AIMS Mathematics, 11(9): 28864–28886.
DOI: 10.3934/math.20261149
Received: 25 June 2026
Revised: 24 August 2026
Accepted: 28 August 2026
Published: 09 September 2026

https://www.aimspress.com/journal/Math

Research article

Lightweight normalization-free GNN-guided local search for logistics routing optimization

Boen Lin, Jing Sun*, and Yinlong Liu*

Faculty of Data Science , City University of Macau, Avenida Padre Tomas Pereira, Taipa, Macao SAR 999078, China

* **Correspondence:** Email: krystalsun@cityu.edu.mo, ylliu@cityu.edu.mo.

Abstract: Routing optimization has grown progressively pivotal in the logistics domain. Recently, learning-based approximate algorithms have emerged as a promising paradigm for solving routing problems. However, they suffer from substantial computational overhead from normalization layers hindering inference on resource-constrained edge devices, and poor cross-scale generalization under fluctuating routing node counts. To address these issues, we propose a novel lightweight normalization-free graph neural network guided local search framework, termed GNN-GLS-DyT, for efficiently solving vehicle routing problems. Specifically, a normalization-free regret approximation module is devised to estimate the regret value of including each edge in the final solution path. All batch normalization layers in the module are replaced by a lightweight dynamic tanh (DyT) operator, which drastically reduces computational complexity while preserving full representational capacity. Taking the predicted edge regret values and original routing graph as adaptive heuristic cues, we embed the proposed graph neural network into the guided local search (GLS) paradigm. This hybrid design bridges deep learning generalization with classical heuristic search capability, enabling efficient discovery of high-quality routing solutions. Extensive experiments on the vehicle routing problems demonstrate that our method consistently outperforms state-of-the-art baselines in overall solution quality, and achieves markedly faster convergence on complex vehicle routing problem variants. Most notably, the framework exhibits superior cross-scale generalization, which trained exclusively on small-scale instances, it generalizes seamlessly to much larger routing problems, validating its strong practical potential for large-scale real-world logistics systems.

Keywords: optimization algorithm; local search; transformer; graph neural network; vehicle routing problem (VRP)
Mathematics Subject Classification: 34A08, 68T07

1. Introduction

Recently, routing optimization problems play a crucial role in intelligent transportation systems, such as industrial warehousing [1], package delivery [2], and demand delivery [3]. For instance, according to a survey in Singapore, over 30% of consumers responded that they made online purchases several times a month. The delivery market and its associated industries contribute to 1% of the nation's gross domestic product with around 7% of the domestic employment [4]. As an representative and realistic problem in the logistic systems, vehicle routing problem (VRP) have practical applications to processes in transportation, logistics, and automation [5, 6]. The theoretical importance of vehicle routing problem extends beyond its computational complexity classification; it continues to stimulate fundamental research in polyhedral combinatorics, integer programming, and probabilistic analysis of algorithms. Conceptually, VRP describes a fleet of vehicles departing from a shared depot to serve geographically scattered customers with specified commodity demands, subject to each vehicle's maximum loading capacity constraint before returning to the depot [6].

The VRP is widely acknowledged to be NP-hard [7], rendering the exact solution of large-scale instances computationally prohibitive. Consequently, the problem serves as a canonical benchmark for evaluating algorithmic paradigms, including exact methods, approximation algorithms, and metaheuristic approaches [5, 6]. One of the earliest landmark solutions employed linear programming techniques to solve a 49-city instance [8]. While exact algorithms provide provable optimality guarantees in theory, they rapidly become computationally intractable as problem scales grow. While heuristic and meta-heuristic methods deliver better scalability, they suffer from slow convergence and depend heavily on tedious, problem-specific handcrafted rules, leaving considerable room for performance improvement.

In recent years, deep reinforcement learning (DRL) has emerged as a promising alternative, enabling end-to-end training without manual heuristics. Bello et al. [9] pioneered neural combinatorial optimization by utilizing Pointer Networks with policy gradient methods to directly learn constructive heuristics. Kool et al. [10] proposed an attention model (AM) employing an encoder-decoder architecture that encodes node features via self-attention and decodes tours sequentially. Kwon et al. [11] designed the POMO framework, which samples multiple solutions in parallel through a shared Transformer encoder and enhances performance via policy optimization. Xin et al. [12] introduced a multi-decoder attention model employing multiple decoder heads to augment exploration of the solution space. Fu et al. [13] constructed a hierarchical Transformer that recursively decomposes large-scale traveling salesman problem into subproblems for scale generalization. DRL alone is not sufficient to solve combinatorial optimization problems; instead, it can provide auxiliary information to augment traditional optimization algorithms. In this paradigm, a master algorithm controls the high-level procedure while repeatedly calling a machine learning model to make lower level decisions. To combine the advantages of DRL and traditional metaheuristics, hybrid frameworks that use graph neural network to guide local search have been proposed [14–17]. Peng [14] integrated residual gated graph convolutional networks with transformers to explicitly model topological graph information. Hudson et al [15] use a combination of GNN and transformer modules, and achieve better results. This and follow-up works have cemented learning-based approximate algorithms as a viable, competitive paradigm for logistics routing, unlocking data-driven adaptability that traditional handcrafted heuristics cannot readily achieve. Despite these advances in

solution quality, existing approximate algorithms for logistics routing problems suffer from high computational overhead in the normalization layer of the Transformer, which limits their deployment on edge devices; poor generalization across different problem scales when the number of logistics nodes fluctuates; and slow convergence, rendering them inadequate for real-time scheduling demands in logistics systems.

To address the aforementioned limitations, this paper proposes a novel framework, GNN-GLS-DyT, which seamlessly integrates lightweight normalization-free GNN with guided local search (GLS) to tackle practical routing problems in complex real-world logistics environments. Specifically, we develop a normalization-free based regret approximation module to predict the edge inclusion regret of optimal routing solutions. To enhance computational efficiency, all batch normalization layers in the transformer GNN encoder are replaced with the lightweight dynamic tanh (DyT) operator [18]. This design preserves the essential nonlinear squashing property critical for stable training on line graph representations of routing problems, while simultaneously reducing memory footprint and eliminating the quadratic scaling overhead inherent to traditional normalization operations. The proposed framework empirically demonstrates that normalization-free architectures can effectively capture the combinatorial structure of routing problems, achieving superior cross-scale generalization across different problem scales over existing state-of-the-art methods. Our contributions are summarized as follows:

- We propose a novel lightweight graph neural network GNN-guided local search approach with a normalization-free Transformer backbone. Extensive experiments on the TSP and capacitated vehicle routing problem (CVRP) are carried out to demonstrate the effectiveness of our method.
- The model is designed for the geometric characteristics of logistics graphs, and the line graph-based GNN encoder effectively models edge features. All batch normalization layers in the GNN encoder are replaced with DyT to reduce computational and memory overhead, making the model suitable for real-time inference on resource-constrained logistics scheduling terminals in logistic system.
- The proposed GNN-GLS-DyT framework integrates the powerful feature representation of normalization-free GNNs into GLS, organically merging representation learning with heuristic optimization. Benefiting from such hybrid design, our method yields prominent cross-scale generalization. Empirical experiments verify that GNN-GLS-DyT surpasses existing state-of-the-art approaches on diverse complex VRP variants in comprehensive optimization performance while attaining distinctly faster convergence toward near-optimal solutions. More crucially, the framework possesses remarkable transferability. The model trained on small-scale VRP instances can generalize to efficiently solve larger-scale tasks, which validates its practical deployment value for real-world large-scale logistics routing.

This paper is organized as follows. Section 2 provides literature on neural combinatorial optimization for the TSP, including the learning-based and hybrid metaheuristic frameworks. In Section 3, we formalize the problem and detail the overall framework. Section 4 delineates our proposed methodology. Section 5 presents extensive empirical validation that substantiates our approach's superiority in solution quality, convergence acceleration, and robust generalization relative to existing transformer-based solvers. Finally, Section 6 concludes the paper and presents future works.

2. Related work

This section presents a systematic review of three key research streams directly pertinent to our work, with a particular focus on their practical applicability and inherent limitations when applied to intelligent transportation system (ITS)-enabled logistics routing problems. We structure our discussion along three primary research directions: (1) classical methods for VRPs; (2) DRL-based approaches for vehicle routing; (3) GNN-guided local search for the routing optimization.

