Travelling Salesman Problem

1805069 - Sumonta Nandy Amit
1805094 - Sheikh Hasanul Banna
1805100 - Utchchhwas Singha
1805107 - Partho Kunda
1805109 - Udayon Paul Dhrubo

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology

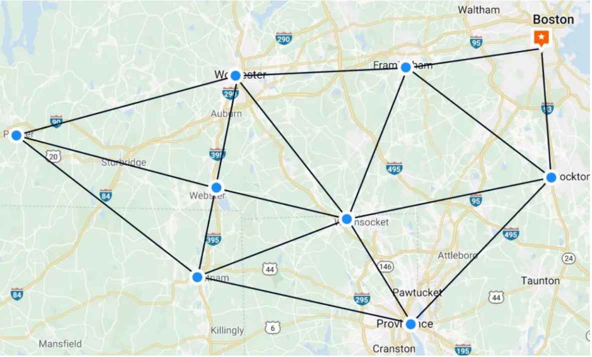
March 9, 2024

Table of Contents

Next Topic

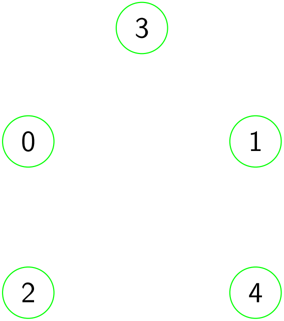
Introduction

- Beginning with a city, a salesman wants to visit all the cities and end up in the starting city
- But travelling costs money
- So how to find a cheap tour?



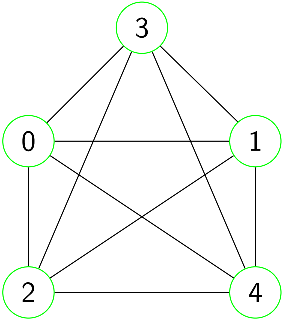
TSP graph formulation

- Represent each city as a vertex



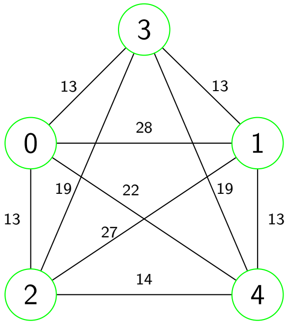
TSP graph formulation

- Represent each city as a vertex
- Each city is connected to all the other cities by edges (complete graph)



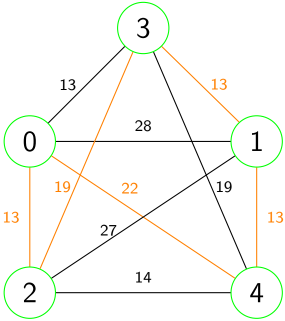
TSP graph formulation

- Represent each city as a vertex
- Each city is connected to all the other cities by edges (complete graph)
- Cost of traveling between cities is represented by weights of the edges.



TSP graph formulation

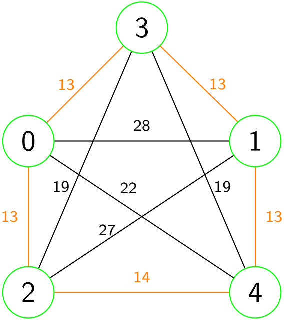
- Represent each city as a vertex
- Each city is connected to all the other cities by edges (complete graph)
- Cost of traveling between cities is represented by weights of the edges.
- A tour is a hamiltonian cycle of the graph. Cost of the tour is sum of the edges in the hamiltonian cycle



$\boxed{\text{cost of tour} = 80}$

TSP Optimization Version

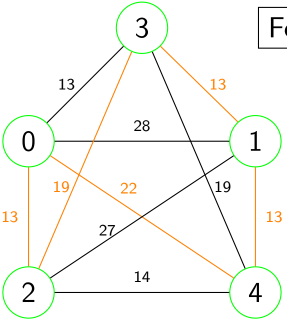
- Find the tour with the minimum cost
- NP-hard but not NP-complete. Because there's no efficient way to verify if the solution is optimal.



cost of optimal tour = 66

TSP Decision Version

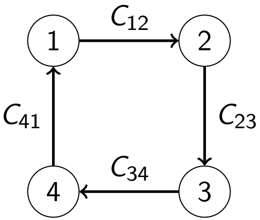
- for a certain value k, is there a tour with cost $\leq k$?
- NP-Complete.[NP-hard and a given instance can be verified in polynomial time]



For $k = 85$: cost of tour $= 80 < k$

TSP Formal Definition

- Cost: c_{ij} = cost of travelling between cities i and j
- Number of cities: n
- Cyclic permutation of cities: $k(1), k(2), \ldots k(n), K(1)$ where each city j is listed as a unique node $k(j)$.
- Then, cost of the tour is
  $$C(\pi) = C_{k(n)k(1)} + \sum_{i=1}^{n-1} C_{k(i)k(i+1)}$$
- Goal of the problem: Find permutation π such that $C(\pi)$ is minimum



Next Topic

Different types of TSP

- **Symmetric TSP(sTSP)**: For this type of TSP problem, weights of edges between two vertices are symmetric[$c_{ij} = c_{ji}$ where c_{ij} denotes cost of travelling from i to j]
- **Asymmetric TSP(aTSP)**: For this type of TSP problem, weights of edges between two vertices are not always symmetric[$c_{ij} \neq c_{ji}$ where c_{ij} denotes cost of travelling from i to j]

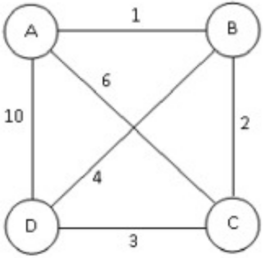


Figure: Symmetric TSP

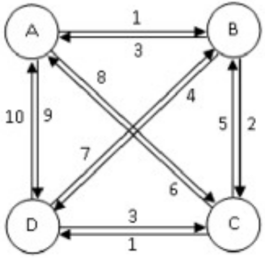


Figure: Asymmetric TSP

Different Types of TSP

- **Multiple TSP(mTSP)**: In a given set of nodes, let there are m salesmen located at a single depot node. The remaining nodes (cities) that are to be visited are intermediate nodes. Then, the mTSP consists of finding tours for all m salesmen, who all start and end at the depot, such that each intermediate node is visited exactly once and the total cost of visiting all nodes is minimized.

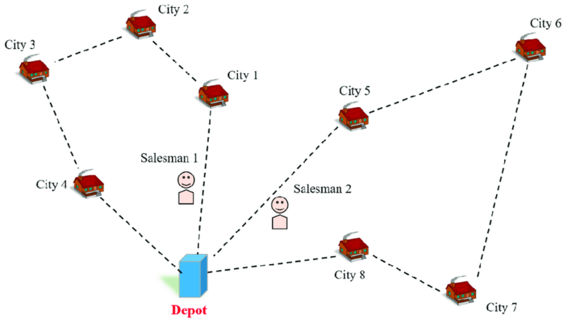


Figure: Multiple TSP

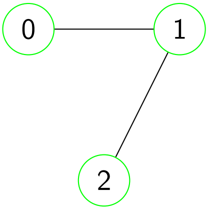
No constant Factor Approximation for TSP

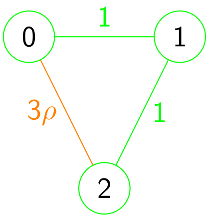
There is no constant factor approximation for TSP, unless P=NP

- We will proof this Theorem using contradiction.

- Suppose, There is a ρ approximation algorithm for TSP. Let G=(V,E) be an unweighted graph. $|V| = n$
- Construct G' from G by adding edges and assigning weights to them as follows:

$$c_{ij} = \begin{cases} 1 & \text{if (i,j)} \in \text{E} \\ n\rho & \text{if (i,j)} \notin \text{E} \end{cases}$$



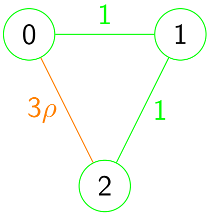


No constant Factor Approximation for TSP

There is no constant factor approximation for TSP, unless P=NP

- We will proof this Theorem using contradiction.

- Now Suppose we run an approximate TSP solver on G'. If G has a hamiltonian cycle, then G' has a TSP of total cost n, otherwise, TSP will give a total cost greater than $n\rho$.
  So using approximation algorithm, we solved the hamiltonian cycle problem in polynomial time. But this is not possible unless P=NP.
- Hence, The theorem is proved.



$\boxed{\text{cost of TSP} > 3\rho. \quad \text{No hamiltonian cycle}}$

Different Types of TSP

- **Metric TSP**: The input to TSP is called metric if for each triplets of vertices, the triangle inequality holds.
  That means, if $i, j, k \in V$, $c_{ik} \leq c_{ij} + c_{jk}$.
  - Approximate Algorithms exist for metric TSP.

