Towards Comparative Analysis of The Complexity of Tour Construction Heuristics for Solving the Travelling Salesman Problem

OGBULOKO VINCENT ECHE[1], ADEREMI ELISHA OKEYINKA[2], IBRAHIM ABDULLAHI[3], ABDULGANIYU ABDULRAHMAN[4]

[1, 2, 3, 4]Department of Computer Science, Faculty of Natural Sciences, Ibrahim Badamasi Babangida University, Lapai, Niger State, Nigeria.

Abstract - The Travelling Salesman Problem (TSP) is a Combinatorial Optimization Problem (COPs) which has gained wide attention of computer scientists specifically because it is simple to define but very difficult to solve. For instance, when traveling (delivering) to seven cities, there could be up to 720 possible routes to consider, so finding the most efficient and feasible route requires evaluating every possible route, which is a computationally challenging task. TSP is NP - hard and does not have an effective polynomial - time solution, so effective heuristic methods are needed to solve it. This paper presents a comparative analysis of complexity of five tour construction algorithms for solving the travelling salesman problem.

Indexed Terms- Combinatorial Optimization, TSP, NP - hard, Heuristics, NNH, NIH, FIH, CIH, RIH, Nodes, Edges.

I. INTRODUCTION

Combinatorial optimization problems are optimization problems in the discrete space which have different types of solutions comparing to the problems in the continuous space. Many of these types of problems are NP-hard and do not have an effective polynomial-time solution. So, effective methods are needed to solve these problems. In the last 30 years, meta-heuristic approaches have been used extensively for solving related problems. Combinatorial optimization problems can be encountered in many fields such as routing, scheduling, planning, decision-making processes, transportation and telecommunications, etc.

The Travelling Salesman Problem (TSP) is a popular example of combinatorial optimization problem. It is an algorithmic and graph computational problem where one needs to visit a set of cities (nodes in a graph) exactly once and the distances (edges in a graph) between all the cities are known. The solution that is needed for this problem is the shortest route in which one visits all the cities and returns to the starting city.

Mathematically, TSP is formulated as a graph G = (V, E), where V is a set of nodes or vertices, E is a set of edges or arcs and let d = (dij) be a distance matrix associated with E.

The objective function for the TSP is to minimize the total distance travelled, which can be represented as:

$$\sum_{i,j=1}^{n} d_{ij} x_{ij} \qquad (1)$$

Subject to constraints that ensure:

i. Each city is visited exactly once
ii. The salesman returns to the starting city
iii. There are no sub tours

Many researchers have tried to solve TSP using different algorithms such as branch & bound, dynamic programming, artificial bee colony (ABC), ant colony optimization (ACO), etc.

This paper gives a comparative analysis of time complexity of five tour construction algorithms in solving the travelling salesman problem. These heuristics are; Nearest Neighbor Heuristic (NNH), Nearest Insertion Heuristic (NIH), Farthest Insertion Heuristic (FIH), Cheapest Insertion Heuristic (CIH) and Random Insertion Heuristic (RIH).

II. RESEARCH MOTIVATION i.

The following are some of the motivations for this research study;

i. While there are numerous formulated Combinatorial Optimization Problems, ithe TSP is perhaps the most central to the field of combinatorial optimization.
ii. Work on the TSP has been a driving force for the emergence and advancement of many important research areas, such as integer programming, as well as for the development of complexity theory.
iii. Additionally, the TSP has also become a standard test bed for new algorithmic ideas; many of the most important techniques for solving combinatorial optimization problems were developed using the TSP as an example application.

III. RESEARCH PROBLEM

i. Given a set of nodes and edges between every pair of nodes, the problem is to find the shortest possible route (path) that passes every node exactly once and returns to the starting node. In theory, solving the Traveling Salesman Problem (TSP) is simple as it involves finding the shortest route for every trip within a city. However, as the number of cities increase, manually solving TSP becomes increasingly difficult. With just n cities, the permutations and combinations are already numerous. Adding just one more cities can exponentially increase the number of possible solutions.
ii. Notwithstanding, the numbers of computational heuristics, many real life problems of great significant remain largely unsolvable within the constraints of polynomial time due to limitations of exact algorithms in solving Combinatorial Optimization Problems.

IV. THE GOAL OF THE STUDY

The goal of this study is to consider some combinatorial optimization heuristics with special focus on tour construction approximation heuristics, with the following specific objectives:

To evaluate the time complexity of nearest neighbor, nearest insertion, farthest insertion, cheapest insertion and random insertion algorithms.

To compare the complexities obtained in (i) above

V. RELATED WORKS

The travelling salesman problem has been extensively studied in the field of combinatorial optimization due to its relevance in various real – world applications. This literature review focuses on the complexity analysis of algorithms in the context of the travelling salesman problem.

Rahman, et al (2024). Presented the improvement of the Nearest Neighbor Heuristic Search Algorithm for travelling salesman problem. The improved NNA sorts all edges and then it takes a short edge with two vertices. The improved NNA began its route with a short distance and repeatedly sorts all edges and takes the shortest routes.

Farid, et al (2022). In their study stated that the first step in implementing the Cheapest Insertion Algorithm in determining shortest delivery route is to collect address data for which the route of delivery of the goods will be searched. The second step is to create a graph representation of the data that has been obtained. After that, compute the total distance for shipping goods from the route used by SiCepat Express Baleendah. The next step is to find the route and total distance for shipping goods using the Asymmetric Travelling Salesman- Cheapest Insertion Heuristic (ATSP CIH) application. The result obtained are then compared with the route used by SiCepat Express Baleendah to determine the efficiency of using the Cheapest Insertion Algorithm (CIH) in the ATSP CIH application.

Lity, *et al* (2022). Modelled the product ordering process of the incremental Software Product Line (SPL) analysis as a Travelling Salesman Problem (TSP). The aim was to optimize product orders and improve the overall SPL analysis. Products were modelled as a node in a graph, and the solution-space information defined edge weights between products nodes. Existing graph route-finding heuristics were used to obtain the path with minimal cost. The first heuristic deployed was the nearest neighbor heuristic. The nodes were analyzed according to their similarity, so the nearest neighbor heuristic path was built by adding the product (node) most similar to the last node. However, it was observed that the approximation quality was poor because it first greedily added all the similar nodes and later suffered the curse of dimensionality when not-so-similar nodes were to be added. To overcome this, a lookup was introduced to examine the next node to be added to the computed path. Thereafter, two insertion heuristics, namely Nearest Insertion and Farthest Insertion were deployed to insert the remaining product into the existing path created by the Nearest Neighbor Heuristic. The proposed method was simulated on a prototype and evaluated for applicability and performance; a significantly more optimized SPL process was reported.

Nemani, et al. 2021). Tested and evaluated the performance of three algorithms: Simulated Annealing, Ant Colony Optimization, and Genetic Algorithm by setting up analogous environments of n cities. The traveling salesman problem (TSP) is one of the most extensively studied optimization problems in the computer science and computational mathematics field given that there is yet an optimal solution for it to be discovered. This algorithmic issue requests the shortest possible route that visits each city precisely once and returns to its initial starting point if a list of n places and the distances between each pair are given. This paper conducts a comparative study to test and evaluate the performance of three algorithms: Simulated Annealing, Ant Colony Optimization, and Genetic Algorithm. With the traveling salesman problem classifying under NP-hard computational complexity, the proposed research work will examine the runtime as well as the shortest distance computed by each of these algorithms by setting up analogous environments of n cities.

Ono, et al (2020). Stated that Travelling Salesman Problem can be summarized by the following: a list of destinations and the distances between each pair of destinations, determine the shortest path that visits each destination only once. TSP is an obvious application of a minimum cost network flow.

Sundar, & Rathinam, (2016). Investigated the generalized multiple depots traveling salesmen problem (GMDTSP). They were motivated by several applications in network design, health-care logistics and scheduling. The problem was defined as follows. Given a set of customers divided into a predefined number of clusters and a set of K depots, search for a collection of at most K cycles having the following properties. Each cycle begins and ends at a certain depot, at least one customer from each cluster is visited by some cycle, and the total costs of the collection of cycles are minimized.

Amarbir, (2016). Undertakes the review of algorithms used to solve Multiple Travelling Salesman Problem. The techniques are categorized into heuristic, meta-heuristics and exact algorithms and out of these the exact approaches are used for relatively small problems.

Yulianto, (2018). Stated that the Cheapest Insertion Heuristic method builds a tour from small cycles with minimal weight and successively adds new points. The selection of the new point is carried out simultaneously with the selection of the edge so that the minimum insertion value is obtained. Then the new point is inserted between the two points that make up the side that has been selected.

Fadhillah, (2017). Opined that Travelling Salesman Problem (TSP) is one of the distribution problems that been discussed for a long time in optimization studies that usually occur in everyday life. The TSP problem is about someone who has to visit all cities exactly once and return to the initial city with minimum distance.

Hoel, (2015). Offered a greedy algorithm-based solution to the travelling salesman problem. The greedy algorithm is like the Nearest Neighbor Algorithm and the route starts from that particular sub-route with two cities, which has the shortest distance among all such feasible sub-routes.

Asani, et al (2024), in their research work titled A Novel Insertion Solution for the Travelling

Salesman Problem proposed an insertion method referred to as the Half Max Insertion Heuristic (HMIH). Their motivation was to explore some strategies with the possibility of improving tour accuracy. The design of HMIH was based on two observations, namely;

i. The superior solution quality of insertion techniques based on the use of polygon as an initial tour
ii. The limitation of FIH's accuracy due to the distance between its initial circuits and the next node to be inserted. *Huang et al (n.d.)* argued that although FIH performs relatively well, the distance between its circuit and new nodes to be inserted impedes accuracy.

Anitha, & Sandeep (2015). In their paper titled "Literature survey on travelling salesman problem using genetic algorithms." In the introduction of the paper stated that optimization is the process of making something better. An optimization problem is a problem which boosts the solution of finds the better solution from all available solution spaces. The terminology:" best" solution implied that there is more than one solution. The travelling salesman problem also results in more one solution, but the aim is to find the best solution in a reduced time and the performance is also increased.

Kumar, et al (2012). Stated that genetic algorithm is one of the best methods which is used to solve various NP-hard problem such as TSP. The natural evolution process is always used by genetic Algorithm to solve the problems. They presented a critical survey to solve TSP problem using genetic algorithm methods that are proposed by researchers. They observed that there is requirement to design new genetic operators that can enhance the performance of the GA used to solve TSP. There is lot of scope for the researcher to do work in this field in future.

Krari, & Benani, (2019). In their pre-processing technique selects from every cluster the closest vertices to the other clusters and removes the vertices that have never been chosen to reduce the solution search space size. The proposed method had very small running times, while the rate of the reduction is up to 98%, therefore being very competitive against the reduction algorithms proposed by *Gutin & Karapetyan (2009*). Different GTSP solvers were applied to the reduced instances in order to assess their performance. The results showed that reduced instances helped the solvers find good feasible solutions in very short computational times, but are not guaranteed to find optimal solutions.

Bernardino, & Paias (2018). Presented several compact and non-compact models of the problem, while Pop, Matei, and Pintea (2018) described an innovative technique to solve the FTSP. They split the problem into two smaller sub problems operating at the macro and micro levels, and solved them individually. The macro level sub problem is aimed at providing a collection of tours visiting the families while employing a classical genetic algorithm (GA) and a diploid GA (i.e., *GA with individuals consisting of two coupled chromosomes Petrovan, Matei, & Pop, 2023a*). The micro level sub problem is aimed at finding the minimum-cost tour, associated with each generated tour at the macro level that visits a given number of vertices belonging to each family.

Yuan, et al (2020). Motivated by applications in the field of delivery services. The GTSP-TW is defined on directed graphs with the set of vertices divided into clusters with the following properties. One cluster includes only the depot and for every vertex we associate a time window, during which the visit must take place if the corresponding vertex is visited. The goal of the GTSP-TW is to find the shortest Hamiltonian tour beginning and ending at the depot so that each cluster is visited exactly once and the time constraints are fulfilled, i.e., for each cluster the selected vertex is visited within its time window.

VI. RESEARCH METHODOLOGY

We have examined the complexities of five (5) approximation algorithms with emphasis on tour construction algorithms focusing on solving the travelling salesman problem (TSP). our primary objective was to conduct a comparative analysis to determine the efficiency of these algorithms when utilized in the context of the travelling salesman problem. To achieve this , we synthesized the complexities and subjected them to experimental runs in the same programming environment.

