A Dynamic-start, Dynamic-end Insertion Heuristic for Solving the Traveling Salesperson Problem

Poorab Gangwani[1], Uzair Lodhi[1,2], Muhammad Owais[1], Afifa Farooq[1], Qazi Farrukh[1], Khalid Rasheed[3], and Syed Samar Yazdani[4*]

[1]Department of Computer Science, Shaheed Zulfikar Ali Bhutto Institute of Science and Technology, Karachi, Pakistan

[2]National Institute of Oceanography, Karachi, Pakistan

[3]Denning, Karachi, Pakistan

[4]Computer Science Department, SZABIST, Karachi, Pakistan

*__Correspondence__: dr.syedsamar@szabist.edu.pk

Citation | Gangwani. P, Lodhi. U, Owais. M, Farooq. Afifa, Farrukh. Q, Rasheed. K, Yazdani. S, "A Dynamic-start, Dynamic-end Insertion Heuristic for Solving the Traveling Salesperson Problem", IJIST Vol. 08 Issue. 01 pp 307-319, February 2026

Received| December 30, 2025 **Revised|** January 31, 2026 **Accepted|** February 04, 2026 **Published|** February 07, 2026.

The traveling salesperson problem (TSP) is one of the most extensively studied NP-hard combinatorial optimization problems with wide applicability in logistics, routing, manufacturing, and network design, making efficient heuristic solutions crucial for practical applications. This paper introduces the Ideal Link Nearest Insertion (ILNI) heuristic, a dynamic-start, dynamic-end insertion method that addresses the structural limitations of classical Nearest Insertion (NI) through dynamic route reconfiguration and path-based route initialization, maintaining an open path structure that allows endpoint insertions throughout the construction process. The proposed heuristic was evaluated on 1,100 randomly generated symmetric TSP instances ranging from 50 to 150 cities, repeated across ten independent iterations (11,000 total comparisons). Using LKH-3 as a reference solver, ILNI achieved a 4.3% improvement in average approximation ratio (2.3753 vs. 2.4819) and a 15.9% reduction in execution time (0.0127s vs. 0.0151s per instance) compared to classical NI. ILNI produced lower-cost tours in 67.3% of instances, demonstrating consistent superiority across all tested problem sizes. Paired t-tests conducted on all 11,000 paired observations confirmed statistically significant improvements in both solution quality and computational efficiency ($p < 0.001$). The proposed heuristic maintains $O(n^2)$ time complexity while improving practical performance, making ILNI a computationally efficient and structurally robust alternative to classical nearest insertion heuristics.

Keywords: Traveling Salesperson Problem; Insertion Heuristic; Constructive Algorithms; Approximation Ratio; Combinatorial Optimization;

Introduction:

The traveling salesperson problem is one of the most studied NP-hard combinatorial optimization problems because it can be used in many areas, such as logistics, routing, manufacturing, and network design. It is a problem in which a salesperson wants to visit a certain number of cities, starting and ending in their hometown, while taking the least amount of time to travel [1]. To find the shortest travel path, you need to figure out the best way to get through the cities while making sure you only visit each city once before going back to the start point [2][3][4][5]. This finished tour is known as a Hamiltonian cycle. It is assumed that the cost of getting from one city to another is already set. The cost usually has to do with distance, but it could also mean other things, like time or money. As shown in Figure 1, a Hamiltonian cycle refers to a graph cycle that visits each of the graph's vertices exactly once before going back to the starting vertex. The traveling salesperson must go through cities 1 to N in a Hamiltonian cycle, which means they start in city 1, go through the other N-1 cities in a certain order, and then return to city 1, touching each city only once and at the lowest possible cost.

The number of cities in TSP makes it exponentially harder to solve, so it's not possible to find exact solutions for big problems. If there are n cities, there are $(n-1)!/2$ possible tours, which leads to a combinatorial explosion. This rapid growth has led to the creation of many heuristic and approximation algorithms that can find almost the best solutions in a reasonable amount of time. Many people have used construction heuristic techniques to solve traditional combinatorial optimization problems, and one of these methods is the nearest insertion heuristic [6].

TSP has many real-world uses that go beyond just being a theory in computer science. TSP solutions optimize delivery routes in logistics and supply chain management, which lowers fuel use and operational costs. In manufacturing, TSP algorithms help arrange production tasks so that setup times are as short as possible. TSP solutions help with network routing and circuit design in telecommunications. TSP's adaptability as a modeling framework has guaranteed its ongoing significance and spurred continual investigation into effective solution techniques.

TSP is used in many different fields today, such as DNA sequencing, where figuring out the order of DNA fragments can be thought of as a TSP problem. In robotics, TSP solutions help robots plan the best routes to take when they have to go to more than one place. TSP algorithms optimize the drilling sequence for holes in the making of printed circuit boards, which cuts down on production time by a lot. These applications show how efficient TSP solution methods can help solve problems in the real world.

The nearest insertion heuristic is a type of insertion heuristic that is used to build tours. The insertion heuristics begin at a random point to create a sub-tour or partial circuit. Then, nodes that are not already in the sub-tour are added based on a set of rules that make sure the total distance of the sub-tour doesn't go up too much. Given the sub-tour T, and given that V is the next node to be inserted, the insertion method puts city x between cities x_i^* and x_j^* in T_i based on the following:

$$(x_i^*, x_j^*) = \underset{(xi,xj) \in Ti}{arg\ min}\ c(xi, xj, x)$$

where $c(x_i,x_j,x)$ is the extra cost of putting city x between cities x_i and x_j. The nearest insertion heuristic can be used to find good tour construction solutions. It is fast, easy to use, and can handle complicated constraints [6]. But it has some structural problems that make it less effective at solving the traveling salesperson problem (TSP). First, the nearest insertion heuristic is very sensitive to the initial starting configuration. This has been shown in real-world studies and can cause the lead to instability in the final tour [7]. Second, nearest insertion

is susceptible to greedy bias, and its short-sighted decision-making frequently leads the algorithm to prematurely commit to suboptimal tour structures that are irreversible [8].

Insertion-based heuristics for the traveling salesperson problem (TSP) have been extensively examined as effective constructive heuristics owing to their simplicity and minimal computational expense. Recently, [6] published a study that introduced a new constructive insertion heuristic. This heuristic improved upon the selection and insertion strategies of classical insertion heuristics like nearest and farthest insertion by using a customized half-max criterion. The problem of start sensitivity in insertion heuristics has been widely discussed in the literature. [8] showed that the approximation ratio for nearest insertion has a constant upper limit of 2, but they also showed that the algorithm's performance can vary significantly depending on the starting city. Recent efforts have sought to mitigate these limitations through diverse methodologies, but these methods usually require more computing power or don't really solve the problem of cyclic initialization's structural rigidity.

This paper is organized as follows: Section 2 presents the related work and describes the proposed ILNI heuristic in detail. Section 3 discusses the material and methods, including the experimental setup and evaluation methodology. Section 4 presents comprehensive results and discussion from the experimental evaluation. Section 5 provides conclusions, limitations, and future work directions.

Objectives of the Study:

The objectives of this research are:

To design a dynamic-start, dynamic-end insertion heuristic that reduces the structural rigidity inherent in classical nearest insertion methods.

To evaluate the performance of the proposed ILNI heuristic against classical nearest insertion (NI) across diverse TSP instances.

To demonstrate the superiority of ILNI in terms of solution quality and computational efficiency using approximation ratios, execution time metrics, and win rate.

To compare ILNI with additional baseline (e.g., LKH-3, Christo des) where data exist.

Novelty and Research Contributions:

Dynamic Open-path Construction: Introduction of an open-path route structure during tour construction, delaying cycle closure until all cities are inserted.

Endpoint Insertion Mechanism: Extension of insertion evaluation to include route endpoints, enabling bidirectional expansion and structural flexibility.

Reduced Start Sensitivity: Mitigation of dependence on initial city selection through dynamic route reconstruction.

Maintained Computational Complexity: Preservation of $O(n^2)$ time complexity equivalent to classical nearest insertion.

Comprehensive Empirical Validation: Large-scale experimental evaluation across 11,000 paired comparisons demonstrating statistically significant improvements ($p < 0.001$).

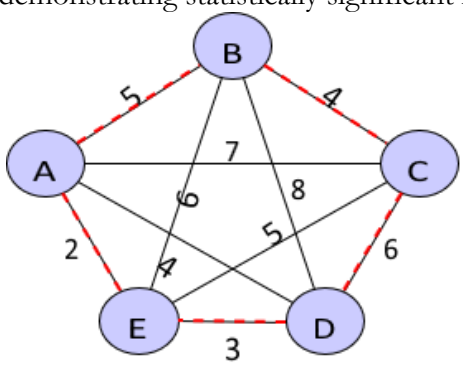

Figure 1. A Hamiltonian weighted graph that shows a TSP instance. The red dashed line A-B-C-D-E-A shows a complete Hamiltonian cycle that visits each vertex only once.

Despite the long-standing study of insertion-based heuristics, a critical structural limitation remains insufficiently addressed: the rigid cyclic initialization of the classical nearest insertion heuristic. Existing enhancements primarily refine city selection or insertion cost criteria, yet they retain early cycle closure behavior, which enforces irreversible structural commitments and amplifies sensitivity to the starting city. The structural rigidity restricts adaptive route evolution during construction and may lead to suboptimal global configurations. To date, limited attention has been given to insertion heuristics that maintain dynamic structural flexibility throughout the tour construction process. Addressing this gap requires rethinking the initialization and growth strategy of constructive heuristics rather than solely modifying insertion criteria.