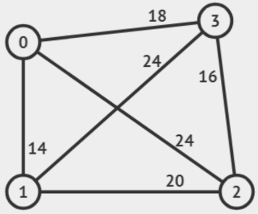
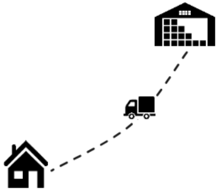


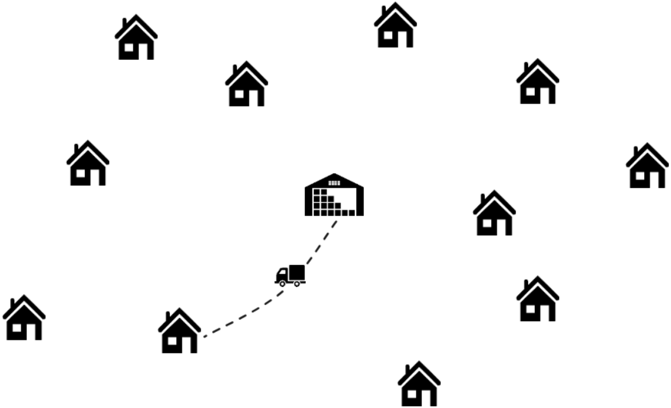
Figure: Metric TSP

Next Topic

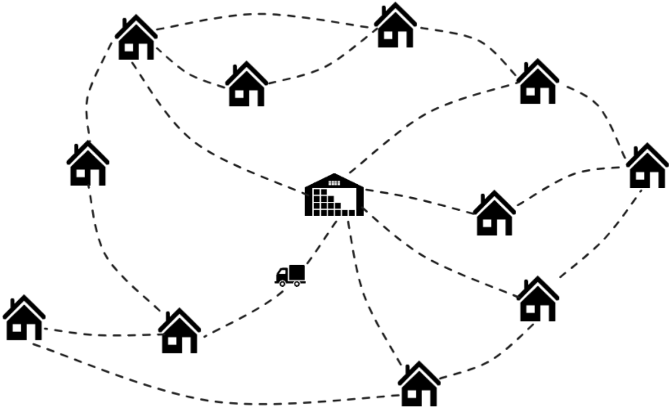
Motivation



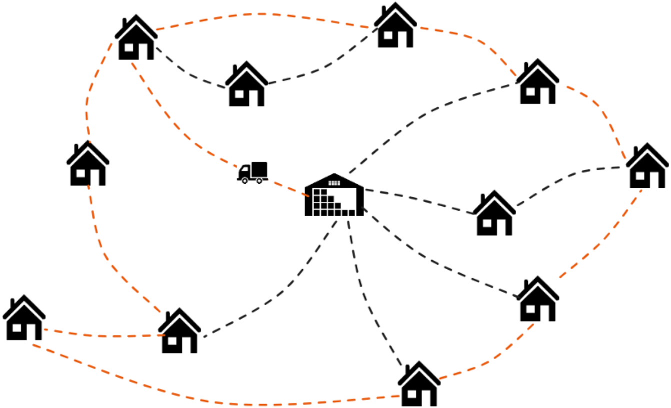
Motivation



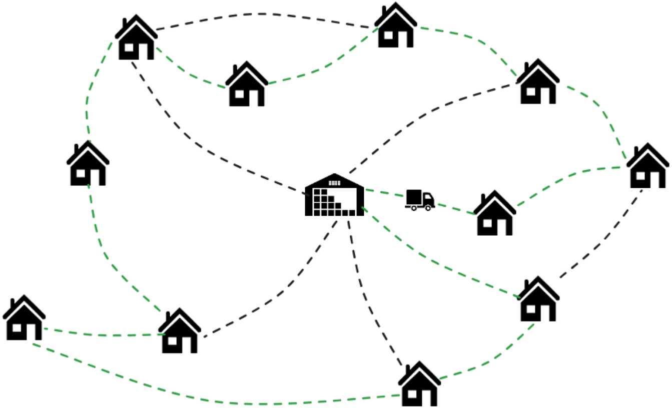
Motivation



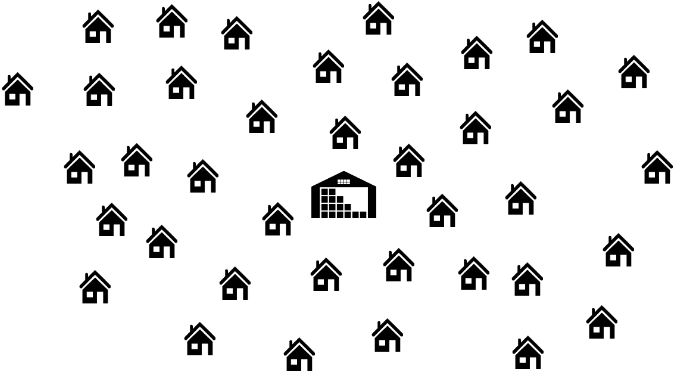
Motivation



Motivation



Motivation



Next Topic

Exact Algorithms For TSP

Brute-force search

Approach: Try all permutations
Complexity: $O(n!)$
Suitable only for a small network with few cities

