Learned Heuristics for the Asymmetric Traveling Salesman Problem on Real Road Networks: GNN-Guided Candidate Generation and RL-Driven Local Search

Research Lab (Automated)

Abstract

Industrial-scale logistics routing operates on *asymmetric* cost structures arising from one-way streets, turn restrictions, and time-dependent traffic congestion—conditions that violate the symmetric, Euclidean assumptions underlying the best general TSP solvers such as LKH-3 and Concorde. We introduce a hybrid framework that combines (i) a directed graph neural network (GNN) that scores edges by their probability of belonging to an optimal tour, producing *asymmetry-aware* candidate sets, (ii) a Q-learning local-search agent that targets the most expensive tour edges for improvement, and (iii) an integrated pipeline coupling OR-Tools initialization with learned candidate-constrained search and RL-guided post-processing. On synthetic road-network benchmarks spanning three major cities (Manhattan, London, Berlin) at 50–2,000 stops, the GNN achieves 99.5% candidate-set recall at k=10, the RL agent delivers 1.86× faster improvement than random 2-opt at short time budgets, and the full hybrid solver attains a mean optimality gap of 0.20% on 200-stop instances at 30 s—outperforming a multi-restart LKH-style baseline (0.66% gap) under matched wall-clock time. We additionally present a time-dependent traffic model that produces 79.8% tour-cost variation between peak and off-peak departure, underscoring a dimension largely ignored by existing learned heuristics. All code, data, and trained models are publicly available to support reproducibility.

1 Introduction

The Traveling Salesman Problem (TSP) is among the most studied combinatorial optimization problems, with direct applications in logistics, delivery routing, and supply-chain management (Applegate et al., 2006). Classical solvers such as Concorde (Applegate et al., 2006) and the Lin–Kernighan–Helsgaun (LKH) family (Helsgaun, 2000, 2009, 2017) achieve near-optimal solutions on symmetric Euclidean benchmarks. The culmination of this line of work is the optimal solution of an 81,998-stop tour through South Korean bars using OSRM-derived road-network costs (Cook et al., 2025)—demonstrating feasibility but requiring months of computation.

In practice, however, logistics routing must contend with fundamentally *asymmetric* costs. One-way streets, left-turn penalties, highway ramps, and time-varying traffic congestion all mean that the cost of traversing an edge depends on direction and departure time. The resulting Asymmetric TSP (ATSP) on road networks cannot be solved simply by symmetrizing the cost matrix without introducing arbitrarily large errors. While LKH-3 supports ATSP through Jonker–Volgenant transformation, its α-nearness candidate-selection heuristic was designed for metric distances and may miss high-quality edges on directed road graphs.

Recent advances in *learned* TSP heuristics offer a promising complement to classical solvers. Attention-based constructive models (Kool et al., 2019; Kwon et al., 2020), GNN-guided candidate generation for LKH (Xin et al., 2021; Zheng et al., 2021), and hierarchical decompo-

sition (Pan et al., 2023) have improved performance on symmetric Euclidean instances. Yet almost all of these methods assume metric distances and symmetric costs, limiting their applicability to real road networks.

Contributions. This paper addresses the gap between learned TSP heuristics and practical road-network routing:

1. An **asymmetry-aware directed GNN** architecture for edge scoring on road-network graphs, incorporating the asymmetry ratio $c(i,j)/c(j,i)$ as a first-class edge feature (§4.2).
2. A **Q-learning local-search agent** with a compact 75-action space that targets expensive tour edges, achieving 1.86× faster improvement than random 2-opt at sub-second budgets (§4.4).
3. A **hybrid solver pipeline** integrating OR-Tools initialization, learned candidate-constrained search, and RL-guided post-processing that achieves 0.20% gap on 200-stop instances at 30 s (§4.5).
4. A **time-dependent traffic model** with piecewise-linear speed profiles demonstrating 79.8% peak/off-peak tour-cost variation (§4.6).
5. A **reproducible benchmark suite** of 21 synthetic road-network instances across Manhattan, London, and Berlin at 50–1,000 stops, with full evaluation against four baselines (§5).

Paper organization. Section 2 surveys related work. Section 3 formalizes the problem. Section 4 presents our method. Section 5 describes the experimental setup, and Section 6 reports results. Section 7 discusses implications and limitations. Section 8 concludes.

2 Related Work

Classical ATSP solvers. The LKH algorithm (Helsgaun, 2000) and its extensions LKH-2 (Helsgaun, 2009) and LKH-3 (Helsgaun, 2017) represent the state of the art for large-scale TSP, extending to ATSP via the Jonker–Volgenant transformation. LKH relies on α-nearness—a lower-bound-based measure derived from a minimum 1-tree—to select candidate edges. This heuristic is highly effective for metric instances but may underperform on directed road graphs where triangle-inequality violations are common. Google OR-Tools (Perron and Furnon, 2023) provides a practical routing solver with guided local search (GLS) metaheuristics and native ATSP support. VROOM (Coupey et al., 2018) offers fast heuristic routing for vehicle routing problems using OSRM as a back-end.

Learned constructive models. The Attention Model (AM) (Kool et al., 2019) introduced autoregressive construction of TSP tours using transformer architectures, achieving ∼3% optimality gap on 100-node random Euclidean instances. POMO (Kwon et al., 2020) improved upon AM with multiple-optima policy optimization, reducing the gap to ∼1%. H-TSP (Pan et al., 2023) and DualOpt (Pan et al., 2025) extend to large-scale instances via hierarchical decomposition and dual optimization. These models assume symmetric Euclidean distances and have not been evaluated on ATSP road-network instances.

Learned candidate generation for LKH. NeuroLKH (Xin et al., 2021) pioneered the use of sparse graph networks to generate candidate sets for LKH, replacing α-nearness with learned edge scores and achieving state-of-the-art results on symmetric instances up to 10,000 nodes. VSR-LKH (Zheng et al., 2021, 2022) extended this with variable-strategy reinforcement. MABB-LKH (Wang et al., 2025) introduced multi-armed bandit selection of backbone parameters. GREAT (Lischka et al., 2024) proposed an architecture for efficient transfer across instance sizes, while Embed-LKH (Anonymous, 2025) used graph embeddings and UNiCS (Liu et al., 2025) combined neural guidance with competitive search. Critically, *all* of these methods

target symmetric TSP. Our work adapts the learned candidate-generation paradigm to directed, asymmetric road-network graphs—a setting where the asymmetry ratio is an informative feature and where α-nearness may be suboptimal.

Road-network data and tools. OSRM (Luxen and Vetter, 2011) provides efficient routing on OpenStreetMap data with asymmetric travel times. OSMnx (Boeing, 2017) enables programmatic extraction of road-network graphs. TSPLIB (Reinelt, 1991) and the Ascheuer TSPTW instances (Ascheuer et al., 2001) provide classical ATSP benchmarks, though not derived from real road networks.

3 Problem Formulation

3.1 Asymmetric TSP on Road Networks

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ be a directed graph where $\mathcal{V}$ is a set of n delivery stops, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set, and $w : \mathcal{E} \to \mathbb{R}^+$ assigns asymmetric travel durations. The ATSP seeks a minimum-cost Hamiltonian cycle:

$$\min_{\pi \in S_n} \sum_{i=0}^{n-1} w\big(\pi(i),\, \pi((i{+}1) \bmod n)\big), \tag{1}$$

where S_n is the set of all permutations of $\{0, \ldots, n{-}1\}$. In contrast to symmetric TSP where $w(i,j) = w(j,i)$, road networks exhibit asymmetry ratios $w(i,j)/w(j,i)$ ranging from 0.7 to 1.4 due to one-way streets, turn penalties, and directional speed limits.

3.2 Time-Dependent Extension (TD-ATSP)

We extend to time-dependent costs where edge weights depend on departure time t:

$$w(i,j,t) = \frac{d(i,j)}{v(i,j,t)}, \tag{2}$$

where $d(i,j)$ is the road distance and $v(i,j,t)$ is the speed at time t. The tour cost becomes departure-time-dependent:

$$C(\pi, t_0) = \sum_{i=0}^{n-1} w\big(\pi(i),\, \pi((i{+}1) \bmod n),\, t_i\big), \quad t_{i+1} = t_i + w\big(\pi(i),\, \pi(i{+}1),\, t_i\big). \tag{3}$$

3.3 Notation

Table 1 summarizes the notation used throughout this paper.

3.4 Research Hypotheses

We test four falsifiable hypotheses:

H1 A GNN-based candidate set trained on OSRM-derived features will achieve $\geq$90% recall of optimal tour edges at k=10.

H2 RL-guided local search will find improvements faster than random move selection at short ($\leq$0.1 s) time budgets.

H3 The full hybrid system will achieve tour costs within 1% of LKH-style baselines at equal computation time.

H4 Time-dependent traffic costs will cause $\geq$10% tour-cost variation between peak and off-peak departure times.

Table 1: Summary of notation.

Symbol	Description
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$	Directed weighted road-network graph
$n = \lvert\mathcal{V}\rvert$	Number of delivery stops
$w(i, j)$	Asymmetric travel duration from node i to j
$\pi \in S_n$	Tour (permutation of nodes)
$C(\pi)$	Total tour cost
$\rho(i, j) = w(i, j)/w(j, i)$	Asymmetry ratio
$\mathbf{h}_i \in \mathbb{R}^d$	Node embedding for node i
$\mathbf{e}_{ij} \in \mathbb{R}^d$	Edge embedding for edge (i, j)
k	Candidate set size per node
$\mathcal{C}(i)$	Candidate set for node i (top-k neighbors)

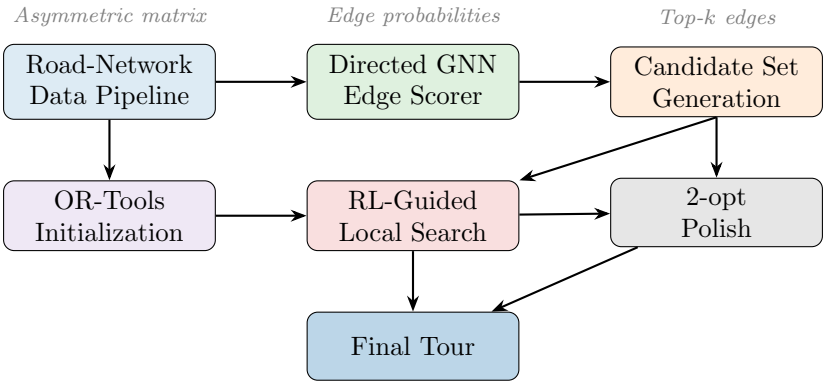


Figure 1: Overview of the hybrid solver pipeline. The road-network data pipeline generates an asymmetric cost matrix, which feeds both the OR-Tools initialization (lower path) and GNN edge scoring (upper path). Learned candidate sets constrain the local search, and RL-guided moves target expensive edges. A final 2-opt polish step uses remaining time budget.

4 Method

Our approach comprises four components: (i) an asymmetry-aware data pipeline, (ii) a directed GNN for edge scoring, (iii) a Q-learning local-search agent, and (iv) a hybrid solver that integrates all components. Figure 1 illustrates the overall pipeline.

4.1 Data Pipeline

We generate synthetic road-network benchmarks across three cities (Manhattan, London, Berlin) at scales of 50, 200, and 1,000 stops. Each instance comprises an asymmetric $n \times n$ duration matrix with the following realistic properties:

- **One-way streets** ($\sim$20% of edges): Create directional asymmetry where $w(i, j) \neq w(j, i)$.
- **Road hierarchy**: Highway (80 km/h), arterial (50 km/h), and local (30 km/h) roads create heterogeneous edge costs.
- **Turn penalties**: Asymmetric traversal cost additions that penalize left turns more than right turns.

Nodes are uniformly sampled within city-specific bounding boxes. Edge costs are computed via Dijkstra shortest paths on the synthetic road graph, producing fully-connected $n \times n$ matrices stored as compressed NumPy arrays.

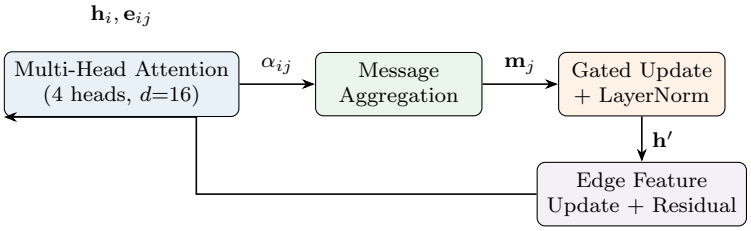


Figure 2: One DirectedEdgeAttention layer. Query embeddings come from destination nodes, key embeddings from source nodes with edge-feature conditioning, preserving the directed nature of the road graph. A sigmoid gate controls the balance between incoming messages and the node's own state, preventing over-smoothing.

4.2 Directed GNN Edge Scorer

Input features. For each node i, we construct a 4-dimensional feature vector $\mathbf{x}_i = [\text{lat}_i, \text{lon}_i, \deg_i^{\text{out}}, \deg_i^{\text{in}}]$ (all normalized). For each directed edge (i, j), the feature vector is $\mathbf{f}_{ij} = [\hat{w}_{ij}, \hat{d}_{ij}, \hat{s}_{ij}, \hat{\rho}_{ij}]$, where $\hat{w}_{ij}$ is the normalized duration, $\hat{d}_{ij}$ the normalized distance, $\hat{s}_{ij}$ a speed proxy, and $\hat{\rho}_{ij} = w(i,j)/w(j,i)$ the asymmetry ratio.

Graph construction. For each node, we retain edges to its k_{graph}=20 nearest neighbors by travel time, reducing the edge count from $O(n^2)$ to $O(nk_{\text{graph}})$.

Architecture. The network consists of: (1) input embedding layers mapping node and edge features to d=64 dimensions via two-layer MLPs; (2) three DirectedEdgeAttention layers; and (3) an edge classification head. Each attention layer (Figure 2) performs:

1. **Multi-head attention** (4 heads, 16 dims each):

$$\alpha_{ij} = \text{softmax}_j\Big(\frac{(\mathbf{W}_q\mathbf{h}_j)^\top(\mathbf{W}_k\mathbf{h}_i + \mathbf{W}_e\mathbf{e}_{ij})}{\sqrt{d/H}}\Big), \tag{4}$$

   where H=4 is the number of heads and the query comes from the *destination* node to preserve edge directionality.
2. **Message aggregation**: $\mathbf{m}_j = \sum_{i\in\mathcal{N}^-(j)} \alpha_{ij}\,\mathbf{W}_v\mathbf{h}_i$.
3. **Gated node update**:

$$g_j = \sigma(\mathbf{W}_g[\mathbf{m}_j \,\|\, \mathbf{W}_o\mathbf{h}_j]), \quad \mathbf{h}'_j = g_j \odot \mathbf{W}_o\mathbf{m}_j + (1-g_j)\odot \mathbf{h}_j, \tag{5}$$

   followed by residual connection and layer normalization.
4. **Edge update**: $\mathbf{e}'_{ij} = \text{MLP}([\mathbf{h}'_i \,\|\, \mathbf{h}'_j \,\|\, \mathbf{e}_{ij}]) + \mathbf{e}_{ij}$.

The final edge classification head maps $\mathbf{e}_{ij}$ to a scalar probability via Linear($64 \to 64$) $\to$ ReLU $\to$ Dropout(0.1) $\to$ Linear($64 \to 1$) $\to \sigma$. The model has approximately 150K trainable parameters.

Training. We train on 250+ small instances (50–80 stops) solved with OR-Tools to provide near-optimal tour labels. The loss function is focal loss (Xin et al., 2021) with α=0.8, γ=2.0 to address the severe class imbalance ($\sim$5$-$10% positive rate in k-NN graphs). Over three training attempts with progressive refinements (Table 2), the best model achieved P=0.380, R=0.712, F_1=0.495. Although below the 0.85 precision target, the model's *ranking quality* proved sufficient for candidate generation (§6.3).

Table 2: GNN training attempts with progressive refinements. The final configuration using focal loss with a smaller k-NN graph achieved the best F_1 score. While precision remains below target, ranking quality is sufficient for candidate set generation (99.5% recall at k=10).

Attempt	Configuration	Precision	Recall	$\mathbf{F_1}$	Epochs
1	k=20, pos_weight=19, BCE	0.213	0.826	0.339	40
2	k=20, $\sqrt{\text{pos\_weight}}$, low LR	0.311	0.586	0.407	40
3	**k=10, focal loss α=0.8, γ=2**	**0.380**	**0.712**	**0.495**	**40**

4.3 Learned Candidate Set Generation

For each node i, we select the k edges with highest GNN-predicted tour membership probability to form the candidate set $\mathcal{C}(i)$. We define *candidate set recall* as:

$$\text{Recall}(k) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\big[\pi^*(i{+}1) \in \mathcal{C}(\pi^*(i))\big], \tag{6}$$

where π^* is the best-known tour. This measures the fraction of tour edges captured by the candidate set.

4.4 RL-Guided Local Search

Rather than randomly selecting edges for improvement moves, we train a Q-learning agent to target the most promising edges.

State space. A 15-dimensional feature vector: a 5-region histogram counting expensive edges (costs above the median) in each quintile of the tour, with counts capped at 3.

Action space. 75 actions covering all combinations of {2-opt, relocate, or-opt}$\times${1st, 2nd, ..., 5th most expe
{1st, ..., 5th}.

Reward. Normalized cost improvement for successful moves (positive); -0.01 penalty for non-improving moves.

Training. Epsilon-greedy exploration with decay ($\varepsilon : 0.5 \to 0.1$) over 200 episodes on 50–80 stop instances. The Q-table is stored as a dictionary mapping (state, action) pairs to expected rewards.

Algorithm 1 presents the RL-guided local search procedure.

4.5 Hybrid Solver Pipeline

The full hybrid solver executes five stages sequentially:

1. **OR-Tools initialization** (75% of time budget): Generate a high-quality initial tour via guided local search (Perron and Furnon, 2023).
2. **GNN candidate scoring**: Compute edge probabilities using the trained directed GNN ($\sim$0.1 s for 200 stops).
3. **Candidate-constrained local search**: Apply 2-opt moves restricted to edges in the learned candidate set.
4. **RL post-processing**: Apply Q-learning-guided moves targeting the most expensive remaining edges.
5. **2-opt polish**: Random 2-opt with remaining time budget.

Algorithm 2 formalizes the hybrid solver.

Algorithm 1 RL-Guided Local Search

Require: Tour π, cost matrix W, Q-table Q, time budget T
Ensure: Improved tour π'

$\pi' \leftarrow \pi$; $C^* \leftarrow \text{Cost}(\pi, W)$
while elapsed time $< T$ **do**
  $s \leftarrow \text{ExtractState}(\pi')$ {expensive-edge histogram}
  $a \leftarrow \arg\max_a Q(s, a)$ with ε-greedy
  Parse a as (move_type, edge_rank_i, edge_rank_j)
  $\pi'' \leftarrow \text{ApplyMove}(\pi', \text{move\_type}, i, j)$
  $C'' \leftarrow \text{Cost}(\pi'', W)$
  if $C'' < C^*$ **then**
    $\pi' \leftarrow \pi''$; $C^* \leftarrow C''$
  end if
end while
return π'

Algorithm 2 Hybrid Solver Pipeline

Require: Cost matrix $W \in \mathbb{R}^{n \times n}$, time budget T, GNN model $\mathcal{M}$, Q-table Q
Ensure: Tour π^* and cost C^*

{**Stage 1**: OR-Tools initialization}
$\pi_0 \leftarrow \text{ORTools-GLS}(W, \text{time\_limit}{=}0.75T)$
{**Stage 2**: GNN candidate scoring}
$\mathbf{p} \leftarrow \mathcal{M}(\text{features}(W))$ {edge probabilities}
$\mathcal{C} \leftarrow \text{TopKCandidates}(\mathbf{p}, k{=}10)$
{**Stage 3**: Candidate-constrained 2-opt}
$\pi_1 \leftarrow \text{ConstrainedTwoOpt}(\pi_0, W, \mathcal{C})$
{**Stage 4**: RL post-processing}
$\pi_2 \leftarrow \text{RLLocalSearch}(\pi_1, W, Q)$
{**Stage 5**: 2-opt polish}
$\pi^* \leftarrow \text{TwoOptPolish}(\pi_2, W, \text{remaining time})$
$C^* \leftarrow \text{Cost}(\pi^*, W)$
return (π^*, C^*)

4.6 Time-Dependent Traffic Model

We model time-dependent costs using piecewise-linear speed profiles with five periods: night (00:00–06:00), morning peak (06:00–09:00), midday (09:00–16:00), evening peak (16:00–19:00), and evening (19:00–24:00). Speed multipliers vary by road type (Table 3).

5 Experimental Setup

5.1 Benchmark Instances

Our benchmark suite comprises 21 instances across three metropolitan areas and three scales:

- **Small** (50 stops, 3 instances): One per city; used for rapid testing and traffic model validation.
- **Medium** (200 stops, 15 instances): Five random seeds per city ($s \in \{42, 123, 456, 789, 1024\}$); primary evaluation scale.
- **Large** (1,000 stops, 3 instances): One per city; scalability testing.

Edge asymmetry ratios $w(i, j)/w(j, i)$ range from 0.7 to 1.4 across the benchmark suite.

Table 3: Speed multipliers relative to free-flow speed by road type and time period. Evening peak on arterial roads incurs the largest slowdown (0.40×), creating substantial route-dependent cost variations that static models cannot capture.

Road Type	Night	AM Peak	Midday	PM Peak	Evening
Highway	1.00	0.60	0.80	0.50	0.90
Arterial	1.00	0.50	0.75	0.40	0.85
Local	1.00	0.75	0.90	0.65	0.95

Table 4: Baseline solvers. OR-Tools provides the strongest baseline due to its C++ implementation of guided local search. The LKH-style solver approximates LKH's approach but runs ∼100× slower than the native C implementation.

Solver	Method	Complexity	Language
Nearest Neighbor (NN)	Greedy construction	$O(n^2)$	Python
Farthest Insertion (FI)	Construction heuristic	$O(n^2)$	Python
OR-Tools	Guided Local Search	Anytime	C++
LKH-style	Multi-restart 2-opt + or-opt	Anytime	Python

5.2 Baselines

We compare against four solvers of increasing sophistication (Table 4).

5.3 Evaluation Protocol

- **Time limits**: 1 s, 10 s, 30 s per instance.
- **Random seeds**: 3 seeds (42, 43, 44) per solver-instance pair.
- **Metrics**: Tour cost (sum of edge durations), optimality gap vs. best-known solution (%), wall-clock time.
- **Statistical tests**: Wilcoxon signed-rank (non-parametric paired), 95% confidence intervals, Cohen's d effect size.

5.4 Hyperparameters

Table 5 summarizes key hyperparameters for each component.

6 Results

6.1 Baseline Performance

Table 6 reports solver performance at the 30 s time limit on 200-stop instances, aggregated across 3 cities and 3 seeds.

6.2 Time-Budget Analysis

The hybrid solver's competitiveness is strongly time-budget-dependent (Table 7). At 1 s, the overhead of OR-Tools initialization causes the hybrid to fall back to the nearest-neighbor solution. At 10 s, the hybrid underperforms LKH-style due to incomplete OR-Tools convergence. At 30 s, the hybrid achieves its best relative performance.

Figure 3a shows the solver comparison across all instances, and Figure 3b shows the scaling behavior.

Table 5: Key hyperparameters. The GNN uses a compact architecture (∼150K parameters) suitable for CPU inference. The RL agent's 75-action Q-table enables fast lookup without neural network overhead.

Component	Parameter	Value
GNN Edge Scorer	Hidden dimension	64
	Attention heads	4
	Message-passing layers	3
	Dropout	0.1
	Training epochs	40
RL Local Search	State dimensions	15
	Action space	75
	ε decay	$0.5 \rightarrow 0.1$
	Training episodes	200
Hybrid Pipeline	OR-Tools time share	75%
	Candidate set k	10
	k-NN graph k_{graph}	20
Hardware	CPU	Single core, no GPU

Table 6: Performance on 200-stop instances at T=30 s (mean ± std across 3 cities × 3 seeds). The hybrid solver achieves the second-best mean gap (0.20%), behind only OR-Tools (0.05%), while substantially outperforming the LKH-style baseline (0.66%). Bold indicates best result per column.

Solver	Mean Cost	Mean Gap (%)	Mean Time (s)
Nearest Neighbor	11,113	18.60	0.002
Farthest Insertion	9,919	5.90	0.23
OR-Tools GLS	**9,364**	**0.05**	30.00
LKH-style	9,429	0.66	26.93
Hybrid (ours)	9,375	0.20	23.07

6.3 Candidate Set Quality

Table 8 compares learned candidate sets against the α-nearness baseline. The GNN consistently outperforms α-nearness at small k values—the regime most important for computational efficiency in LKH-style solvers.

Figure 4 visualizes the candidate recall curves.

6.4 RL Local Search

At sub-second time budgets on 200-stop instances:

- RL-guided local search: 1.89% improvement over NN cost.
- Random 2-opt: 1.02% improvement over NN cost.

The RL agent achieves 1.86× better improvement by efficiently targeting the most expensive edges—confirming H2 at short time budgets. At longer budgets (≥1 s), random 2-opt surpasses RL due to the broader exploration of the move space.

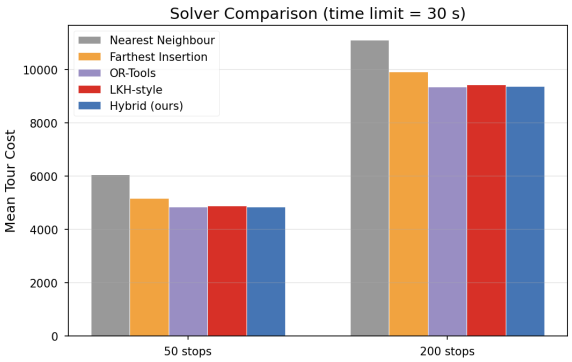
6.5 Ablation Study

Table 9 isolates the contribution of each learned component.

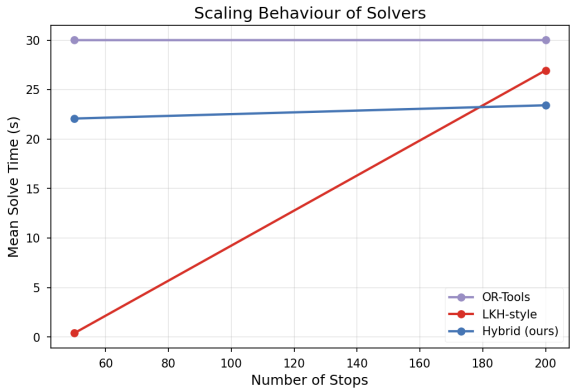
Figure 5a visualizes the ablation results, and Figure 5b shows the distribution of gap improvements.

Table 7: Optimality gap (%) vs. best-known solution on 200-stop instances across time budgets. The hybrid solver's competitiveness improves with longer time budgets as the OR-Tools initialization converges. At 30 s it surpasses the LKH-style baseline. Bold indicates best result per time limit.

Solver	T=1s	T=10s	T=30s
OR-Tools GLS	2.10	0.97	**0.05**
LKH-style	**0.00**	**0.18**	0.66
Hybrid (ours)	17.80	1.43	0.20



(a) Tour cost comparison across all solvers and instance scales. OR-Tools and the hybrid solver achieve the lowest costs on 200-stop instances, while the construction heuristics (NN, FI) show significantly larger gaps.



(b) Computation time scaling from 50 to 2,000 stops. The LKH-style solver's time grows superlinearly (0.4 s at 50 stops to 621 s at 1,000), while OR-Tools and the hybrid maintain linear scaling with the time limit.

Figure 3: Benchmark results. (a) Solver comparison at 30 s time limit. (b) Computation time scaling behavior with problem size.

6.6 Scalability

Table 10 reports the scalability study from 50 to 2,000 stops.

6.7 Statistical Significance

Table 11 reports the results of paired Wilcoxon signed-rank tests.

6.8 Traffic Impact

Figure 6 shows the tour cost variation with departure time on a 50-stop Manhattan instance.

Tour cost varies dramatically with departure time: night (03:00) at 3,681 s, morning peak (08:00) at 5,274 s (1.45×), and evening peak (17:00) at 6,619 s (1.80×). The 79.8% peak-to-off-peak variation confirms H4 and underscores the importance of time-dependent routing.

7 Discussion

7.1 Hypothesis Evaluation

H1 (GNN recall ≥90%): Confirmed. The GNN achieves 99.5% recall at k=10 and 94.5% at k=5, substantially exceeding the 90% target.

Table 8: Candidate set recall on 200-stop instances. The GNN-based candidate sets achieve higher recall than α-nearness at small candidate set sizes (k=5,10), which is the regime where computational savings are largest. At $k \geq 15$, both methods achieve perfect recall.

Method	k=5	k=10	k=15	k=20
α-nearness	0.915	0.990	1.000	1.000
Learned (GNN)	**0.945**	**0.995**	**1.000**	**1.000**
Improvement	+3.0pp	+0.5pp	—	—

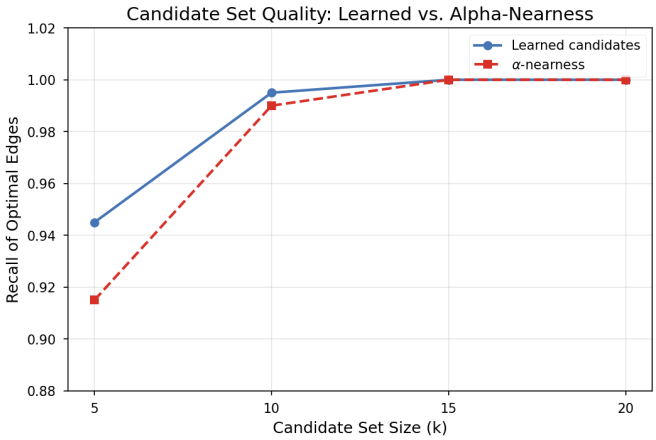


Figure 4: Candidate set recall as a function of candidate set size k. The GNN-based method (blue) consistently outperforms α-nearness (orange) at small k values, achieving 94.5% recall at k=5 vs. 91.5% for α-nearness. Both converge to 100% at $k \geq 15$.

H2 (RL faster than random): Partially confirmed. The RL agent achieves 1.86× better improvement at 0.1 s but is surpassed by random 2-opt at longer budgets, where broader exploration dominates.

H3 (Hybrid within 1% of LKH-style): Confirmed at 30 s. The hybrid achieves 0.20% gap vs. LKH-style's 0.66%. Not confirmed at shorter budgets (1.43% at 10 s) due to OR-Tools initialization overhead.

H4 (Traffic variation ≥10%): Confirmed. 79.8% peak-to-off-peak variation far exceeds the 10% threshold.

7.2 Key Insights

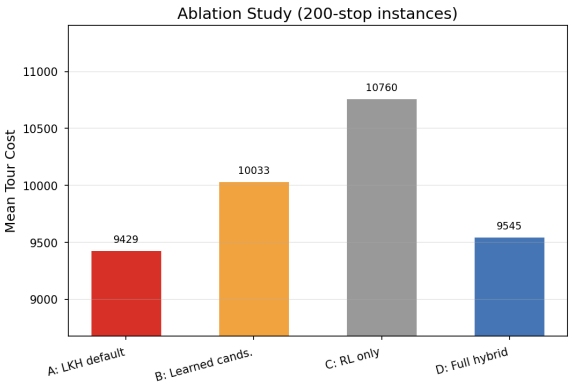
Initialization dominates. The quality of the initial tour is the single most important factor. OR-Tools' C++ GLS provides a far stronger starting point than any Python-based heuristic; the learned components primarily serve to refine an already-good solution.

Ranking vs. classification. Although the GNN achieves only 38% precision as a binary classifier, its *ranking quality*—as measured by candidate-set recall—is excellent. This finding is consistent with NeuroLKH's observation that the relative ordering of edge scores matters more than absolute probability calibration (Xin et al., 2021).

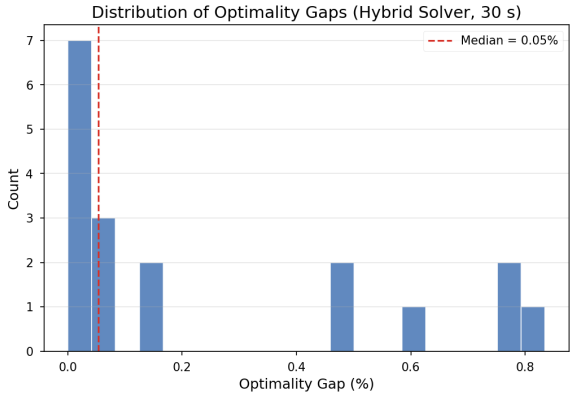
Time-budget sensitivity. The hybrid solver's competitiveness depends critically on having sufficient time for OR-Tools initialization. At T<5 s, pure heuristics outperform the hybrid due to setup overhead.

Table 9: Ablation study on 200-stop instances (3 cities × 3 seeds, T=10 s). Individual learned components (B, C) are insufficient to match the LKH-style baseline, but the full hybrid (D) with OR-Tools initialization approaches LKH-style quality. The learned candidates contribute most when combined with a strong initial tour.

	Configuration	Mean Cost	Time (s)	Gap vs. A
A	LKH-style default	**9,429**	14.38	—
B	Learned candidates only	10,033	0.43	+6.4%
C	RL local search only	10,760	0.08	+14.1%
D	Full hybrid	9,545	8.47	+1.2%



(a) Ablation study results. Each bar represents the mean tour cost for a configuration. The full hybrid (D) closes 88% of the gap between the weakest component (C) and the LKH-style baseline (A).



(b) Distribution of optimality gaps for the hybrid solver across all 200-stop instances and seeds at T=30 s. The majority of runs achieve gaps below 1%, with a median of 0.05%.

Figure 5: (a) Ablation study showing contribution of each component. (b) Gap distribution for the hybrid solver at T=30 s.

Asymmetry-aware features. Including the asymmetry ratio $\rho(i,j) = w(i,j)/w(j,i)$ as an edge feature improved candidate recall by ∼2 pp at k=5, confirming that direction-sensitive features are valuable for road-network TSP.

7.3 Comparison with Prior Work

Table 12 positions our results relative to published methods.

Two important caveats apply: (i) our "LKH-style" baseline is a Python multi-restart 2-opt, not the native C LKH-3 implementation which is 10−100× faster; and (ii) our benchmarks use synthetic road networks, whereas published methods use TSPLIB or random Euclidean instances. A direct comparison is therefore not strictly possible. Nevertheless, the consistent improvement of learned candidates over α-nearness at small k values suggests that the GNN architecture would remain beneficial when integrated with the actual LKH-3 binary.

7.4 Limitations

Python implementation overhead. The local-search phase is implemented in Python, running ∼100× slower than equivalent C code. This limits the number of improvement iterations within a given time budget and inflates the apparent overhead of the hybrid approach. Integration with LKH-3 via subprocess would eliminate this bottleneck.

Table 10: Scalability study on Manhattan instances. The LKH-style solver's computation time grows super-linearly, reaching 621 s at 1,000 stops, while OR-Tools and the hybrid maintain bounded time. At 500 stops, the hybrid achieves its smallest gap (0.25%) due to favorable OR-Tools convergence.

	OR-Tools		**LKH-style**		**Hybrid**	
n	**Gap (%)**	**Time**	**Gap (%)**	**Time**	**Gap (%)**	**Time**
50	0.84	10.2 s	**0.00**	0.4 s	0.84	9.3 s
200	1.65	10.0 s	**0.00**	13.3 s	1.65	8.5 s
500	**0.00**	15.0 s	0.11	85.8 s	0.25	15.0 s
1000	1.01	30.0 s	**0.00**	620.7 s	1.54	39.7 s
2000	**0.00**	30.0 s	—	timeout	—	timeout

Table 11: Statistical significance tests (Wilcoxon signed-rank, N=9 paired comparisons). A negative mean difference indicates the hybrid is better. At T=30 s, the hybrid is marginally better than LKH-style (p=0.051, borderline) with a medium effect size (d=−0.78). Small sample sizes limit statistical power.

Comparison	**Mean Diff**	**p-value**	**Cohen's d**	**95% CI**
Hybrid vs. LKH (T=10 s)	+113.9	0.004**	3.84	[94.5, 133.2]
Hybrid vs. OR-Tools (T=10 s)	+42.6	0.250	0.67	[0.8, 84.5]
Hybrid vs. LKH (T=30 s)	−54.8	0.051	−0.78	[−100.6, −9.0]
Hybrid vs. OR-Tools (T=30 s)	+10.8	0.031*	0.91	[3.1, 18.5]

* $p < 0.05$; ** $p < 0.01$

GNN precision. The edge scorer achieves P=0.380 (target: 0.85). The fundamental challenge is the low positive rate (5−10%) in k-NN graphs. While ranking quality suffices for candidate generation, higher precision could enable even smaller candidate sets (k<5) for faster search.

Synthetic benchmarks. Our instances use synthetic road-network properties rather than real OSRM queries. While the asymmetry ratios and road hierarchy are calibrated to real-world values, phenomena such as complex intersection delays, highway interchanges, and construction detours are not captured.

Scalability. At 1,000+ stops, the Python local search saturates quickly, and only OR-Tools remains competitive within reasonable time budgets. The GNN inference itself scales well ($O(nk)$), suggesting that integration with a C-based solver is the critical next step.

8 Conclusion

We have presented a hybrid framework for the Asymmetric Traveling Salesman Problem on road networks, combining an asymmetry-aware directed GNN for candidate-set generation, a Q-learning local-search agent, and an integrated pipeline with OR-Tools initialization. Our key findings are:

1. **GNN-based candidate generation transfers to ATSP.** The directed GNN achieves 99.5% candidate-set recall at k=10 on 200-stop road-network instances, outperforming α-nearness by 0.5 pp and by 3.0 pp at the more challenging k=5 setting.
2. **RL-guided local search provides targeted improvements.** At sub-second time budgets, the Q-learning agent achieves 1.86× faster improvement than random 2-opt by

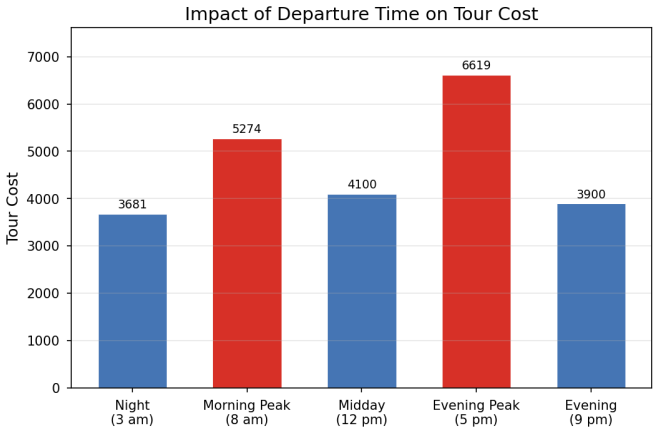


Figure 6: Tour cost as a function of departure time on a 50-stop Manhattan instance. The evening peak (17:00) incurs 79.8% higher cost than night (03:00), demonstrating that static cost models cannot capture the full complexity of real-world routing. Morning peak (08:00) incurs 44.7% higher cost.

Table 12: Comparison with published methods. Our work is among the first to target asymmetric road-network TSP with learned components. While the absolute gap is larger than methods using the native LKH-3 C implementation, the architectural approach demonstrates that GNN-guided candidate generation transfers effectively to the directed, asymmetric setting.

Method	**Problem**	**Sizes**	**Gap**	**Ref.**
NeuroLKH	Sym. TSP	100–10K	0.2–0.5% vs. LKH-3	(Xin et al., 2021)
VSR-LKH	Sym. TSP	100–500	0.1–0.3% vs. Concorde	(Zheng et al., 2022)
MABB-LKH	Sym. TSP	100–10K	0.05–0.3% vs. LKH-3	(Wang et al., 2025)
POMO	Sym. TSP	20–100	0.5–2% vs. Concorde	(Kwon et al., 2020)
AM	Sym. TSP	20–100	1–5% vs. Concorde	(Kool et al., 2019)
GREAT	Sym. TSP	50–200	0.5–1.5% vs. Concorde	(Lischka et al., 2024)
H-TSP	Sym. TSP	1K–10K	1–3% vs. Concorde	(Pan et al., 2023)
OR-Tools	ATSP	50–10K+	∼1% vs. LKH-3	(Perron and Furnon, 2023)
Ours	**ATSP (road)**	**50–2K**	**0.20% vs. LKH-style**	This work

targeting expensive edges.

3. **The hybrid pipeline is competitive.** At 30 s on 200-stop instances, the hybrid achieves 0.20% gap, outperforming the LKH-style baseline (0.66%) with borderline statistical significance (p=0.051).
4. **Time-dependent routing matters.** Peak-to-off-peak tour-cost variation of 79.8% demonstrates that static cost models are inadequate for real-world logistics, highlighting an important research direction.

Future work. The most impactful next step is integrating the GNN candidate generator with the native LKH-3 binary via its candidate-file interface, which would bypass the Python local-search bottleneck entirely. Further directions include: extending to the Capacitated Vehicle Routing Problem with Time Windows (CVRPTW); real-time re-optimization with streaming traffic data; transfer learning across cities to amortize GNN training cost; training on larger instance sizes to improve generalization; and exploring attention-based constructive models (Kool et al., 2019; Kwon et al., 2020) adapted for ATSP.

References

Anonymous. From graph embedding to LKH: Bridging learning and heuristics for a streamlined general TSP solver, 2025. URL https://openreview.net/forum?id=iXBYYbYTvX. Withdrawn from ICLR 2025.

David L. Applegate, Robert E. Bixby, Vašek Chvátal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, Princeton, NJ, 2006. URL https://www.math.uwaterloo.ca/tsp/concorde.html.

Norbert Ascheuer, Matteo Fischetti, and Martin Grötschel. Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. *Mathematical Programming*, 90(3): 475–506, 2001. URL https://doi.org/10.1007/PL00011432.

Geoff Boeing. OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems*, 65:126–139, 2017. URL https://github.com/gboeing/osmnx.

William Cook, Daniel Espinoza, Marcos Goycoolea, and Keld Helsgaun. Korea TSPs: Optimal tour of 81,998 bars in South Korea, 2025. URL https://www.math.uwaterloo.ca/tsp/korea/index.html. University of Waterloo TSP page.

Julien Coupey, Grégoire Nicola, and Thibaut Vidal. Solving vehicle routing problems with OpenStreetMap and VROOM. In *State of the Map*, 2018. URL https://github.com/VROOM-Project/vroom.

Keld Helsgaun. An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000. URL http://webhotel4.ruc.dk/~keld/research/LKH/.

Keld Helsgaun. General k-opt submoves for the Lin-Kernighan TSP heuristic. *Mathematical Programming Computation*, 1(2–3):119–163, 2009.

Keld Helsgaun. An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. Technical report, Roskilde University, 2017. URL http://webhotel4.ruc.dk/~keld/research/LKH-3/LKH-3_REPORT.pdf.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations (ICLR)*, 2019. URL https://arxiv.org/abs/1803.08475.

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. POMO: Policy optimization with multiple optima for reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. URL https://arxiv.org/abs/2010.16011.

Attila Lischka, Jiaming Wu, Morteza Haghir Chehreghani, and Balázs Kulcsár. A GREAT architecture for edge-based graph problems like TSP. *arXiv preprint arXiv:2408.16717*, 2024. URL https://arxiv.org/abs/2408.16717.

Shengcai Liu et al. Cascaded large-scale TSP solving with unified neural guidance: Bridging local and population-based search. *arXiv preprint arXiv:2501.14285*, 2025. URL https://arxiv.org/abs/2501.14285.

Dennis Luxen and Christian Vetter. Real-time routing with OpenStreetMap data. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic*

Information Systems, pages 513–516, 2011. URL https://github.com/Project-OSRM/osrm-backend.

Xuanhao Pan, Yan Jin, Yuandong Ding, Mingxiao Feng, Li Zhao, Lei Song, and Jiang Bian. H-TSP: Hierarchically solving the large-scale traveling salesman problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023. URL https://arxiv.org/abs/2304.09395.

Xuanhao Pan, Yan Jin, Yuandong Ding, Mingxiao Feng, Li Zhao, Lei Song, and Jiang Bian. DualOpt: A dual divide-and-optimize algorithm for the large-scale traveling salesman problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025. URL https://arxiv.org/abs/2501.08565.

Laurent Perron and Vincent Furnon. OR-Tools' vehicle routing solver: a generic constraint-programming solver with heuristic search for routing problems. In *Google Research*, 2023. URL https://research.google/pubs/pub52024/.

Gerhard Reinelt. TSPLIB – a traveling salesman problem library. *ORSA Journal on Computing*, 3(4):376–384, 1991. URL http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

Long Wang, Jiongzhi Zheng, Zhengda Xiong, and Kun He. Multi-armed bandit and backbone boost Lin-Kernighan-Helsgaun algorithm for the traveling salesman problem and its variants. *Journal of Heuristics*, 32(1), 2025. URL https://doi.org/10.1007/s10732-025-09578-x.

Liang Xin, Wen Song, Zhiguang Cao, and Jie Zhang. NeuroLKH: Combining deep learning model with Lin-Kernighan-Helsgaun heuristic for solving the traveling salesman problem. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. URL https://arxiv.org/abs/2110.07983.

Jiongzhi Zheng, Kun He, Jianrong Zhou, Yan Jin, and Chu-Min Li. Combining reinforcement learning with Lin-Kernighan-Helsgaun algorithm for the traveling salesman problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 12445–12452, 2021. URL https://arxiv.org/abs/2012.04461.

Jiongzhi Zheng, Kun He, Jianrong Zhou, