TOUR LENGTH ESTIMATION GUIDED VEHICLE ROUTING

Stephen Feldman

Submitted in Partial Fulfillment of the Requirements
for the Degree of

MASTER OF SCIENCE

Approved by:
Dr. James Bailey, Chair
Dr. Bahar Çavdar, Co-Chair

Department of Industrial Engineering
Rensselaer Polytechnic Institute
Troy, New York

November 2025
(For Graduation December 2025)

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

I would first like to express my deepest gratitude to my committee chair and advisor, Dr. James Bailey. I extend my sincere appreciation to my co-chair, Dr. Bahar Çavdar, who also served as an invaluable advisor. Their guidance and support were instrumental in the development of this research. I am particularly grateful for their insightful contributions, which enabled this work to come together successfully under a very short timeline. Their encouragement to explore this topic area profoundly shaped the direction of this thesis, leading directly to the contributions presented herein.

I would also like to thank the other members of my committee, Dr. Jennifer Pazour and Dr. John E. Mitchell, for their time and valuable feedback.

I am especially thankful to the entire committee for their quick responses and flexibility, which were essential in allowing this thesis to be completed under a significant time constraint. Their collective support has been invaluable.

ABSTRACT

The Vehicle Routing Problem (VRP) is a fundamental challenge in logistics, with most state-of-the-art heuristics relying on operators that directly manipulate explicit route sequences. This work proposes a heuristic method based on the 2-Stage Assignment-Routing formulation, which reframes the VRP as the problem of first partitioning customers among vehicles and then solving for the routes. The primary challenge of this formulation is the intractability of evaluating an assignment's cost, which requires solving a Traveling Salesman Problem (TSP). We address this challenge by replacing the exact TSP cost with a fast tour length estimator, creating a tractable objective function to guide a metaheuristic search.

We evaluate a variety of existing estimators and introduce a novel Geometrically Assisted Regression Tree (GART), which employs machine learning to predict the TSP-to-MST cost ratio using a rich set of spatial and topological features. A comprehensive benchmark demonstrates that GART is significantly more precise than existing estimator models across a wide range of TSP instances. When integrated into a powerful VRP metaheuristic, the estimator-driven search improves upon initial solutions generated by constructive heuristics. However, a local optimizer of the estimator is not always consistent with the true VRP objective function.

Our numerical experiments demonstrate that high-quality VRP solutions can be obtained by guiding the second stage with TSP tour length estimations rather than solving the full TSP directly. The effectiveness of this approach, however, depends critically on the precision of the estimation method: while reliable estimators enable strong performance, medium- to large-scale errors can cause the search to diverge from the true objective.

1. Introduction

The Vehicle Routing Problem (VRP) is a cornerstone of combinatorial optimization, representing a generalized fusion of the Traveling Salesman Problem and the Bin Packing Problem. At its core, the VRP seeks to determine the optimal set of routes for a fleet of vehicles to serve a geographically dispersed set of customers, subject to constraints on vehicle capacity and customer demand, while minimizing total operational cost. As an NP-hard problem (Lenstra and Kan 1981), the VRP has been the subject of sustained academic research for over six decades, not only for its theoretical richness but also for its profound practical importance in logistics and supply chain management. The overarching goal in addressing the VRP is to develop methods that achieve strong solution quality while remaining computationally efficient, particularly for large and complex instances. This work explores a new heuristic framework for the VRP that decomposes the problem into two stages—assignment of nodes to vehicles followed by routing—and investigates the efficacy of using fast, precise tour length estimators for the Traveling Salesman Problem (TSP) to guide the search for high-quality VRP solutions.

1.1 Motivation

The Vehicle Routing Problem (VRP) remains one of the most widely studied problems in combinatorial optimization due to its central role in logistics and distribution systems. With logistics accounting for up to 20% of a product's final cost, routing optimizations routinely yield savings of 5–20%, making the VRP a focal point for achieving economic efficiency (Moghdani et al. 2021). At the same time, the VRP is computationally challenging: exact algorithms quickly become intractable for realistic instance sizes, necessitating the use of heuristics and metaheuristics to obtain high-quality solutions within practical time limits. Academic inquiry into the VRP is therefore motivated by two factors: its immense economic relevance, reflected in the potential for substantial cost savings, and its computational intractability, which continues to drive the development of efficient heuristics and metaheuristics.

In this work, we focus on a VRP with the following characteristics: a single depot, a homogeneous fleet of vehicles with fixed capacity, and customers each requiring a unit demand. The objective is to minimize the total distance traveled while ensuring that every customer is served exactly once and that no vehicle exceeds its capacity. By restricting attention to unitary demand, we avoid conflating routing with bin-packing subproblems, thereby isolating the routing challenge itself.

1.2 Our Contributions

Our approach is predicated on an Assignment–Routing formulation of the VRP, which decomposes the problem into two stages: first assigning customers to vehicles, and then finding the optimal tour for each vehicle's assignment. The primary obstacle to this strategy is that evaluating the true cost of any assignment requires solving a Traveling Salesman Problem (TSP), which is itself NP-hard (Lenstra and Kan 1981).

We propose to circumvent this bottleneck by replacing the exact TSP cost with a computationally inexpensive tour length estimator that bypasses the need to construct explicit routes. Rather than solving the TSP for each assignment decision, we use a fast approximation method to estimate the optimal TSP tour length, thereby creating a new, tractable objective function to guide the search.

The fundamental research question motivating this work is whether a solution partition found by optimizing this estimated objective function can serve as a high-quality proxy for an optimal VRP solution. To investigate this hypothesis, we first develop a new TSP tour length estimation model. We then integrate this estimator into the Assignment–Routing formulation of the VRP, where a neighborhood search method determines the assignment of customers to routes in the first stage, and the cost of each route is evaluated using the estimator in place of an exact TSP solution.

The primary contributions of this study are as follows:

1. We develop a regression tree-based tour length estimator for the TSP, specifically tailored to handle the diverse geometric properties of customer assignments found in VRP solutions. This model is systematically benchmarked against existing tour length estimators in the literature and offers significant improvements in precision.

2. We propose a new two-stage solution method for the VRP, where the first stage assigns nodes to routes and the second stage evaluates these assignments using our TSP tour length estimation model rather than explicitly solving the routing subproblems. Our

results validate that TSP tour length estimations can successfully guide node assignment decisions in the VRP.

1.3 Outline

The remainder of the thesis is organized as follows. Chapter 2 reviews the relevant literature on VRP heuristics and TSP tour length estimation, identifying the research gap. Chapter 3 formally introduces an arc-flow mathematical formulation for the VRP, the Assignment–Routing formulation, and establishes their equivalence. Chapter 4 introduces our new regression tree-based estimator for TSP estimation and benchmarks it against various models in the literature. Chapter 5 presents our computational study, structured around two experiments that evaluate the feasibility of searching along the estimated VRP objective function. Finally, Chapter 6 concludes the thesis with a summary of findings and a discussion of avenues for future research.

2. Literature Review

The research presented in this thesis builds upon two distinct but complementary bodies of work: the extensive literature on heuristics for the Vehicle Routing Problem, and the more specialized field of Tour Length Estimation for the Traveling Salesman Problem. This chapter surveys the key concepts from both domains to establish the context and motivation for the proposed estimator-driven, assignment-based heuristic framework.

2.1 Heuristic Approaches for the Vehicle Routing Problem

The NP-hard nature of the VRP makes finding provably optimal solutions computationally intractable for the large-scale instances encountered in practice (Lenstra and Kan 1981). Consequently, the field has been dominated by the development of heuristics—methods that seek high-quality, near-optimal solutions within a reasonable time frame. VRP heuristics can be broadly categorized into constructive methods, which build a solution from scratch, and improvement methods, which iteratively refine an existing solution.

2.1.1 Constructive Heuristics

Constructive heuristics are typically fast, single-pass algorithms used to generate an initial feasible solution. One of the earliest and most influential is the Clarke–Wright Savings algorithm, which begins with each customer being served by a dedicated vehicle and progressively merges routes if doing so produces a cost savings (Clarke and Wright 1964). While effective, the Clarke–Wright heuristic does not permit specifying the number of vehicles in advance. Another major class is insertion heuristics, which begin with empty routes and incrementally add unassigned customers. The selection rule for which customer to insert next is the defining feature; for instance, the Regret-k heuristic selects the customer that would suffer the greatest cost penalty if it were not assigned to its best possible route, thereby providing a measure of look-ahead (Potvin and Rousseau 1993). While constructive heuristics are effective for generating good starting points, their primary role is to provide high-quality initial solutions for more advanced improvement heuristics.

2.1.2 Improvement Heuristics and Metaheuristics

Improvement heuristics form the core of nearly all high-performance VRP solvers. They begin with a feasible solution and iteratively apply moves to transition to better solutions in a local neighborhood until no further improvements can be found. Their power derives from the direct manipulation of explicit route structures—that is, ordered sequences of customers.

A traditional VRP solution is represented as a set of routes, where each route is a permutation of the customers it serves. This representation enables a rich set of well-studied neighborhood operators that make small, well-defined changes to the solution structure. These operators can be classified as either intra-route (modifying a single route) or inter-route (modifying multiple routes).

Intra-route operators are primarily adapted from the TSP literature. The most common is the *k-opt* family of moves, such as 2-opt or 3-opt, which modify a route by deleting k edges and reconnecting the resulting segments in a new, potentially better way (Lin 1965). Other moves, such as Or-opt, relocate a contiguous chain of customers within the same route (Ropke and Pisinger 2006). These operators are essential for improving the efficiency of individual vehicle tours.

Inter-route operators are the principal mechanisms for reallocating workload between vehicles. At the simplest level, single-customer transfers and pairwise exchanges between routes effect targeted, low-cost reassignment of service points and are widely used because their cost deltas can be computed very efficiently. These operators are effective at correcting local imbalances in load or eliminating short detours but are limited in their ability to escape deeper local optima (Toth and Vigo 2014). To achieve broader, nonlocal restructuring, research has developed a hierarchy of more powerful inter-route operators that act on contiguous segments or chains of customers. Segment-exchange operators swap subsequences between routes, and the CROSS-exchange family in particular swaps route tail segments to rewire two routes simultaneously, producing large improvements per move at the expense of a much larger neighborhood that must be pruned by candidate selection or heuristic guidance (Ropke and Pisinger 2006; Shaw 1998). Ejection-chain and chain-relocation operators generalize sequential relocations by propagating a vacancy through several routes: a relocated customer creates a hole that is filled by another relocation, and so on, allowing complex redistributions across many vehicles while maintaining incremental cost updateability (Glover 1996; Ropke and Pisinger 2006). Cyclic exchanges extend pairwise swaps to cycles involving

k routes, moving one or more customers (or subsequences) around the cycle to rebalance loads and resolve intertwined routing conflicts that pairwise moves cannot address (Toth and Vigo 2014).

The advantage of these traditional operators is their efficiency: the change in cost (the delta) resulting from a move can often be calculated very quickly without re-evaluating the entire solution.

A simple local search will quickly become trapped in a local optimum. Metaheuristics are higher-level strategies designed to guide local search operators to explore the solution space more effectively. Tabu Search introduces a memory structure (a tabu list) that forbids the reversal of recent moves, pushing the search to explore new regions and escape local optima (Glover and Taillard 1993). Large Neighborhood Search (LNS), takes a more aggressive approach by destroying a large part of the current solution and then using a constructive heuristic to repair it, effectively jumping to a distant and potentially more promising area of the solution space (Shaw 1998; Pisinger and Ropke 2007). The strategic combination of different neighborhoods, as seen in Variable Neighborhood Descent (VND), further enhances exploration by systematically escalating from simple, fast operators to more complex and powerful ones (Mladenović and Hansen 1997).

Across these approaches, the practical trade-offs are consistent: more powerful inter-route operators can overcome deeper local optima and yield larger objective gains per accepted move, but their raw neighborhood sizes demand algorithmic controls—bounded segment lengths, limited chain depth, candidate lists, and fast incremental delta evaluations—to remain computationally tractable in large instances. Modern implementations therefore pair high-impact inter-route operators with focused search strategies and pruning heuristics so that the expected improvement per evaluated move remains high while full-solution recomputation is avoided (Pisinger and Ropke 2018; Toth and Vigo 2014).

2.1.3 The Challenge for Assignment-Routing Heuristics

The common thread among these powerful, state-of-the-art heuristics is their fundamental reliance on an explicit, sequenced route structure. Knowledge of this route structure enables the efficient computation of the objective function when applying operators and helps guide the application of operators by indicating where and which routes should be modified.

This reliance presents a significant challenge for the Assignment–Routing formulation.

Because the representation is an unordered set of customers assigned to a vehicle (a partition), rather than a sequenced tour, standard operators are not directly applicable. To evaluate even a simple swap of two customers between two vehicles, one would need to re-solve two separate TSP instances to determine the new optimal routing costs. Repeating this for every potential move in a neighborhood is computationally inefficient.

This exposes a critical methodological gap: to effectively search the space of customer partitions, a new objective function is needed—one that can accurately and efficiently evaluate the quality of an assignment without requiring the explicit construction of its corresponding optimal route. This necessity directly motivates the investigation into TSP tour length estimators as a computationally inexpensive proxy for the true routing cost.

