An Approximation of The Traveling Salesman Problem in Determining The Shortest Route Across The Continent of America

Abdullah Farhan - 13523042[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]farhanjuneadi213@gmail.com, 13523042@std.stei.itb.ac.id

***Abstract*— The Traveling Salesman Problem (TSP) is a classical optimization challenge that seeks to determine the shortest route visiting each city exactly once before returning to the starting point. Due to its NP-hard complexity, approximation methods are necessary for solving the TSP when the number of cities is substantial. This paper examines the implementation of the Nearest Neighbor algorithm alongside the 2-Opt optimization technique to find an approximate solution for the shortest route covering all countries in the Americas. Testing results indicate that the combination of Nearest Neighbor and 2-Opt yields more optimal solutions than employing the Nearest Neighbor algorithm alone.**

***Keywords*— Travelling Salesman Problem, Approximation, America, Nearest Neighbor, 2-opt, Route Optimization**

I. INTRODUCTION

The American continent is home to numerous beautiful countries that attract a significant number of tourists. Renowned for its rich history, diverse cultures, impressive architecture, and stunning natural landscapes, America offers a wide array of destinations. From modern cities such as New York, Los Angeles, and Toronto to natural wonders like the Rocky Mountains and the Grand Canyon, the continent presents a plethora of appealing sites for travelers from around the globe.

Figure 1. Map of the Americas
Source: https://id.wikipedia.org/wiki/Daftar_bursa_efek_di_Benua_Amerika

Geographically, America is a relatively large continent characterized by varying population densities, comprising approximately 35 countries distributed across its expanse. Each country possesses unique characteristics and distinct attractions, making inter-country travel within America an exceptionally engaging experience. The available transportation systems, including domestic and international flight networks as well as intercity bus services, facilitate ease of movement for tourists exploring this vast continent.

Traveling across America can be quite costly, particularly for those wishing to visit all the countries in one journey, which can lead to significant expenses. Therefore, it is essential to identify routes that minimize costs. It is assumed that shorter travel distances will generally result in lower expenses, as airlines often set prices based on distance.

This paper will explore the shortest travel route for a tour across the American continent, visiting all 35 countries, utilizing one of the approximation techniques for the traveling salesman problem (TSP). Given the complexity of finding the shortest tour, employing an exact match TSP algorithm is impractical. The best-known exact match algorithm for solving TSP, Held-Karp, operates with exponential time complexity, making it infeasible to solve TSP instances with more than 34 cities within a reasonable timeframe.

II. THEORITICAL FRAMEWORK

A. Traveling Salesman Problem (TSP)

The Traveling Salesman Problem (TSP) is defined as follows: "Given a list of cities and the distances between each pair of cities, find the shortest route that visits each city exactly once and returns to the origin city." This problem is a classic example in the fields of combinatorial optimization and graph theory, and it holds significant relevance in various applications such as travel planning and circuit design.

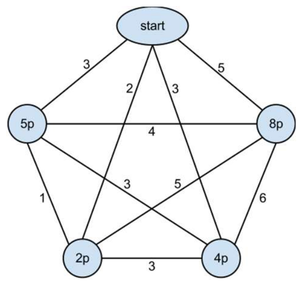


Figure 2. Complete Graph
Source: https://i.sstatic.net/VP1OS.png

For instance, the TSP can be illustrated using the graph above, where the shortest route is determined to be 16, traversing the nodes in the order of start → 8p → 5p → 2p → 4p → start. TSP is classified as an NP-hard problem, which currently lacks a polynomial-time algorithm for its resolution. Consequently, exact solutions for TSP become impractical in scenarios involving a large number of cities due to the exceedingly high computational time required. For example, solving TSP exactly for 50 cities using the Held-Karp algorithm, which has a time complexity of $O(n^2 2^n)$, would take approximately 3255334.81 days or 8918.7255 years, This calculation is based on the following formula:

$$2^{50} * 50^2 * \frac{0.1}{(3{,}156e+7) * 1000} = 8918.7255$$

Assuming:

- Each computation takes 0.1 ms
- One year is equal to 3.156×10^7 seconds

Algorithms designed to tackle TSP can be categorized into two types: exact algorithms and approximation algorithms. Exact algorithms yield results that are guaranteed to be minimal and accurate; examples include branch and bound, dynamic programming (Held-Karp), and brute force methods. In contrast, approximation algorithms provide results that are not precise but offer reasonable approximations, with the advantage of significantly reduced computational time compared to exact algorithms. Examples of approximation algorithms include the Nearest Neighbor and Minimum Spanning Tree methods. In this paper, the author will employ the Minimum Spanning Tree approach to address the TSP.