Python programming language is the language used to code the algorithms. All the algorithms were tested using arrays (nodes) of different sizes. Ten (10) datasets were used in the testing in the range of 4 - 50 nodes. The experiments were run on Intel core15 Hp laptop with 12 GB memory and Windows 10 (64 bits) operating system.

The basic process of tour construction heuristics can be summarized as follows;

i. Sub-tour establishment rule
ii. Selection rule
iii. Expansion rule
iv. Repeated application of steps (ii) and (iii) until a complete tour is obtained.

The expansion rules can be categorized into two (2) types; insertion and addition. An insertion - based expansion rule chooses where in the permutation to place the new city on the basis of the cost of the resulting sub-tour, whereas an addition - based expansion rule bases this decision on next - hop distance.

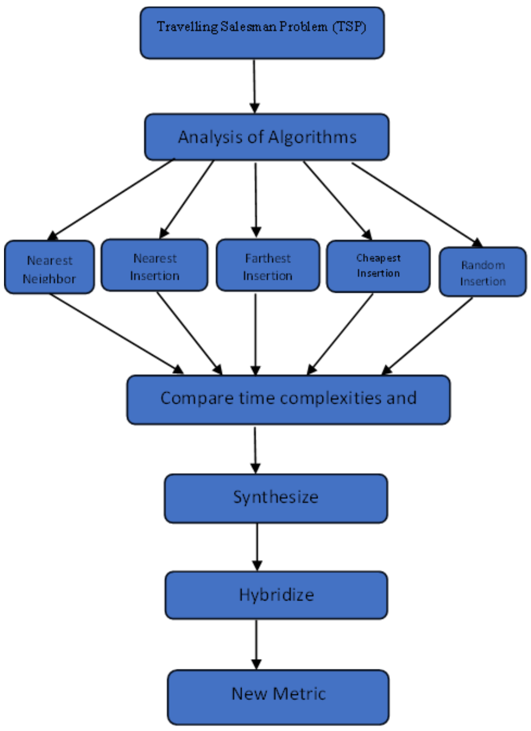

Figure 1: Conceptual framework

The Heuristics for solving the Travelling Salesman Problem (TSP):

The five (5) tour construction approximation algorithms used in this research study are;

i. Nearest Neighbor Heuristic (NNH)
ii. Nearest Insertion Heuristic (NIH)
iii. Farthest Insertion Heuristic (FIH)
iv. Cheapest Insertion Heuristic (CIH)
v. Random Insertion Heuristic (RIH)

Nearest Neighbor Heuristic (NNH)

Nearest Neighbor Algorithm is a simple and intuitive approximation for the TSP. This is a greedy approach and the greedy criterion is in selecting the nearest city. It starts at an arbitrary city and repeatedly selects the nearest unvisited city until all cities have been visited.

Algorithm 1: Nearest-Neighbor Algorithm (NNH)

i. Start at any node
ii. Select the nearest unvisited neighbor and add it to end of tour.
iii. Repeat step (ii) until all nodes are added to the tour.

Time complexity = $O(n^2)$

NNH - code

```
import NumPy as np
 def nearest_neighbor (points, current_point):
      distances = np.linalg.norm (points - current_point, axis=1)
return points [np.argmin (distances)]

def tsp_nn(points): num_points = len(points)
      solution = [points [0]] # Start at the first point
            current_point = points [0]
              for _ in range (num_points - 1):
                 next_point = nearest_neighbor (points, current_point)
             solution.append (next_point)
      current_point = next_point
return solution
points = np.array ([(15, 0), (20, 0), (35, 0), (30, 0)])
solution = tsp_nn(points)
```

Nearest Insertion Heuristic (NIH)

Nearest Insertion Algorithm is still greedy but not as greedy as nearest neighbor. It allows partial tour to be modified

Algorithm 2: Nearest Insertion Algorithm (NIH)

I. Start the tour at any node

II. Pick the nearest unvisited - neighbor of the selected node in the tour
III. Insert it into the tour T = t_1, …..,t_k so that the total tour distance (cost) is minimized. i.e., find (i,j,k) = w(i, k) + w(k, j) - w(i, j) is minimize
IV. Repeat steps (ii) and (iii) until all nodes are added to the tour.
V. Time complexity = O(n^2)