2.2 Tour Length Estimation for the TSP

The Assignment-Routing framework reframes the VRP by creating numerous TSP subproblems. To evaluate any potential assignment of customers to a vehicle, we must determine the cost of the optimal tour for that assignment. While exact TSP solvers like Concorde exist, a more computationally efficient yet highly accurate approach is the LKH-3 solver (Helsgaun 2017). LKH-3 has consistently found optimal solutions for standard TSPLIB instances and produced near-optimal results for very large problems, with deviations typically well below 1% (Helsgaun 2000). However, repeatedly solving the TSP for every candidate move within a larger VRP search remains a significant computational bottleneck. This motivates the study of tour length estimators—methods that efficiently approximate optimal tour cost. We survey three prominent classes of estimators.

2.2.1 Spatial Estimators

The earliest tour length estimation methods are analytical formulas derived from principles of geometric probability. These models typically rely on asymptotic assumptions, estimating the tour length based on the number of nodes, n, and the geometric properties of the area in which the nodes are distributed. A notable modern example from Vinel and Silva (2018) found that the distribution of optimal tour lengths for random points in a square approaches a normal distribution, even for a small number of nodes ($n \geq 10$). While powerful, the limitation of such models is their reliance on strong assumptions about the underlying node distribution (e.g., uniform in a square), which rarely holds for real-world customer locations.

Addressing this limitation, the distribution-free model by Çavdar and Sokol (2015) represents a logical progression. Instead of assuming a specific distribution, their regression model incorporates geometric properties of the actual point set, primarily the overall dispersion of customers in region. This provides greater robustness to arbitrary node distributions. However, the model was designed for and validated on dense TSP instances, with its accuracy generally improving for instances with over 100 nodes, potentially limiting its applicability for the smaller, more varied customer assignments often found within a VRP solution.

2.2.2 Machine Learning-Based Estimators

A more recent and robust direction leverages machine learning, particularly neural networks, to approximate the optimal tour length of TSP instances. As shown by Varol, Özener, and Albey (2024), these models are trained on diverse, synthetically generated instances that replicate real-world logistics networks and morphologies, where the true optimal tour lengths are available. Instead of relying solely on simple instance statistics, their approach constructs a richer feature space that integrates multiple abstraction levels: instance-level descriptors (e.g., the width and height of the tightest bounding rectangle), node-level information (e.g., normalized coordinates and nearest neighbor distances), and solution-level predictors (e.g., lower bounds and partial solutions derived from a dendrogram-based heuristic) (Varol, Özener, and Albey 2024). This hybridization of domain knowledge and data-driven learning enables the neural estimators to achieve prediction errors of less than 0.7% from optimality, significantly outperforming prior estimation models (Varol, Özener, and Albey 2024).

While ML-based estimators can achieve high accuracy across arbitrary distributions, they come with significant practical trade-offs. The primary limitation is computational expense; training requires a massive upfront investment to generate and solve tens of thousands of TSP instances, and even inference (making a prediction) is comparatively expensive relative to other machine learning architectures.

2.2.3 Graph-Based Bounds and Feature Engineering

A simpler and computationally cheaper approach to approximation is the use of graph-based bounds to improve existing estimators. The Minimum Spanning Tree (MST), which connects all nodes with the minimum possible total edge weight, is trivial to compute relative to solving for the TSP. The length of the MST serves as a natural lower bound for the TSP tour length and twice the length of the MST is a guaranteed upper bound (Christofides 2022). The Christofides algorithm, which provides a worst-case performance guarantee of 1.5 times the optimal tour length, further solidifies the theoretical link between the MST and the TSP (Christofides 2022). The MST can be computed efficiently in $O(n^2)$ using Prim's algorithm (Prim 1957). While tightening the bounds requires additional computation (Christofides 2022), this strong correlation suggests that the MST length is itself a powerful predictive feature. Its value lies not only as a bound but as a source of valuable geometric information for more advanced estimators.

3. Mathematical Formulations for VRP

This chapter provides an overview of the VRP through two fundamental mathematical viewpoints. We begin by presenting a standard arc-flow formulation, which serves as a classical and direct integer programming model of the problem. Following this, we present an Assignment-Routing formulation that decomposes the problem into two distinct stages: assigning customers to vehicles subject to Traveling Salesman Problem (TSP) tour length and then finding the routes for each vehicle by solving the TSP.

3.1 Problem Definition and Notation

The VRP is defined on a complete directed graph $G = (V', A)$, where $V' = \{0, 1, \dots, n\}$ is the set of $n + 1$ vertices and $A = \{(i, j) \mid i, j \in V', i \neq j\}$ is the set of arcs. Vertex 0 represents the depot, while the vertex set $V = V' \setminus \{0\}$ is the set of n customers. We assume each customer $i \in V$ generates demand $q_i = 1$ and a fleet of m identical vehicles, each with capacity Q, is available at the depot. A non-negative cost d_{ij}, which we assume is defined by the Euclidean distance between i and j, is associated with each arc $(i, j) \in A$. The goal is to design exactly m vehicle routes starting and ending at the depot such that each customer is visited exactly once to meet the demand, the total demand on each route does not exceed Q, and the total distance traveled by the vehicles is minimized.

3.2 An Arc-Flow Formulation

A common way to model the VRP is through a two-index arc-flow formulation, where the decisions focus on the arcs traversed by the vehicles (Toth and Vigo 2014). Let x_{ij} be a binary variable equal to 1 if any vehicle travels directly from node i to node j, and 0 otherwise, for all $(i, j) \in A$. Additionally, let u_i be a continuous variable representing the cumulative load delivered up to and including customer $i \in V$ (a fictional commodity flow to enforce sub-tour elimination and capacity constraints).

The formulation is as follows:

$$\begin{aligned}
\min \quad & z = \sum_{(i,j)\in A} d_{ij}x_{ij} & & \\
\text{s.t.} \quad & \sum_{j\in V', j\neq i} x_{ij} = 1, & \forall i \in V \quad & (1) \\
& \sum_{i\in V', i\neq j} x_{ij} = 1, & \forall j \in V \quad & (2) \\
& \sum_{j\in V} x_{0j} = m & & (3) \\
& \sum_{i\in V} x_{i0} = m & & (4) \\
& u_j \geq u_i + q_j - Q(1 - x_{ij}), & \forall i,j \in V, i\neq j \quad & (5) \\
& q_i \leq u_i \leq Q, & \forall i \in V \quad & (6) \\
& x_{ij} \in \{0,1\}, & \forall (i,j) \in A. \quad & (7)
\end{aligned}$$

The objective function minimizes the total travel cost. Constraints (1) and (2) are degree constraints ensuring each customer is entered and exited exactly once. Constraints (3) and (4) enforce that exactly m vehicles depart from and return to the depot. Constraints (5) and (6) together prevent subtours and ensure capacity limits are respected. Specifically, if $x_{ij} = 1$, then constraint (5) implies $u_j \geq u_i + q_j$, forcing the cumulative load to increase by at least q_j along the route. Since $u_i \leq Q$ from (6), the total demand on any route cannot exceed the vehicle capacity Q. Moreover, in any potential subtour (a cycle not including the depot), the strict increase in u values would lead to a contradiction, as the values cannot cycle back consistently. The lower bound in (6) ensures feasibility for individual demands. This is a compact formulation with a polynomial number of variables and constraints, though its linear relaxation is often weak, necessitating branch-and-cut enhancements for large instances (Toth and Vigo 2014).

3.3 An Assignment-Routing Formulation (AR)

The VRP can also be reformulated by decomposing it into two interconnected subproblems: assigning customers to vehicles while respecting capacity constraints, and routing each vehicle through its assigned customers to form a minimum-cost tour.

3.3.1 Conceptual Framework

A feasible solution to the VRP consists of a partition of the customer set V into exactly m disjoint subsets $\{P_1, P_2, \ldots, P_m\}$, where each subset P_k represents the customers assigned to vehicle k, and the union of all P_k equals V. For feasibility, each assignment P_k must satisfy the capacity constraint: $\sum_{i \in P_k} q_i \leq Q$. Let S denote the set of all such feasible partitions.

The total cost of a partition s is the sum of the minimum routing costs for each assignment. Specifically, let $T(P_k)$ be the cost of the optimal TSP tour on the vertex set $\{0\} \cup P_k$, which determines the shortest Hamiltonian cycle starting and ending at the depot. The total cost of the VRP routes in partition s can then be computed as follows:

$$f(s) = \sum_{k=1}^{m} T(P_k).$$

The VRP is equivalent to finding the optimal partition $s^* \in \mathcal{S}$ that minimizes $f(s)$:

$$s^* = \arg\min_{s \in S} f(s)$$

and solving for the best route within each partition.

3.3.2 Formalizing the Subproblems

This decomposition can be expressed through two coupled integer programs: a set partitioning model for the assignment and a TSP model for the routing within each partition.

The Assignment Problem (Set Partitioning View) To formulate the assignment as an integer program, let binary variables $y_{ik} = 1$ if customer $i \in V$ is assigned to a subset P_k, and 0 otherwise. Then the minimum-cost assignment problem can be modeled as:

$$\begin{aligned}
\min \quad & f(s) = \sum_{k=1}^{m} T(\{i \in V \mid y_{ik} = 1\} \cup \{0\}) & & \\
\text{s.t.} \quad & \sum_{k=1}^{m} y_{ik} = 1, & \forall i \in V \quad & (8) \\
& \sum_{i \in V} q_i y_{ik} \leq Q, & \forall k \in \{1, \ldots, m\} \quad & (9) \\
& y_{ik} \in \{0, 1\}, & \forall i \in V, k \in \{1, \ldots, m\}. \quad & (10)
\end{aligned}$$

Constraint (8) ensures each customer is assigned to exactly one vehicle. Constraint (9) enforces the capacity limit of each vehicle. The number of customers assigned to a vehicle cannot exceed the vehicles capacity for each vehicle.

The Routing Subproblem (TSP) For each partition of customers P_k from the assignment stage, the routing subproblem is to find the minimum cost tour. This is a classic Traveling Salesman Problem (TSP) defined on the subgraph induced by the vertex set $V_k = \{0\} \cup P_k$.

Let z_{ij} be a binary variable equal to 1 if the tour for vehicle k traverses arc (i, j), and 0 otherwise. To prevent subtours, we use the Miller-Tucker-Zemlin (MTZ) formulation (Miller, Tucker, and Zemlin 1960), which requires an auxiliary variable. Let w_i be a continuous variable representing the position of customer $i \in P_k$ in the sequence of the tour. The cost $T(P_k)$ is the optimal value of the following integer program:

$$
\begin{aligned}
T(P_k) = \min \quad & \sum_{(i,j)\in A_k} d_{ij} z_{ij} & & \\
\text{s.t.} \quad & \sum_{j\in V_k, j\neq i} z_{ij} = 1, & \forall i \in V_k \quad & (11)\\
& \sum_{i\in V_k, i\neq j} z_{ij} = 1, & \forall j \in V_k \quad & (12)\\
& w_i - w_j + |P_k| z_{ij} \leq |P_k| - 1, & \forall i, j \in P_k, i \neq j \quad & (13)\\
& 1 \leq w_i \leq |P_k|, & \forall i \in P_k \quad & (14)\\
& z_{ij} \in \{0, 1\}, & \forall (i, j) \in A_k, \quad & (15)
\end{aligned}
$$

where $A_k = \{(i, j) \mid i, j \in V_k, i \neq j\}$ is the set of arcs connecting the depot and the customers assigned to vehicle k.

The objective function minimizes the tour length for the given set of customers. Constraints (11) and (12) are degree constraints ensuring that each customer and the depot are entered and exited exactly once within the tour. Constraints (13) and (14) are the MTZ subtour elimination constraints. They work by assigning an increasing position number w_i to each customer along the path. If a vehicle travels from customer i to j (i.e., $z_{ij} = 1$), constraint (13) enforces that $w_j \geq w_i + 1$, which is impossible to satisfy for all arcs in a cycle that do not include the depot, thus eliminating subtours. The depot (vertex 0) is not given a position variable, effectively anchoring it as the start and end of the sequence.

3.4 Equivalence with the Set Partitioning Formulation

To establish the validity of the Assignment-Routing (AR) formulation, we can demonstrate its equivalence to the well-known Set Partitioning (SP) formulation for the VRP provided by Baldacci, Christofides, and Mingozzi (2008).

The SP formulation requires the enumeration of every feasible route a single vehicle could take. A route is considered feasible if it starts and ends at the depot and the total demand of the customers it serves does not exceed the vehicle capacity Q. The SP model then seeks to select a combination of exactly m routes from this vast collection with two primary conditions: the total cost of the selected routes is minimized, and every customer is served by exactly one of the chosen routes.

The AR formulation expands on this intuition. While the SP model makes decisions over an exponential set of individual routes, the AR model makes decisions over the set of customer partitions. The equivalence is rooted in a simple observation: an optimal VRP solution must, for any chosen subset of customers P_k, serve them using the minimum-cost sequence. This minimum cost is, by definition, the optimal TSP tour cost, $T(P_k)$.

The SP model finds this solution by selecting the best routes from a collection that implicitly contains the optimal tour for every feasible customer subset. The AR formulation achieves the same result more directly by finding the best partition of customers, where the cost of serving each partition element P_k is defined as its optimal routing cost, $T(P_k)$. Because both approaches must converge on the same combination of customer groupings and their corresponding best routes, they are fundamentally equivalent.