B. Greedy Algorithm

The greedy algorithm is a widely utilized approach for addressing various optimization problems, wherein decisions are made locally with the expectation that these choices will lead to a globally optimal solution. This methodology proves particularly effective in scenarios where a problem can be decomposed into a series of smaller decisions, and each decision can be made independently without requiring knowledge of future choices. A prominent example of the application of the greedy algorithm is found in the Traveling Salesman Problem (TSP). In this problem, a salesman is tasked with visiting a set number of cities and returning to the starting point while minimizing the total distance traveled. The Nearest Neighbour algorithm represents one of the greedy methods employed to tackle this issue. In this algorithm, the salesman consistently selects the nearest unvisited city as the next destination, disregarding potentially more advantageous routes that may become available later. The key components of the greedy algorithm include:

a) Candidate Set (C)

This refers to the collection of all potential candidates that can be selected at each step. In the context of TSP, the candidate set may consist of nodes (cities) within a graph that represents the travel routes. In other problems, candidates could include tasks to be completed, jobs to be performed, coins to be selected, objects to be picked, or characters to be chosen in a game.

b) Solution Set (S)

This set contains the candidates that have been selected and are part of the solution being constructed. In the case of TSP, the solution set will include the sequence of cities visited by the salesman.

c) Solution Function

This function is employed to ascertain whether the selected candidate set yields the desired solution. In TSP, this function will verify if all cities have been visited and whether the resulting route meets the criteria for the shortest distance.

d) Selection Function

This function is responsible for choosing candidates based on specific criteria.

C. Hamiltonian Path and Circuit

A Hamiltonian path is a path in a graph that visits each vertex exactly once without returning to the starting vertex. When this path returns to the starting vertex, it is referred to as a Hamiltonian circuit. In the context of the Traveling Salesman Problem (TSP), the optimal solution is identified as the Hamiltonian circuit with the minimum total weight.

D. Haversine Formula

The Haversine formula is used to calculate the great-circle distance between two points on a sphere given their latitudes and longitudes. It is particularly useful for computing distances between cities on Earth. The formula is:

$$d = 2R * \arcsin\left(\sqrt{[} sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + cos(\varphi_1) * cos(\varphi_2) * sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)) \right]$$

Where:

- d is the distance between the two points along the sphere's surface
- R is the radius of the sphere (for Earth, approximately 6,371 kilometers)
- φ_1, φ_2 are the latitudes of point 1 and point 2 in radians
- λ_1, λ_2 are the longitudes of point 1 and point 2 in radians

This formula accounts for the Earth's spherical shape and provides more accurate distance calculations compared to the Euclidean distance when measuring large distances across the Earth's surface.

E. The 2-Opt Algorithm

The 2-Opt algorithm is a widely recognized method employed to enhance solutions for the Traveling Salesman Problem (TSP). This algorithm operates by refining an existing route through a series of straightforward exchanges known as 2-Opt swaps. The fundamental principle behind this algorithm is to eliminate any crossings present in the TSP route, thereby resulting in a more efficient and shorter path.

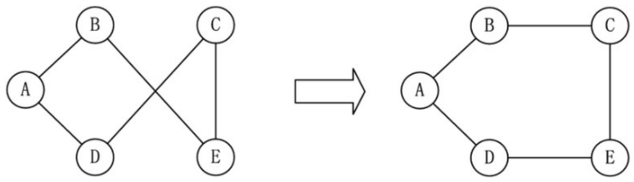


Figure 3. Algoritma 2-Opt Process
Source: https://www.researchgate.net/publication/281412705/figure/fig3/AS:668978481475599@1536508307852/Local-search-2-opt-and-Or-opt.png

As illustrated in the accompanying figure, the image depicts a route before and after the application of the 2-Opt algorithm. Initially, the route is represented as A-B-E-C-D-A, which contains a crossing. By reversing the sub-route E-C to C-E, a new, more optimal route is generated: A-B-C-E-D-A. The operational procedure of the 2-Opt algorithm can be summarized as follows:

1. Consider a route R and nodes v1 and v2, where the nodes between v_1 and v_2 are to be reversed.
2. Extract the segment of the route from the start to v_1 and append it to the new route in the same order.
3. Extract the segment from v_{1+1} to v_{2+1} and append it to the new route in reverse order.
4. Finally, extract the segment from v_{2+1} back to the start and append it to the new route in the same order.

F. Nearest Neighbor Algorithm

The Nearest Neighbor algorithm is one of the straightforward heuristics employed to address the Traveling Salesman Problem (TSP). This method utilizes a greedy approach, whereby at each step, the next destination selected is the nearest unvisited city. The steps of this algorithm are as follows:

1. Begin at a designated starting city.
2. Identify the closest unvisited city.
3. Move to that city.
4. Repeat steps 2 and 3 until all cities have been visited.
5. Return to the initial city.

III. Method

The author, as previously indicated in the introduction, seeks to establish an estimation of the most efficient complete tour route encompassing all nations within the Americas, using the capitals of these nations as key reference points. The Americas are categorized into distinct regions: North America, the Caribbean, South America, and Central America. In this framework, the author has assembled a comprehensive list of each country paired with its capital within the respective regions of Europe, a portion of which is presented below.

```
countries = {
    'North America': [
        "Washington DC United States",
        "Ottawa Canada",
        "Mexico City Mexico",
        "Guatemala City Guatemala",
        "San Salvador El Salvador",
        "Tegucigalpa Honduras",
        "Managua Nicaragua",
        "San Jose Costa Rica",
        "Panama City Panama"
    ],
    'Caribbean': [
        "Havana Cuba",
        "Santo Domingo Dominican Republic",
        "Port-au-Prince Haiti",
        "Kingston Jamaica",
        "Nassau Bahamas",
        "San Juan Puerto Rico",
        "Port of Spain Trinidad and Tobago",
        "Bridgetown Barbados"
    ],
    'South America': [
        "Brasilia Brazil",
          ...
          ...
```

Subsequently, the author utilized the Geopy library available in Python to obtain the coordinates of latitude and longitude. The acquired latitude and longitude coordinates were then stored in JSON format.

```
{
    "North America": {
        "Washington DC United States": {
            "latitude": 38.8950368,
            "longitude": -77.0365427
        },
        "Ottawa Canada": {
            "latitude": 45.4208777,
            "longitude": -75.6901106
        },
        "Mexico City Mexico": {
            "latitude": 19.4326296,
            "longitude": -99.1331785
        },
        "Guatemala City Guatemala": {
            "latitude": 14.6416142,
            "longitude": -90.5132836
        },
        "San Salvador El Salvador": {
            "latitude": 13.6989939,
            "longitude": -89.1914249
        },
          ...
          ...
```

Upon obtaining the coordinates of each national capital, the author calculated the geographical distances between one country and all others using the Haversine formula, which is employed to determine the distance between two points on a sphere (in this case, the Earth). The distances between each pair of countries were recorded in a weighted adjacency matrix. This weighted adjacency matrix serves as the complete graph representation for the 35 countries in question. From this complete graph, the shortest tour was identified using the Nearest Neighbour algorithm, as outlined in the following pseudocode:

```
procedure NearestNeighbor;
var
  path: array [0..n-1] of integer;
  unvisited: array [1..n-1] of integer;
  current: integer;
  nearest: integer;
  i: integer;

begin
```

```
  path[0] := 0; // Start from city 0
  for i := 1 to n-1 do
    unvisited[i] := i;

  current := 0;
  while unvisited != [] do
  begin
    // Find nearest unvisited city
    nearest := unvisited[1];
    for i := 2 to n-1 do
      if adj_matrix[current, unvisited[i]] <
adj_matrix[current, nearest] then
        nearest := unvisited[i];

    path[length(path)] := nearest;
    delete(unvisited, nearest);
    current := nearest;
  end;

  path[length(path)] := 0; // Complete the cycle
  Result := path;
end;
```