Classic methods for VRPs. The VRP is a canonical combinatorial optimization task that seeks a minimum-length Hamiltonian cycle over a complete weighted graph. The exact methods were put forward for solving VRP [19, 20]. However, trying to generate exact optimality suffers from heavy computation given the NP-hard complexity. For TSP, Concorde [21] serves as the state-of-the-art exact solver. For CVRP, LKH-3 [22] is widely recognized as the gold-standard heuristic solver and is used as the baseline for optimality gap calculation in this work. Furthermore, these exact solvers exhibit poor extensibility when practical side constraints, such as, time windows, vehicle capacity, pickup-and-delivery requirements are introduced, which severely limits the flexibility of practitioners in adapting to diverse real-world logistics scenarios. Conversely, while heuristic methods can produce high-quality near-optimal solutions in a computationally efficient manner, they are inherently dependent on hand-engineered rules and their performance is heavily bounded by the domain expertise of human designers.

DRL-based approaches for vehicle routing. To circumvent the brittleness of hand-crafted rules, a growing body of work casts routing problems as sequential decision-making tasks and solves them with DRL. Pointer networks [23] first adopted an LSTM-based sequence-to-sequence model trained with supervised labels from Concorde. Subsequent methods replaced the recurrent core with the transformer [24] architecture, yielding stronger empirical performance on TSP and the capacitated vehicle routing problem (CVRP) [10, 11]. A common backbone of these DRL solvers is a multi-head attention encoder that repeatedly applies *layer normalization* (LN) inside residual blocks; LN is known to stabilize training, accelerate convergence, and improve zero-shot generalization [25]. However, the heavy memory footprint and sequential nature of attention layers become bottlenecks when the number of nodes scales, and the empirical gain of LN diminishes when the training data distribution shifts [26].

GNN-guided local search for the routing optimization. Instead of learning a construction policy, a complementary paradigm employs a graph neural network (GNN) to guide a traditional local-search meta-heuristic. Hudson et al. recently proposed to learn the global regret of each edge via a GNN operating on the line graph and feed the predicted regret to GLS [27]. Their hybrid approach achieves state-of-the-art anytime performance, but still inherits the Transformer-style normalization layers inside the GNN, incurring non-negligible overhead on GPU memory and latency.

Normalization-free deep learning is a new research direction that challenges the indispensability of normalization layers [18]. Recent works [18] demonstrate that DyT can effectively substitute standard normalization in Transformers, achieving comparable performance with lower computational overhead.

While the lightweight DyT architecture is inherently well-suited for inference on resource-constrained edge devices, its potential within graph neural network-guided local search (GNN-GLS) paradigms for logistics routing optimization remains a largely untapped research avenue. To bridge this research gap, we pioneer the integration of the DyT design into the GNN-GLS

framework, constructing a normalization-free hybrid solver that simultaneously addresses the three core limitations of prevailing learning-based routing methods, which are prohibitive computational overhead, weak cross-scale generalization, and sluggish convergence. In the following, to establish a solid foundation for the detailed exposition of our proposed methodology, we first formalize the specific vehicle routing problem formulation addressed in this work and introduce key algorithmic concepts in the following preliminary section.

3. Preliminary

Table 1. Notations of the problem formulation and algorithm.

Notation	Description	Notation	Description
$\mathcal{G}$	The graph representation	$\mathcal{V}$	The set of ports $\{v_0, v_1, \ldots, v_n\}$
$\mathcal{E}$	The set of edges and arcs	d_i	The non-negative demand of port v_i
l_i	The maximum draft of port v_i	p_i	The profit obtained from visiting node i
Q	The maximum vessel capacity	x_{ij}	Binary variable (1 if arc (i, j) is used)
z_i	Binary variable (1 if port i is visited)	y_{ij}	amount of vehicle flow from node i to j
α_i	The current cumulative load at port v_i	c_{ij}	The cost associated with edge $\{i, j\}$
n_i	The 2D location vector of node i	$D(x_i, x_j)$	The distance between node i and node j
s_t	The state at time step t	a_t	The action at time step t
π_θ	The policy parameterized by θ	r	The reward function
R	The total cumulative reward function	τ	The state transition rule
γ	The discount factor	B	The batch size for training
T	The total number of steps/episodes	$h_i^{(l)}$	The node embeddings in layer l
$Z_{l,y}$	The attention value for head y in layer l	W_l^Q, W_l^K, W_l^V	Trainable weight matrices in layer l
W_l^O	Trainable output projection matrix	M	Masking vector for invalid nodes
L, H	Number of layers and attention heads	η	The learning rate for optimization

In this section, we formalize the specific VRP formulation adopted in this study and elaborate on the foundational concepts of regret, GLS, and the DyT operator. All definitions are contextualized within ITS-based logistics routing scenarios to ensure their direct practical relevance to our research problem.

3.1. Vehicle routing problem

A CVRP instance with n customer nodes is defined on a complete undirected weighted graph $G = (V, E)$ with $|V| = n + 1$ nodes. The vertex set $V = \{0, 1, \ldots, n\}$ consists of one central depot node (indexed by 0) and n geographically scattered customer nodes. The edge set $E = \{(i, j) \mid 0 \leq i < j \leq n\}$ connects all node pairs, with each edge weighted by the Euclidean travel distance between its two endpoints. Each customer node $i \in V \setminus \{0\}$ is associated with a non-negative commodity demand d_i; a homogeneous fleet of vehicles, each with a maximum load capacity Q, are dispatched from the depot to fulfill customer demands.

A feasible CVRP solution is a set of vehicle routes satisfying three core constraints, including (1) every customer node is visited exactly once by exactly one vehicle, (2) the total demand of customers on any single route does not exceed the vehicle capacity Q, and (3) all routes depart from the depot and return to the depot after service. The core optimization objective is to find the feasible solution with the minimum total travel distance across all vehicles.

The weight of each edge (i.e., the travel distance between nodes i and j) is calculated as:

$$w_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2, \quad (i, j) \in E \tag{3.1}$$

where $\mathbf{x}_i, \mathbf{x}_j$ denote the 2D coordinate vectors of nodes i and j, respectively.

3.2. Regret

Regret quantifies the potential future cost of an action that offers an immediate reward, serving as a widely used mechanism to mitigate the myopia inherent in greedy constructive procedures. Instead of optimizing solely for instantaneous gain, the algorithm evaluates the marginal loss incurred over a finite lookahead horizon by comparing the optimal insertion against its best alternative [28, 29]. Since exhaustively enumerating all downstream sequences is computationally prohibitive, this horizon is restricted to a small, fixed number of steps. Theoretically, were it feasible to compute regret over an unbounded horizon, a purely greedy strategy that consistently selects the minimum-regret alternative would inherently yield an optimal solution.

3.3. GLS

GLS is a metaheuristic strategy designed to assist local search algorithms in escaping local optima by dynamically modifying the objective function [30]. As illustrated in Figure 1, the mechanism operates by altering the search landscape based on specific solution features. The standard local search process (Figure 1, left) iteratively explores the neighborhood of a current solution s to minimize an objective function $g(s)$. This greedy approach inevitably leads to a local optimum s^*, where no neighbor offers a better objective value. To overcome this stagnation, GLS employs a penalty-based scheme. Let F be a set of features defining the solution space (e.g., edges in a routing problem [27, 31]). Each feature $i \in F$ is associated with a penalty counter p_i, initially set to zero. When the local search settles at a local optimum s^*, GLS identifies the features present in s^* that contribute most to the cost and increments their respective penalty counters. The search then continues by minimizing an augmented objective function $h(s)$, defined as:

$$h(s) = g(s) + \lambda \sum_{i \in F} p_i \cdot I_i(s), \tag{3.2}$$

where $g(s)$ is the original cost, λ is a regularization parameter controlling the weight of penalties, and $I_i(s)$ is an indicator function that equals 1 if feature i is present in solution s, and 0 otherwise. By increasing the augmented cost of the current local optimum (Figure 1, middle), the algorithm effectively fills the "basin of attraction" of the local minimum [30]. This deformation of the landscape renders the current solution less attractive and forces the search agent to move toward new, unexplored regions of the solution space (Figure 1, right) in pursuit of the global optimum.