NIH - code

```
import NumPy as np

def nearest_point (points, point):
   distances = np.linalg.norm (points - point, axis=1)
return points [np.argmin (distances)]

def insert_point (solution, point):
     nearest = nearest_point (solution, point)
       index = np.where (solution == nearest) [0][0]
     solution = np.insert (solution, index + 1, point, axis=0)
return solution

def tsp_ni(points):
     num_points = len(points)
       solution = [points [0]] # Start with the first point
         for i in range (1, num_points):
   solution = insert_point (solution, points[i])
return solution
points = np.array ([(30, 0), (15, 0), (20, 0), (35, 0)])
solution = tsp_ni (points)
print(solution) # Output: [(30, 0), (15, 0), (20, 0), (35, 0)]
```

Farthest Insertion Heuristic (FIH)
Farthest - Insertion Algorithm Start with a tour consisting of the two cities that are farthest apart. Repeat the following: Among all cities not in the tour, choose the one that is farthest from any city already in the tour. Insert it into the tour in the position where it causes the smallest increases in the tour distance.

Algorithm 3: Farthest Insertion Algorithm (FIH)
i. Start the tour at any node
ii. Pick the nearest farthest unvisited neighbor of the selected node.
iii. Insert it into the tour T = t_1, ….., t_k so that the total tour distance (cost) is minimized. i.e., find (i,j,k) = w(i, k) + w(k, j) - w(i, j) is minimized
iv. Repeat steps (ii) and (iii) until all nodes are added to the tour.
v. Time complexity = O(n^2)

FIH - code

```
import NumPy as np
def farthest_point (solution, points):
     max_distance = 0
        farthest = None
          for point in points:
            distance = np.min (np.linalg.norm (solution - point, axis=1))
        if distance >max_distance:
     max_distance = distance
  farthest = point
return farthest

def insert_point (solution, point):
     solution = np.append (solution, point, axis=0)
return solution

def tsp_fi(points):
     num_points = len(points)
       solution = [points[0]]  # Start with the first point
           remaining_points = points[1:]
               for i in range(1, num_points):
           farthest = farthest_point(solution, remaining_points)
       solution = insert_point(solution, farthest
remaining_points = np.delete(remaining_points, np.where(remaining_points == farthest)[0][0], axis=0)
return solution

points = np.array([(30, 0), (15, 0), (20, 0), (35, 0)])
solution = tsp_fi(points)
print(solution)  # Output: [(30, 0), (15, 0), (20, 0), (35, 0)]
```

Cheapest Insertion Heuristic(CIH):
The cheapest insertion algorithm is a heuristic method that build a tour from small cycles with minimal weight and successively adds new nodes.

Algorithm 4: Cheapest Insertion Algorithm (CIH):

I. Start with a partial tour from a node
II. Create a sub tour relationship; a sub tour link is created between two (2) places. It is a journey from the first place and ends in the first place.
III. Change the direction of the relationship (insertion) . One of the directions of the relationship (arc) of two places with a combination of two arcs, namely arc (i,j) is change to arc (i,k) and arc (k,j) where k is the insertion point with the smallest additional distance which is obtained from (i,j,k) = w(i,k) + w(k,j) - w(i,j).
IV. Repeat steps (ii) and (iii) until all nodes are added to the tour.
V. Time complexity = O(n^2 log n)

CIH - code

```
import NumPy as np
def cheapest_point (solution, points):
    min_cost = float('inf')
        cheapest = None
            for point in points:
            cost = np.min (np.linalg.norm (solution
                         - point, axis=1))
              if cost <min_cost:
      min_cost = cost
  cheapest = point
return cheapest

def insert_point (solution, point):
  solution = np.append (solution, point, axis=0)
return solution

def tsp_ci (points):
    num_points = len(points)
        solution = [points [0]] # Start with the first
point
            remaining_points = points [1:]
                for i in range (1, num_points):
        cheapest = cheapest_point (solution,
            remaining_points)
    solution = insert_point (solution, cheapest)
  remaining_points = np.
delete(remaining_points, np. where
(remaining_points == cheapest) [0][0], axis=0)
 return solution

points = np.array([(30, 0), (35, 0), (20, 0), (15,
0)])
solution = tsp_ci (points)
print(solution)  # Output: [(30, 0), (35, 0), (20, 0),
(15, 0)]
```

Random Insertion Heuristic (RIH)

The random insertion algorithm is a heuristic method used to construct an approximate solution for the Traveling Salesman Problem (TSP) by randomly selecting the start node as an initial tour and continue selecting randomly until all nodes are added to the tour.

