THE TRAVELING SALESPERSON PROBLEM

Addis Ababa University School of Information Technology and Engineering

Department of Artificial Intelligence

Abstract

In this paper the most famous Traveling Sales man problem will be defined, and Brute force and Ant pheromone Optimization algorithms will be analyzed to solve it, finally algorithm will be used to solve the TSP problem.

Instructor: Dr Beakal Gizachew

Author: Thomas Kitaba

1. Introduction

These days, cities are growing rapidly with size and the number of commuters, new transportation routes are being added to major cities continuously, ecommerce is becoming the best choice of sales requiring delivery of service or products so solving the traveling sales problem will solve many problems related to the traveling sales man problem.

1.1. Define the TSP:

TSP can be defined as "the challenge of finding the shortest possible route that visits a set of destinations exactly once and returns to the starting point ".

1.2 Practical Relevance of TSP:

TSP can be applied to different areas of our day-to-day life some of them are

- Logistics and transportation planning.
- Network design and optimization.
- Routing for meter reading or delivery services.
- Tourist route design

2. Problem Definition

The main problem of the TSP is the increase in computational time and resource usage when the number of cities increase.

2.1 Inputs:

In general, we will have 6 cities named, [Addis Ababa, Adama, Bahir Dar, Hawassa, Gondar, Mekele]

Additionally, distance and heuristic data will be used as inputs

2.1.1 inputs for the brute force algorithm

The distance and Heuristics between each city are given as matrix for the brute force algorithm.

```
Distance Matrix
[[ 0.  125.  150.  175.] # Addis Ababa to: itself, Adama, Bahir Dar, Hawassa
[125.  0.  250.  150.] # Adama to: Addis Ababa, itself, Bahir Dar, Hawassa
[150.  250.  0.  200.] # Bahir Dar to: Addis Ababa, Adama, itself, Hawassa
[175.  150.  200.  0.]] # Hawassa to: Addis Ababa, Adama, Bahir Dar, itself
```

City Mapping:

The order of cities corresponds to their alphabetical order:

- Index 0: Addis Ababa
- Index 1: Adama
- Index 2: Bahir Dar
- Index 3: Hawassa

2.1.2 inputs for the Ant Pheromone Optimization algorithm

- **Adjacency matric or graph**

Data structure will be used to represent the adjacency matric between cities, within the adjacency matric relationship between city and neighboring city is represented as object made of key value pair Where the key represents the current city and value represents information about the neighboring cities or nodes.

The values themselves are represented as object containing, 2 key value pairs representing distance and heuristics information.

Adjacency matric represented using a variable named graph

```
graph = {
    "Addis Ababa": {
        "Adama": {"distance": 125},
        "Bahir Dar": {"distance": 150},
        "Hawassa": {"distance": 175},
    },
    "Adama": {
        "Addis Ababa": {"distance": 125},
        "Bahir Dar": {"distance": 250},
        "Hawassa": {"distance": 150},
    },
    "Bahir Dar": {
        "Addis Ababa": {"distance": 150},
        "Adama": {"distance": 250},
        "Hawassa": {"distance": 200},
    },
    "Hawassa": {
        "Addis Ababa": {"distance": 175},
        "Adama": {"distance": 150},
        "Bahir Dar": {"distance": 200},
    },
}
```