The Nearest Neighbour algorithm utilizes lists to store the nodes that have been visited. In its implementation, this algorithm employs two primary lists: one for maintaining the current route being formed (path) and another for keeping track of the nodes that have yet to be visited (unvisited). Each time the algorithm selects the nearest node, that node is removed from the unvisited list and added to the path list. This process ensures that each node is visited only once, facilitating the tracking of which nodes remain available for selection.

In the context of finding an optimal route across the continent of America, the algorithm initiates from Washington DC as the starting city and subsequently selects the nearest unvisited capital based on distances calculated using the Haversine formula. Although this greedy approach does not always yield the optimal solution, it is effective in providing reasonable solutions with better time complexity compared to exact algorithms. This is evidenced by implementation results that can resolve the problem for 35 countries in America in under 1 milliseconds. The selection of the nearest node at each iteration also aids in minimizing the likelihood of routes deviating significantly from the optimal solution, although there remains a possibility of route crossings, which can subsequently be optimized using the 2-Opt algorithm.

```
TSP Using NN

Washington DC United States -> Ottawa Canada ->
Nassau Bahamas -> Havana Cuba -> Kingston Jamaica ->
Port-au-Prince Haiti -> Santo Domingo Dominican
Republic -> San Juan Puerto Rico -> Basseterre Saint
Kitts and Nevis -> St. John's Antigua and Barbuda ->
Roseau Dominica -> Castries Saint Lucia -> Kingstown
Saint Vincent and the Grenadines -> St. George's
Grenada -> Port of Spain Trinidad and Tobago ->
Bridgetown Barbados -> Georgetown Guyana ->
Paramaribo Suriname -> Caracas Venezuela -> Bogota
Colombia -> Quito Ecuador -> Panama City Panama ->
Panama City Panama -> San Jose Costa Rica -> San Jose
Costa Rica -> Managua Nicaragua -> Managua Nicaragua
-> Tegucigalpa Honduras -> Tegucigalpa Honduras ->
San Salvador El Salvador -> San Salvador El Salvador
-> Guatemala City Guatemala -> Belmopan Belize ->
Mexico City Mexico -> Lima Peru -> La Paz Bolivia ->
Asuncion Paraguay -> Buenos Aires Argentina ->
Montevideo Uruguay -> Santiago Chile -> Brasilia
Brazil -> Washington DC United States
```

```
Total distance traveled: 33774.61743080288
Time taken to execute: 0.2088000183 ms
```