Related Work:

Insertion-based heuristics for the traveling salesperson problem (TSP) have been extensively examined as effective constructive heuristicss owing to their simplicity and minimal computational expense. Recent studies have aimed to enhance traditional insertion techniques, including nearest and farthest insertion, by refining the criteria employed to select and incorporate cities into a partial tour.

Recently, [6] published a study that introduced a new constructive insertion heuristic. This heuristic improved upon the selection and insertion strategies of classical insertion heuristics like nearest and farthest insertion by using a customized half-max criterion. As a result, it consistently achieved better tour costs than the standard nearest and farthest insertion strategies on benchmark instances. The Half Max Insertion Heuristic (HMIH) showed that tour costs could be cut by as much as 16% compared to traditional methods. This indicates the potential for further improvement in insertion strategies.

The aspect of tour initialization has been investigated as a determinant for enhancing the efficacy of insertion heuristics. The research in [9] advanced the insertion technique of convex hull cheapest insertion and advocated for the initialization of insertion heuristics utilizing a convex hull structure. This approach mitigates suboptimal early insertions and enhances solution quality for non-Euclidean TSP instances through intelligent tour initialization and selective insertion. This method highlights the importance of effective initialization strategies for making heuristics work better.

Several other studies related to TSP heuristics have also stressed the need to look into ways to improve the decision-making process of TSP heuristics by refining insertion strategies. Bentley's [12] seminal work on rapid heuristic methods for geometric TSP instances underscored that optimizing the insertion location of a city can yield significant enhancements in tour quality without escalating asymptotic complexity [10]. Gutin and Punnen's [13] survey of TSP heuristics also stated that better evaluation of insertion positions is a key part of making constructive heuristics better while keeping them fast [11].

The problem of start sensitivity in insertion heuristics has been widely discussed in the literature. [8] showed that the approximation ratio for nearest insertion has a constant upper limit of 2, but they also showed that the algorithm's performance can change a lot depending on the starting city. [7] examined this phenomenon in greater detail, observing that the inflexible cyclic initialization of nearest insertion imposes structural constraints that hinder the algorithm's capacity to rectify suboptimal initial choices.

Our Contribution:

The suggested Ideal Link Nearest Insertion (ILNI) heuristic fixes some of the problems with the classical nearest insertion heuristic. The classical nearest insertion heuristic uses a cyclical method to start the tour. This means picking a random city (or an arbitrary city) and finding its closest neighboring city. From the start, these two cities make up the first subtour, which is treated as a closed cycle. Once this first subtour is set up, the starting city of the structure is set in stone. This makes it impossible to change the starting point to optimize

the tour. Also, this setting lets the start city configuration have a big effect on the final tour outcome. This is why the nearest insertion heuristic is very sensitive to the choice of starting city, which can make its performance very unstable.

The Ideal Link Nearest Insertion (ILNI) heuristic changes the tour initialization and city insertion strategies of the nearest insertion heuristic, but it keeps the basic city selection strategy of the nearest insertion heuristic. The Ideal Link Nearest Insertion (ILNI) heuristic begins the tour with just one starting city instead of two, which means that the tour starts as an open path instead of a closed cycle. The Ideal Link Nearest Insertion (ILNI) heuristic treats both the starting and ending positions of the current route as valid insertion candidates, along with all internal edges, as cities are added one at a time. This makes it possible to change the starting city as new cities are added, which makes it easier to optimize the tour. The Ideal Link Nearest Insertion (ILNI) heuristic avoids the structural rigidity that comes with an early two-city subtour by waiting until all cities have been added to close the cycle. It also has a dynamic start-city configuration that makes it less sensitive to the choice of starting city.

The source code of the Ideal Link Nearest Insertion (ILNI) heuristic is available at https: //github.com/uzairlodhi/Ideal-Link-Nearest-Insertion-TSP-Algorithm

Heuristic Description:

Notation and Definitions:

Let $G = (V, E)$ be a complete weighted graph, where $V = \{v_1,v_2,...,v_n\}$ denotes the set of cities and $d(i,j)$ denotes the cost of traveling between cities i and j. At any iteration, ILNI keeps:

A partial route $R = (u_1,u_2,...,u_k)$, where $k \leq n$, is a set of cities that haven't been visited U = V \ R yet.

The route R is kept as an open path, which means that it has different start and end points.

This is different from the closed cycle that classical nearest insertion keeps. ILNI's ability to change the route on the y depends on this open path structure.

Nearest City Selection:

At each iteration, ILNI picks the next city $v^* \in U$ that is closest to the current route R, defined as:

$$v* = \arg \min_{v \in U} \min_{u \in R} d(u, v)$$

This step is the same as the city selection strategy used in the classical Nearest Insertion heuristic, which makes sure that ILNI uses the same greedy selection criterion to decide which city to insert next. The selection process considers all cities currently in the route and finds the one that has not been visited yet and is closest to any of the cities on the route.