3.5 The Role of the TSP Cost Estimation

The equivalence highlights the challenges of using the assignment-based formulation; solving it requires calculating the optimal TSP cost for an exponential number of potential assignments. However, this challenge can be addressed heuristically by using a TSP tour length estimation model for each partition instead of solving the actual TSP. By replacing the exact TSP cost $T(P_k)$ with a computationally inexpensive estimator $\hat{T}(P_k)$, we define a tractable objective function, $\hat{f}(s)$:

$$\hat{f}(s) = \sum_{k=1}^{m} \hat{T}(P_k)$$

The optimization problem then becomes finding the partition $\hat{s}^*$ that minimizes this estimated cost:

$$\hat{s}^* = \arg\min_{s \in \mathcal{S}} \hat{f}(s)$$

The core of our solution strategy is to search for $\hat{s}^*$, under the hypothesis that a good estimator $\hat{T}$ will lead to an estimated optimal partition $\hat{s}^*$ that corresponds to a high-quality solution for f. If the hypothesis holds, the TSP is solved only once to determine the actual routes on the final set of nodes assigned to m vehicles. In Chapter 4 we propose a precise $\hat{T}$ to power our solution approach.

4. Design and Evaluation of a TSP Tour Length Estimator

The Assignment-Routing formulation developed in Chapter 3 hinges on the ability to evaluate the cost of a customer assignment, a task that requires solving the TSP. To make the formulation tractable for a heuristic search, we replace the exact TSP cost, $T(P)$, with an efficient and precise estimator, $\hat{T}(P)$. While there are several high-quality estimation models available, it is not difficult to find a type of instance where the estimation quality of each model is subpar. Therefore, we first propose a new TSP tour length estimation model. This chapter details the process of developing a new tour length estimation model for TSP.

Due to the shortcomings of available estimation models, we combine them into piecewise estimators. The purpose of a piecewise estimator is to ensure that each model is only used when its underlying assumptions are met. We further improve the quality of the piecewise estimators by truncating predictions that fall outside of the MST bounds. However, experimental results show that the piecewise approach is insufficient to cover the wide range of instances that can occur. Consequently, we develop a Geometrically Assisted Regression Tree (GART) that forecasts the relationship between the MST and the TSP based on the properties of the provided tour. We compare the precision of two regression-based estimators, a piecewise estimator constructed using the two models, GART, and a piecewise estimator that includes GART. Benchmarking shows that the GART estimator is significantly more precise than the other models.

4.1 Regression-Based Estimators from Literature

Analytical estimators approximate tour length using formulas derived from geometric probability and empirical regression. While these models are powerful and fast, their underlying assumptions can limit their precision across diverse instance types.

4.1.1 The BHH-Based Estimator ($\hat{T}_{VS}$)

The Beardwood-Halton-Hammersley (BHH) theorem provides a foundational result in geometric probability, stating that the length of an optimal TSP tour, T, for a large number of nodes n drawn from a probability distribution over a region of area A, is asymptotically proportional to $\sqrt{nA}$ (Beardwood, Halton, and Hammersley 1959). For the special case of nodes distributed uniformly over a region, the theorem simplifies to the well-known form $T \approx \beta\sqrt{nA}$ (Vinel and Silva 2018).

As this study specifically utilizes the empirically derived constant from Vinel and Silva (2018), we refer to this model as $\hat{T}_{VS}$, for brevity. The formula is:

$$\hat{T}_{VS} = b\sqrt{n \cdot A}$$

where n is the number of nodes and b is the constant (0.768) from Vinel and Silva (2018). While computationally efficient, this model's accuracy is predicated on the assumption of spatial uniformity, which may not hold for clustered or irregular node distributions.

4.1.2 Çavdar and Sokol's (2015) Distribution-Free Estimator ($\hat{T}_{ÇS}$)

To improve robustness for arbitrarily distributed nodes, Çavdar and Sokol (2015) developed a distribution-free regression model that incorporates more detailed sample statistics of the node set. The formula consists of two main terms:

$$\hat{T}_{ÇS} = a_o\sqrt{n \cdot \sigma_{cx}\sigma_{cy}} + a_1\sqrt{\frac{n \cdot \sigma_x\sigma_y \cdot A}{\bar{c}_x\bar{c}_y}}$$

Here, σ_x and σ_y are the standard deviations of the node coordinates, while $\bar{c}_x, \bar{c}_y, \sigma_{cx}, \sigma_{cy}$ are statistics related to the mean and standard deviation of the absolute deviation of nodes from their centroid. The constants $a_0 = 2.791$ and $a_1 = 0.2669$ are empirically fitted. To improve accuracy for smaller instances ($n < 1000$), the authors introduced a correction factor, C_n, by which the final estimate is divided:

$$C_n = 0.9325e^{0.00005298n} - 0.2972e^{-0.01452n}$$

This model is more computationally intensive than Vinel's but is designed to provide consistent estimates without prior assumptions about the spatial layout of the customers. However, its performance on strongly clustered distributions or for small tour sizes ($n \leq 100$) has not been validated.

4.2 Proposed Estimators

To address the limitations of the literature models, we propose three new estimators designed to offer improved precision across a wider range of TSP instances.

4.2.1 The Baseline Composite Estimator (BCE)

As a reasonable baseline, we define a piecewise estimator that leverages the respective strengths of the regression-based estimators. Based on empirical evidence that $\hat{T}_{VS}$ performs well on smaller n and $\hat{T}_{ÇS}$ is more robust for large n, we define the Baseline Composite Estimator (BCE) as:

$$\hat{T'}_{BCE} = \begin{cases} \hat{T}_{VS} & \text{if } n \leq 100 \\ \hat{T}_{ÇS} & \text{if } n > 100 \end{cases}$$

To prevent extreme prediction errors from irregular distributions, the final estimate is truncated to lie within the valid theoretical bounds defined by the Minimum Spanning Tree (MST) length. The final bounded estimate, $\hat{T}_{BCE}$, is computed as:

$$\hat{T}_{BCE} = \max(MST, \min(\hat{T'}_{BCE}, 2 \cdot MST))$$

4.2.2 Geometrically Assisted Regression Tree (GART)

The GART model is a supervised machine learning model trained to predict the optimal TSP tour length for mid-sized instances ($n \leq 100$), to bridge the gap between exact methods and $\hat{T}_{ÇS}$. While designed for mid-sized instances, GART generalizes well across distributions and larger instances. Neural network–based tour length estimation, such as the one by (Varol, Özener, and Albey 2024), is accurate but an expensive framework. GART seeks to take advantage of the computationally inexpensive features proposed by Varol, Özener, and Albey (2024) and Çavdar and Sokol (2015) in a regression tree framework. By taking advantage of the relationships between geometric features and tour length found in the literature, a broad training set, and a novel intuition, we produce a model that is a significantly more resilient estimator at an acceptable computational cost.

4.2.2.1 Foundational Model

The core principle of GART is to learn the instance-specific relationship between the TSP tour length and its corresponding Minimum Spanning Tree (MST) length. This relationship is hypothesized based on the principle that the MST length already serves as a reasonable bound on the TSP. Tightening that bound based on a geometric understanding proves a promising method to build a simple but effective approximation technique. This intuition powers a multiplicative model:

$$T \approx \alpha \cdot MST$$

GART's task is to predict the unique scalar ratio, α, for a specific partition P, using a rich set of geometric and topological features derived from the node set, for which we chose LightGBM. LightGBM is a high-performance gradient boosting library that constructs an ensemble of decision trees to minimize a specified loss function (Ke et al. 2017). It employs a histogram-based splitting mechanism, which bins continuous features for efficient computation, and a leaf-wise tree growth strategy that prioritizes splits with the highest loss reduction. This approach enables the handling of large, high-dimensional datasets with reduced memory usage and faster training times compared to traditional pre-sorted boosting methods.

Prediction involves feeding the feature vector into the ensemble. Each decision tree traverses from root to leaf based on feature thresholds, outputting a value; these are summed (scaled by the learning rate) to produce α. With 2000 trees, each limited to 286 leaves (implying average depths of approximately $\log_2(286) \approx 8$), the prediction complexity is $O(T \cdot D)$, where $T = 2000$ is the number of trees and D is the average depth. This ensures logarithmic-time traversals per tree and efficient inference.

4.2.2.2 Feature Selection

To accurately predict tour length, GART is trained on a feature set designed to capture the nuanced spatial properties of a customer assignment. These features are grouped into several categories:

- **Convex Hull Features:** Metrics such as area, perimeter, and the ratio of hull vertices to total nodes, which describe the overall spatial footprint and boundary complexity (Çavdar and Sokol 2015).

- **Nearest-Neighbor Statistics:** The mean and standard deviation of the distance from each node to its closest neighbor, capturing the degree of local clustering or dispersion (Varol, Özener, and Albey 2024).

- **PCA-based Anisotropy:** The ratio of the eigenvalues from a Principal Component Analysis (PCA) on the node coordinates, which measures the elongation or directional bias of the point set (Varol, Özener, and Albey 2024).
- **MST Topology Features:** Statistics on the degree distribution of the MST, such as its mean, maximum, and the fraction of leaf nodes, which encode information about the graph's connectivity (Varol, Özener, and Albey 2024).
- **Coordinate and Depot Dispersion:** Standard deviations of the coordinates and the average and maximum distance of customers from the depot.

The complete list is provided in Table 4.1. All features are calculated for the set of customer nodes together with the depot. When the depot is not defined, a depot is chosen arbitrarily.

Table 4.1: Complete feature set for the GART model.

Category	**Feature name**	**Description**
Instance	`n`	Number of nodes in the instance (customers plus depot).
Convex Hull	`convex_hull_area`	Area of the convex hull enclosing all nodes, in coordinate units.
	`convex_hull_perimeter`	Perimeter of the convex hull (sum of edge lengths along its polygonal boundary).
	`hull_vertex_count`	Number of nodes that lie on the convex hull polygon (count of hull vertices).
	`hull_ratio`	Ratio of hull vertices to total nodes, defined as $\frac{\texttt{hull\_vertex\_count}}{\texttt{n}}$.

Continued on next page

Table 4.1 (continued)

Category	**Feature name**	**Description**
Bounding Box	`bounding_box_area`	Area of the axis-aligned minimum bounding rectangle covering all nodes (width $\times$ height).
Coordinate Dispersion	`coord_std_dev_x`	Standard deviation of node x-coordinates (horizontal spread).
	`coord_std_dev_y`	Standard deviation of node y-coordinates (vertical spread).
Nearest-Neighbor	`one_nn_dist_mean`	Mean Euclidean distance from each node to its nearest neighbor.
	`one_nn_dist_std`	Standard deviation of nearest-neighbor distances (local density variability).
Depot-Customer Metrics	`avg_dist_from_depot`	Average Euclidean distance of customers from the depot (excluding the depot itself).
	`max_dist_from_depot`	Maximum Euclidean distance of any customer from the depot.
PCA Features	`pca_eigenvalue_ratio`	Ratio of the primary to secondary PCA eigenvalues on node coordinates (elongation/directionality).
MST Topology	`mst_degree_mean`	Mean node degree in the minimum spanning tree (MST).
	`mst_degree_max`	Maximum node degree observed in the MST.
	`mst_degree_std`	Standard deviation of node degrees in the MST (degree heterogeneity).

Continued on next page

Table 4.1 (continued)

Category	Feature name	Description
	`mst_leaf_nodes_fraction`	Fraction of nodes with degree 1 in the MST, i.e., leaf nodes divided by `n`.

Feature importance, computed as the average split gain across trees, highlights the model's reliance on nearest-neighbor and dispersion metrics (e.g., `one_nn_dist_mean` with importance 53,950; `pca_eigenvalue_ratio` with 48,414). The full ranking is visualized in Figure 4.1, which presents a bar plot of descending importance. Conversely, the plot reveals that `mst_degree_mean` (the mean node degree of the minimum spanning tree) was not a significant predictor, registering a feature importance near zero. This finding suggests a potential area for refinement in future iterations of the GART model.

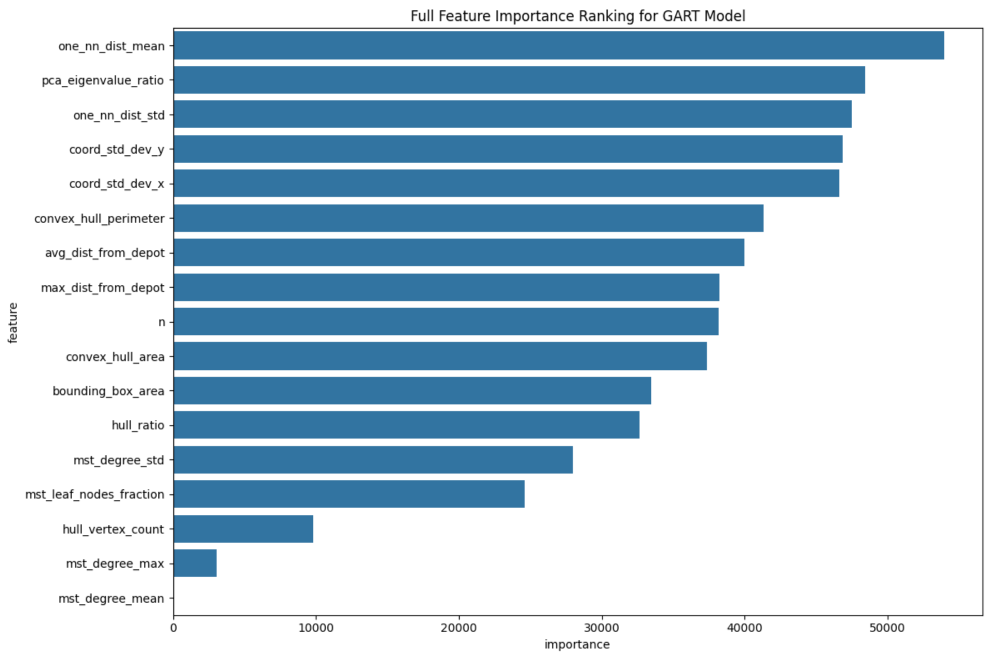

Figure 4.1: Full Feature Importance Ranking for GART.