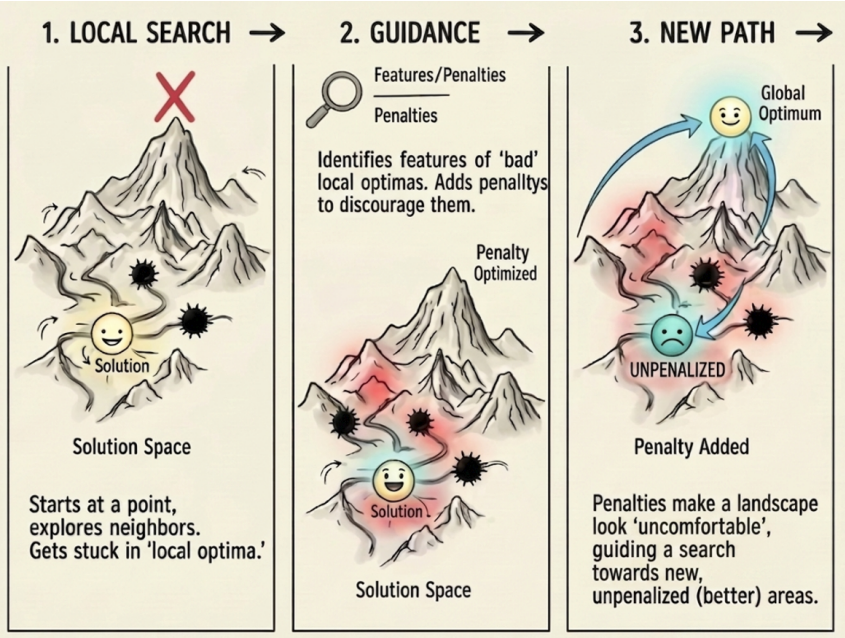


Figure 1. Schematic illustration of the GLS mechanism.

3.4. Transformer

Transformer is a purely attention-based sequence model that replaces recurrence with multi-head self-attention and positional encodings, enabling parallel training and capturing long-range dependencies in $O(n^2)$ time. The Transformer encodes an input sequence $\mathbf{X} \in \mathbb{R}^{n\times d}$ by first adding sinusoidal positional encoding to its token embeddings, then passing the sum through a stack of identical encoder layers; each layer first computes multi-head self-attention that jointly weighs all positions via scaled dot-products, next applies a residual connection followed by layer normalization, subsequently feeds the result into a position-wise two-layer feed-forward network with ReLU activation and another residual-plus-norm block, and finally produces a context-aware representation $\mathbf{H}$ that the decoder—structured analogously but inserting masked self-attention and encoder-decoder cross attention—uses to autoregressively predict the output sequence one token at a time by linearly projecting the last decoder state and applying a softmax distribution [24].

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}. \tag{3.3}$$

This equation performs scaled dot-product attention: similarities between queries $\mathbf{Q}$ and keys $\mathbf{K}$ are scaled by $1/\sqrt{d_k}$, normalized with softmax, and used to weight the values $\mathbf{V}$, producing the context-aware output.

Although normalization layers have long been regarded as indispensable components in deep learning architectures, their foundational role is being challenged by normalization-free frameworks. Analysis reveals that a key function of layers like L is to induce tanh-like, S-shaped activation mappings that scale inputs and suppress outliers. Building on this insight, recent work demonstrates that simpler, learnable alternatives, such as DyT, can effectively substitute standard normalization in models like transformers, achieving comparable performance [18]. The layer, termed DyT, is a

lightweight, element-wise operation that can serve as a drop-in replacement for normalization layers, with no need for running statistics, mean-centering, or variance normalization. It re-scales the input tensor element-wise through a learnable scalar α, compresses extreme values with the tanh non-linearity, and finally applies a per-channel affine transformation:

$$\mathrm{DyT}(\mathbf{x}) = \boldsymbol{\gamma} \odot \tanh(\alpha\, \mathbf{x}) + \boldsymbol{\beta}, \tag{3.4}$$

where $\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^C$ are learnable gain and bias vectors and $\odot$ denotes channel-wise multiplication. DyT emulates the S-shaped input-output mapping of normalization layers, enabling stable optimization of deep models with lower computational and memory overhead—this is the core lightweight design for our model for real-time logistics routing.

4. Method

This section illustrates the proposed framework for the routing problem, which is a lightweight, normalization-free hybrid framework combining a GNN encoder for edge regret estimation and a GLS for efficient logistics routing search. We first present the overall framework, then detail the normalization-free GNN encoder for regret estimation, and finally introduce the GLS framework guided by predicted regret for logistics routing optimization.

As depicted in Figure 2, our GNN-based model learns an approximation of the global regret associated with including each edge of the problem graph in the solution. To improve the training efficiency and stability, we adopt the normalization-free mechanism with DyT operator in the approximation process. The metaheuristic, GLS, utilizes this learned regret in conjunction with the original problem graph to rapidly identify high-quality solutions.

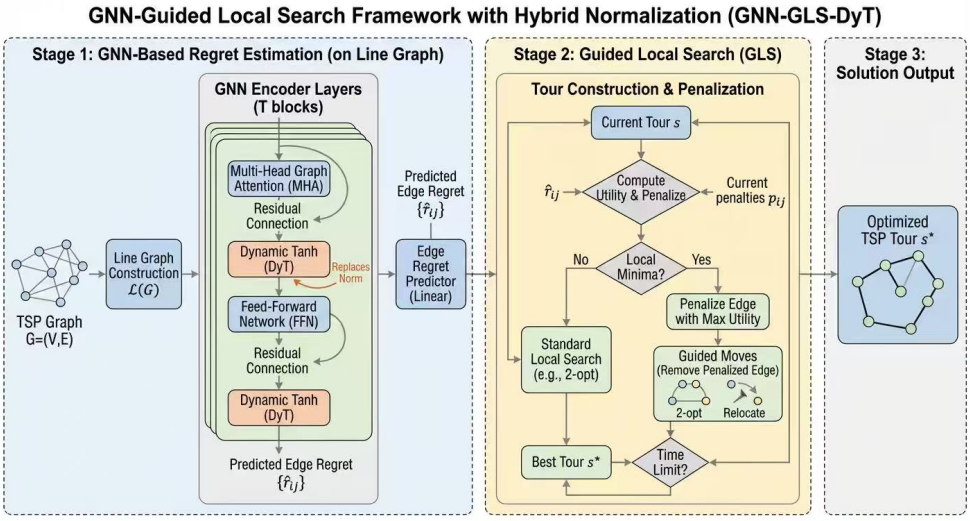


Figure 2. Overview of the proposed GNN-GLS-DyT framework. Stage 1: the line-graph GNN encoder with DyT units predicts edge regret; Stage 2: the predicted regret drives GLS to iteratively penalize high-utility edges; Stage 3: the best tour found is returned once the time budget is reached.

In the model of the edge-side regret estimation, the line-graph GNN encoder with DyT units takes

the routing graph $G = (V, E)$ as input, and outputs the predicted regret $\hat{r}_{ij}$ for each edge. The line graph $L(G)$ maps each edge of G to a node, which effectively models the edge features. The learned regret values effectively distinguish between edges with high and low inclusion costs, providing informative heuristic guidance for the underlying local search procedure. This guidance significantly enhances the algorithm's capability to escape local minima and direct the search process toward more promising regions of the solution space. Furthermore, the GLS model in Stage 2 leverages the predicted regret values to guide the GLS module, which iteratively computes edge utility, penalizes high-utility edges, and performs standard local search operations to refine the routing solution. Finally, Stage 3 outputs the best feasible routing solution once the predefined time budget is exhausted. In the following subsections, we detail the core components of our framework, including the normalization-free GNN regret estimator and the regret-guided GLS optimization module.

4.1. Global-regret target with Nnormalization-free mechanism

We define the global regret of a decision as the additional cost incurred by forcing that decision into the solution, relative to a globally optimal solution. Consequently, decisions that belong to an optimal solution have zero regret, while all other decisions yield positive regret. The global regret of fixing an edge (i, j) to be inside the tour is:

$$r_{ij} = \frac{g(s^*_{ij})}{g(s^*)} - 1, \tag{4.1}$$

where s^* is an optimal tour of the instance and s^*_{ij} is an optimal tour constrained to contain edge (i, j). Because computing (4.1) exactly for every edge is intractable, we use GNNs to achieve this, as they are universal function approximators that operate on graph-structured data.