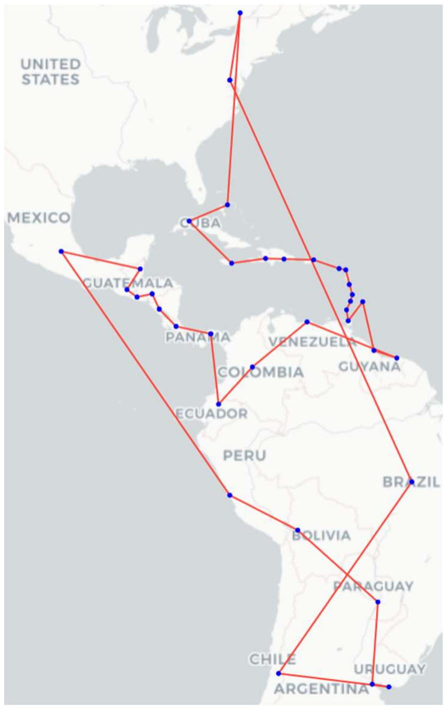


Figure 4. TSP Tour with Nearest Neighbour
Source: Author's Document

The approximation route for the Traveling Salesman Problem (TSP) indicates a total distance of 33.774.6 kilometers required to visit all 35 countries in the Americas and return to the starting point. However, it is evident that this route contains several suboptimal paths and intersections, such as the segment connecting the USA to Brazil and the one from Canada to the Bahamas, among others. There remains potential for further optimization of this route to eliminate all crossings by employing the 2-Opt crossover algorithm. The pseudocode for generating a new route using the 2-Opt algorithm is as follows:

```
function TwoOptSwap(tour: array of integer; i, j:
integer): array of integer;
var
  new_tour: array of integer;

begin
  new_tour := Copy(tour, 1, i-1) +
              Reverse(Copy(tour, i, j-i+1)) +
              Copy(tour, j+1, Length(tour)-j);
  Result := new_tour;
end;
```

```
function ImproveTour2Opt(tour: array of integer;
tour_cost: integer): array of integer;
var
  best_tour: array of integer;
  best_cost: integer;
  improved: boolean;
  i, j: integer;
  new_tour: array of integer;
  new_cost: integer;
```

```
begin
  best_tour := tour;
  best_cost := tour_cost;
  improved := False;

  while True do
  begin
    for i := 1 to Length(tour)-3 do
    begin
      for j := i+2 to Length(tour) do
      begin
        new_tour := TwoOptSwap(tour, i, j);
        new_cost := CalculateTspCost(new_tour);

        if new_cost < best_cost then
        begin
          best_tour := new_tour;
          best_cost := new_cost;
          improved := True;
          Break;
        end;
      end;

      if improved then
        Break;
    end;

    if not improved then
      Break;

    tour := best_tour;
    tour_cost := best_cost;
  end;

  Result := best_tour;
end;
```

After changing the tour with Nearest Neighbor by adding the 2-Opt algorithm, the tour results are as follows:

```
TSP Using NN and 2-Opt

Washington DC United States -> Nassau Bahamas ->
Havana Cuba -> Kingston Jamaica -> Port-au-Prince
Haiti -> Santo Domingo Dominican Republic -> San Juan
Puerto Rico -> Basseterre Saint Kitts and Nevis ->
St. John's Antigua and Barbuda -> Roseau Dominica ->
Castries Saint Lucia -> Bridgetown Barbados ->
Kingstown Saint Vincent and the Grenadines -> St.
George's Grenada -> Caracas Venezuela -> Bogota
Colombia -> Port of Spain Trinidad and Tobago ->
Georgetown Guyana -> Paramaribo Suriname -> Brasilia
Brazil -> Asuncion Paraguay -> Montevideo Uruguay ->
Buenos Aires Argentina -> Santiago Chile -> La Paz
Bolivia -> Lima Peru -> Quito Ecuador -> Panama City
Panama -> Panama City Panama -> San Jose Costa Rica -
> San Jose Costa Rica -> Managua Nicaragua -> Managua
Nicaragua -> Tegucigalpa Honduras -> Tegucigalpa
Honduras -> San Salvador El Salvador -> San Salvador
El Salvador -> Guatemala City Guatemala -> Belmopan
Belize -> Mexico City Mexico -> Ottawa Canada ->
Washington DC United States

Total distance traveled: 28004
Time taken to execute: 26.049100008094683 ms
```