4.2.2.3 Training Pipeline

A large dataset of over 600,000 unique tour instances is generated by randomly sampling routes from the benchmark XML100 CVRP dataset by Queiroga et al. (2021). This data is comprised of tours between 10 and 100 nodes in length only. The data was split into training

(70%), validation (15%), and testing (15%) sets. Model hyperparameters are optimized using the Optuna framework to minimize Mean Absolute Error (MAE) on a validation set. The final parameters used for training the model are listed in Table 4.2.

Table 4.2: Optimized LightGBM hyperparameters for the GART model.

Parameter	**Value**	**Description**
`objective`	`regression_l1`	L1 loss (Mean Absolute Error) for the regression task.
`metric`	`mae`	Evaluation metric used during training.
`boosting_type`	`gbdt`	Gradient Boosting Decision Tree.
`n_estimators`	2000	Total number of boosting rounds (trees).
`learning_rate`	0.04995	Step size shrinkage used to prevent overfitting.
`num_leaves`	286	Maximum number of leaves in one tree.
`max_depth`	-1	No limit on tree depth.
`min_child_samples`	20	Minimum number of data points required in a leaf.
`random_state`	42	Seed for reproducibility.
`n_jobs`	1	Number of parallel threads used for training.

Testing shows GART has an R^2 of 0.9454, indicating that the model explains 94.54% of the variance in α values, and a Mean Absolute Error (MAE) of 0.0396. We also verify regression validity via diagnostic plots:

- **Normality of Residuals**: The histogram of residuals (Figure 4.2) approximates a normal distribution, centered near zero with a slight right skew, supporting the assumption of normality.

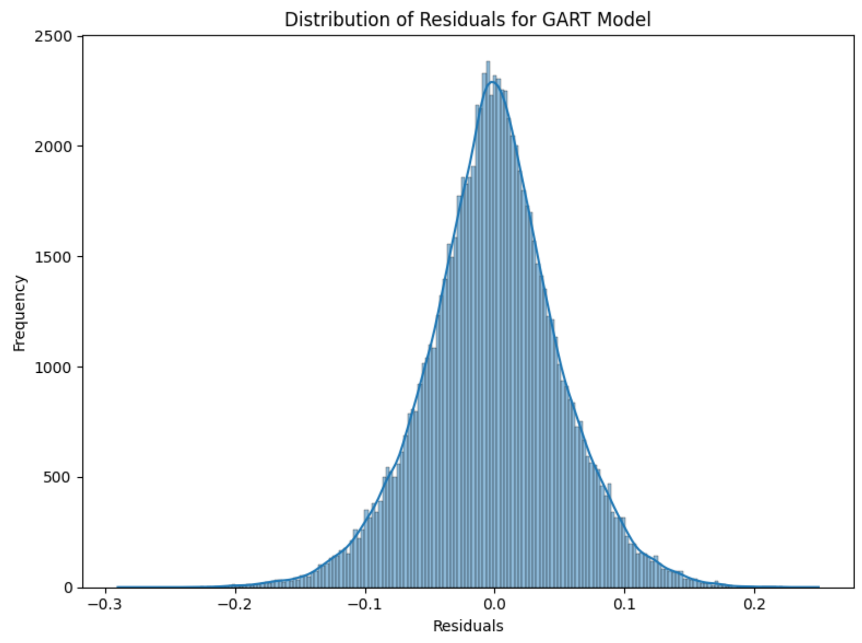

Figure 4.2: Distribution of Residuals for GART.

- **Homoscedasticity**: The residual plot (Figure 4.3) shows residuals scattered evenly around zero across predicted values (1.2 to 2.0), with no funneling, indicating constant variance.

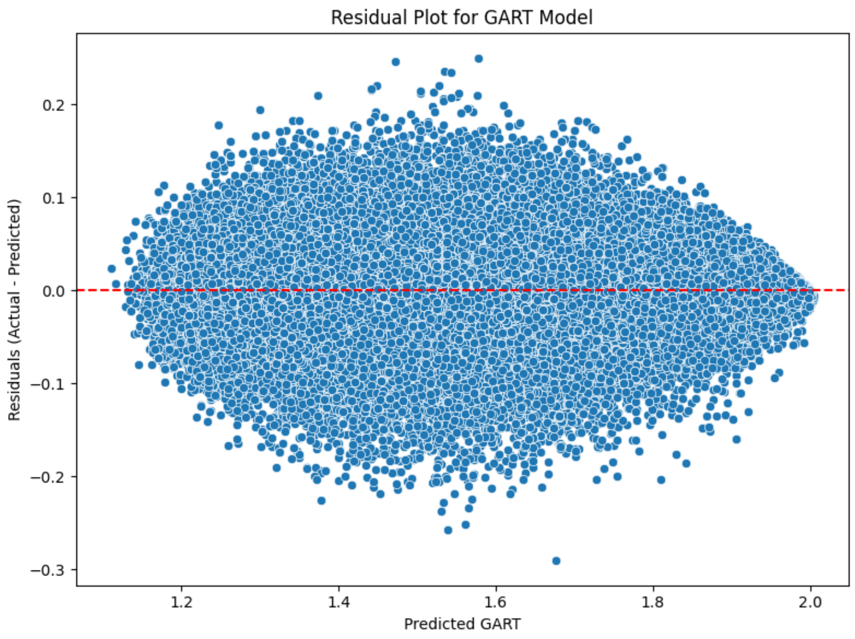

Figure 4.3: Residual Plot for GART.

These diagnostics confirm the model's robustness for TSP estimation in VRP contexts.

4.2.3 The Composite Machine Learning Assisted Estimator (CMLA)

We propose an advanced composite model that integrates our GART estimator into the piecewise framework. The Composite Machine Learning Assisted Estimator (CMLA) replaces the less robust $\hat{T}_{VS}$ estimator with $\hat{T}_{GART}$ for small- to mid-sized instances, while retaining $\hat{T}_{ÇS}$ for larger instances:

$$\hat{T}'_{CMLA} = \begin{cases} \hat{T}_{GART} & \text{if } n \leq 100, \\ \hat{T}_{ÇS} & \text{if } n > 100. \end{cases}$$

This architecture combines the finely tuned accuracy of GART on smaller instances with the scalability of $\hat{T}_{ÇS}$ for larger problem sizes. As with the Baseline Composite Estimator (BCE), the predictions of CMLA are bounded by the Minimum Spanning Tree (MST) to improve robustness:

$$\hat{T}_{CMLA} = \max\left(MST,\ \min\left(\hat{T}'_{CMLA},\ 2 \cdot MST\right)\right).$$

This bounding ensures that the estimator remains consistent with theoretical limits, preventing extreme over- or under-estimation across diverse instance types.

4.3 Experimental Design for Benchmarking

To select the best tour length estimation model $\hat{T}$ for TSP, we design a comprehensive benchmarking experiment. Through these experiments, we compare the candidate estimators, namely $\hat{T}_{VS}$, $\hat{T}_{ÇS}$, $\hat{T}_{GART}$, $\hat{T}_{BSE}$, and $\hat{T}_{CMLA}$ on a diverse dataset of TSP instances.

4.3.1 Benchmark Dataset

To ensure a comprehensive and unbiased evaluation, we generate a large synthetic dataset comprising 1,685 unique TSP instances. The dataset's structure is primarily based on a factorial experimental design, systematically crossing instance size (n) with spatial point distribution.

Factor 1: Instance Size (n) The number of nodes, n, varies across thirty distinct levels to assess estimator performance on problems of different scales. The selected sizes span several orders of magnitude, from very small instances to large-scale problems common in logistics. The specific values for n used in the experiment are: 5, 8, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000.

Factor 2: Spatial Node Distributions To model a diverse range of scenarios, we generate instances from twelve different spatial distributions. All point sets are generated within a standardized 1000×1000 coordinate plane. The experimental design includes 11 standard distributions that are fully crossed with the 30 instance sizes. The 12th distribution, 'clustered', is generated using a separate, specific set of configurations. For each configuration, 5 replicates are generated to account for stochastic variability. Table 4.3 provides a complete list of each distribution type.

4.3.2 Performance Metric

The primary metric for evaluating estimator performance is percentage error. For a given prediction $\hat{T}$ and known cost T, the error is calculated as:

Table 4.3: Spatial Distributions of Nodes for Benchmark Dataset Generation.

Distribution Type	**Description**
Random (Uniform)	Nodes are drawn from a uniform distribution over $[0, 1000]^2$.
Normal	Nodes are drawn from a bivariate normal distribution with $\mu = (500, 500)$ and $\sigma \approx 166.7$.
Triangular	Nodes are generated using a triangular probability density function centered on the plane.
Squeezed Uniform	A non-uniform distribution where node density increases towards the corner $(1000, 1000)$.
Uniform Triangular	Nodes are uniform along the x-axis and have a triangular distribution along the y-axis.
Triangular Squeezed	Nodes have a triangular distribution on the x-axis and a squeezed distribution on the y-axis.
Boundary	A non-uniform distribution where node density is highest near the boundaries of the plane.
X-Central	A non-uniform distribution where node density is highest along the central x-axis.
Exponential	Nodes generated from an exponential distribution, with coordinates wrapped into the $[0, 1000]^2$ plane.
Grid	Nodes are placed on a regular grid with slight random jitter.
Correlated	The y-coordinate is linearly dependent on the x-coordinate, following $y = x + \mathcal{N}(0, 100)$.
Clustered	Nodes grouped into k clusters of radius r. Uses the following 7 specific configurations, not the full factorial design: **n=100**: (k=5, r=50), (k=10, r=50), (k=10, r=100) **n=500**: (k=10, r=50), (k=20, r=100) **n=1000**: (k=20, r=100) **n=5000**: (k=50, r=100)

$$\text{Percentage Error } = \frac{\hat{T} - T}{T} \times 100\%$$

A positive error indicates an overestimation, while a negative error indicates an underestimation. For the purposes of this study, we use LKH-3 (Helsgaun 2017) to approximate T.

4.4 Results and Analysis

This section presents the results of the comprehensive benchmark of the candidate TSP tour length estimators. The analysis is designed to identify the most suitable estimator $\hat{T}$, among our proposed models and those in the literature, to power the estimated objective function $\hat{f}$ in our VRP search heuristic.

When selecting an estimator to guide a metaheuristic search, it is crucial to distinguish between its accuracy (or bias) and its precision (or variance). Accuracy, measured by the mean error, reflects the tendency to systematically over- or underestimate the true cost. Precision, measured by the variance of the error, reflects the estimator's consistency. For a search algorithm, an estimator's precision is critical. A predictable, consistently biased estimator (e.g., one that is always 5% too high) still preserves the relative ordering of solutions in a neighborhood and provides a reliable "sense of direction" for the search. In contrast, an imprecise (high-variance) estimator is noisy and unpredictable, providing an unreliable signal that can hinder a heuristic's convergence. Therefore, our primary goal is to identify the most precise TSP tour length estimator. In the following sections, we present our numerical results.

4.4.1 Overall Estimator Performance

The primary goal of the benchmark is to identify the most precise estimator across the 1,520 instances in our test set where $n > 10$. Figure 4.4 presents the distribution of percentage errors for each evaluated method. The plot provides a visual summary of both the accuracy and the precision of each estimator. According to the plot, $\hat{T}_{GART}$ exhibits a significantly smaller interquartile range and shorter whiskers than all other estimators except $\hat{T}_{CMLA}$, while having fewer outliers, suggesting superior precision.

Table 4.4 provides the summary statistics for each estimator. While $\hat{T}_{GART}$ shows a mean error of 6.94%, indicating an overestimation on average, its standard deviation of 6.17% is by far the lowest. In contrast, the other estimators have means closer to zero but suffer from standard deviations that are two to three times larger, confirming their lower precision. It is important to note that the MST bounds added to $\hat{T}_{CBE}$ and $\hat{T}_{CMLA}$ reduce the presence of outliers, which shows that the MST bounds are effective at limiting the impact of irregular distributions.