Determination of the Ideal Link Insertion Position:

After finding the nearest city, v^*, ILNI finds the best place to insert the link that will lower the cost of the tour as much as possible. NI only looks at inserting between adjacent nodes of a closed subtour, but ILNI looks at all possible positions, such as: inserting at the start of the route, putting it between any two nodes that are next to each other, and putting it at the end of the route.

Let $R = (u_1,u_2,...,u_k)$ be the current part of the route. The incremental cost Δ for putting v^* at position i is:

$$\Delta i = \begin{cases} d(v*, u1), \text{if inserted at the beginning } (i = 0), \\ d(ui, v*) + d(v*, ui + 1) \\ d(ui, ui + 1), \text{ if inserted between nodes } (1 \leq i < k), \\ d(uk, v*), \text{if inserted at the end } (i = k). \end{cases} \quad (1)$$

The city v^* is added at the position $i^* = \arg\min_i \Delta_i$, which yields the minimum incremental cost. ILNI is different from classical nearest insertion because it can dynamically

evaluate all possible insertion positions, including endpoints. This allows the route to be dynamically modified. The ability to insert at endpoints is very important because it lets the route grow in either direction, which is not possible with cyclic structures.

Route Expansion and Termination:

After insertion, v^* is taken out of the unvisited set U, and the route R is changed to reflect this. This process continues until all the cities have been added. Finally, the tour ends by connecting the last city to the first city, which makes the Hamiltonian cycle complete. The total cost of the tour is computed as:

$$C_{final} = \sum_{i=1}^{n-1} d(u_i, u_{i+1}) + d(u_n, u_1)$$

where $u_1, u_2, \ldots, u_n$ is the last route sequence. The closure step is only done after all the cities have been added, which keeps the route structure flexible during the building process.

Algorithm 1 Ideal Link Nearest Insertion (ILNI)

Require: Complete weighted graph $G = (V, E)$ with distance matrix d, starting city s

Ensure: Hamiltonian cycle R with total cost C

Initialize route $R \leftarrow [s]$

Initialize unvisited cities $U \leftarrow V \setminus \{s\}$

while $U \neq \emptyset$ do

  Find nearest city: $v* \leftarrow \arg\ min_{v \in U}\ min_{u \in R}\ d(u, v)$

  Initialize $\Delta_{min} \leftarrow \infty$, $pos_{best} \leftarrow -1$

  for each position i in route R do

  if $i = 0$ then

  $\Delta i \leftarrow d(v*, R[0])$

  else if $i = |R|$ then

  $\Delta i \leftarrow d\ (R\ [|R|\ -\ 1], v*)$

  else

  $\Delta i \leftarrow d\ (R\ [i\ -\ 1], v*)\ +\ d(v*, R[i])\ -\ d\ (R\ [i\ -\ 1], R[i])$

  end if

  if $\Delta_i < \Delta_{min}$ then

  $\Delta_{min} \leftarrow \Delta_i$

  $pos_{best} \leftarrow i$

  end if

end for

  Insert $v*$ into R at position pos_{best}

  $U \leftarrow U \setminus \{v*\}$

end while

Close cycle: $C \leftarrow \sum_{i=0}^{|R|-2} d(R[i], R[i+1]) + d(R[|R|-1], R[0])$

return R, C

Flow Diagram of ILNI:

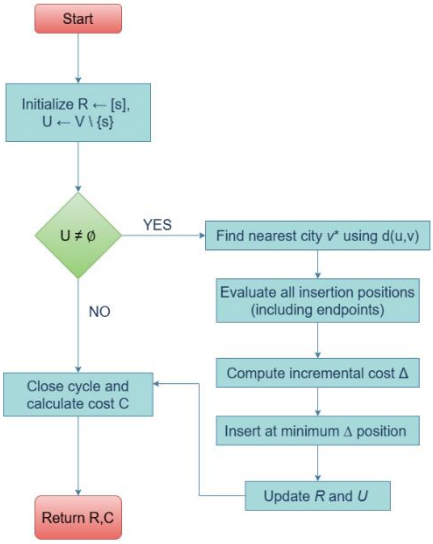

Figure 2. Algorithmic Workflow Diagram of ILN

Materials and Methods:

Problem Definition:

Let $G = (V, E)$ be a complete weighted graph where $V = v_1, v_2, v_3, \ldots, v_n$ represents the set of cities and $d(i,j)$ denotes the symmetric distance (cost) between cities i and j. The Traveling Salesperson Problem (TSP) seeks to determine a permutation π of the vertices such that the total tour cost is minimized:

$$\min \sum_{i=1}^{n-1} d(\pi_i, \pi_{i+1}) + d(\pi_n, \pi_1)$$

The objective is to construct a Hamiltonian cycle that visits each city exactly once and returns to the starting city while minimizing total travel cost.

In this study, the focus of the problem is not on exact optimization, but on improving constructive heuristic performance by reducing structural rigidity and start-city sensitivity inherent in classical insertion-based approaches.