Held–Karp algorithm

Approach: Dynamic programming
Complexity: $O(n^2 2^n)$
Suitable for an intermediate number of cities

Concorde TSP solver

Approach: Branch-and-cut-and-price
Can solve for a network of 85,900 cities but taking over 136 CPU-years

Next Topic

Approximation Algorithms For TSP

Nearest Neighbour

Approach: Greedy search
Complexity: $O(n^2)$
Approximation ratio: 2

Double Tree

Approach: Constructs Eulerian circuit by doubling edges of min spanning tree
Complexity: $O(n^2)$
Approximation ratio: 2

Christofides Algorithm

Approach: Combines the min spanning tree with minimum-weight matching
Complexity: $O(n^3)$
Approximation ratio: $\frac{3}{2}$

Next Topic

Heuristics Algorithms For TSP

Greedy heuristic

Approach: Selecting shortest edge
Complexity: $O(n^2 \log_2(n))$

Match Twice and Stitch

Approach: Combines two sequential matching(min-cost edge cycle)
Complexity: $O(n^2)$

Next Topic

Metaheuristics Algorithms For TSP

Lin-Kernighan Algorithm

Approach: Local Search
Complexity: $O(n^2.2)$

Tabu Search

Approach: Local Search with tabu-list
Complexity: $O(n^3)$

Simulated annealing

Approach: Explores the solution space by combining local search with random jumps, governed by a cooling schedule
Complexity: $O(n^2)$

Metaheuristics Algorithms For TSP

Genetic algorithm

Approach: population of solutions evolves over generations through operations like selection, crossover, and mutation
Complexity: $O(n^3)$

Memetic algorithm

Approach: Combines the global search capability of genetic algorithms with local search heuristics
Complexity: $O(n^3)$

Ant colony optimization

Approach: probabilistically guides future ants towards promising solutions
Complexity: $O(n^3)$

Next Topic

TSPLib95

- For our experiments, we used TSPLib95 dataset [link].
- Includes real life data. So, **Metric TSP**
- We took **Symmetric TSP** only.
- The distance from node i to node j is the same as from node j to node i.

Specification

- **NAME:** burma14
- **TYPE:** TSP
- **COMMENT:** 14-State in Burma (Zaw Win)
- **DIMENSION:** 14
- **EDGE_WEIGHT_TYPE:** GEO
- **EDGE_WEIGHT_FORMAT:** FUNCTION
- **DISPLAY_DATA_TYPE:** COORD_DISPLAY
- **NODE_COORD_SECTION**

  1: 16.47 96.10

  2: 16.47 94.44

  ...

  14: 20.09 94.55

  EOF

Next Topic

Christofides Algorithm - Pseudocode

Algorithm Christofides algorithm

Require: A complete weighted $G = (V, E, w)$ graph with Metric Property

Ensure: Hamiltonian cycle H with approximate minimization of cost

1: Create a minimum spanning tree(MST) T of the given graph G.

2: Let O be the set of vertices with odd degree in T.

3: Find a minimum weight perfect matching M in the induced subgraph given by the vertices from O.

4: Combine the edges of M with the edges of T to form a multigraph H in which each vertex has even degree.

5: Form an Eulerian circuit in H.

6: Make the circuit found in previous step into a Hamiltonian circuit by skipping visited vertices (shortcutting).

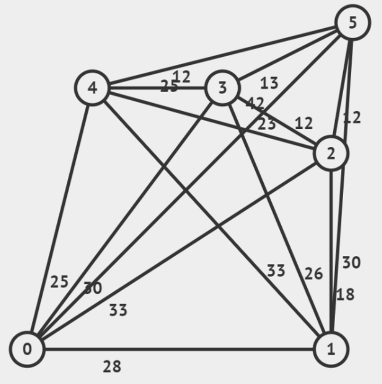


Figure: Original Graph

Initially $G(V, E)$ is complete.