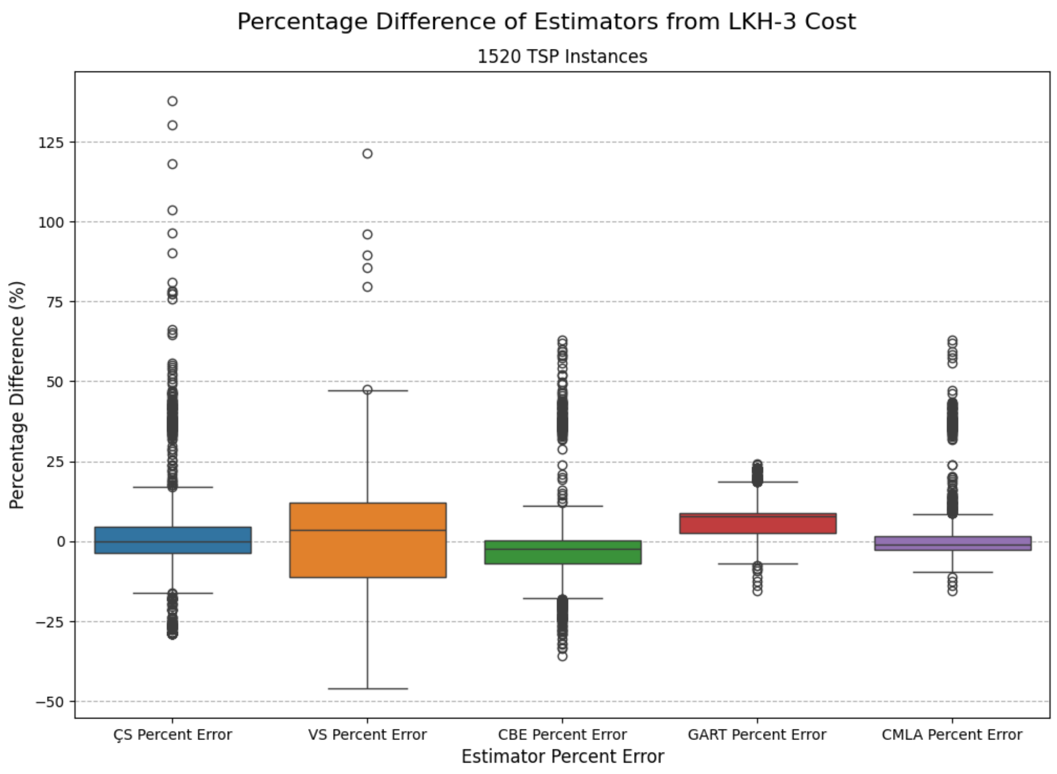

Figure 4.4: Overall estimation error distribution by method.

Table 4.4: Summary Statistics of Percentage Error for All Estimators ($n > 10$).

Estimator	Mean (%)	Median (%)	Standard Deviation (%)
$\hat{T}_{VS}$	0.22	3.65	16.86
$\hat{T}_{ÇS}$	2.90	-0.11	17.28
$\hat{T}_{CBE}$	-1.31	-2.58	14.23
$\hat{T}_{GART}$	**6.94**	**7.71**	**6.17**
$\hat{T}_{CMLA}$	2.03	-0.97	10.94

4.4.2 Statistical Analysis

To rigorously compare the precision of the estimators, we perform statistical tests for normality and homogeneity of variances to identify the most precise estimator.

Normality of Error Distributions The Shapiro-Wilk test is applied to the error distributions of each estimator. The null hypothesis (H_0), that the errors are drawn from a normal distribution, is rejected for all five estimators ($p < 0.0001$), as detailed in Table 4.5. Given that we cannot assume the results follow the normal distribution, we use Levene's test and the Fligner-Killeen test to assess the homogeneity of variance.

Table 4.5: Results of the Shapiro-Wilk Test for Normality on Estimator Percentage Errors.

Estimator	Statistic (W)	p-value
$\hat{T}_{VS}$	0.9427	< 0.0001
$\hat{T}_{ÇS}$	0.7828	< 0.0001
$\hat{T}_{CBE}$	0.7671	< 0.0001
$\hat{T}_{GART}$	0.9315	< 0.0001
$\hat{T}_{CMLA}$	0.8872	< 0.0001

Homogeneity of Variances Given the non-normal nature of the data, we use two robust methods, Levene's test and the Fligner-Killeen test, to assess the equality of variances across all estimators. The null hypothesis (H_0) for these global tests is that all estimators share a common error variance. As shown in Table 4.6, both tests return highly significant results, allowing us to reject H_0 and conclude that there are significant differences in precision among the estimators.

Table 4.6: Global Tests for Homogeneity of Variances.

Test	Statistic	p-value	Conclusion
Levene's Test	168.41	< 0.0001	Variances are unequal
Fligner-Killeen Test	612.95	< 0.0001	Variances are unequal

Pairwise Variance Comparisons and Ranking To identify a definitive hierarchy of precision, we conduct post-hoc pairwise comparisons using both Levene's and Fligner-Killeen tests with a Holm correction for multiple comparisons. The raw variance values establish a clear order of performance, from most to least precise: $\hat{T}_{GART}$ (38.07), $\hat{T}_{CMLA}$ (119.63), $\hat{T}_{CBE}$ (202.46), $\hat{T}_{VS}$ (284.22), and $\hat{T}_{ÇS}$ (298.44).

The pairwise test results, presented in Table 4.7, largely confirm this ranking. However, they reveal a noteworthy conflict regarding the top two estimators. While Levene's test finds that $\hat{T}_{GART}$ is significantly more precise than $\hat{T}_{CMLA}$ ($p = 0.0001$), the Fligner-Killeen test finds their variances to be statistically indistinguishable ($p = 0.4357$). All other adjacent comparisons in the ranking are confirmed to be statistically significant by both tests.

Table 4.7: Full Pairwise Tests for Equality of Variances with Holm Correction.

Comparison	Levene's p-value	Fligner-Killeen p-value
$\hat{T}_{GART}$ vs. $\hat{T}_{CMLA}$	0.0001	0.4357
$\hat{T}_{GART}$ vs. $\hat{T}_{CBE}$	< 0.0001	< 0.0001
$\hat{T}_{GART}$ vs. $\hat{T}_{VS}$	< 0.0001	< 0.0001
$\hat{T}_{GART}$ vs. $\hat{T}_{ÇS}$	< 0.0001	< 0.0001
$\hat{T}_{CMLA}$ vs. $\hat{T}_{CBE}$	< 0.0001	< 0.0001
$\hat{T}_{CMLA}$ vs. $\hat{T}_{VS}$	< 0.0001	< 0.0001
$\hat{T}_{CMLA}$ vs. $\hat{T}_{ÇS}$	< 0.0001	< 0.0001
$\hat{T}_{CBE}$ vs. $\hat{T}_{VS}$	< 0.0001	< 0.0001
$\hat{T}_{CBE}$ vs. $\hat{T}_{ÇS}$	0.0002	0.0003
$\hat{T}_{VS}$ vs. $\hat{T}_{ÇS}$	< 0.0001	< 0.0001

4.4.3 Selected Tour Length Estimators

The results of the benchmark analysis are definitive. $\hat{T}_{GART}$ demonstrates statistically significant or practically superior precision compared to all other candidate estimators. While it exhibits a consistent positive bias, its low variance makes it the most reliable model for providing a stable cost signal to a search heuristic. The other estimators, even if more accurate, suffer from high variance, rendering them noisy and less suitable for guiding an optimization algorithm.

Therefore, we select $\hat{T}_{GART}$ as the primary estimator to power the objective function $\hat{f}$ in our main VRP experiments. For the large-scale $n > 100$ instances, we also consider $\hat{T}_{ÇS}$ for comparative purposes, as its lower computational costs for long routes enables increased exploration.

In the case of small tours, we encode a fallback into our estimator function. For all cases where $n \leq 10$, $\hat{T}$ is replaced by computing the TSP tour length via the Held-Karp Algorithm (Held and Karp 1962). With a time complexity of $O(n^2 2^n)$, it is feasible only for the smallest vehicle assignments (Held and Karp 1962). This option enables independent tuning of heuristics, independent of the performance of the TSP tour length estimator. In the following chapter, we utilize the estimators to construct heuristics that are compatible with the Assignment-Routing formulation and examine their effectiveness.

5. Computational Analysis of the Assignment-Routing Formulation

In this chapter, we design and execute two experiments to investigate the relationship between the true optimal VRP solution, $s^* = \arg\min_{s \in \mathcal{S}} f(s)$, and the local optimizer of $\hat{f}(s)$. This comparison provides direct insight into the efficacy of using the estimated objective function $\hat{f}$ as a proxy to guide a search for high-quality VRP solutions.

5.1 Experimental Design

To evaluate the alignment between the true and estimated objective functions, we design two distinct experiments. Experimental performance is evaluated by two metrics. The 'true gap' measures the quality of a found solution relative to the known optimum, while the 'estimator gap' measures the quality relative to the estimated cost of the optimum.

$$\text{Estimator Gap} = \frac{\hat{f}(s_{best}) - \hat{f}(s^*)}{\hat{f}(s^*)} \times 100\%$$
$$\text{True Gap} = \frac{f(s_{best}) - f(s^*)}{f(s^*)} \times 100\%$$

A divergence between these two metrics—for instance, if the estimator gap decreases while the true gap increases—provides strong evidence that the estimator's perception of a good solution is misaligned with the true VRP objective, suggesting that an optimizer of $\hat{f}$ does not find s^* or high-quality solutions relative to f.

5.1.1 Datasets and Test Instances

For both experiments, we construct a stratified sample of 72 instances from the XML100 dataset (Queiroga et al. 2021), for which proven optimal solutions are known, and for the second experiment, 36 additional instances from the larger XML1000 dataset, for which best-known solutions are used as a benchmark (see Appendix B for information on the XML1000 dataset).

5.1.2 Heuristic Approaches

We develop two heuristics with different search strategies to test the estimator's behavior under different conditions.

2-Opt Tabu Search The first experiment utilizes a simple Tabu search designed to perform local refinement. Its neighborhood is defined by two fundamental operators:

- **Relocation:** A single customer is moved from its current vehicle to a different vehicle. This move requires spare capacity to be available on the destination vehicle.
- **Swap:** A customer belonging to one vehicle is exchanged with a customer belonging to a different vehicle.

The heuristic iteratively generates the complete neighborhood of the current solution, evaluates each candidate using the cost estimator $\hat{f}$, and selects the move with the greatest estimated cost improvement. A standard Tabu list with a tenure of $\sqrt{N}$ (where N is the number of customers) prevents cycling, and an aspiration criterion overrides a tabu move if it leads to a new global best solution (Glover and Taillard 1993). The search terminates after a fixed time limit or if no valid moves exist (see Appendix A for pseudocode). This heuristic approach is not feasible for larger instances, as the number of candidate moves grows exponentially with the number of customers.

Ad Hoc Search Heuristic: Multi-Campaign Search To test the estimator's performance in a more realistic global search scenario, we develop a second, more powerful ad hoc heuristic. The development of this 'Multi-Campaign Search' is necessitated by the fact that our assignment-based formulation precludes the use of standard routing heuristics. This heuristic aggressively explores the solution space guided by the estimator, starting from an arbitrary feasible solution. Its ability to explore a wide variety of neighborhoods makes it a suitable instrument for finding solutions close to the estimator's optimum, $\hat{s}^*$. Further implementation details are provided in Appendix C.

5.1.3 Experimental Procedure

The two experiments are designed to test different aspects of the estimator's performance.

- **Experiment 1 (Local Divergence Test):** This experiment applies the 2-Opt Tabu Search. Each run is initialized with the known optimal solution, s^*. The objective is to test whether optimizing along $\hat{f}$ causes the search to move away from the true optimum. A total of 72 instances from XML100 are each explored for 2 hours.

- **Experiment 2 (Global Guidance Test):** This experiment applies the Multi-Campaign Search heuristic. Each run is initialized with an arbitrary feasible solution generated by a regret-based heuristic. The objective is to evaluate how effectively the estimator can guide a powerful search from scratch toward a high-quality final solution. The same 72 instances in the XML100 dataset are explored for 2 hours each using $\hat{f}_{GART}$, where $\hat{f}_{GART}$ is computed via $\hat{T}_{GART}$. In addition, 36 instances from the XML1000 dataset are explored for 6 hours each, with one test using $\hat{f}_{GART}$ and another using $\hat{f}_{ÇS}$ calculated via $\hat{T}_{ÇS}$.

5.2 Experiment 1: Local Divergence from the True Optimum on XML100

This experiment is designed to answer the question: when starting at the true optimum s^*, does optimizing along the estimated objective function $\hat{f}$ lead the search to a different, sub-optimal solution? The results show that it does, demonstrating a fundamental misalignment between the estimator's cost landscape and the true VRP objective.

Starting from the optimal solution (where the true gap is 0%), the 2-Opt Tabu Search heuristic, guided by the estimator, moves to a new solution in nearly every case. As summarized in Table 5.1, the search concludes with an average true gap of 2.389%, meaning the final solutions are significantly worse than the starting optimum. Concurrently, the heuristic achieves an average estimator gap of -2.668%, indicating that it successfully finds solutions it perceives as substantial improvements. This divergence, where minimizing the estimated cost leads to a quantifiable increase in the true cost, is the key finding of this experiment.

This experiment demonstrates a clear local divergence between the estimated and true objective functions. The estimator's local optimum in the neighborhood of s^* is demonstrably different from s^* itself. The next experiment investigates the consequences of this misalignment in a more realistic global search scenario.