Study Location:

This research was conducted at the Department of Computer Science, Shaheed Zulqarnain Bhutto Institute of Science and Technology (SZABIST), Karachi, Pakistan. The experimental setup and computational analysis were performed using the Google Colab cloud computing platform, with all experiments executed remotely. The study location coordinates are approximately 24.8607 N, 67.0011 E (Karachi, Sindh, Pakistan).

Investigation Site:

The experimental evaluation was conducted using Google Colab as the computational platform. All experiments were executed on an HP EliteBook 850 G1 laptop, equipped with an Intel Core i7-4600U CPU at 2.10 GHz and 8 GB of RAM, operating Python 3 on Google Colab. This environment provided the necessary computational resources for generating and evaluating TSP instances across multiple iterations.

Material and Methods:

The study's overall methodology consists of three main steps: baseline heuristic selection, experimental setup, and heuristic evaluation. The baseline heuristic selection is a critical component of the research method used in the study. The goal of this step was to choose a construction-based insertion heuristic that would be an appropriate method to evaluate the ideal link nearest insertion heuristic. The ideal link nearest insertion heuristic as shown in Figure 2 is an improved version of the nearest insertion heuristic. To evaluate the

performance improvement the ideal link nearest insertion heuristic is than the nearest insertion heuristic, we used the nearest insertion heuristic as the baseline heuristic. This choice makes sure that both heuristics are compared fairly because they both use the same city selection strategy. This enables evaluation of how the dynamic route configuration mechanism affects things.

At this point, the experimental setting and environment for the heuristic evaluation were put together. Google Colab was the platform where the experiment was set up, and Python 3 was the programming language used to do it. At first, the nearest insertion and ideal link heuristics were programmed so that they could be tested later.

To compare the baseline heuristic with our proposed method, a dataset was generated of 1100 randomly generated, symmetric TSP graphs. Each instance contained between 50 and 150 cities in each graph instance, with 10 cities added at a time. Specifically, the dataset consisted of 100 instances each for city counts ranging from 50 to 150 cities in increments of 10. This distribution allows analysis of how performance changes as problems get more complicated. We made the distance matrices by picking random integers between 1 and 50, ensuring that the cost matrices were symmetric (i.e., $d(i,j) = d(j, i)$ for all city pairs). The random generation process used a uniform distribution to make sure that the test instances were different from each other and not biased toward certain graph structures.

The LKH-3 heuristic, which is a cutting-edge TSP solver, was added to the environment to compute average approximation ratios for each heuristic, both per instance and across the entire dataset. For very accurate execution time measurements, we used the perf_counter function from Python's time module. Figure 3 shows the structure of the experimental framework, and you can find the full source code at https: //github.com/uzairlodhi/Ideal-Link-Nearest-Insertion-TSP-Algorithm.

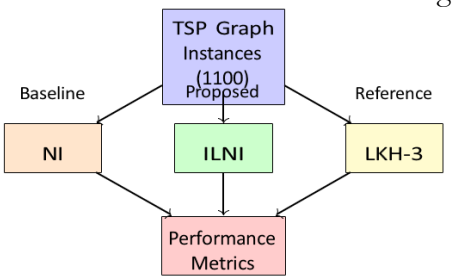

Figure 3. The Layout of the Experimental Setting. Three algorithms handle the TSP graph instances: NI (baseline), ILNI (proposed), and LKH-3 (reference solver). We gather and compare performance metrics.

The nearest insertion and ideal link nearest insertion heuristics were initially tested on 100 randomly generated TSP graphs, each containing 50 cities. Likewise, both heuristics were subsequently utilized on 100 instances of randomly generated TSP graphs, each comprising 60 cities, and this procedure continued until both heuristics were ultimately assessed on 100 instances of randomly generated TSP graphs, each containing 150 cities. For each instance, both heuristics were applied on a TSP graph, their respective approximation ratios were computed. Finally, the average approximation ratio of each heuristic on all 1100 TSP graph instances was measured and compared. The average execution time of each heuristic was also measured and compared in the same way. The number of instances in which the ideal link nearest insertion heuristic found a route that cost less than the one found by the nearest insertion heuristic on the same TSP graph. A statistical record was maintained regarding the frequency with which the ideal link nearest insertion heuristic provided a more expensive route compared to the nearest insertion heuristic on the same graph. The entire detailed process described above was manually repeated ten times for the comparative evaluation of both heuristics.