Algorithm 5: Random Insertion Algorithm (RIH)

i. Select a random node (or a pre-specified node) as the initial tour.
ii. Choose the next node to join the tour randomly from the remaining nodes (not yet connected to the tour).
iii. Calculate the cost of inserting a node between two existing nodes in the tour. The cost is defined as: (i,j,k) = w(i, k) + w(k, j) - w(i, j) is minimize Insert the selected node at the location that minimizes the insertion cost.
iv. Repeat steps (ii) and (iii) until all nodes have been added into the tour.
v. Time complexity = O(n^2).

RIH - code

```
import NumPy as np
def insert_point (solution, points):
random_index = np.random.choice(len(points))
    solution = np.insert (solution, np.
        random.randint (0, len(solution)),
  points[random_index], axis=0)
return solution

def tsp_ri(points):
    num_points = len(points)
        solution = [points [0]] # Start with the first
point
            remaining_points = points [1:]
          for i in range (1, num_points):
        solution = insert_point (solution,
            remaining_points)
  remaining_points = np. delete
(remaining_points, np. random.randint (0,
len(remaining_points)), axis=0)
return solution

points = np.array ([(30, 0), (15, 0), (20, 0), (35,
0)])
solution = tsp_ri(points)
```

```
print(solution) # Output: a random permutation of the points
```

Tour path visualization:

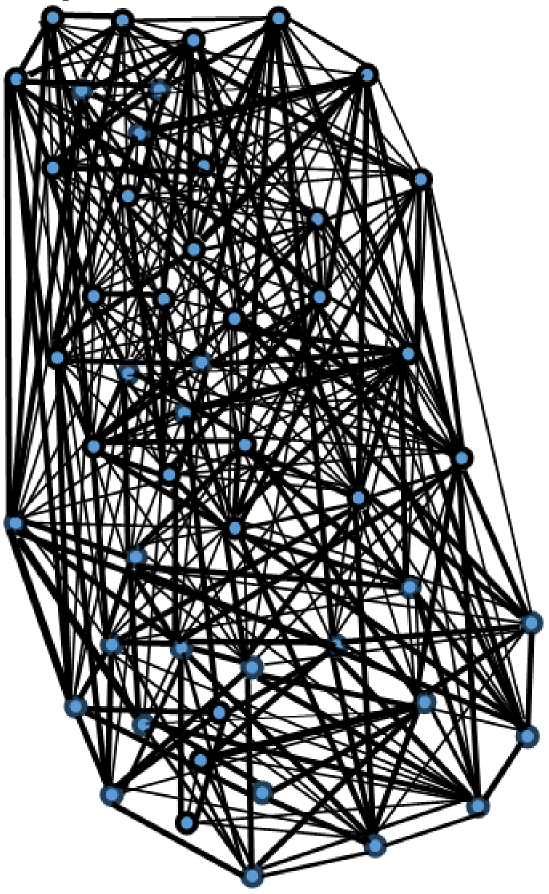

Figure 2: A graph showing 50 nodes tsp

VI. RESULTS AND FINDINGS

We provide details comparative analysis of the five (5) tour construction algorithms using time complexity measurement in seconds.

Table 1: Computational speed analysis of 5 tour construction algorithms

No of Nodes	Nearest Neighbor Heuristic	Nearest Insertion Heuristic	Farthest Insertion Heuristic	Cheapest Insertion Heuristic	Random Insertion Heuristic
4	0.042633	0.011111	0.012018	0.010111	0.011903
10	0.053338	0.009997	0.021974	0.007988	0.011569
15	0.034997	0.015641	0.024966	0.017999	0.021841
20	0.055915	0.010123	0.032691	0.010079	0.011972
25	0.015972	0.012519	0.014222	0.011270	0.013977
30	0.035949	0.021966	0.035849	0.011790	0.028748
35	0.047920	0.032047	0.040691	0.021418	0.033952
40	0.116592	0.031172	0.111411	0.022115	0.101003
45	0.120016	0.071123	0.100100	0.024671	0.100129
50	0.123849	0.100866	0.122849	0.028968	0.112811