Christofides Algorithm - MST

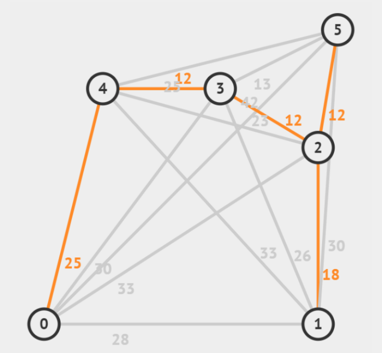


Figure: Minimum Spanning Tree

Use Kruskal's Algorithm
$O(E \log V)$ Time Complexity

Christofides Algorithm - Odd Vertices

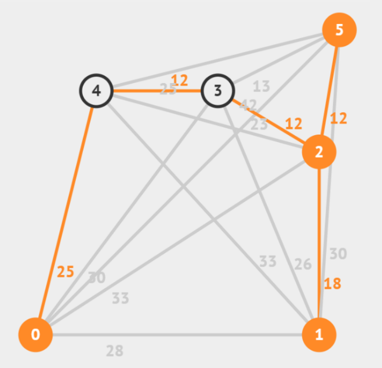


Figure: Odd Vertices in MST

Eulerian path needed.
Each vertex needs to have 2 degree exact.
We find odd degree vertices.
Which are 0, 1, 2, 5.

Christofides Algorithm - Minimum Cost Matching

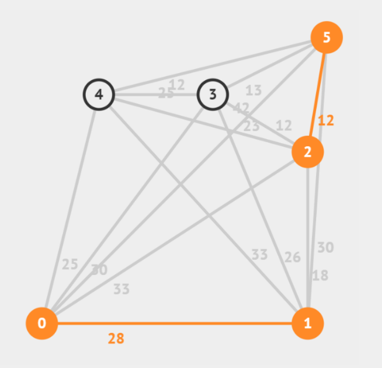


Figure: Minimum Cost Matching

Create their subgraph and find minimum cost matching. $O(n^3)$ time complexity.

Christofides Algorithm - Euler Tour

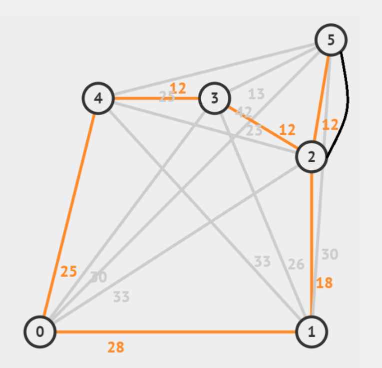


Figure: Multigraph

Combined Graph
Eulerian Tour:
0, 1, 2, 5, 2, 3, 4, 0
Remove Duplicates
(shortcutting)

Christofides Algorithm - Answer

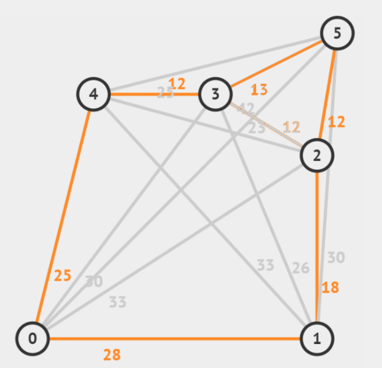


Figure: TSP Solution

TSP answer path = 0, 1, 2, 5, 3, 4, 0
Cost = 108
Overall time complexity $O(n^3)$