This study used two main performance metrics: execution time and approximation ratio. We used Python's perf_counter function to measure execution time. This function provides high-resolution timing suitable for microsecond-level measurements. The approximation ratio measures the quality of a solution by comparing it to a reference solution. We used LKH-3 as the reference solver because it is impossible to find exact solutions for large instances. The metrics are defined as follows:

$$\text{Execution Time} = T_{\text{end}} - T_{\text{start}}$$

$$\text{Heuristic Cost Approximation Ratio} = \frac{\text{Heuristic Cost}}{\text{LKH-3 Cost}}$$

Where T_{start} = perf_counter before the code block in which the heuristic starts and ends execution on a TSP graph, T_{end} = perf_counter after the code block in which the heuristic starts and ends running on a TSP graph, Heuristic Cost = The cost of the last route that the heuristic being tested found for a TSP graph T, and LKH-3 Cost = The LKH-3 heuristic's cost for the TSP Graph T's final route.

Results:

The proposed Ideal Link Nearest Insertion (ILNI)heuristic was compared against the classical Nearest Insertion (NI) heuristic across 1,100 TSP instances repeated over ten independent iterations (11,000 total paired evaluations). Based on their performance in the experimental setting outlined in the heuristic evaluation section, the results have been compiled and presented in Table 1.

Table 1. Performance Comparison of ILNI and NI Across Ten Iterations

Iter.	Avg. Exec. Time (s)		Avg. Approx. Ratio		Wins (Lower Cost)	
	ILNI	NI	ILNI	NI	ILNI	NI
1	0.0128	0.0153	2.3752	2.4792	747	345
2	0.0126	0.015	2.3672	2.4833	759	335
3	0.0126	0.015	2.3723	2.4701	715	376
4	0.0127	0.0152	2.3781	2.4845	754	335
5	0.0126	0.0149	2.375	2.479	733	356
6	0.0127	0.015	2.3791	2.4849	750	335
7	0.0127	0.015	2.3787	2.4868	734	356
8	0.0128	0.0152	2.3755	2.4831	764	323
9	0.0125	0.015	2.3772	2.4826	744	347
10	0.0126	0.015	2.3741	2.4858	746	345
Mean	0.0127	0.0151	2.3753	2.4819	740.2	345.3
Std Dev	0.0001	0.0001	0.0038	0.0053	16.2	15.8

The results show that the ideal link nearest insertion heuristic produced shorter routes in nine out of ten iterations and on more than twice the number of graph instances than the nearest insertion heuristic. Across all iterations, ILNI achieved a mean approximation ratio of 2.3753 compared to 2.4819 for NI, representing a 4.3% improvement in solution quality. ILNI reduced average execution time from 0.0151 seconds (NI) to 0.0127 seconds per instance, corresponding to a 15.9% computational speed improvement. ILNI produced lower-cost tours in 740.2 instances per iteration on average (67.3%), whereas NI achieved Table 1: Performance Comparison of ILNI and NI Across Ten Iterations superiority in only 345.3 instances (31.4%). This corresponds to a 2.14-fold improvement in solution quality. The ideal link nearest insertion heuristic is consistent in its performance, demonstrating consistent performance in both average execution time and approximation ratio compared to NI.

Statistical Validation:

To statistically validate performance differences, paired t-tests were conducted on 11,000 paired approximation ratios and execution time observations.

The null hypotheses tested were:

H_{01}: There is no significant difference in approximation ratios between ILNI and NI.
H_{02}: There is no significant difference in execution times between ILNI and NI.

Results rejected both null hypotheses at the 95% confidence level ($p < 0.001$), confirming that ILNI's improvements are statistically significant.

The low standard deviation (0.0038 for ILNI approximation ratio and 0.0001 for execution time) further demonstrates performance stability across independent experimental runs.

The paired design ensures that each ILNI result is directly compared against NI under identical problem conditions, strengthening internal validity. Extremely low p-values (<0.001) indicate a less than 0.1% probability that these improvements occurred by chance, providing strong empirical evidence for algorithm superiority.

Performance Across Problem Sizes:

ILNI consistently outperformed NI across all tested ranges (50-150 cities), maintaining an average win rate of approximately 67% regardless of problem size.
Table 2 presents performance trends across varying city counts.

Table 2. Performance Comparison Across Different Problem Sizes

City Range	Avg. Exec. Time (s)		Avg. Approx. Ratio		Win Rate (%)	
	ILNI	NI	ILNI	NI	ILNI	NI
50-70	0.0085	0.0102	2.3124	2.4231	68.2	31.8
80-100	0.0112	0.0135	2.3567	2.4678	67.5	32.5
110-130	0.0148	0.0176	2.3892	2.4987	66.8	33.2
140-150	0.0185	0.0219	2.4123	2.5214	66.1	33.9
Overall	0.0127	0.0151	2.3753	2.4819	67.3	32.7

The analysis by problem size shows that ILNI keeps its performance edge in all tested ranges. As the size of the problem grows, both heuristics show longer execution times and higher approximation ratios. However, ILNI always has lower values. The performance advantage of ILNI persists across all tested problem sizes. This suggests that the dynamic route reconstruction mechanism gives benefits that grow with the complexity of the problem.