Table 5.1: Results of Experiment 1: Local Refinement from the True Optimum (s^*) on the XML100 dataset.

Statistic	**Estimator Gap (%)**	**True Gap (%)**
Average	-2.668	2.389
Standard Deviation	1.388	1.290

5.3 Experiment 2: Global Search Guided by the Estimator

Given the local divergence identified in Experiment 1, we now test how well a powerful heuristic performs when guided by this estimator in a global search. The results are partitioned by the XML100 and XML1000 datasets.

On the XML100 benchmark set, for which optimal solutions are known, the Multi-Campaign Search guided by the estimator averages a true gap of 4.976%, as shown in Table 5.2. Simultaneously, it achieves a large negative estimator gap of -3.557%, confirming that the heuristic consistently locates solutions it perceives as high-quality while failing to find near-optimal solutions. This exposes a significant limitation of the two-stage estimator-based formulation, where optimizing for the estimator does not equate to optimizing for the true objective.

Statistic	**Multi-Campaign Search**
Avg. True Gap (%)	4.976
Avg. Estimator Gap (%)	-3.557
Standard Deviation of True Gap (%)	2.079
Standard Deviation of Estimator Gap (%)	2.217

Table 5.2: Multi-Campaign Search Performance on the XML100 Dataset

It is important to note, however, that searching along the estimator does provide improvement over the initial solutions provided to the search. An example of the search trajectory is shown in Figure 5.1. The report, which plots the estimated gap at each improving move and the true gap at each new global best, shows that the true gap improves relative to the initial solution for a time. However, after a certain point (approximately 10% true gap in this example), further improvements to the estimator gap lead to erratic and often worsening behavior of the true gap.

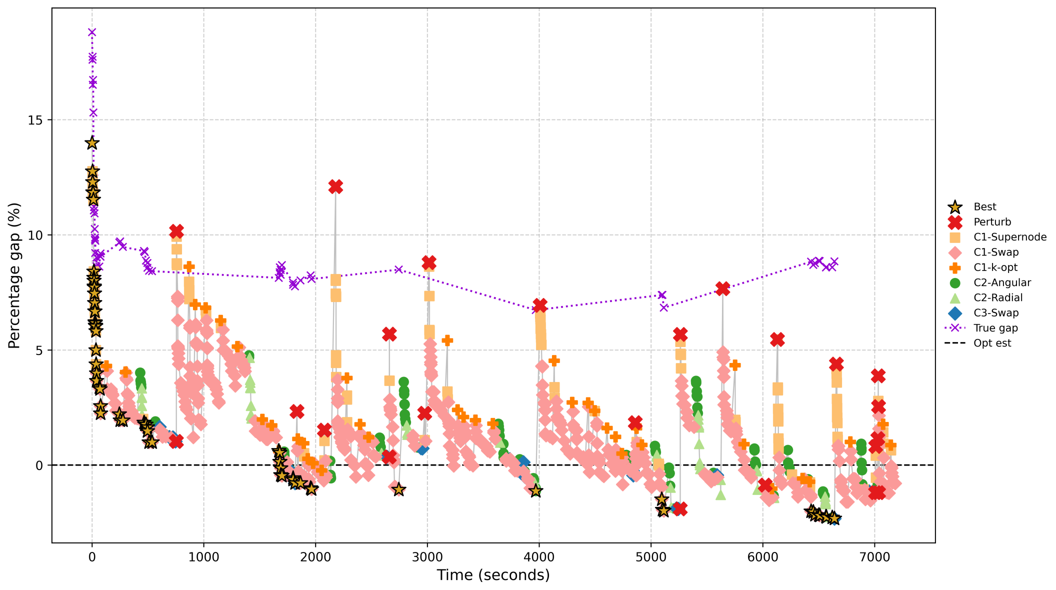

Figure 5.1: Multi-Campaign Search Performance Tracker Example.

The results on the larger and more complex XML1000 instances reveal a significant performance advantage when using the superior $\hat{T}_{GART}$ estimator to guide the search. As summarized in Table 5.3, the Multi-Campaign Search guided by $\hat{T}_{GART}$ achieves an average gap of 1.776% against the best available solutions. This is a marked improvement over the 2.906% gap achieved when the search is guided by the $\hat{T}_{ÇS}$ estimator, demonstrating that $\hat{T}_{GART}$'s superior predictive capabilities translate directly into finding higher-quality final VRP solutions. For completeness, Table 5.3 also includes comparative performance of Multi-Campaign Search against other common constructive heuristics.

While the more precise $\hat{T}_{GART}$ estimator improves heuristic performance, the framework still struggles to consistently outperform a dedicated solver. When compared against 6-hour runs of OR-Tools, the $\hat{T}_{GART}$-guided search produces an average gap of 1.659%, indicating that OR-Tools finds better solutions on average. Nonetheless, the estimator-guided approach proves competitive, with the $\hat{T}_{GART}$-guided search finding a better solution than OR-Tools in 13 out of 36 instances. The minimum gap of -6.265% against the best available solutions further shows that, despite some systemic bias, the estimator-driven framework is powerful enough to discover higher-quality solutions for f.

Table 5.3: Relative performance of the Multi-Campaign Search on the XML1000 dataset.

Performance Metric (% Gap)	**Multi-Campaign Search (MCS)**	
$\frac{f(s^{\widetilde{*}}_{MCS})-f(s^{\widetilde{*}}_{H})}{f(s^{\widetilde{*}}_{H})} \times 100\%$	**Guided by $\hat{f}_{ÇS}$**	**Guided by $\hat{f}_{GART}$**
Versus OR-Tools (6 Hours) ($s^{\widetilde{*}}_{OR}$)		
Mean	2.789%	**1.659%**
Std. Dev.	4.360%	**4.150%**
95% CI	(1.312%, 4.266%)	(0.256%, 3.061%)
Median	2.897%	**1.484%**
Min / Max	-5.580% / 12.624%	-6.265% / 11.655%
MCS Obtained Better Solution	9 / 36	**13 / 36**
Versus KMeans Clustering ($s^{\widetilde{*}}_{KM}$)		
Mean	-10.483%	**-11.462%**
Std. Dev.	4.340%	**4.240%**
95% CI	(-11.951%, -9.015%)	(-12.898%, -10.027%)
Median	-10.290%	**-11.570%**
Min / Max	-21.251% / 0.316%	-21.585% / -3.763%
MCS Obtained Better Solution	35 / 36	**36 / 36**
Versus Nearest Neighbor Heuristic ($s^{\widetilde{*}}_{NN}$)		
Mean	-2.758%	**-3.829%**
Std. Dev.	3.010%	**2.680%**
95% CI	(-3.777%, -1.740%)	(-4.737%, -2.922%)
Median	-2.440%	**-3.620%**
Min / Max	-7.164% / 3.499%	-8.332% / 2.138%
MCS Obtained Better Solution	28 / 36	**32 / 36**
Versus Best Available Solution ($min_{h \in H} f(s^{\widetilde{*}}_{h})$)		
Mean	2.906%	**1.776%**
Std. Dev.	4.200%	**4.030%**
95% CI	(1.484%, 4.327%)	(0.414%, 3.139%)
Median	2.897%	**1.484%**
Min / Max	-5.580% / 12.624%	-6.265% / 11.655%
MCS Obtained Better Solution	9 / 36	**13 / 36**

5.4 Experimental Analysis

The combined results of both experiments provide a clear answer to our central research question. Experiment 1 demonstrates that the local gradient of the estimator at the true optimum s^* is non-zero, indicating that the estimator's optimum, $\hat{s}^*$, is fundamentally different from the true optimum, s^*. Experiment 2 reveals the practical consequences of this misalignment. When a powerful heuristic is guided by a biased estimator, it converges to solutions with a large negative estimator gap but a persistent positive true gap. Consequently, when working with smaller customer sets where traditional methods are more developed, the Assignment-Routing formulation is less attractive.

However, the results for the $\hat{T}_{GART}$-guided search on the XML1000 instances present a more optimistic picture. By providing a more accurate signal to the heuristic, $\hat{T}_{GART}$ facilitates a much more effective search. This superior guidance translates directly into finding higher-quality VRP solutions that significantly outperform those found using $\hat{T}_{ÇS}$. While this approach does not yet consistently outperform a state-of-the-art solver, it serves as a proof of concept for the Assignment-Routing formulation. Its ability to find new best-in-class solutions for several instances shows considerable promise. This highlights that while finding the exact optimal solution using a proxy objective function may not be achievable, high-quality solutions can still be efficiently obtained.

6. Conclusion

This research sets out to investigate the viability of an estimator-driven Assignment-Routing framework for the Capacitated Vehicle Routing Problem (VRP). The central goal is to address the fundamental research question: can a local optimizer of an *estimated* objective function of the VRP, $\hat{f}$, efficiently locate high-quality solutions to f? To answer this, we develop a general tour length estimator, and conduct a series of computational experiments to analyze the relationship between estimator accuracy and final VRP solution quality.

6.1 Summary of Key Findings

Our investigation yields several key findings that span the domains of tour length estimation and VRP heuristic design. First, in developing a suitable estimator, we demonstrate that our novel Geometrically Assisted Regression Tree estimator ($\hat{T}_{GART}$) is significantly more accurate and robust at predicting TSP tour lengths than existing analytical models, without the computational drawbacks of a neural network. We also show that applying the MST as bounds to existing tour length estimators significantly strengthens their performance on irregular distributions.

Second, our VRP experiments provide a clear, albeit nuanced, answer to our central research question. Experiment 1, which performs a local search from the true optimum s^*, shows that the estimator-guided heuristic consistently diverges to a different solution. This provides direct evidence that the estimator's optimum is fundamentally different from the true optimum ($\hat{s}^* \neq s^*$). Experiment 2 confirms the practical consequences of this misalignment in a global search context. The powerful Multi-Campaign Search, when guided by the estimator, consistently converges to solutions with a large negative estimator gap but a persistent positive true gap. This demonstrates a systematic bias in the estimator, where its perception of a high-quality solution does not align with the true VRP cost function.

Finally, despite this demonstrated bias, the results on problems with more customers reveal that the estimator-driven approach is not without merit. The heuristic is competitive with existing VRP heuristics, indicating that even a flawed estimator can effectively guide a search into high-quality regions of the solution space.

6.2 Implications

The findings of this work have significant implications for the future design of VRP heuristics. The primary takeaway is that the quality of a proxy objective function is paramount. Raw predictive accuracy (e.g., low Mean Absolute Percentage Error) on standalone TSP instances is not, by itself, a sufficient condition for an estimator to serve as an effective guide in a VRP context. Our results show that even a small systematic bias can be amplified by a powerful metaheuristic, leading the search to converge on demonstrably suboptimal solutions.

This suggests that the focus of future research in this domain must shift. Instead of solely pursuing accuracy, the development of tour length estimators should prioritize precision. The goal should be to design and train estimators whose cost landscapes more closely mirror that of the true objective function, even if this comes at the expense of a slight decrease in raw predictive accuracy on individual instances.

6.3 Limitations and Future Research

This study, while comprehensive, has several limitations that open clear avenues for future work. First, the $\hat{T}_{GART}$ model is trained on a dataset derived primarily from the XML100 benchmark. Our results show that its performance, while strong, is less stable on out-of-distribution structures such as highly clustered instances. Second, our VRP experiments employ a specific, highly customized metaheuristic, and the interaction between the estimator and other search paradigms remains an open question.

Building on these findings, we propose the following directions for future research:

- **Advanced Estimator Design and Training:** The $\hat{T}_{GART}$ model represents a promising avenue for future development in tour length estimation based on the multiplicative MST relationship. Future work should focus on tuning both the accuracy (mean) and precision (variance) of the estimates to ensure proper alignment with the true VRP objective. This can be achieved through disciplined feature engineering, such as the inclusion of cluster-aware features, and by developing broader, more diverse training datasets that encompass a wider array of VRP route structures.

- **Hybrid Metaheuristic Frameworks:** The Multi-Campaign Search can be substantially improved by embedding it within a hybrid framework. The estimator-driven search proves effective at global exploration and could be used to quickly drive a solution into a near-optimal neighborhood. Once in this promising region, the search could switch to a different paradigm for fine-grained local intensification. This may involve using an exact TSP solver on the small number of active assignments or transitioning to a traditional route-based local search that employs operators such as 2-opt and CROSS-exchange for final solution refinement. The current implementation is limited in terms of computational optimization. Refinements to the operators and implementation would enable the construction of a more powerful heuristic that is not restricted to unitary demand (e.g., a single customer node requiring exactly one unit of vehicle capacity).

- **Integration with Exact Methods:** The estimator could also be used to enhance exact VRP algorithms. For example, in a branch-and-price framework, the estimator could be employed to quickly price columns (routes) or to provide a strong initial upper bound, thereby accelerating the convergence of the exact solver.

In conclusion, this work demonstrates both the promise and the peril of using estimators to guide a VRP search. While showing that the current generation of estimators can serve as a flawed proxy, it simultaneously illuminates a clear and promising path forward for the development of the next generation of intelligent, two-stage VRP heuristics.

Bibliography

Baldacci, Roberto, Nicos Christofides, and Aristide Mingozzi. 2008. "An Exact Algorithm for the Vehicle Routing Problem Based on the Set Partitioning Formulation with Additional Cuts." *Mathematical Programming* 115 (2): 351–385.