Our model takes the line graph of the original problem graph as input. For an undirected graph G, its line graph $L(G)$ maps each edge of G to a node; edges in $L(G)$ connect nodes corresponding to edges in G that share a common node. This line graph representation enables direct message aggregation and state representation on the edges of the original graph, which is a critical advantage for routing optimization problems. Specifically, since the optimal routing solution is intrinsically a subset of edges and our core task is to predict the inclusion regret of each edge, operating on the line graph aligns perfectly with our edge-level prediction objective, avoiding the complexity of modeling edge features indirectly through node representations. Building on this line graph representation, our model consists of an embedding layer, several GNN layers, and an output layer. The embedding layer is an edge-wise fully connected layer that maps d_X-dimensional edge features to d_h-dimensional embeddings.

The raw input edge features are constructed directly from spatial node coordinates and Euclidean distance metrics. During line-graph message passing, graph attention layers aggregate information across neighboring edges via weighted summation. Because edge geometry, node distributions, and local graph connectivities vary substantially across instances, the resulting pre-activation values naturally exhibit highly heterogeneous scales and wide dynamic ranges.

While batch normalization is commonly adopted in deep GNNs to stabilize gradient flow and accelerate training, it attempts to control scaling variations by estimating mini-batch statistics, which introduces strict batch-wise dependencies that fail under scale shift. This flaw becomes critical in routing tasks: when a model trained on small synthetic graphs is transferred to larger TSPLIB instances, the edge feature distributions and graph density undergo a severe domain shift, rendering the running statistics accumulated during training mismatched to the target scale and causing

significant performance degradation. To address this, we replace the BatchNorm layers with the lightweight DyT operator, converting our GNN-GLS-DyT encoder into a strictly normalization-free architecture. DyT completely eliminates reliance on batch statistics, making model inference entirely independent of graph size and batching configurations. In our routing setting, the input edge features are constructed from Euclidean distances and node coordinates, and the graph attention layers aggregate heterogeneous neighborhood information via weighted summation. As a result, the activations naturally exhibit heterogeneous scales due to variations in edge geometry and graph connectivity. DyT provides a lightweight S-shaped nonlinear transformation that suppresses excessively large activations while maintaining the expressive capability required for edge representation learning, making it a well-justified normalization-free alternative for graph attention networks. Formally, the 2D coordinates of the two vertices connected by each edge are concatenated to the corresponding raw edge feature vector prior to embedding. Its forward propagation process is formalized as follows:

$$h_{ij}^0 = W_{x_{ij}} + b, \tag{4.2}$$

where h_{ij}^0 is the initial embedding of edge ij, W is a learnable weight matrix, x_{ij} are the input features of edge ij (including any node features), and b is a set of learnable biases. We use $d_x = 1$ and $d_h = 128$. The edge embeddings are updated using T-layer message-passing architecture. To improve the training efficiency and stability, we adopt the normalization-free mechanism with DyT operator. The update rule for an edge embedding $\mathbf{h}_{ij}^t \in \mathbb{R}^{d_h}$ at layer $t+1$ is

$$\tilde{\mathbf{h}}_{ij}^{t+1} = \mathrm{DyT}\Big(\mathbf{h}_{ij}^t + \mathrm{MHA}^t(\mathbf{h}_{ij}^t, L(G))\Big), \tag{4.3}$$

$$\mathbf{h}_{ij}^{t+1} = \mathrm{DyT}\Big(\tilde{\mathbf{h}}_{ij}^{t+1} + \mathrm{FFN}^t(\tilde{\mathbf{h}}_{ij}^{t+1})\Big), \tag{4.4}$$

where $\alpha \in \mathbb{R}$ is a *learnable* scalar gain, and $\gamma, \beta \in \mathbb{R}^C$ are learnable affine parameters. MHA denotes an 8-head Graph Attention layer, FFN a two-layer MLP with ReLU, and normalization-free mechanism. After T layers a single edge-wise feed-forward layer outputs

$$\hat{r}_{ij} = \mathbf{W}\mathbf{h}_{ij}^T + b. \tag{4.5}$$

Equations (4.3) and (4.4) is applied element-wise, requires no running statistics, and has the same functional signature as the original normalization layers. For the final DyT layer that immediately precedes the regret predictor (4.5), we keep the original affine parameters of the removed BN layer; their initial values are inherited from the pre-trained GNN–GLS checkpoint and remain trainable. The model is trained by minimizing the squared-error loss

$$\mathcal{L}(\theta) = \sum_{(i,j)\in E} (\hat{r}_{ij} - r_{ij})^2. \tag{4.6}$$

This supervised training objective enables our normalization-free GNN encoder to learn accurate edge inclusion regret predictions, which serve as adaptive heuristic cues to guide the subsequent GLS module for high-quality routing solution generation.

4.2. GNN-guided GLS for routing optimization

GLS serves as a high-level metaheuristic that dynamically deforms the search landscape to help local search procedures escape from sub-optimal basins of attraction [30]. Traditionally, GLS relies

on static problem features; however, recent advancements integrate GNNs to provide edge-wise regret predictions $\hat{r}_{ij}$, which act as a learned prior for the search process [15]. In our proposed GNN-GLS-DyT framework, the standard routing cost function $g(s)$ is augmented with penalty terms weighted by the GNN-predicted edge regrets, resulting in the modified objective function $h(s)$ defined as

$$h(s) = g(s) + \lambda \sum_{(i,j)\in E} p_{ij} \cdot I_{ij}(s) \cdot \hat{r}_{ij}. \tag{4.7}$$

Here, $I_{ij}(s)$ is an indicator variable denoting the presence of edge (i, j) in the current tour s, while p_{ij} represents the cumulative penalty counter for that specific edge. The hyperparameter λ (commonly set to 0.01) scales the influence of the learned regret against the actual tour length. To ensure dimensional homogeneity in Eq (4.7), we note that the indicator variable $I_{ij}(s)$, the penalty counter p_{ij}, and the predicted edge regret $\hat{r}_{ij}$ (defined in Eq (4.1) as a normalized relative cost increase) are all dimensionless. Following the standard Guided Local Search paradigm, the penalty parameter λ carries the dimension of routing distance and is scaled proportionally to the average edge length of the instance. Consequently, both the base routing cost $g(s)$ and the weighted penalty term share identical physical dimensions, rendering the augmented objective $h(s)$ mathematically consistent. In this setup, the dimensionless regret $\hat{r}_{ij}$ serves as a scale-invariant dynamic prior that modulates penalty updates without altering the underlying physical units of the search space. The core of the GNN-GLS mechanism lies in its response to local optimality. When a local improvement phase (such as a greedy 2-opt) reaches a state s^* where no further improving moves exist under $g(s)$, the algorithm evaluates the *utility* of each edge in the current tour to identify the most suitable candidate for penalization:

$$\mathsf{util}_{ij} = I_{ij}(s)\frac{\hat{r}_{ij}}{1 + p_{ij}}. \tag{4.8}$$

This utility function explicitly balances two critical factors: the GNN's confidence in the edge's suboptimality, quantified by the predicted edge regret $\hat{r}_{ij}$, and the search history encoded by the penalty counter p_{ij}. This design prevents excessive penalization of the same edges and maintains a proper exploration-exploitation balance throughout the search process. After identifying the edge with the highest utility, the algorithm increments its corresponding penalty counter: $p_{ij} \leftarrow p_{ij} + 1$.

To maximize the efficiency of the landscape deformation, the algorithm then enters a restricted search phase. During this stage, the local search engine is restricted to accepting only those moves (typically 2-opt or relocate operations) that directly remove the newly penalized edge from the current tour. This focused perturbation continues until either $K = 20$ such moves are accepted or the move set is exhausted.