The experimental results and theoretical analysis in this study furnish substantial evidence for the superiority of ILNI compared to classical nearest insertion. ILNI can keep structural flexibility throughout the construction process. Classical nearest insertion fixes a cyclic structure early, imposing constraints on subsequent tour construction. The open path structure of ILNI lets the algorithm change as the problem structure changes, which helps it make better global decisions. Consider a scenario in which an early insertion creates a suboptimal route segment. This segment in NI is fixed due to early cycle closure and cannot be easily modified. ILNI can change the route, though, by adding new cities at the ends. This lets the route grow in different directions and allowing the heuristic to bypass suboptimal segments. This adaptive behavior is especially useful when working with distance matrices that aren't evenly spaced out or cities that aren't evenly spaced out.

Discussion:

Even though endpoint insertions give ILNI more flexibility, it still has the same asymptotic complexity as NI. The main point is that it only takes a constant amount of time to evaluate endpoint insertions for each position, and the number of positions to evaluate grows linearly with the length of the route. So, the total overhead is $O(n)$ per iteration, which doesn't change the overall $O(n^2)$ complexity. The observed decrease in execution time indicates that ILNI's implementation may be more efficient in practice, potentially due to improved cache locality or diminished overhead from managing a simpler data structure.

Implication of the Study:

Theoretical Implications:

This study demonstrates that structural flexibility in constructive heuristics can yield measurable improvements without increasing asymptotic computational complexity. The results suggest that initialization strategy plays a critical role in heuristic performance, and that delaying cycle closure enhances adaptive route evolution.

Practical Implications:

The ILNI heuristic provides a computationally efficient alternative for real-world routing and logistics applications where rapid solution generation is required. Its improved solution quality and reduced execution time make it suitable for medium-scale routing tasks, embedded optimization modules, and resource-constrained environments.

Limitations and Future Work:

While ILNI demonstrated consistent superiority within the evaluated range (50-150+ cities), instances remain to be examined.

Future research directions include:

Evaluation using standardized benchmark datasets such as TSPLIB to enable comparison with state-of-the-art heuristics.

Testing multiple randomized starting cities to further assess robustness against initialization bias.

Extending ILNI within a hybrid metaheuristic framework, such as genetic algorithms or local search refinement methods.

Adapting the heuristic to asymmetric and real-world distance matrices.

Conclusion:

In this study, we introduced the ideal link nearest insertion heuristic, a dynamic-start, dynamic-end variant of the classical nearest insertion heuristic. We conducted a comprehensive performance evaluation, measuring and comparing the efficacy of our proposed heuristic against the nearest insertion heuristic.

The nearest insertion heuristic is very sensitive to how the start city is set up, and it can't handle bad early commitments to sub-optimal structure. This can make it less effective in terms of solution quality and practical efficiency on the traveling salesperson problem. The ideal link nearest insertion heuristic, which has the features of dynamic route reconfiguration and path-based route initialization, gets around these structural problems and mostly gives better answers to the traveling salesperson problem. The experimental evaluation in the study showed that the ideal link nearest insertion heuristic was better than the nearest insertion heuristic in all ten independent iterations of the experiment. It had a better average approximation ratio and average execution time.

The thorough evaluation showed that ILNI had many important benefits, including: (1) Better Solution Quality: ILNI won 67.3% of the time, while NI only won 31.4% of the time, which is a 2.14-fold improvement; (2) Better Approximation Ratios: ILNI had an average approximation ratio of 2.3753, which was 4.3% better than NI's 2.4819; (3) Improved Efficiency: ILNI ran 15.9% faster on average, showing that better solutions can be found with less computational overhead; (4) Consistent Performance: The low standard deviations across iterations show that ILNI's performance is stable and reliable.

The success of ILNI supports the hypothesis that dynamic route reconfiguration can alleviate the constraints of rigid cyclic initialization. ILNI offers a more adaptable framework for tour construction by keeping an open path structure and allowing endpoint insertions. This demonstrates that maintaining structural flexibility during tour construction improves heuristic performance without increasing computational complexity. The fact that ILNI makes these improvements while keeping the same $O(n^2)$ time complexity as classical nearest insertion shows how efficient the method is.

The better performance of ILNI makes it a great choice for situations where you need rapid, high-quality solutions for TSP instances. ILNI is a good choice for practical routing and logistics applications because it offers better solution quality and shorter execution time than classical nearest insertion. The fact that it performs consistently across varying problem sizes, enhancing its practical utility. Potential applications include vehicle routing, delivery optimization, circuit board design, and DNA sequencing, where rapid heuristic solutions are preferable to computationally expensive exact methods.