Christofides APX Ratio Scatterplot

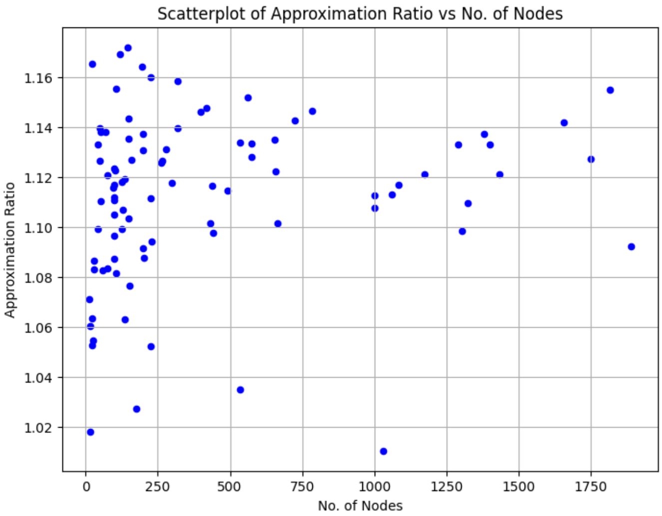


Figure: Scatterplot of APX ratio for Christofides algorithm

Christofides Algorithm Runtime Plot

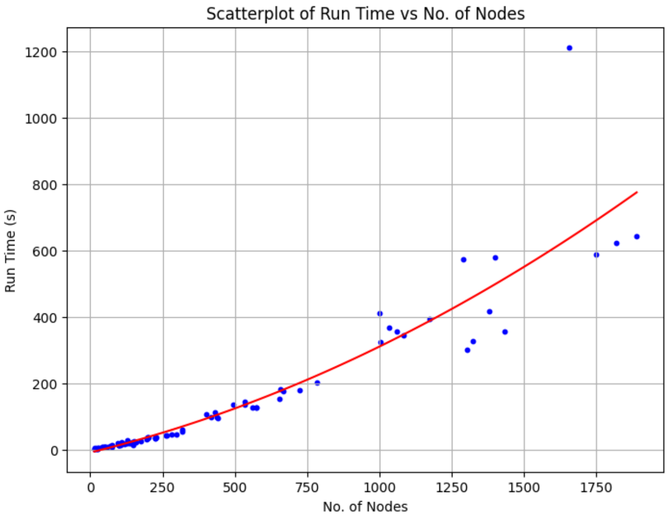


Figure: Christofides algorithm runtime plot

Next Topic

Simulated Annealing Flow Chart

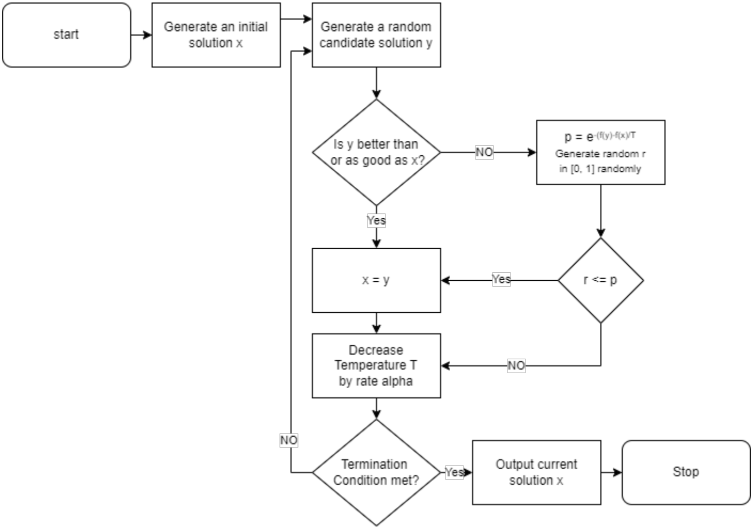


Figure: Simulated Annealing

Initial Solution

- Permutations of $\{1, \ldots, n\}$
- Representation: $[1, 2, 3, 4, 5, 6, 7, 8, 1]$

Algorithm 1 Simulated Annealing Pseudo-code

$nodes \leftarrow$ list of all nodes in the graph
$curr\_node \leftarrow$ randomly choose a node from $nodes$
initialize $current\_solution$ as a list containing $curr\_node$
$available\_nodes \leftarrow$ set of all nodes in the graph
$available\_nodes \leftarrow available\_nodes \setminus \{curr\_node\}$
while $available\_nodes \neq \emptyset$ **do**
    $next\_node \leftarrow$ node in $available\_nodes$ with minimum edge weight from $curr\_node$
    $available\_nodes \leftarrow available\_nodes \setminus \{next\_node\}$
    $current\_solution \leftarrow current\_solution + \{next\_node\}$
    $curr\_node \leftarrow next\_node$
end while

Figure: Initial Solution

Definition of Neighbor

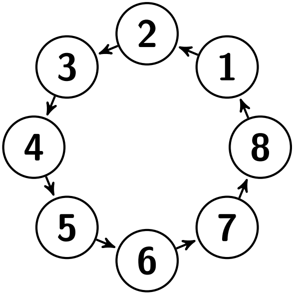


Figure: Current Tour

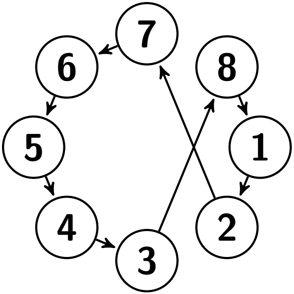


Figure: Candidate Neighbor

Fitness Function

- $f(x)$, measures the performance of the solution candidate
- in our case, the total length of the tour
- e.g.: Candidate Solution, $S = (S_1, \ldots, S_i, \ldots, S_n)$
  $C(S) = \sum d(S_i, S_{i+1})[1 \leq i \leq n-1] + d(s_n, s_1)$