Subsequently, the algorithm reverts to a standard best-improvement local search governed by the original cost $g(s)$, starting from the newly discovered region of the solution space. This iterative cycle of GNN-guided penalization and targeted restricted exploration repeats until the pre-allocated computational time budget is exhausted, effectively harnessing the power of deep learning to navigate the complex combinatorial solution landscapes of routing problems. We select two lightweight local search operators for our framework, where the 2-opt operator swaps two non-adjacent edges to eliminate route crossings, which is particularly effective for urban delivery and port transportation scenarios; the relocate operator moves a single customer node to a different position in the tour, making it well-suited for dynamic logistics node adjustment tasks. Both operators exhibit extremely

low computational complexity, which aligns perfectly with the overall lightweight design philosophy of our GNN-GLS-DyT framework for large-scale routing optimization.

5. Experiment

In this section, we conduct extensive experiments on both randomly generated datasets and real-world datasets to answer the following research questions:

(1) RQ1: How does the proposed framework compare to baselines?
(2) RQ2: How well can our method generalize to different sizes?
(3) RQ3:How does the proposed framework generalize to real-world datasets?

The results addressing RQ1 are presented in Tables 2–3 and illustrated in Figure 3, while RQ2 and RQ3 are examined in Figures 4 and Tables 4–6, respectively.

5.1. Experimental settings

5.1.1. Training data setup

We trained the model on two-dimensional Euclidean TSP and CVRP instances with 20, 50, and 100 nodes, respectively, generating 100,000 samples for each scale. The node coordinates were uniformly sampled within the unit square $[0, 1]^2$. We calculate the target output, the regret of including each edge in the solution. We use Concorde to find an optimal solution and LKH-3 (100 trials, 10 runs) is adopted to compute the optimal solution for instances with a fixed edge. We empirically validate that this configuration for LKH-3 is sufficient to find optimal solutions for problems with 100 nodes for the CVRP. We scale the input features and target outputs to a range of [0, 1].

5.1.2. Network architecture and hyperparameters

Our model employs a graph neural network architecture comprising $T = 3$ message passing layers. Each layer utilizes multi-head attention with 8 heads and a hidden dimension of $d_h = 128$ (16 dimensions per head). The feed-forward network (FFN) following each attention block consists of a single hidden layer with 512 units and ReLU activation. The output layer is a single-neuron linear regression head. The model is trained by minimizing the ℓ_2 loss using the Adam optimizer with an initial learning rate of 10^{-3}, which decays by a factor of 0.99 after each epoch. Training employs a batch size of $B = 32$ for the 20 and 50-node instances, and $B = 15$ for the 100-node instances due to GPU memory constraints. An early stopping mechanism with a patience of 10 epochs is applied, with a maximum training duration of 100 epochs. The GLS procedure is configured as follows. The initial solution is constructed by a nearest neighbor greedy heuristic, in which edges are selected in ascending order of their predicted regret $\hat{r}_{ij}$. Local search alternates between the relocate and 2-opt operators under a best-improvement strategy. Each penalty phase executes $K = 20$ perturbation moves. The coefficient for the augmented objective is set to $\lambda = 0.01$. The complete procedure, including feature computation, model inference, initial construction, and the search, is allotted a runtime budget of 10 seconds per instance. All experiments are conducted on a workstation equipped with an Intel Core i9-10940X CPU and a single NVIDIA GeForce RTX 3090 Ti GPU. We will make all our codes public after the work is accepted.

5.1.3. Baselines methods

The performance of our proposed model is evaluated against both non-learning and learning-based baselines. For non-learning baselines, we select five representative algorithms: (1) nearest neighbor: a classic constructive heuristic that starts from a random depot node and sequentially visits the closest unvisited customer node until all nodes are included in the tour; (2) farthest insertion: an insertion-based heuristic that iteratively selects the farthest unvisited node and inserts it into the optimal position in the current partial tour, which generally produces more balanced and higher-quality solutions than the nearest neighbor heuristic; (3) standard local search: an iterative improvement heuristic that starts from an initial solution and repeatedly applies 2-opt edge swaps to eliminate route crossings and reduce total tour length until no further cost reduction can be achieved; (4) Concorde [32]: the state-of-the-art exact solver, which combines branch-and-bound with cutting-plane methods to provably find the globally optimal solution for small-to-medium scale instances; and (5) LKH-3 (Helsgaun, 2017): the latest implementation of the Lin-Kernighan heuristic, which is widely recognized as the most efficient heuristic solver for large-scale TSP instances and serves as the gold standard for heuristic performance comparison in the routing optimization community.

For learning-based baselines, we consider three widely recognized learning-based methods: (1) GNN-GLS [15]: the original GNN-GLS model with parameters identical to ours but lacking our proposed specialized normalization layer. (2) **The attention model (AM)** [10], as reproduced by the authors for a fair comparison, employs the same network architecture as POMO (N_{layer}=6, d_h=128, etc.) but follows the original training scheme: it uses REINFORCE with a greedy rollout baseline, samples a single trajectory per instance during training, and uses a larger batch size of 256. The learning rate is decayed by a factor of 0.1 once the performance plateaus. (3) **POMO** [11] adopts an attention model (AM) architecture with $N_{\text{layer}} = 6$ encoder layers, $M = 8$ attention heads, a hidden dimension $d_h = 128$, and a feed-forward dimension $d_f = 512$. During training, it generates N parallel rollouts (where N is the number of nodes) per instance and uses the average reward of these rollouts as a shared baseline. It is trained with the Adam optimizer using a learning rate of 10^{-4} and a batch size of 64. For inference, it adopts a multi-greedy strategy by starting from all N nodes and optionally augments each instance with $\times 8$ symmetric transformations, selecting the best from $8N$ candidate solutions.

To ensure a strictly fair comparison, the proposed model and all learning-based baselines are implemented in PyTorch and trained on the same datasets under identical hardware constraints. All baselines were re-implemented and re-trained from scratch using identical training datasets, preprocessing pipelines, and single-GPU hardware environments. Furthermore, for AM and POMO, all evaluations strictly employ standard single-pass greedy decoding, omitting test-time data augmentation and multi-start trajectory sampling. This controlled protocol measures the raw single-pass generalization capacity of constructive policies under equalized computational and time constraints.

5.2. Experiment results

5.2.1. Comparison results

We consolidate the numerical outcomes of TSP and CVRP in Tables 2 and 3, where the proposed GNN-GLS-DyT framework is compared with eight strong baselines of 20, 50 and 100-nodes. Table 2 evaluates all methods on synthetic, uniformly distributed Euclidean TSP instances that share the same node distribution as the training set but span three different scales. Under the in-distribution scale, models perform near-optimally; the table reports the average tour length, optimality gap relative to the Concorde exact solver, and average computation time for each method. The results demonstrate the consistent effectiveness of our proposed GNN-GLS-DyT framework across all problem scales, with significant performance improvements over the original GNN-GLS baseline that employs conventional normalization layers.

Table 2. Solution quality and computation time for various approaches for TSP.

	TSP20			TSP50			TSP100		
Method	Len	Gap (%)	Time	Len	Gap (%)	Time	Len	Gap (%)	Time
Concorde	3.831	0	1.5s	5.69	0	2.3s	7.716	0	5.2s
LKH3	3.833	0.05	1.2s	5.695	0.04	1.7s	7.872	2.02	3.7s
Local Search	3.896	1.70	2.3s	5.751	1.07	1.5s	7.979	3.41	4.5s
Farthest Insertion	3.917	2.24	5.2s	5.768	1.32	2.3s	7.976	3.36	3.7s
Nearest Neighbor	3.911	2.09	4.6s	5.789	1.69	2.7s	7.983	3.46	3.5s
AM	3.853	0.57	2.8s	5.781	1.55	6.4s	8.038	4.07	7.1s
POMO	3.844	0.12	5.3s	5.773	1.41	6.8s	8.094	4.89	8.9s
GNN-GLS	3.842	0.26	8.7s	5.767	1.30	10.1s	7.935	2.84	10.9s
GNN-GLS-DyT	**3.833**	**0.05**	6.8s	**5.749**	**1.00**	6.5s	**7.899**	**2.38**	7.6s

Table 3. Solution quality and computation time for various approaches on CVRP.