Graph

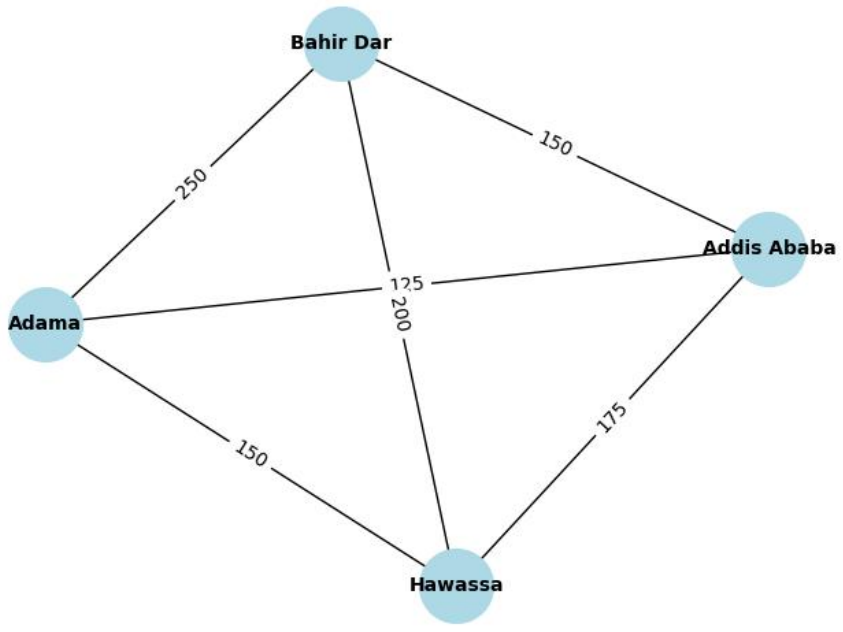

Figure-1: adjacency list for the traveling sales man

2.2 Outputs:

The output will contain

- The optimal route: A sequence of cities that minimizes the total distance while satisfying the TSP conditions.
- Total distance of the optimal route

This will be the output format

Best route: ***['Bahir Dar', 'Addis Ababa', 'Adama', 'Hawassa']***

Best distance: ***987***

3. Solution

In this section two algorithm will be selected as candidates and be analyzed for their time and space complexity, finally we will recommend the best algorithm design that best solves the TSP.

3.1 candidate solutions

The candidate solutions chosen to solve the traveling sales man problem are presented below using Table-1.

Table-1: candidate solutions

	Brute Force	Ant Colony Optimization
Method	Exact Method	Heuristic
definition	A problem-solving method that systematically explores all possible solutions to find the optimal one.	A heuristic optimization algorithm inspired by the behavior of ants searching for the shortest paths to food.
Nature of Exploration	Exhaustive search	Probabilistic
Algorithm Type	Enumerative	Bio-inspired
Implementation Difficulty	Low	Moderate to high
Optimality	Always finds the optimal solution	May not find Optimal, but finds suboptimal solution
Deterministic or None Deterministic	Deterministic	None Deterministic

3.2 Solution Design Analysis:

3.2.1 Analysis for Brute Force Algorithm

General algorithm procedure: for N cities *$(i_1, i_2, \ldots, i_n)$*

```
for $i_{1=1}$ to n
   for $i_2 = 1$ to n
      for $i_3 = 1$ to n
      ....
         ....
            For $i_n = 1$ to n

                  If ($i_1, i_2, \ldots, i_n$ is a valid tour)
               If ($(i_1, i_2, \ldots, i_n)$ is shorter than the current optimal)
               Return TSP = $(i_1, i_2, \ldots, i_n)$
```

Information's necessary to calculate Runing time of the brute force algorithm.

- ***number of times for a single for loop*** = n
- ***total number of for loops = total number of cities*** = n
- ***Time to complete 1 operation*** = **1 nanosecond**
  (1 nanosecond = 0.000000001sec= 10^{-9}seconds)
  Based on the provided information
  time complexity = O (n^n)

Table-2: Brute force algorithms Ruining time

Number of Cities (n^n)	n^n	**Total Operations = 10^{-9} \times n^n seconds**
5	$5^5 = 3{,}125$	$3{,}125\times10^{-9}$ = **3.13×10^{-6} seconds**
10	$10^{10} = 10{,}000{,}000{,}000$	$10{,}000{,}000{,}000\times10^{-9}$= **10 seconds**
12	$12^{12} = 8.9161\times10^{12}$	8.9161×10^{12} x 10^{-9}= **8,916 seconds (~2.476 hours)**
20	$20^{20} = 1.0486\times10^{26}$	$1.0486\times10^{26}\times10^{-9}$=$1.0486\times10^{26}\times10^{-9}$=**$1.0486\times10^{17}$ seconds (~3.32 billion years)**

NB: based on the information provided in Table-2, brute force algorithm is not applicable for greater cities or nodes greater than 10.

Because its running time is an acceptable. As a rule of thumb every algorithm should terminate within 1/30 of a second.