Beardwood, Jillian, John H Halton, and John Michael Hammersley. 1959. "The Shortest Path Through Many Points." *Mathematical Proceedings of the Cambridge Philosophical Society* 55 (4): 299–327.

Çavdar, Bahar, and Joel Sokol. 2015. "A Distribution-Free TSP Tour Length Estimation Model for Random Graphs." *European Journal of Operational Research* 243 (2): 588–598.

Christofides, Nicos. 2022. "Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem." *Operations Research Forum* 3 (1): 20.

Clarke, Geoff, and John W Wright. 1964. "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points." *Operations Research* 12 (4): 568–581.

Cuvelier, Thibaut, Frederic Didier, Vincent Furnon, Steven Gay, Sarah Mohajeri, and Laurent Perron. 2023. "Or-Tools' Vehicle Routing Solver: A Generic Constraint-Programming Solver with Heuristic Search for Routing Problems." In *24e Congrès Annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision.*

Glover, Fred. 1996. "Ejection Chains, Reference Structures and Alternating Path Methods for Traveling Salesman Problems." *Discrete Applied Mathematics* 65 (1-3): 223–253.

Glover, Fred, and Eric Taillard. 1993. "A User's Guide to Tabu Search." *Annals of Operations Research* 41 (1): 1–28.

Held, Michael, and Richard M Karp. 1962. "A Dynamic Programming Approach to Sequencing Problems." *Journal of the Society for Industrial and Applied Mathematics* 10 (1): 196–210.

Helsgaun, Keld. 2000. "An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic." *European Journal of Operational Research* 126 (1): 106–130.

Helsgaun, Keld. 2017. *An Extension of the Lin-Kernighan-Helsgaun TSP Solver for Constrained Traveling Salesman and Vehicle Routing Problems.* Technical report 12:966-980. Roskilde, Denmark: Roskilde University.

Ke, Guolin, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. “LightGBM: A Highly Efficient Gradient Boosting Decision Tree.” In *Advances in Neural Information Processing Systems 30,* edited by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, 3146–3154. Curran Associates, Inc.

Kirkpatrick, Scott, C Daniel Gelatt Jr, and Mario P Vecchi. 1983. “Optimization by Simulated Annealing.” *Science* 220 (4598): 671–680.

Lenstra, Jan Karel, and AHG Rinnooy Kan. 1981. “Complexity of Vehicle Routing and Scheduling Problems.” *Networks* 11 (2): 221–227.

Lin, Shen. 1965. “Computer Solutions of the Traveling Salesman Problem.” *Bell System Technical Journal* 44 (10): 2245–2269.

Lourenço, Helena R, Olivier C Martin, and Thomas Stützle. 2003. “Iterated Local Search.” In *Handbook of Metaheuristics,* edited by Fred Glover and Gary A. Kochenberger, 320–353. Springer.

Miller, Clair E, Albert W Tucker, and Richard A Zemlin. 1960. “Integer Programming Formulation of Traveling Salesman Problems.” *Journal of the ACM (JACM)* 7 (4): 326–329.

Mladenović, Nenad, and Pierre Hansen. 1997. “Variable Neighborhood Search.” *Computers & Operations Research* 24 (11): 1097–1100.

Moghdani, Reza, Khodakaram Salimifard, Emrah Demir, and Abdelkader Benyettou. 2021. “The Green Vehicle Routing Problem: A Systematic Literature Review.” *Journal of Cleaner Production* 279:123691.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. “Scikit-Learn: Machine Learning in Python.” *The Journal of Machine Learning Research* 12:2825–2830.

Pisinger, David, and Stefan Ropke. 2007. "A General Heuristic for Vehicle Routing Problems." *Computers & Operations Research* 34 (8): 2403–2435.

Pisinger, David, and Stefan Ropke. 2018. "Large Neighborhood Search." In *Handbook of Metaheuristics,* edited by Michel Gendreau and Jean-Yves Potvin, 99–127. Cham: Springer.

Potvin, Jean-Yves, and Jean-Marc Rousseau. 1993. "A Parallel Route Building Algorithm for the Vehicle Routing and Scheduling Problem with Time Windows." *European Journal of Operational Research* 66 (3): 331–340.

Prim, Robert Clay. 1957. "Shortest Connection Networks and Some Generalizations." *The Bell System Technical Journal* 36 (6): 1389–1401.

Queiroga, Eduardo, Ruslan Sadykov, Eduardo Uchoa, and Thibaut Vidal. 2021. "10,000 Optimal CVRP Solutions for Testing Machine Learning Based Heuristics." In *AAAI-22 Workshop on Machine Learning for Operations Research (ML4OR).*

Ropke, Stefan, and David Pisinger. 2006. "An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows." *Transportation Science* 40 (4): 455–472.

Rosenkrantz, Daniel J, Richard E Stearns, and Philip M Lewis II. 1977. "An Analysis of Several Heuristics for the Traveling Salesman Problem." *SIAM Journal on Computing* 6 (3): 563–581.

Shaw, Paul. 1998. "Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems." In *International Conference on Principles and Practice of Constraint Programming,* edited by Michael Maher and Jean-Francois Puget, 417–431. Springer.

Taillard, Éric, Philippe Badeau, Michel Gendreau, François Guertin, and Jean-Yves Potvin. 1997. "A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows." *Transportation Science* 31 (2): 170–186.

Toth, Paolo, and Daniele Vigo. 2003. "The Granular Tabu Search and Its Application to the Vehicle-Routing Problem." *INFORMS Journal on Computing* 15 (4): 333–346.

Toth, Paolo, and Daniele Vigo. 2014. *Vehicle Routing: Problems, Methods, and Applications.* Philadelphia: SIAM.

Varol, Taha, Okan Örsan Özener, and Erinç Albey. 2024. “Neural Network Estimators for Optimal Tour Lengths of Traveling Salesperson Problem Instances with Arbitrary Node Distributions.” *Transportation Science* 58 (1): 45–66.

Vinel, Alexander, and Daniel F Silva. 2018. “Probability Distribution of the Length of the Shortest Tour Between a Few Random Points: A Simulation Study.” In *2018 Winter Simulation Conference (WSC),* 3156–3167. IEEE.

APPENDIX A
Simple 2-Opt Tabu Search Heuristic

The algorithm implements a Tabu Search (TS) metaheuristic (Glover and Taillard 1993) designed to optimize customer partitions for the Vehicle Routing Problem (VRP). The search does not operate on sequenced routes, but rather on the assignment of customers to vehicles. It commences with an initial partition, $S_{initial}$, and iteratively seeks to find improved configurations by exploring a defined neighborhood.

The neighborhood structure is composed of two standard move types: a relocation, which transfers a single customer to a different route, and a swap, which exchanges two customers between two different routes (Taillard et al. 1997; Toth and Vigo 2003). All candidate moves are first verified for capacity feasibility.

At each iteration, the algorithm employs a best-improvement strategy by evaluating all feasible moves in the combined relocation and swap neighborhood. The quality (cost) of each potential new partition is evaluated using a provided cost estimator function. To prevent cycling and guide the search from local optima, the heuristic maintains a short-term memory, or Tabu List, implemented as a FILO queue (Glover and Taillard 1993). The size of this list, the tabu tenure, is dynamically set to $\lfloor \sqrt{N} \rfloor$, where N is the total number of customers. When a move is executed, its corresponding attribute (e.g., the customer(s) involved) is added to the tabu list, forbidding its reversal for the duration of the tenure.

A standard Aspiration Criterion is utilized, which overrides the tabu status of a move if that move leads to a new global-best solution (i.e., a partition with an estimated cost lower than $Cost_{best}$) (Glover and Taillard 1993). The search proceeds iteratively, applying the best-allowed move at each step, until a specified time limit is reached or no feasible moves remain. The algorithm returns the best-found partition, S_{best}, encountered during the search.

Algorithm 1 Simple Tabu Search for VRP Partitions

```
function TabuSearch($S_{initial}$, $CostEstimator$, $Timeout$)
    $S_{current}$ and $S_{best} \leftarrow S_{initial}$    ▷ Initialize the current solution and best-so-far solution.
    $Cost_{best} \leftarrow CostEstimator(S_{best})$
    $Tenure \leftarrow \lfloor\sqrt{\text{number of customers}}\rfloor$    ▷ Set the size of the search's memory.
    $TabuList \leftarrow$ new empty queue with max size $Tenure$
    while (current time $-$ $StartTime$) $<$ $Timeout$ do
        $BestMove \leftarrow$ FindBestNeighborhoodMove($S_{current}, TabuList, Cost_{best}$)
        if $BestMove =$ null then    ▷ Search has run out of moves.
            break
        end if
        $S_{current} \leftarrow$ ApplyMove($S_{current}, BestMove$)    ▷ Commit to the best move found.
        add $BestMove.TabuAttribute$ to $TabuList$    ▷ And forbid reversals.
        if $CostEstimator(S_{current}) < Cost_{best}$ then    ▷ Check for a new global best solution.
            $S_{best} \leftarrow S_{current}$
            $Cost_{best} \leftarrow CostEstimator(S_{current})$
        end if
    end while
    return $S_{best}$
end function

function FindBestNeighborhoodMove($S$, $TabuList$, $Cost_{best}$)
    $BestOverallMove \leftarrow$ null with $\Delta = \infty$
    for each move type in {Relocation, Swap} do
        for each specific move $M$ of the current type do
            if $M$ is feasible w.r.t. capacity then
                $\Delta \leftarrow$ CalculateDelta($S, M$)    ▷ Calculate the change in cost.
                $IsTabu \leftarrow M.Attribute \in TabuList$    ▷ Check the search's memory.
                $AspirationMet \leftarrow (Cost(S) + \Delta) < Cost_{best}$    ▷ Check the exception rule.
                if $\neg IsTabu$ or $AspirationMet$ then
                    if $\Delta < BestOverallMove.\Delta$ then
                        $BestOverallMove \leftarrow M$
                    end if
                end if
            end if
        end for
    end for
    return $BestOverallMove$
end function
```

APPENDIX B
XML1000 Dataset

This appendix explains the process used to generate the XML1000 dataset, which was employed to evaluate 2-Stage formulation performance under conditions more favorable for the tour length estimators. Benchmarking of the estimators showed that their accuracy improves as the density of nodes in a tour increases. XML1000 builds on this observation by generating a dataset of VRP instances with 1000 customers distributed over a 100×100 grid. Because the grid size is the same as in XML100 but vehicle capacity is increased, each route is longer and tends to have higher density.

The dataset was generated by modifying code provided by Queiroga et al. (2021). The generator script creates 10 instances for each configuration, iterating over the following factors:

- Depot Position: Random, Centered, Cornered.
- Customer Distribution: Random, Clustered, Random-Clustered.
- Vehicle Capacity: 150, 250, 350, 500.

Table B.1: Factorial design of the XML1000 dataset. Each configuration is generated with 10 replicates.

Factor	**Levels**
Depot Position	Random, Centered, Cornered
Customer Distribution	Random, Clustered, Random-Clustered
Vehicle Capacity	150, 250, 350, 500

Given the increased complexity of these instances, no optimal solutions are available. Consequently, a variety of standard VRP heuristics were used to generate feasible solutions, though without any guarantee of optimality:

- **Clarke-Wright Savings Heuristic** (Clarke and Wright 1964): The number of vehicles cannot be specified, which limits the use of these solutions in experimental analysis.

- **OR-Tools CVRP Solver** (Cuvelier et al. 2023): The full dataset is generated with initial solutions from `PATH_CHEAPEST_ARC`. For the subset of instances in Experiment 2, `GUIDED_LOCAL_SEARCH` is run for 6 hours to improve the initial solutions.
- **K-Means Clustering** (Pedregosa et al. 2011): A standard implementation (via Scikit-learn) with a random reassignment technique to ensure vehicles satisfy capacity constraints.
- **Nearest-Neighbor Heuristic** (Rosenkrantz, Stearns, and Lewis 1977): A standard implementation.

APPENDIX C
Multi-Campaign Search Heuristic

The Multi-Campaign Search heuristic was developed to support evaluation of the research question comparing $\hat{f}$ to f. The use of more common heuristics to find optimal or near-optimal solutions based on $\hat{f}$ is infeasible due to their reliance on information encoded in explicit routes. Without a route embedded within a solution, developing a search strategy aggressive enough to explore the solution space for $\hat{f}$ required an experimental approach. This heuristic is guided by the intuition that poor assignments contribute disproportionately to the objective function value. This marginal cost serves as an identifier of which customers should be targeted by swaps or other types of movement. While effective, this approach is prone to local optima, necessitating the construction of additional operators and new search techniques to ensure thorough exploration.

The heuristic is composed of five distinct components. The first is a regret-based method for constructing an initial feasible assignment of customers to vehicles, described in Section C.1. The remaining four components are operators that iteratively improve the initial solution and together compose the overall search strategy, described in Section C.2. Each component is triggered in sequence if the previous component fails to yield improvement. The first campaign targets nodes based on marginal cost and contributes the majority of improvements. The other three campaigns are designed to perturb the solution with different moves and selection criteria, enabling the first operator to discover new local minima.