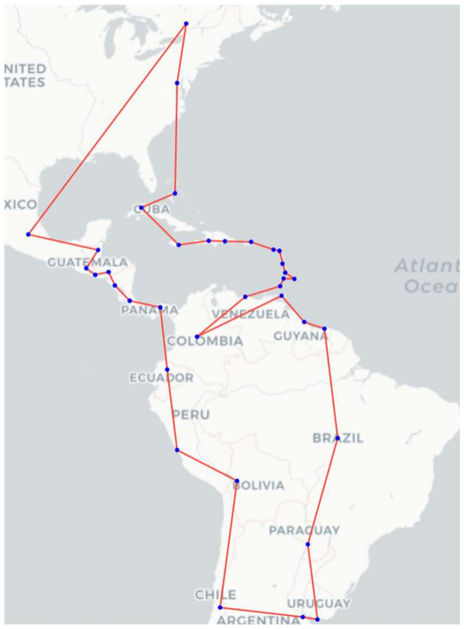


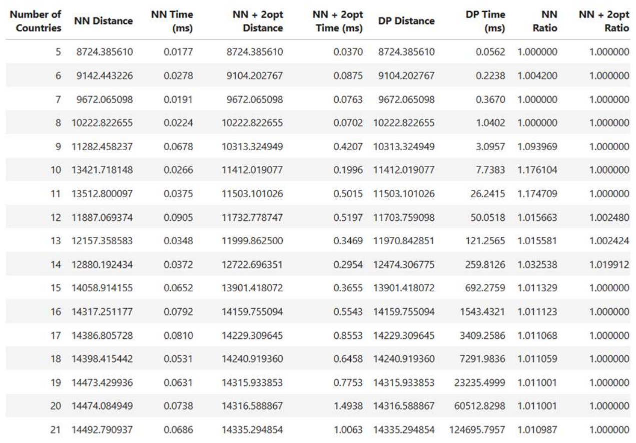
Figure 5. TSP Tour with Nearest Neighbour and 2-Opt
Source: Author's Document

The results indicate a significant difference in costs, as the implementation of the Nearest Neighbour algorithm combined with the 2-Opt method yields a considerably lower cost (28,004 compared to 33,774.6). Furthermore, the resulting route no longer exhibits any intersections. The execution time required enhance quite significant, recorded at approximately 26.04 milliseconds.

IV. RESULTS AND DISCUSSION

The application of approximation algorithms such as Nearest Neighbour and 2-Opt does not always yield optimal solutions; however, they can provide solutions within reasonable bounds. In the worst-case scenario, the Nearest Neighbour algorithm, when not combined with 2-Opt, may produce a cost that is twice that of the optimal solution. When the 2-Opt algorithm is incorporated, the cost generated by the Nearest Neighbour method can be optimized by eliminating crossing routes.

The author has conducted an analysis comparing the solutions produced by three algorithms: the Traveling Salesman Problem (TSP) solved with the Held-Karp method (exact solution), the TSP using Nearest Neighbour, and the TSP employing both Nearest Neighbour and 2-Opt. This comparison was performed for a range of countries from 5 to 21. For instances involving more than 21 countries, the Held-Karp algorithm requires significant time due to its exponential complexity.



Number of Countries	NN Distance	NN Time (ms)	NN + 2opt Distance	NN + 2opt Time (ms)	DP Distance	DP Time (ms)	NN Ratio	NN + 2opt Ratio
5	8724.385610	0.0177	8724.385610	0.0370	8724.385610	0.0562	1.000000	1.000000
6	9142.443226	0.0278	9104.202767	0.0875	9104.202767	0.2238	1.004200	1.000000
7	9672.065098	0.0191	9672.065098	0.0763	9672.065098	0.3670	1.000000	1.000000
8	10222.822655	0.0224	10222.822655	0.0702	10222.822655	1.0402	1.000000	1.000000
9	11282.458237	0.0678	10313.324949	0.4207	10313.324949	3.0957	1.093969	1.000000
10	13421.718148	0.0266	11412.019077	0.1996	11412.019077	7.7383	1.176104	1.000000
11	13512.800097	0.0375	11503.101026	0.5015	11503.101026	26.2415	1.174709	1.000000
12	11887.069374	0.0905	11732.778747	0.5197	11703.759098	50.0518	1.015663	1.002480
13	12157.358583	0.0348	11999.862500	0.3469	11970.842851	121.2565	1.015581	1.002424
14	12880.192434	0.0372	12722.696351	0.2954	12474.306775	259.8126	1.032538	1.019912
15	14058.914155	0.0652	13901.418072	0.3655	13901.418072	692.2759	1.011329	1.000000
16	14317.251177	0.0792	14159.755094	0.5543	14159.755094	1543.4321	1.011123	1.000000
17	14386.805728	0.0810	14229.309645	0.8553	14229.309645	3409.2586	1.011068	1.000000
18	14398.415442	0.0531	14240.919360	0.6458	14240.919360	7291.9836	1.011059	1.000000
19	14473.429936	0.0631	14315.933853	0.7753	14315.933853	23235.4999	1.011001	1.000000
20	14474.084949	0.0738	14316.588867	1.4938	14316.588867	60512.8298	1.011001	1.000000
21	14492.790937	0.0686	14335.294854	1.0063	14335.294854	124695.7957	1.010987	1.000000