	CVRP20			CVRP50			CVRP100		
Method	Len	Gap (%)	Time	Len	Gap (%)	Time	Len	Gap (%)	Time
LKH3	3.533	0	**1.2s**	5.852	0	1.7s	8.672	0	1.8s
Local Search	3.979	12.62	0.1s	6.095	4.15	1.5s	9.002	3.81	0.5s
Farthest Insertion	3.818	8.07	0.2s	6.076	3.83	0.2s	9.005	3.84	0.7s
Nearest Neighbor	3.851	9.00	0.1s	6.489	10.89	0.2s	9.082	4.73	0.5s
AM	3.853	9.06	1.8s	6.024	2.93	2.4s	9.091	4.83	3.1s
POMO	3.834	8.52	1.3s	5.971	2.03	2.8s	9.194	6.02	3.9s
GNN-GLS	3.911	10.70	1.2s	6.113	4.46	2.1s	9.112	5.07	4.7s
GNN-GLS-DyT	**3.694**	**4.56**	1.1s	**5.977**	**2.14**	1.3s	**8.774**	**1.18**	3.1s

On TSP instances, the performance of GNN-GLS-DyT is on par with the state-of-the-art classical heuristic LKH3 and closely approximates the global optimum obtained by Concorde, achieving an

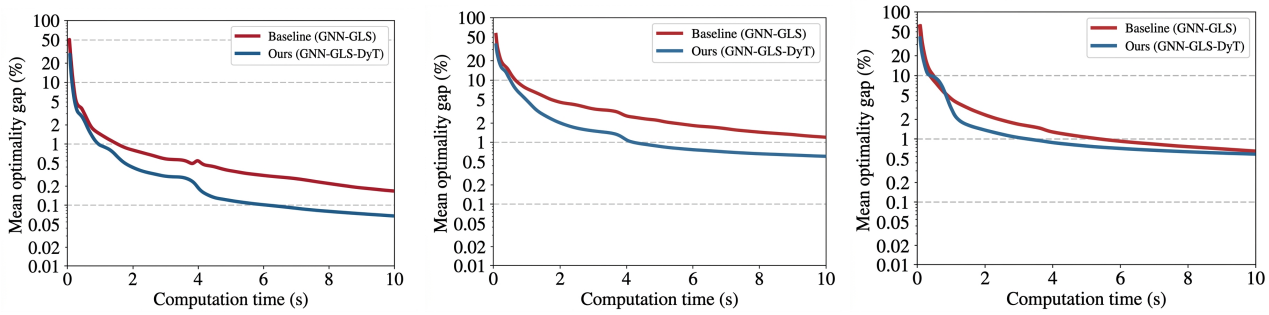
average tour length of 3.833 with an optimality gap of merely 0.05%. Since the compared heuristic algorithms involve random components, we further examine their solution stability by repeating multiple independent runs under the same standardized inference budget. We find that these heuristics exhibit not only larger mean optimality gaps but also significantly higher standard deviations than our proposed method. For example, on the CVRP20 benchmark, the standard deviation of nearest neighbor reaches 11. 209, while our method maintains a standard deviation of only 0.121 with a better mean gap. This variance confirms that the heuristics frequently produce unstable solutions under tight time budgets, whereas our model, benefiting from robust regret predictions, delivers both more accurate and more reliable solutions. Compared with the vanilla GNN-GLS framework, the proposed method reduces the optimality gap and computation time. These results corroborate that the DyT operator effectively preserves the training stability and representational capacity required for high-quality solution approximation, while eliminating the computational overhead incurred by batch normalization layers. Pure end-to-end constructive models inherently suffer from error accumulation during step-by-step autoregressive generation, making their solution quality brittle when scaling up without costly multi-sample search. By decoupling local representation learning from combinatorial state space search, GNN-GLS-DyT consistently delivers superior solution quality and robust inference efficiency, outperforming both pure constructive methods and traditional search baselines. Most remarkably, on the largest 100-node test instances, GNN-GLS-DyT delivers an optimality gap of 2.38%. This constitutes a 0.46 percentage point improvement over the original GNN-GLS framework and substantially outperforms all other learning-based baselines, as well as conventional heuristics, such as local search and insertion methods. While the highly-engineered classical solver LKH3 still maintains a slight performance lead at this scale, our normalization-free design significantly closes the performance gap between data-driven solvers and traditional handcrafted heuristics, validating the effectiveness of the proposed architecture for large-scale routing optimization.

As presented in Table 3, experiments on CVRP benchmarks consistently demonstrate the superiority of the GNN-GLS-DyT framework across all problem scales, with more remarkable performance gains on larger instances. The normalization-free DyT architecture eliminates the computational overhead of batch normalization, yielding even shorter inference times than the vanilla GNN-GLS while delivering substantially better solution quality. Under the exact same constrained computation budget and search protocol, GNN-GLS-DyT dramatically reduces the gap on CVRP20 compared to vanilla GNN-GLS. This performance leap demonstrates that our proposed normalization-free DyT architecture yields far more accurate regret predictions per search step, allowing GLS to converge to high-quality solutions much faster without requiring excessive iteration budgets. Notably, on CVRP100 instances, our method reduces the optimality gap to 1.18%, significantly outperforming all learning-based baselines and closely approaching the classical state-of-the-art solver LKH3. These findings verify the effectiveness of the proposed method.

5.2.2. Ablation study

Furthermore, we conduct a more in-depth analysis of the influence exerted by each component of our method on overall performance enhancement. We analyze the training curve of the proposed GNN-GLS-DyT framework and compare it with the original GNN-GLS baseline by evaluating the mean optimality gap with respect to computation time, as illustrated in Figure 3. Empirical results

demonstrate that GNN-GLS-DyT consistently outperforms the baseline across all three TSP instance scales throughout the entire training process, achieving both faster convergence rates and lower final optimality gaps. These experimental findings provide compelling evidence that normalization-free architectures leveraging learnable element-wise activation scaling can effectively guide local search heuristics for combinatorial routing problems. The proposed GNN-GLS-DyT framework not only alleviates the memory and computational bottlenecks inherent to standard normalization operations but also delivers substantial performance improvements over the original GNN-GLS baseline across diverse problem settings.

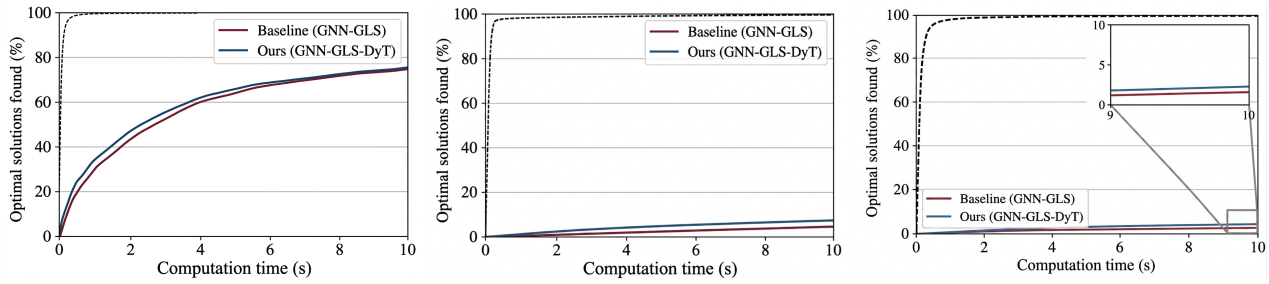


(a) Training Curve for `TSP20`. **(b)** Training Curve for `TSP50`. **(c)** Training Curve for `TSP100`.

Figure 3. Training comparison between the baseline GNN-GLS and the proposed GNN-GLS-DyT on TSP instances.

5.2.3. Generalization results

This section systematically investigates the cross-scale generalization capability of the proposed method. As shown in Figures 4, the proposed GNN-GLS-DyT method demonstrates excellent generalization performance across varying problem scales. On small-scale problems (20–50 nodes), it converges rapidly to near-optimal solutions. On medium- and large-scale problems (50–100 nodes), it maintains a significant performance advantage. Moreover, on the mixed-scale test set (20–100 nodes), the method exhibits smooth and stable performance curves. These results indicate that the learned strategy possesses strong scale adaptability and transferability, enabling it to effectively handle optimization problems of unseen sizes, thus showing promising potential for practical applications.



(a) Generation results for `20-50`. **(b)** Generation results for `50-100`. **(c)** Generation results for `20-100`.