Controlling Temperature

- denoted by T
- $T = T - \alpha * T$; α is the cooling parameter

Termination Condition

- Maximum number of iterations
- $T \leq 0$

Results: Tour Cost vs Iteration

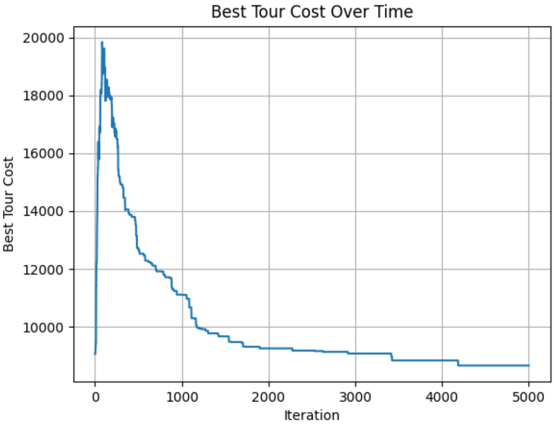


Figure: Tour Cost vs Iteration

Dataset: Berlin52

Results: Tour Cost vs Temperature

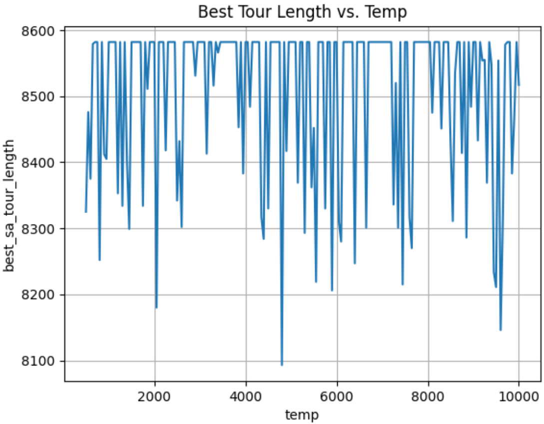


Figure: Best Tour Cost vs Temperature

Dataset: Berlin52

Simulated Annealing Results

Name	Dimension	C_Ratio	SA_Ratio	Improvement
fri26	26	1.054429	1.03095	0.023479
bays29	29	1.083168	1.006436	0.076733
swiss42	42	1.098979	1.087981	0.010998
hk48	48	1.139429	1.013088	0.126342
brazil58	58	1.082693	1.025517	0.057177
st70	70	1.137778	1.084444	0.053333
kroB150	150	1.135438	1.104669	0.030769
pr152	152	1.076477	1.046389	0.030089
rat195	195	1.164012	1.158846	0.005166
d657	657	1.122240	1.122240	0.000000

Empirical Ratios for Simulated Annealing

Name	Min_Ratio	Max_Ratio	Avg_Ratio
Christofides	1.0544290	1.1640120	1.1094644
C+SA	1.0064356	1.1588463	1.0680559

Next Topic

Revisiting Christofides

- Christofides algorithms is deterministic.
- May use Kruskal's algorithm or Prim's algorithm as a subroutine to obtain MST.

KRUSKAL-MST

Algorithm 1 KRUSKAL-MST

Require: A connected, undirected, weighted graph $G = (V, E)$

Ensure: A minimum spanning tree T of G

```
T ← ∅
for v ∈ G.V do
    MAKE-SET(v)
end for
E ← LIST(G.E)
Sort E in non-decreasing order by weight
i ← 0
while |T| < |G.V| − 1 do
    e ← E[i]
    if FIND-SET(e.u) ≠ FIND-SET(e.v) then
        T ← T ∪ {e}
        UNION(e.u, e.v)
    end if
    i ← i + 1
end while
return T
```

Figure: Pseudocode For Kruskal's MST

Time Complexity Analysis of KRUSKAL-MST

- $|V|$ MAKE-SET operations take $O(V \log V)$
- Sorting E takes $O(E \log E) = O(E \log V)$
- $|E|$ FIND-SET and UNION operations take $O(E \log E) = O(E \log V)$
- For complete graphs, $|E| = O(V^2)$
- Thus, total time complexity is $O(E \log V) = O(V^2 \log V)$

Trying Other STs

- MST may not give optimal tour.
- May try with other STs.
- **Idea:** Randomly sample an ST.
- But, does a completely random ST perform well?