Figure 6. TSP Approximation optimality comparison table
Source: Author's Document

Based on the conducted tests, it is evident that the TSP algorithm utilizing Nearest Neighbour combined with 2-Opt yields great results, with an average ratio of the solution compared to the exact solution being approximately 1.0014597531865865. This would mean that the result from the aforementioned method has a relatively small deviation from the exact solution as expected for an optimal solution. In contrast, the TSP algorithm using Nearest Neighbour alone, without the 2-Opt enhancement, has an average ratio of 1.0347254478760672.

The implementation of the Nearest Neighbour algorithm entails a time complexity of $O(n^2)$. This complexity arises from the need to search for the nearest unvisited city during each iteration. For each city (n iterations), the algorithm must evaluate the distances to all unvisited cities (up to n-1 evaluations). Consequently, the overall time complexity is $O(n^2)$. The route modification using 2-Opt has a time complexity of $O(n^2)$ for each improvement attempt, and in the worst case, it may require $O(n)$ improvements. Therefore, the actual time complexity for 2-Opt is $O(n^3)$. Thus, the total time complexity for constructing an approximate solution to the TSP using both Nearest Neighbour and 2-Opt is $O(n^2) + O(n^3) = O(n^3)$, as the higher-order term dominates.

V. Conclusion

The American continent comprises numerous countries, and determining the optimal route to visit all these nations at minimal cost presents a challenge that can be addressed through the Traveling Salesman Problem (TSP). The TSP can be approached through various methods, one of which involves the use of approximation algorithms. Employing approximation algorithms such as Nearest Neighbour and 2-Opt can significantly expedite the solution process, enabling the resolution of the TSP for a number of cities reaching into the thousands; however, this comes with the trade-off that the solutions obtained may not be optimal.

VI. Appendix

Link Grihub: https://github.com/Farhanabd05/makalah-matdis-nn

VII. Acknowledgment

I express my profound gratitude to the Almighty God for all the blessings and grace that have enabled me to successfully complete this paper. I would also like to extend my heartfelt thanks to:

1. Mr. Ir. Rila Mandala, M.Eng., Ph.D., my instructor for the Discrete Mathematics course, for his invaluable guidance;
2. Mr. Dr. Ir. Rinaldi Munir, M.T., for his website, which greatly facilitated the information-gathering process related to the topic discussed in this paper;
3. My friends and family for their unwavering support.

References

[1] Munir, Rinaldi. 2023. Greedy (Bag. 1): Bahan Kuliah Strategi Algoritma. Accessed 6 January 2025

[2] GeeksforGeeks, "Traveling Salesman Problem,". Avalaible: https://www.geeksforgeeks.org/travelling-salesman-problem-using-dynamic-programming/ . Accessed 1 January 2025

[3] Jazib, M. (2023, June 22). List of countries in Americas: Geographic locations, full list, Important facts. *Jagranjosh.com*. https://www.jagranjosh.com/general-knowledge/list-of-countries-in-americas-1687427717-1. Accessed 6 January 2025

[4] Some important heuristics for the TSP" https://ocw.mit.edu/courses/1203j-logistical-and-transportation-planning-methods-fall 2006/03634d989704c2607e6f48a182d455a0_lec16.pdf, Accessed 1 January 2025

[5] Munir, Ranldi. 2024. Graf (Bag. 1): Bahan Kuliah Matematika Diskrit. Accessed 6 January 2025

Statement

Hereby, I declare that this paper I have written is my own work, not a reproduction or translation of someone else's paper, and not plagiarized.

Bandung, 26 Desember 2024

[illegible]

Abdullah Farhan 13523042