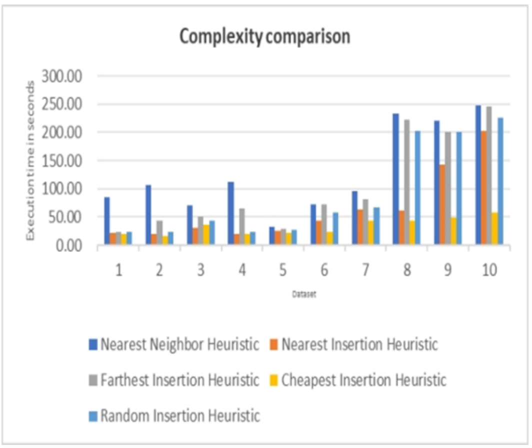

Figure 3: A graph showing time complexities of NNH, NIH, FIH, CIH and RIH

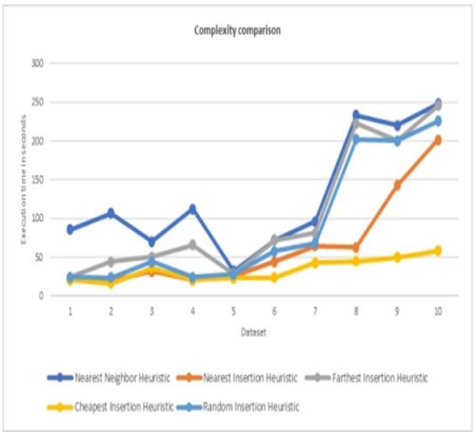

Figure 4: A graph showing time complexity between NNH, NIH, FIH, CIH and RIH

Table 2: Total time complexity of individual algorithms.

S/N	ALGORITHM	TOTAL TIME COMPLEXITY (TTC)
1	Nearest Neighbor Heuristic	0.647181
2	Nearest Insertion Heuristic	0.332096
3	Farthest Insertion Heuristic	0.516781
4	Cheapest Insertion Heuristic	0.166409
5	Random Insertion Heuristic	0.447899

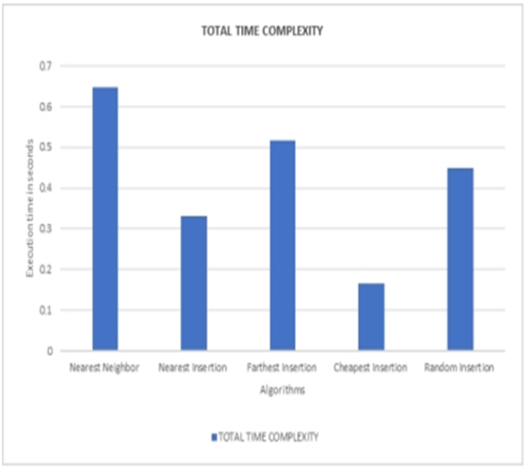

Figure 5: A graph showing total time complexity of the five algorithms

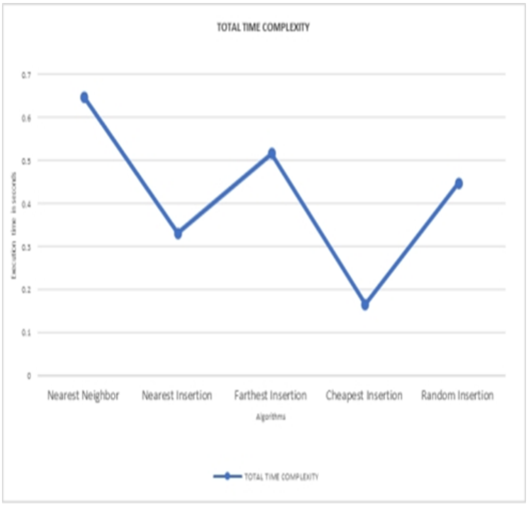

Figure 6: Comparison of total time complexity

VII. THE FINDINGS

The research study compares the time complexity and execution speed of five tour construction heuristics namely; Nearest Neighbor Heuristic (NNH), Nearest Insertion Heuristic (NIH), Farthest Insertion Heuristic (FIH), Cheapest Insertion Heuristic (CIH) and Random Insertion Heuristics RIH).