Problem with Random STs

Name	Dimension	C_Ratio	R_Ratio
fri26	26	1.054429029	3.186766275
bays29	29	1.083168317	3.20049505
swiss42	42	1.09897879	3.825608798
hk48	48	1.139429369	4.826978449
brazil58	58	1.082693444	4.869895649

k-RANDOMIZED-ST

Algorithm 2 RANDOMIZED-ST

Require: A connected, undirected, weighted graph $G = (V, E)$

Ensure: A spanning tree T of G and a randomization parameter k

$T \leftarrow \emptyset$
for $v \in G.V$ **do**
    MAKE-SET(v)
end for
Sort E in non-decreasing order by weight
while $|T| < |G.V| - 1$ **do**
    $j \leftarrow$ MIN$(|E|, k)$
    $W \leftarrow [e.w$ **for** $e \in E[0, j+1]]$
    $i \leftarrow$ SAMPLE-INDEX(W)
    $e \leftarrow E[i]$
    if FIND-SET$(e.u) \neq$ FIND-SET$(e.v)$ **then**
        $T \leftarrow T \cup \{e\}$
        UNION$(e.u, e.v)$
    end if
    ERASE(E, i)
end while
return T

SAMPLE-INDEX

Algorithm 3 SAMPLE-INDEX

Require: A list of weights W

Ensure: Sample a random index $i \in [0, |W|]$

1: Z-Normalize the values in W
2: $P \leftarrow [e^{-w}$ **for** $w \in W]$
3: $s \leftarrow \text{SUM(P)}$
4: Sample $j \in [0, |W| - 1]$ using the probability distribution P
5: **return** j

Time Complexity Analysis of RANDOMIZED-ST

- Additional $|E|$ calls to SAMPLE-INDEX take $O(kE) = O(kV^2)$
- $|E|$ ERASE operations take $O(kE) = O(kV^2)$
- Thus, total time complexity is $O(V^2 \log V + kV^2)$

Randomized Christofides

- Use k-RANDOMIZED-ST instead of KRUSKAL-MST

Randomization Improves Performance

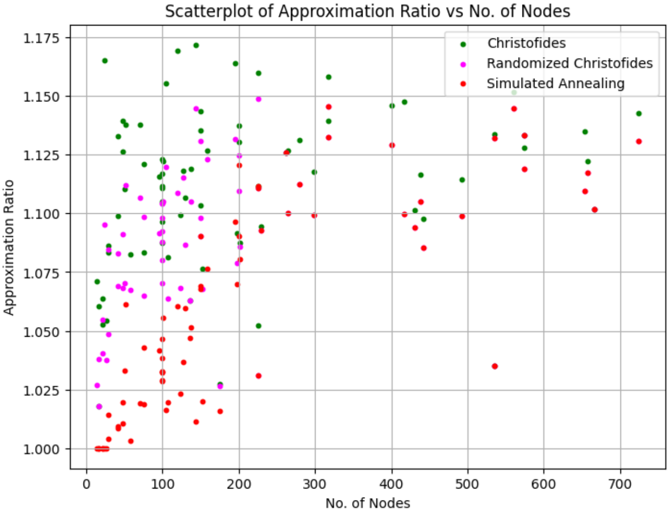


Figure: Scatterplot of Ratio vs No. of Node for randomization

Randomization Results for Christofides

Name	Dimension	C_Ratio	RC_Ratio	Best_k	Improvement
fri26	26	1.054429	1.037353	10	0.017076
bays29	29	1.083168	1.048515	11	0.034653
swiss42	42	1.098979	1.069128	16	0.029851
hk48	48	1.139429	1.091179	13	0.048251
brazil58	58	1.082693	1.067533	2	0.015160
st70	70	1.137778	1.106667	9	0.031111
kroB150	150	1.135438	1.130884	6	0.004554
pr152	152	1.076477	1.067941	14	0.008537
rat195	195	1.164012	1.131726	13	0.032286
d657	657	1.12224	1.117170	9	0.005070

Randomization Results for Simulated Annealing

Name	Dimension	C_Ratio	SA_Ratio	Best_k	Improvement
fri26	26	1.054429	1.000000	7	0.054429
bays29	29	1.083168	1.003960	15	0.079208
swiss42	42	1.098979	1.009427	19	0.089552
hk48	48	1.139429	1.010383	4	0.129046
brazil58	58	1.082693	1.003150	2	0.079543
st70	70	1.137778	1.019259	14	0.118519
kroB150	150	1.135438	1.067700	19	0.067738
pr152	152	1.076477	1.019761	14	0.056717
rat195	195	1.164012	1.096427	1	0.067585
d657	657	1.122240	1.117170	9	0.005070

Empirical Ratios for Randomization

Name	Min_Ratio	Max_Ratio	Avg_Ratio
Christofides	1.0179326	1.1718571	1.1115747
RC	1.0179326	1.1487638	1.0929558
RC+SA	1.0000000	1.1456040	1.0602919