuristics

Nearest neighbor heuristic

```
1: Input: Cities N, costs c
2: Output: Tour T and its cost cost(T)
3: procedure NearestNeighbor(N, c)
4:     T ← φ; cost(T) ← ∞
5:     for i ∈ N do
6:         s ← i                          ▷ Each starting point may output different tour
7:         SE ← N\{i}                                          ▷ Unexplored nodes
8:         T' ← {s}; cost(T') ← ∞
9:         while SE do
10:            n ← argmin_{j∈SE}{c_sj}                          ▷ Nearest neighbor
11:            append n to T';
12:            cost(T') ← cost(T') + c_sn
13:            U ← U\{n}
14:            s ← n
15:        end while
16:        append i to T'; cost(T') ← cost(T') + c_si
17:        if cost(T) > cost(T') then
18:            cost(T) ← cost(T')
19:            T ← T'
20:        end if
21:    end for
22: return T, cost(T)
```

Greedy heuristic

- Select the edges in increasing order of their cost as long. Ties broken arbitrarily.
- Avoid selecting edges that will result in a subtour and only two edges associated to any node can be selected.

Sweep heuristic

- The algorithm starts by finding coordinates of the "center".
  - $x_c = 0.5 \times (\max_i x_i + \min_i x_i); y_c = 0.5 \times (\max_i y_i + \min_i y_i)$
  - $x_c = \sum_{i=1}^{|N|} x_i; y_c = \sum_{i=1}^{|N|} y_i$
- A ray is rotated around (x_c, y_c) and cities are added to the tour in the sequence in which the ray traverses different cities.
- Ties are broken arbitrarily.

Savings heuristic

- Select starting city s. Let $SE = N \backslash \{s\}$ be the unexplored nodes.
- Find the pair of cities such that $p, q \leftarrow \text{argmax}_{i \in U, j \in U} \{c_{is} + c_{sj} - c_{ij}\}$. Add them to the tour. Remove p, q from SE
- The next point is selected by finding the point with largest savings to the current end points of the partial tour.
- Let p, q be the end points of the partial tour. Find h such that $h \in \text{argmax}_{h'} \{\max_{j=p,q} \{c_{js} + c_{sh} - c_{jh}\}\}$
- Continue while all points have been added to the tour.

Insertion heuristics

- In each iteration, they add an unexplored node to the partial tour and decide between which pair of nodes (edge in the partial tour) to add this node.

1. Cheapest insertion

$$\min_{k} \left\{ \min_{i,j} \{c_{ik} + c_{kj} - c_{ij}\} \right\}$$

2. Priciest insertion

$$\max_{k} \left\{ \min_{i,j} \{c_{ik} + c_{kj} - c_{ij}\} \right\}$$

Insertion heuristics

3. Nearest insertion It first finds a point to insert by finding the unexplored point closest to any point on the partial tour. let T be the partial tour.

$$\min_{k \notin T} \left\{ \min_{j \in T} c_{kj} \right\}$$

Then, it determines the best link to insert this point.

$$\min_{i,j \in T} \{c_{ik} + c_{kj} - c_{ij}\}$$

4. Farthest insertion It first finds determines for every unexplored point the smallest distance to any point on the partial tour. Then, it inserts the unexplored point with maximum smallest distance to any point on the tour. let T be the partial tour.

$$\max_{k \notin T} \left\{ \min_{j \in T} c_{kj} \right\}$$

Then, it determines the best link to insert this point.

$$\min_{i,j \in T} \{c_{ik} + c_{kj} - c_{ij}\}$$

2-approximation algorithm for Euclidean TSP[2]

1. Find an MST T^* of G (complete graph) rooted at the first node.
2. Perform a pre-order traversal of T^* from the root. Let W be the list of nodes in the order they are visited.
3. Scan each node one-by-one in T^* and delete all the occurrences other than first occurrence of any node. Let C be the resulting Hailtonian.
4. Return C as the approximate TSP tour.

[2]Finding an approximation algorithm for general TSP is NP-Hard

2-approximation algorithm for Euclidean TSP

Theorem

Let C^* and C be optimal tour and approximate tour (given by above algorithm) respectively. Then $\frac{cost(C)}{cost(C^*)} \leq 2$.

Proof.

We know that the cost of MST T^* provides a LB on $cost(C^*)$ (why because removing any edge in $cost(C^*)$ will give a spanning tree $T^{'}$ whose cost $cost(C^*) \geq cost(T^{'}) \geq cost(T^*)$), i.e.,

$$cost(C^*) \geq cost(T^*) \tag{2}$$

The walk (maybe using DFS) traverses every edge twice, therefore,

$$cost(W) = 2 \times cost(T^*) \tag{3}$$

Combining (2) and (3), we obtain

$$cost(W) \leq 2 \times cost(C^*) \tag{4}$$

Further,C is obtained by deleting the repeated nodes in W and connecting the nodes directly. This operation will not increase the cost of W due to triangular inequality. So, we have $cost(C) \leq cost(W^*)$, together with (4), we prove that

$$cost(C) \leq 2 \times cost(C^*)$$

□

Christofides' 1.5-approximation algorithm for Euclidean TSP

1. Find an MST T^* of G (complete graph) rooted at the first node.
2. Find the minimum weight perfect matching M of nodes with odd degree in T^*[3]. Let $H = T^* \cup M$. Note that there may be multiple links between two nodes in H.
3. Find an Euler tour C' in H. Convert C' into TSP tour C by skipping the repeated nodes (shortcutting) if any .