Figure 4. Generalization performance. The smaller the performance gap, the stronger the generalization.

On the generalization test set comprising problems with 20–50 nodes, the proposed GNN-GLS-DyT method demonstrates outstanding generalization performance. Compared to the baseline, its "mean optimality gap" curve decreases more rapidly and reaches a lower level, converging close to the theoretical optimum within a short timeframe. Its "optimal solutions found" rate is also significantly higher. This indicates that the learned strategy or dynamic threshold mechanism can effectively transfer to such small-scale, yet unseen, specific problems without performance degradation due to changes in problem scale.

The advantage of GNN-GLS-DyT remains evident even on more challenging problems with 50–100 nodes. Although this problem scale typically exceeds the common range used in training, where training sets are often smaller or fixed in size, the method maintains a low optimality gap and a high rate of optimal solutions found. This confirms that its core mechanism does not overfit to the characteristics of small-scale problems. Instead, it learns scalable, general principles for search and optimization, enabling it to handle the increased complexity associated with larger problem sizes effectively.

As shown in Table 4, the performance of GNN-GLS-DyT on the mixed-scale test set covering 20–100 nodes further validates its strong generalization ability and robustness. Across this broader and more varied problem set, it consistently outperforms the baseline method comprehensively. Both the "mean optimality gap" and "optimal solutions found" metrics exhibit smooth and efficient performance curves. This demonstrates the method's ability to adaptively manage challenges arising from continuous variations in problem scale, indicating that its performance is not dependent on a specific scale range and possesses favorable scale-invariant characteristics.

Table 4. The mean optimality gap and standard deviation, as well as the percentage of optimally solved problems are reported. Our approach converges to better solutions at a faster rate than the baseline approach.

Problem	Method	Optimality gap (%)	Optimal solutions (%)
TSP20-50	Baseline (GNN-GLS)	0.160±0.020	75.0000
	Ours (GNN-GLS-DyT)	**0.070±0.010**	**75.5300**
TSP50-100	Baseline (GNN-GLS)	1.300±0.150	7.2600
	Ours (GNN-GLS-DyT)	**0.840±0.080**	**8.3000**
TSP20-100	Baseline (GNN-GLS)	1.500±0.200	1.7000
	Ours (GNN-GLS-DyT)	**1.490±0.150**	**2.2500**

A critical challenge for deploying DL-based routing in real logistics is the generalization of real instances, which necessitates training on synthetic data followed by real-world transfer. This simulation-to-real cross-domain generalization capability is a key metric to evaluate the practical applicability of learning-based routing methods. Table 5 reports a zero-shot cross-domain evaluation on TSPLIB (50–200 nodes), where models trained exclusively on random 100-node TSP instances are tested directly without fine-tuning. It reports the mean optimality gap (%), which is calculated as $Gap(\%) = [(Length_{method} - Length_{optimal})/Length_{optimal}] \times 100\%$, with the optimal solutions obtained by LKH3. Due to domain shift in spatial distributions and instance scales, overall optimality gaps naturally increase, with baseline performance degrading significantly as problem size grows. In

contrast, GNN-GLS-DyT consistently achieves the lowest optimality gaps across virtually all TSPLIB instances, demonstrating exceptional out-of-distribution robustness. This superiority stems primarily from its normalization-free architecture, which effectively suppresses performance fluctuations induced by distribution shifts. Consequently, GNN-GLS-DyT maintains high accuracy across all test scales: it yields ultra-low gaps on small-to-medium instances and, more remarkably, retains a gap of only 2.018% on the 200-node instance (kroA200), doubling the training scale while substantially outperforming all competing baselines. Regarding computational efficiency, Table 6 reports the average computation time in seconds (s). While simple constructive method, AM, attain lower computation time, GNN-GLS-DyT maintains a highly stable and practical runtime profile, consistently completing execution within 7s across all instances. Notably, GNN-GLS-DyT strictly outperforms the vanilla GNN-GLS baseline in inference speed across all benchmarks. This confirms that our normalization-free lightweight design introduces no additional computational overhead, but rather streamlines model inference while boosting generalization performance. Overall, GNN-GLS-DyT offers a highly compelling trade-off between solution accuracy and computational efficiency, demonstrating strong potential for real-world routing deployments.

Table 5. Solution quality for learning-based methods (Gap(%)).

Instance	AM	POMO	GNN-GLS	GNN-GLS-DyT
eil51	8.400	6.635	1.397	**0.433**
berlin52	5.189	3.220	0.441	**0.150**
st70	3.733	3.700	1.245	**0.760**
eil76	3.004	2.410	1.197	**0.160**
pr76	1.830	0.700	0.298	**0.040**
rat99	2.643	1.637	1.130	**0.553**
kroA100	4.043	2.809	1.300	**0.720**
kroB100	4.169	4.680	1.160	**1.150**
kroC100	1.975	5.663	2.041	**1.580**
rd100	3.492	3.436	1.005	**0.459**
eil101	6.990	6.700	0.386	**0.211**
lin105	4.735	3.990	1.848	**0.620**
pr107	3.932	1.560	0.895	**0.449**
pr124	3.673	3.147	1.324	**0.750**
ch150	4.584	4.750	3.314	**2.124**
kroA150	4.786	4.410	3.728	**2.980**
kroB150	3.438	3.770	3.885	**3.250**
pr152	7.495	5.929	4.028	**3.230**
u159	7.581	7.300	3.055	**2.020**
rat195	9.805	7.962	3.712	**1.660**
d198	9.025	6.351	5.537	**4.770**
kroA200	7.191	5.880	4.468	**2.018**

Table 6. Computation time for learning-based methods (Time(s)).

Instance	AM	POMO	GNN-GLS	GNN-GLS-DyT
eil51	**0.127**	3.012	8.040	7.078
berlin52	**0.131**	3.062	8.878	7.118
st70	**0.201**	4.039	8.962	7.058
eil76	**0.220**	4.321	8.555	7.157
pr76	**0.232**	4.323	9.152	7.043
rat99	**0.350**	5.556	9.184	6.943
kroA100	**0.457**	5.718	9.029	7.055
kroB100	**0.362**	5.218	9.138	7.010
kroC100	**0.361**	5.663	9.168	7.178
rd100	**0.551**	3.754	8.967	7.129
eil101	**0.476**	5.798	9.845	7.276
lin105	**0.382**	5.934	9.545	7.037
pr107	**0.392**	5.968	9.037	6.975
pr124	**0.492**	6.051	9.578	7.369
ch150	**0.694**	6.148	8.517	7.108
kroA150	**0.691**	6.157	9.959	7.338
kroB150	**0.692**	6.121	9.071	7.013
pr152	**0.781**	6.135	9.006	7.266
u159	**0.761**	6.219	9.966	7.428
rat195	**1.113**	6.228	9.414	7.297
d198	**1.152**	6.591	9.866	7.593
kroA200	**1.160**	6.706	9.832	7.075

6. Conclusions

This work presents GNN-GLS-DyT, a novel lightweight normalization-free hybrid framework that synergizes graph neural networks with guided local search for efficient vehicle routing optimization. At the core of our contribution is a normalization-free regret approximation module, designed to estimate the regret value of including each edge in the final solution path. Within this module, all conventional batch normalization layers are systematically replaced by the lightweight learnable DyT operator. This architectural redesign drastically reduces computational complexity and memory footprint, while fully preserving the model's representational capacity and training stability. By feeding the predicted edge regret values and original routing graph as adaptive heuristic cues into the guided local search pipeline, our hybrid framework seamlessly bridges the generalization capability of deep learning and the robust search performance of classical heuristics, enabling efficient discovery of high-quality routing solutions. Extensive empirical evaluations on canonical routing benchmarks, such as the TSP and CVRP, have shown that the proposed method delivers substantially lower optimality gaps than the mainstream learning-based competitors, with its performance advantage

growing more pronounced as problem complexity increases. The normalization-free DyT design eliminates the computational overhead of standard normalization operations, accelerating both model inference and the heuristic search process. High-quality solutions can be obtained within a shorter search horizon, making the framework well suited for time-sensitive logistics scheduling scenarios. GNN-GLS-DyT demonstrates exceptional cross-scale generalization capability. When trained exclusively on small-scale instances, it generalizes seamlessly to much larger, unseen routing problems without severe performance degradation. This finding empirically validates that the DyT operator captures the intrinsic structural patterns of routing problems more effectively than standard normalization techniques. For future work, we will extend this normalization-free paradigm to more complex constrained routing problems, such as vehicle routing with time windows and pickup-and-delivery requirements, to further verify its generalizability. We also plan to investigate the theoretical foundations of DyT-based graph learning for combinatorial optimization, aiming to unravel the intrinsic mechanisms behind its superior generalization performance. Furthermore, inspired by stronger normalization-free transformers [33], we will explore more powerful normalization-free mechanisms for graph neural network guided local search, in order to design even more lightweight neural solvers.