Table-3: Brute force algorithms analysis summary

	Brute Force
Time Complexity	n^n
Space Complexity	n
Optimality	Exact optimal: always returns the best optimal solution
Usage for large number of N	Unapplicable for cities number larger than 11 (as you can see 12 cites require 2.4 operation hours)

3.2.2 Analysis of Ant Pheromone Optimization

- Formulas 1: Probability of moving from i-city to j-city

$$P_{ij} = \frac{\tau_{ij}^{\sigma}\eta_{ij}^{\beta}}{\sum \tau_{im}^{\sigma}\eta_{im}^{\beta}}$$

$$m \in allowed$$

- P_{ij} = the probability of moving from i-city to j-city

Meaning: the probability of moving from i-city and j-city is directly proportional to the amount of pheromone on the path and the value of the proximity (distance) between the cities.

$\tau_{ij}^{\sigma}\eta_{ij}^{\beta}$ = the desire of moving from i-city to j-city

τ_{ij}^{σ} = amount of pheromone between i-city to j-city

$\eta_{ij}^{\beta} = \frac{1}{d_{ij}}$ reciprocal of the distance between i-city and j-city (proximity between i-city and j-city) or 1 can be integer appropriate for visibility

and j-city)

Amount of pheromone between **i-city** and **j-city**

- $\sum \tau_{im}^{\sigma}\eta_{im}^{\beta}$ = the sum of desires of moving from i-city to all allowed cities

  m = unvisited allowed cities
- σ : - tells us how much ants trust the pheromone trial
- β : - shows how much they trust the direct attractiveness of a path
- **Formulas 2**: Distribution of pheromones across the path

  Pheromone update

$$\Delta \tau_{ij}, k^{(t)} = \frac{Q}{L_k^{(t)}}, if\,(i, j) \in T_k^{(t)}$$

$$\Delta \tau_{ij}, k^{(t)} = 0, if\,(i, j) \notin T_k^{(t)}$$

Global Pheromone update.

$$\tau_{ij}^{(t+1)} = p.\tau_{ij}^{(t)} + \sum_{k=1}^{m} \Delta \tau_{ij}, k^{(t)}$$

Meaning: the amount of pheromone left from (**city-i, city-j**) on the next iteration **(t + 1)**

Depends on the amount of pheromone at current iteration **(t)** multiplied with coefficient of pheromone evaporation **(p)** summed with the total amount of pheromones left by all ants at iteration **(t)**.

- $\Delta \tau_{ij}, k^{(t)}$ = represents the change in pheromone level on edge (i, j) by ant k at time t.

  Meaning: The Pheromone addition that the ant **k**, adds at iteration **t** when moving from **i-city** to **j-city** is equal to a constant **Q** divided by the length **L** made by **K** on that iteration.

- **Q** A **constant** that represents the total pheromone quantity an ant has available to deposit.

- **t** iteration number

- $L_k^{(t)}$ is the length of the tour constructed by ant **k** at time **t**.

- $T_k^{(t)}$ is the tour constructed by ant **k** at time **t**.

- $\tau_{ij}^{(t+1)}$ is the pheromone level on edge **(i, j)** at time **t + 1**,

Meaning: the amount of pheromone on the next iteration at that specific edge **(i, j)**

- **p** is the pheromone evaporation rate.

(1-p) = coefficient of evaporation.

- m is the total number of ants.

Time Complexity Analysis for Ant Pheromone Optimization

1. Probability Equation

- **Key operations**: Iterating over all allowed edges **O(n)**.
  - To calculate the sum of all desires for a single iteration, we compute **n** desires for **m** (allowed ants).
  - Total n iterations require **$O(n^2)$**.
- **For every ant**: **O(n)** summations are calculated. Therefore:
- Time complexity=O(m×n2)

2. Global Pheromone Update

- **Pheromone updates**: For **n** cities, **n** calculations are made: **$O(n \times n) = O(n^2)$**
- **For m ants**: Total time complexity becomes: **$O(m \times n^2)$**
- **Dominant factor**: **$O(n^2)$**.