The heuristic is tuned on small test cases where $\hat{f}$ is replaced by f. Verification on instances with low vehicle capacity shows that the heuristic is able to achieve strong solutions within 15 minutes when guided by f. Due to the low vehicle capacity, the computational cost of evaluating $T(A)$ is low, enabling a thorough search. The coded implementation is available on GitHub.

C.1 Initial Solution Generation

To initialize the optimization process, we require a feasible starting partition of customers into vehicle routes. We utilize the Regret-2 insertion heuristic inspired by Potvin and Rousseau (1993) to generate a strong initial solution. In our implementation, the algorithm is simplified by calculating the ‘regret’ value of adding a node based on its nearest neighbor rather than its true contribution to the solution.

C.2 The 4-Campaign Search Metaheuristic

The overall heuristic is structured as a deterministic, multi-stage metaheuristic that strategically balances intensification and diversification. The architecture combines principles from several established frameworks, primarily Variable Neighborhood Descent (VND) (Mladenović and Hansen 1997) and Iterated Local Search (ILS) (Lourenço, Martin, and Stützle 2003). The core philosophy is to employ a powerful and systematic local search engine as the default operator, escalating only to more disruptive or computationally expensive campaigns when the search becomes trapped in a local optimum.

The search process begins by generating a single, high-quality initial solution using a parallel Regret-2 insertion heuristic (Potvin and Rousseau 1993). This solution then becomes the incumbent for the main search loop, which operates under a strict, sequential escalation logic. Campaign 1, the primary intensification engine, is always executed first. If it discovers any improving move, the search immediately restarts the campaign sequence from the beginning, forcing Campaign 1 to exploit the new, superior solution. This “first improvement, restart” policy ensures that the heuristic remains in a state of intensive local optimization whenever progress is being made.

Only when Campaign 1 completes a full run without finding any improvement does the heuristic engage its diversification campaigns. These are triggered in a fixed sequence of escalating power: Campaign 2 (Polar Geometric Tuning), Campaign 3 (Granular Border Refinement), and finally Campaign 4 (Global Search and Perturbation). Each of these campaigns acts as an escape mechanism, modifying the solution in ways that create new opportunities for the primary local search engine. If any of Campaigns 2, 3, or 4 find an improvement, control is immediately returned to Campaign 1. The entire search terminates only when a full pass through all four campaigns yields no improvement, or the time limit is reached. This structure leverages computationally cheaper campaigns to escape shallow local optima, reserving the most powerful perturbation mechanism—the Ruin-and-Recreate strategy of Campaign 4, a form of Large Neighborhood Search (LNS) (Shaw 1998; Pisinger and Ropke 2007)—for escaping the deepest basins of attraction.

C.2.1 Systematic Local Search - Campaign 1

Campaign 1 is the primary intensification engine of the heuristic, designed to aggressively exploit the local search landscape of a given solution S. It operates as a Variable Neighborhood Descent (VND) strategy, progressing through a fixed sequence of four distinct local search operators, or tiers (Mladenović and Hansen 1997). The campaign follows a strict first-improvement policy: upon finding any cost-reducing move, the modified solution is accepted, and the campaign restarts its entire four-tier sequence from the beginning. This ensures that simpler operators are always prioritized after any change to the solution state. The search is guided by two memory structures: a tabu list to prevent recent moves from being reversed (Glover and Taillard 1993), and a node-locking mechanism. Nodes belonging to the top 25% of best-performing routes (by estimated cost) are temporarily "locked," exempting them from most move considerations and preserving well-optimized solution components. Candidate selection for all operators is heavily biased toward "worst" nodes, which are identified by their marginal cost contribution: for a node i in route R_k, its contribution is calculated as $\hat{T}(R_k) - \hat{T}(R_k \setminus \{i\})$. This focuses computational effort on the least efficient parts of the solution.

Tier 1: Supernode Operator The initial operator is a cluster-based move inspired by concepts from Granular Search, where expensive search operators are restricted to a small, promising subset of moves (Toth and Vigo 2003). A "supernode" is a spatially aware cluster of k customers, constructed around a high-cost "anchor" node and its $k-1$ nearest unlocked neighbors. Anchor nodes are selected from the top third of worst-contributing nodes within a given route. The operator first attempts a computationally cheap *supernode-relocate*, where the entire cluster is moved to a target vehicle if capacity permits. If this fails, it attempts a more complex *supernode-swap*, where it exchanges the supernode with a similarly sized supernode from a target vehicle. This operator is powerful because it can move a coherent group of geographically close customers at once, correcting larger structural flaws in the solution.

Tier 2: Probabilistic Relocate Polish If the supernode operator fails to find an improvement, the campaign proceeds to a probabilistic 1-opt (relocate) polish. This operator first identifies the complete set of all feasible, improving, and non-tabu relocate moves available in the current solution. Instead of greedily applying the best one, it employs a fitness-proportionate selection strategy. A probability distribution is constructed where the likelihood of selecting a particular move is proportional to the magnitude of its cost improvement (its Δ). A single move is then chosen from this distribution and applied. This probabilistic approach provides a measure of diversification, allowing the search to explore different improvement pathways instead of repeatedly selecting the most obvious greedy move.

Tier 3: Targeted 1-for-1 Swap The third operator is a more deterministic and focuses on 1-for-1 swap. It prioritizes entire routes based on "pressure," defined as the sum of the marginal contributions of their top three worst nodes. The operator iterates through the highest-pressure routes first. For each high-contribution node in the source route, it attempts to swap it with a candidate node from a target route. This provides a highly focused mechanism for exchanging poorly placed nodes between adjacent or interacting routes.

Tier 4: 3-Route Cyclic Exchange The final and most complex operator in Campaign 1 is a specialized 3-opt move that performs a cyclic exchange of single nodes between three distinct routes. While named k-opt for consistency in logging, its function is to execute a move where a node from route R_1 moves to R_2, a node from R_2 moves to R_3, and a node from R_3 moves to R_1. To manage the combinatorial complexity, candidate nodes for the cycle are drawn exclusively from a pre-filtered list of high-contribution nodes. This operator is capable of resolving complex, interdependent route configurations that cannot be fixed by simpler relocate or swap moves. Its tabu tenure is temporarily shortened to encourage more exploration with this powerful operator. Failure to find an improving cycle concludes Campaign 1, signaling that the solution is in a local minimum with respect to this set of operators.

C.2.2 Polar Geometric Tuning - Campaign 2

When the systematic local search of Campaign 1 stalls, the heuristic escalates to Campaign 2, a diversification strategy designed to escape the current local optimum by fundamentally altering the geometric structure of the routes. This campaign reframes the solution in a polar coordinate space relative to the depot and executes two sequential phases to improve the global shape of the routes.

Phase 1: Angular Optimization The first phase targets the angular partitioning of customers to resolve route crossovers and create more logical, wedge-shaped sectors. All customers are sorted globally by their polar angle θ. The algorithm then iterates through this sorted list, identifying adjacent customers that belong to different vehicles. Such a pair defines a geometric "border" where two routes interleave. A small "border window" of neighboring unlocked nodes is constructed around this pair. The algorithm then performs an exhaustive search within this window, evaluating every possible reassignment of its nodes between the two routes. This is a powerful geometric operator for untwisting overlapping routes. If an improvement is found, the phase restarts to operate on the new, improved solution structure.

Phase 2: Radial Optimization If the angular phase fails to find an improvement, the radial optimization phase begins. This phase aims to improve route compactness and reduce overall travel distance by targeting geographic outliers. First, the geometric centroid is calculated for each route. For a given route, its "bad" nodes are identified as the unlocked customers located farthest from their own route's centroid. The operator then attempts to swap these outlier nodes with nodes from neighboring routes. An ideal swap partner is a node that, while in a different route, is geographically closer to the outlier's original centroid. This targeted swapping systematically pulls distant customers into more appropriate routes, reducing the radial spread and improving the overall clustering quality of the solution. If any swap succeeds, control returns to the main heuristic manager to re-engage the intensive local search of Campaign 1.

C.2.3 Granular Border Refinement - Campaign 3

Campaign 3 is engaged when the broader, shape-based tuning of Campaign 2 fails. This campaign implements a strategy analogous to Granular Search, executing a computationally intensive, fine-grained search focused exclusively on the border regions between adjacent routes (Toth and Vigo 2003). The core strategy is to intelligently construct a small "border window" of candidate nodes that lie on the boundary between a pair of routes and then apply powerful improvement operators only within this constrained set.

The construction of the border window is a multi-faceted process designed to identify the most promising nodes for exchange. For a given pair of routes, a node is considered a candidate if it meets at least one of three geometric criteria inspired by computational geometry:

- **Convex Hull Membership:** The node lies on the convex hull of its own route's customer set, indicating it is an external-facing point.
- **Voronoi Proximity:** The node is located near the Voronoi boundary, meaning it is approximately equidistant from the two competing route centroids, making its route assignment inherently ambiguous.
- **k-Nearest Neighbor Proximity:** The node is one of the k-nearest neighbors to a customer in the opposing route, indicating a strong local interaction across the route boundary.

Nodes satisfying these criteria form a high-potential candidate set, which is then filtered and expanded to create a final, size-limited window for intensive optimization.

Once the border window is defined, the campaign applies two local search operators in a strict, sequential "first-improvement" order. The primary operator is a 1-for-1 swap, which exhaustively evaluates all pairs of nodes within the window that belong to different routes. If no improving swap is found, the secondary operator—a relocate (1-opt) move—is attempted. By restricting its search to these high-potential border regions and prioritizing balanced swap moves over relocates, Campaign 3 discovers complex, localized improvements missed by the preceding campaigns.

C.2.4 Global Search and Perturbation - Campaign 4

Campaign 4 serves as the final and most powerful stage of the heuristic, combining principles from Iterated Local Search (ILS) and adaptive metaheuristics (Lourenço, Martin, and Stützle 2003). It is activated only when all prior campaigns are exhausted, indicating that the search is trapped in a deep local optimum. Its purpose is to enable a global search capability and provide a mechanism for escaping the current basin of attraction.

Prioritized Global Search The campaign's primary operator is a global 3-route cyclic exchange, identical in mechanics to the one in Campaign 1. However, instead of a localized search, it considers combinations of three customers selected from a global priority list. This list is dynamically generated by scoring all customers based on a weighted sum of their marginal cost contribution and their Euclidean distance from the solution's global centroid. This focuses the computationally expensive 3-opt search on customers that are both costly to serve and structurally significant (i.e., peripheral nodes).

Heat-Based Perturbation Campaign 4 also incorporates an adaptive memory mechanism to trigger a large-scale perturbation. A "heat" metric accumulates each time a non-improving move is evaluated by the 3-opt operator, with the amount of heat added proportional to the move's positive (worsening) delta. This concept of using a metric to control the acceptance of non-improving or diversifying moves is conceptually related to the temperature parameter in Simulated Annealing (Kirkpatrick, Gelatt Jr, and Vecchi 1983). As heat builds, a logistic function increases the probability of triggering a perturbation. When triggered, the heuristic employs a *Ruin and Recreate* strategy, a cornerstone of Large Neighborhood Search (LNS) (Shaw 1998; Pisinger and Ropke 2007). A percentage of the solution's nodes are removed (ruin) and then reinserted using a fast, greedy insertion heuristic (recreate). This drastically restructures the solution, allowing the entire four-campaign search to restart from a new, potentially more promising region of the solution space.

APPENDIX D
Hardware and Dependencies

All computational experiments presented in this work were performed using commercially available hardware and open-source software. This appendix details the computing environment and key software dependencies to ensure reproducibility. All code, together with datasets and results, can be found on GitHub.

D.1 Hardware

The experiments were conducted on a Gigabyte Aero 16 notebook computer (2022 model) running Microsoft Windows 11. The system was equipped with an Intel Core i7-12700H processor (12th Generation Alder Lake-H), which provides 14 physical cores (6 performance cores and 8 efficiency cores) and 20 hardware threads. The machine was custom-modified to include 64 GB of DDR5 RAM, exceeding the standard retail configuration. This high memory capacity was critical for handling large-scale VRP datasets and parallelized benchmarking experiments.

To accelerate the computational studies, experiments were heavily parallelized. Python scripts, particularly those for benchmarking and solving large sets of VRP instances, were designed to leverage the `concurrent.futures` library. This allowed multiple independent problem instances to be processed simultaneously, with each instance assigned to a dedicated thread. This design maximized utilization of the 20-thread architecture and significantly reduced the total time required to complete the large-scale computational studies presented in Chapters 4 and 5.

D.2 Dependencies

The primary solution framework was developed in Python 3.11.9. The choice of the LKH-3 solver for computing ground-truth TSP costs was motivated by its state-of-the-art performance and straightforward compatibility with the Windows environment via a pre-compiled C executable (Helsgaun 2017).

A key aspect of the experimental timing methodology was the separation of heuristic search time from final solution evaluation time. The main heuristic algorithms (e.g., the Multi-Campaign Search) were run under a strict time limit, and their runtime was recorded. The calculation of $f(s)$ for the best-found solution—a process that involves calling the LKH-3 solver for each route—was performed as a separate, post-processing step. This ensures that the reported heuristic performance metrics, particularly runtime, purely reflect the efficiency of the estimator-guided search, without being conflated with the time required to compute the ground-truth benchmark cost.