Author contributions

Boen Lin: Conceptualization, Methodology, Software, Formal analysis, Investigation, Writing–original draft, Visualization; Jing Sun: Methodology, Validation, Resources, Supervision, Project administration, Funding acquisition, Writing–review and editing; Yinlong Liu: Data curation, Validation, Supervision, Project administration, Writing–review and editing. All authors have read and approved the final version of the manuscript for publication.

Use of Generative-AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article, and that only the Grammarly Professional Version was used to assist with language refinement, grammar correction, and paraphrasing.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant No. 62503013.

Conflicts of interest

The authors declare that they have no conflicts of interest.

References

1. B. Rouwenhorst, B. Reuter, V. Stockrahm, G. J. van Houtum, R. J. Mantel, W. H. M. Zijm, Warehouse design and control: Framework and literature review, *Eur. J. Oper. Res.*, **122** (2001), 515–533. https://doi.org/10.1016/S0377-2217(99)00020-X
2. M. W. P. Savelsbergh, The vehicle routing problem with time windows: Minimizing route duration, *ORSA J. Comput.*, **4** (1992), 146–154.
3. G. Berbeglia, J. F. Cordeau, G. Laporte, Dynamic pickup and delivery problems, *Eur. J. Oper. Res.*, **202** (2010), 8–15. https://doi.org/10.1016/j.ejor.2009.04.024
4. P. S. K. Jayasinghe, T. Kelly, N. Madhavika, S. Enalapitiya, D. S. De Costa, M. K. Liyanage, et al., Unveiling the robustness of apparel exporters' supply chains: Exploring the influence of IT capability, supply chain collaborations and government support, *Int. J. Logist. Manag.*, **37** (2026), 397–429. https://doi.org/10.1108/IJLM-10-2024-0642 .
5. E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, New York: Wiley, 1985.
6. D. L. Applegate, R. E. Bixby, V. Chvátal, W. J. Cook, *The Traveling Salesman Problem: A Computational Study*, Princeton: Princeton University Press, 2006.
7. C. H. Papadimitriou, K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Englewood Cliffs: Prentice-Hall, 1982.
8. G. Dantzig, R. Fulkerson, S. Johnson, Solution of a large-scale traveling-salesman problem, *J. Oper. Res. Soci. Amer.*, **2** (1954), 393–410. https://doi.org/10.1287/opre.2.4.393
9. I. Bello, H. Pham, Q. V. Le, M. Norouzi, S. Bengio, Neural combinatorial optimization with reinforcement learning, arXiv preprint, arXiv:1611.09940, 2016. https://doi.org/10.48550/arXiv.1611.09940
10. W. Kool, H. van Hoof, M. Welling, Attention, learn to solve routing problems! arXiv preprint, arXiv:1803.08475, 2019. https://doi.org/10.48550/arXiv.1803.08475
11. Y. D. Kwon, J. Choo, B. Kim, I. Yoon, Y. Gwon, S. Min, POMO: Policy optimization with multiple optima for reinforcement learning, In: *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, **33** (2020), 21188–21198.
12. L. Xin, W. Song, Z. Yao, Z. Zhang, Multi-decoder attention model with embedding glimpse for vehicle routing problems, In: *Proceedings of the AAAI Conference on Artificial Intelligence*, **35** (2021), 12042–12049. https://doi.org/10.1609/aaai.v35i13.17430
13. Z. H. Fu, Q. Wu, J. K. Hao, Generalize to large-scale traveling salesman problems via transformer with hierarchical encoding, arXiv preprint, arXiv:2103.16947, 2021.
14. P. Huang, H. Jiang, S. Wang, J. Huang, Enhancing human behavior recognition with dynamic graph convolutional networks and multi-scale position attention, *Int. J. Intell. Comput. Cyber.*, **18** (2025), 236–253. https://doi.org/10.1108/IJICC-09-2024-0414

15. B. Hudson, M. Malencia, Q. Li, A. Prorok, Graph neural network guided local search for the traveling salesperson problem, arXiv preprint, arXiv:2110.05291, 2022. https://doi.org/10.48550/arXiv.2110.05291

16. R. Yang, C. Fan, Neural combinatorial optimization for time-dependent traveling salesman problem, In: *Advances in Neural Information Processing Systems*, **38** (2026), 72870–72893. https://doi.org/10.52202/085713-2442

17. W. Guettala, Á. L. Holló-Szabó, L. Gulyás, J. Botzheim, Heterogeneous graph neural networks for scalable asymmetric traveling salesman problem optimization, *Neurocomputing*, **671** (2026), 132718. https://doi.org/10.1016/j.neucom.2026.132718

18. J. Zhu, X. Chen, K. He, Y. LeCun, Z. Liu, Transformers without normalization, In: 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville: IEEE, 2025. https://doi.org/10.1109/CVPR52734.2025.01388

19. R. Baldacci, A. Mingozzi, A unified exact method for solving different classes of vehicle routing problems, *Math. Program.*, **120** (2009), 347–380. https://doi.org/10.1007/s10107-008-0218-9

20. M. Haimovich, A. H. Rinnooy Kan, Bounds and heuristics for capacitated routing problems, *Math. Oper. Res.*, **10** (1985), 527–542. https://doi.org/10.1287/moor.10.4.527

21. D. Applegate, R. Bixby, V. Chvátal, W. Cook, *The Traveling Salesman Problem: A Computational Study*, Princeton: Princeton University Press, 2006.

22. K. Helsgaun, An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems, Technical report, *Roskilde University*, 2017.

23. O. Vinyals, M. Fortunato, N. Jaitly, Pointer networks, In: *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, 2015.

24. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, et al., Attention is all you need, In: *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, 2017.

25. J. L. Ba, J. R. Kiros, G. E. Hinton, Layer normalization, arXiv preprint, arXiv:1607.06450, 2016. https://doi.org/10.48550/arXiv.1607.06450

26. M. Deudon, P. Cournut, A. Lacoste, Y. Adulyasak, L. M. Rousseau, Learning heuristics for the TSP by policy gradient, In: *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, Cham: Springer, 2018. https://doi.org/10.1007/978-3-319-93031-2_12

27. C. Voudouris, E. Tsang, Guided local search and its application to the traveling salesman problem, *Europ. J. Oper. Res.*, **113** (1999), 469–499. https://doi.org/10.1016/S0377-2217(98)00099-X

28. J. Potvin, J. M. Rousseau, An exchange heuristic for routeing problems with time windows, *J. Oper. Res. Soc.*, **46** (1993), 1433–1446. https://doi.org/10.1057/jors.1995.204

29. R. Hassin, A. Keinan, Greedy heuristics with regret, with application to the cheapest insertion algorithm for the TSP, *Oper. Res. Lett.*, **36** (2008), 243–246. https://doi.org/10.1016/j.orl.2007.05.001

30. C. Voudouris, E. Tsang, Guided local search, Technical report, *Department of Computer Science, University of Essex*, CSM-247, 1996.

31. F. Arnold, K. Sörensen, Knowledge-guided local search for the vehicle routing problem, *Comput. Oper. Res.*, **105** (2019), 32–46. https://doi.org/10.1016/j.cor.2019.01.002

32. D. L. Applegate, R. E. Bixby, V. Chvatal, W. J. Cook, *The Traveling Ssalesman Problem: A Computational Study*, Princeton: Princeton university press, 2006.

33. M. Chen, T. Lu, J. Zhu, M. Sun, Z. Liu, Stronger normalization-free transformers, In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 27418–27428, 2026.

AIMS Press