One limitation of the study is the specificity of the evaluation setting; the performance assessment of the heuristic was conducted on TSP graph instances with some cities ranging from 50 to 150. Consequently, the generalizability of the results to instances with fewer than 50 or more than 150 cities is limited. To mitigate this limitation, subsequent research should entail evaluating the heuristic on TSP graph instances encompassing a broader spectrum of cities, specifically both smaller instances (10-50 cities) and larger instances (150-500 cities), to ascertain comprehensive performance characteristics. An additional limitation is the use of a uniform starting city across all instances, which may limit assessment of ILNI's robustness to initial conditions. Future research should concentrate on replicating the experiment from the study with varied initial city configurations, possibly utilizing multiple random starts to further substantiate the robustness of ILNI. Another limitation is that the instances were randomly generated instead of using established TSP benchmark instances like those in TSPLIB. Future research should assess ILNI using standard benchmark instances to facilitate direct comparison with other cutting-edge
heuristics.

Acknowledgment:

The authors acknowledge that this manuscript has not been previously published, nor is it currently under consideration for publication in any other journal. All authors have contributed significantly to this work and agree with the content of the manuscript. The authors would like to thank the Department of Computer Science at the Shaheed Zulqarnain Bhutto Institute of Science and Technology for giving them the computer resources and help they needed to do this research. We also thank the open-source community for giving us tools and libraries that made this work easier, especially the LKH-3 solver and the Python scientific computing ecosystem.

Author's Contribution:

Poorab Gangwani, Uzair Lodhi, Muhammad Owais, and AfiFa Farooq contributed to the algorithm development, experimental design, data collection, and analysis. Qazi Farrukh contributed to the experimental setup and data validation. Syed Samar Yazdani (corresponding author) provided overall supervision, research guidance, methodology refinement, manuscript review, and final approval of the work.

Conflict of Interest:

The authors declare that there is no conflict of interest in publishing this manuscript.

Project Details:

This research was conducted as an independent academic study. No external project funding or project number is associated with this work.

References:

[1] Traveling salesman problem | Research Starters | EBSCO Research." [Online]. Avail-able: https://www.ebsco.com.

[2] E. O. Asani, A. E. Okeyinka, and A. A. Adebiyi, "A construction tour technique for solving the travelling salesman problem based on convex hull and nearest neigh-bour heuristics," in the 2020 International Conference in Mathematics, Computer Engineering, and Computer Science (ICM-CECS), 2020, pp. 1_4. [3] H. A. Me and Abdulkarim. F. Alshammari, "Comparison of algorithms for solving the traveling

salesman problem," International Journal of Engineering and Advanced Technology, vol. 4, no. 4, pp. 76_78, 2015.

[4] Z. Xiao, H. Fang, H. Jiang, J. Bai, V. Havyarimana, H. Chen, and L. Jiao, "Understanding the private car aggregation e_ect through spatio-temporal analysis of trajectory data," IEEE Transactions on Cybernetics, vol. 53, no. 4, pp. 2346_2357, 2023.

[5] Z. Xiao, H. Li, H. Y. Jiang Li, M. Alazab, Y. Zhu and S. Dustdar, "Predicting urban region heat via learning arrive-stay-leave behaviors of private cars," IEEE Transactions on Intelligent Transportation Systems, vol. 24, no. 10, pp. 10843_10856, 2023.

[6] E. O. Asani, A. E. Okeyinka, S. A. Ajagbe, A. A. Adebiyi, R. T. Ogundokun, O. S. P. Mudali, M. Adekunle, and O. Adigun, "A Novel Insertion Solution for the Travelling Salesman Problem," Computers, Materials & Continua, vol. 79, no. 1, pp. 1581_1597, 2024. [Online]. Available: https://www.techscience.com/cmc/v79n1/56285

[7] W. Huang and J. X. Yu, "Investigating TSP Heuristics for Location-Based Services," Data Science and Engineering, vol. 2, no. 1, pp. 71_93, Mar. 2017. [Online]. Available: https://doi.org/10.1007/s41019-016-0030-0.

[8] Z. A. Ali, "Concentric Tabu Search Algorithm for Solving Traveling Salesman Problem (TSP)."

[9] G. Reinelt, The Traveling Salesman: Computational Solutions for TSP Applications, ser. Computer Science Lecture Notes. Springer, 1994, vol. Berlin, Heidelberg 840.

[10] D. J. Rosenkrantz, R. E. Stearns and P. M. Lewis, II, "An Analysis of Several Heuristics for the Traveling Salesman Problem," SIAM Journal on Computing, vol. 6, no. 3, pp. 563_581, 1977. [Online]. Available: https://doi.org/10.1137/0206041.

[11] M. M. Goutham Menon, S. Garrow and S. Stockar, "A Convex Hull Cheapest Insertion Heuristic for the Non-Euclidean TSP," 2024. [Online]. Available: https://arxiv.org/abs/2302.06582.

[12] J. Bentley, "Fast algorithms for geometric traveling salesman problems," INFORMS J. Comput., vol. 4, pp. 387_411, 1992. [Online]. Available: https://api.semanticscholar.org/CorpusID:207225550

[13] G. Gutin and A. P. Punnen, The Traveling Salesman Problem and Its Variations. Springer, 2007, in Boston, MA.