3. Pheromone Deposition

- For **n** nodes, each ant contributes **n** pheromones: **O(m×n)**

Space Complexity Analysis for Ant Pheromone Optimization

1. Probability Computation

- Temporary storage for probabilities: O(m×n)
- Storing pheromone levels: O(n2)
- Total space complexity: $O(m \times n^2)$

2. Global Pheromone Update

- To store pheromone levels (all n^2 edges): $O(n^2)$
- To temporarily store pheromone deposited by ant k at iteration t: O(m×n)
- Total space complexity: $O(m \times n^2)$

3. Pheromone Deposition

- To store pheromone for all edges: $O(n^2)$
- To store pheromone left by ant k for n nodes: O(n)
- For a total of m ants: O(m×n)
- Total space complexity: $O(n^2)$

Complexity generalization

Table-4 Time and space complexity

	Time complexity	Space complexity
Probability formula	**O (m x n^2)**	**O (n^2)**
Global Pheromone update	**O (m x n^2)**	**O (n^2)**
Pheromone update	**O (m x n)**	**O (n^2)**
Dominating complexity	**O (m x n^2)**	**O (n^2)**

Table-5: Ant Pheromone Optimization analysis summary

	Ant Pheromone Optimization
Time Complexity	O (m x $n^{2)}$
Space Complexity	O (n^2)
Optimality	suboptimal
Usage for large number of N	applicable

- **Design Choices (Algorithm Choice):**

Table-6: Comparison: Brute Force vs. Ant Colony Optimization

Property	**Brute Force**	**Ant Colony Optimization**
Time Complexity	Exponential O(n^n) infeasible for large n.	**O(m×n^2)** feasible for large n
Space Complexity	Low	Moderate
Optimality	Guaranteed exact solution.	Near-optimal, approximates close solutions.
Scalability	Poor for large n	High scalability for large, complex problems.

After analyzing the candidate algorithms, I have chosen APO as a design choice for the TSP problem.

Why Ant Pheromone Optimization? For large-scale real-world problems that contain larger number of agents and for environment whose horizon is invisible Ant Pheromone optimization (APO) is better because it provides faster and sub optimal solutions.

4. Contributions

- **Design Decisions:**

My decision to use APO as a design choice is driven by these factors

1. Considering that e commerce in our country will blossom and lead to the design of AI systems that create jobs related to delivery of goods.
2. APO is selected because it will easily adjust to an increase in number of cities or nodes, in other words scalability has been considered for APO to be our best choice.
3. As our instructor mentioned “Hardware and memory are fast and cheap. Computing getting cheaper and cheaper. Intelligent manpower is expensive” in the lecture slide algorithm that best uses this resource is selected.

- **Optimizations:**
  - Most codes I encountered didn’t have heuristic, so I added heuristics so as to make local search more dependent on values rather than deepening only on the level of pheromone.

- **Unique Contributions:**
  - The contribution I had on this project, is selecting appropriate formulas and making them available here by transferring them from images to text. As we all

    Creating the graph in such a way that it is understandable by readers.

5. Comparison with Other Solutions

Strength and Weakness:

- weaknesses of My Project.
  - I have selected only 2 algorithms as candidates, but it would have been eye opening for me If I selected 4 or more candidates, because while analyzing them I could have enlightened myself about new algorithms designed to solve the TSP problem.
- Strength of My Project.

  I have selected the 2 most extreme algorithms, enabling me understand what not to choose for on the worst case, and what to look for in algorithms so that they not only result in optimal answers but also applicable to real world problem solving.

6. Conclusions

- **Summary of Learning:**

  algorithmic concepts that helped me best from lecture materials and instructors.

  Table 3 that shows the total operation for n cities is inspired by the table presented on lecture slide 1 titled “lecture 1 INTRODUCTION Marriage Problem”, the decision of choosing APO is also made by considering the point made on this slide that says “Algorithms must terminate in 1/30 of a sec” which made the brute force technique in applicable.

- **Challenges:**

  The part of the project that was time consuming was understanding and compiling the APO formulas.

- **Feedback:**

  I wouldn’t change anything from Dr Beakals Gizachew’s Advanced Problem-solving course, I am looking forward to learn more and see the world of AI with our respected Instructor.

Source

- Github repository for APO:
  https://github.com/thomaskitaba/traveling_sales_man_using_APO.git

Reference

- Biron, D. (2006). *The traveling salesman problem: Deceptively easy to state; notoriously hard to solve* (Senior thesis, Liberty University). Retrieved January 19, 2025, from https://digitalcommons.liberty.edu/honors/188