i. Figures 3 and 4 shows clearly that Nearest Neighbor Heuristic (NNH) had the fastest computational speed followed by Farthest Insertion Heuristic (FIH), Random Insertion Heuristic (RIH), Nearest Insertion Heuristic (NIH) and Cheapest Insertion Heuristic (CIH). This is consistent with literature findings that insertion techniques requires more computational time than the addition techniques to complete a tour..*(Laha, et al 2016, Babel 2020)*

ii. According to *(Lity, et al 2017, Babel 2020)* computational speed is affected by the insertion criteria for computation

iii. Farthest Insertion Heuristic (FIH) generally performs best in the community of insertion heuristics of O(n^2) or low time complexity.

iv. Figures 5 and 6 using the total execution time and speed also demonstrate that the Nearest Neighbor Heuristic (NNH) out-performed the four other heuristics in this research study.

REFERENCES

[1] Amarbir, (2016). A Review on Algorithms used to solve Multiple Travelling Salesman Problem. *International Research Journal of Engineering and Technology (IRJET)*, *3*(4), 598–603.

[2] Anitha, and Sandeep, (2015). Literature Survey om Travelling Salesman Problem Using Genetic Algorithms. *International Journal of Advanced Research in Education Technology (IJARET)*, *2*(1).

[3] Asani, et al (2020). A construction tour technique for solving the travelling salesman problem based on convex hull and nearest neighbor heuristics. *International Conference on Mathematics, Computer Engineering and Computer Science*, 1–4.

[4] Asani, et al (2024). A Novel Insertion Solution for the Travelling Salesman Problem. *Computers, Materials & Continua*.

[5] Babel, (2020). New heuristic algorithms for the Dublin traveling salesman problem. Journal of Heuristics

[6] Bernardino and Paias (2018). Solving the family traveling salesman problem. European Journal of Operational Research. 267 (2).

[7] Fadhillah, et al (2017). Solving travelling salesman problem using heuristics approaches. *Journal of Global Research in Computer Science/*

[8] Farid, et al (2022). Implementation of Cheapest Insertion Heuristic Algorithm in Determining Shortest Delivery Route. *International Journal of Global Research*, *3*(2), 37–45.

[9] Krari, et al (2021). A pre – processing reduction method for the generalized travelling salesman problem. European Journal of Operation Research. 21(11)

[10] Kumar, et al (2012). A Genetic Algorithm Approach to study travelling salesman problem. Journal of Global Research in Computer Science. 3(3):33-8

[11] Laha, et al (2016). Nature – Inspired Metaheuristics for optimizing Information Dissemination in Vehicular Networks. TECNALIA Research and Innovation, Derio, Spain.

[12] Lity, et al (2022). Travelling salesman problem. An overview of Application. *International Journal of Advanced Research in Education Technology.*

[13] Malik and Muhammad (2015). Heuristic Approaches to solve travelling salesman problem. TELKOMNIKA Indonesian Journal of Electrical Engineering. 15(2). Pp. 390-396

[14] Nemani, et al (2021). Algorithms and Optimization techniques for solving traveling salesman problem

[15] Ono, et al (2020). A criteria – based approach to the travelling salesman problem. Western Decision Science Journal. Librarian Publications and Presentations. 143.

[16] Rahman, et al (2024). Improvement of the Nearest Neighbor Heuristic Search Algorithm for Travelling Salesman Problem. *Journal of Engineering Advancements*, *5*(1), 19–26.

[17] Rao, et al (2024). Literature survey on travelling salesman problem using genetic algorithms/ *International Journal of Advanced Research in Education Technology.* 26, pp. 503 – 530.

[18] Sathya and Muthukumaravel (2015). A review of the optimization algorithms on travelling salesman problem. *Indian journal of science and technology. 8(29)*

[19] Sharma and Dutta (2015). Review of Algorithms to solve travelling salesman problem. *Journal of Basic and Applied Engineering Research. 2(18), pp. 1612-1616.*

[20] Sundar and Rathinam (2016). Generalized multiple depots traveling salesman problem. Computers and Operation Research. 70

[21] Yuan, et al (2020). A branch and Cut algorithms for the generalized travelling salesman problem. *European Journal of Operational Research. 286(3)*