[3] Remember that the no. of odd degree nodes are even and the graph is complete.

Theorem

Let C^* and C be optimal tour and approximate tour (given by the Christofides' algorithm) respectively. Then $\frac{cost(C)}{cost(C^*)} \leq 1.5$.

Proof.

In step 3,

$$cost(C^{'}) = cost(T^*) + cost(M) \tag{5}$$

since you traverse each edge only once in Euler tour. We know that $cost(T^*) \leq cost(C^*)$ (from (2)). Let O be the set of nodes with odd degree that were matched by M. Generate two perfect matching M_1 and M_2 using alternating edges. We know that $cost(M) \leq cost(M_1)$ and $cost(M) \leq cost(M_2)$ (as they are not minimum weight matching). Therefore,

$$cost(M) \leq \frac{1}{2}\left(cost(M_1) + cost(M_2)\right) = \frac{1}{2}\left(cost(C^O)\right)$$

where, C^O is the hamiltonion cycle produced for set of nodes with odd degree. Since $O \subset N$, we have $cost(C^O) \leq cost(C^*)$. From (5),

$$cost(C) \leq cost(C^{'}) \leq cost(C^*) + \frac{1}{2}cost(C^*) = \frac{3}{2}cost(C^*) \tag{6}$$

Nicos Christofides invented his algorithm in 1976. For 44 years, there was no improvement until recently when the approximation ratio was improved to 1.5-ϵ for some $\epsilon > 10^{-36}$!

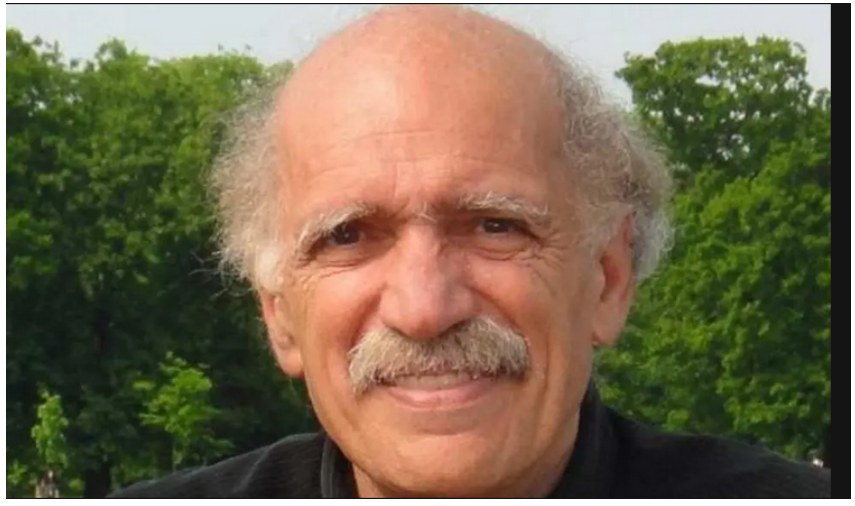


Figure: Nicos Christofides

A (Slightly) Improved Approximation Algorithm for Metric TSP

Anna R. Karlin
karlin@cs.washington.edu
University of Washington
USA

Nathan Klein
nwklein@cs.washington.edu
University of Washington
USA

Shayan Oveis Gharan
shayan@cs.washington.edu
University of Washington
USA

ABSTRACT

For some $\epsilon > 10^{-36}$ we give a randomized $3/2 - \epsilon$ approximation algorithm for metric TSP.

CCS CONCEPTS

• **Theory of computation → Routing and network design problems.**

In contrast, there have been major improvements to this algorithm for a number of special cases of TSP. For example, polynomial-time approximation schemes (PTAS) have been found for Euclidean [3, 36], planar [4, 25, 35], and low-genus metric [17] instances. In addition, the case of graph metrics has received significant attention. In 2011, the third author, Saberi, and Singh [39] found a $\frac{3}{2} - \epsilon_0$ approximation for this case. Mömke and Svensson [37] then obtained a combinatorial algorithm for graphic TSP with an approximation

Final thoughts

- There are many other heuristics for tour finding
- Branch-and-cut with heuristics is commonly used for solving it exactly.
- I recommend the following resource for further study on TSP
  - Applegate, David L., et al. "The traveling salesman problem: a computational study." The Traveling Salesman Problem. Princeton university press, 2011.
- Also the wonderful solver Concorde.

Thank you!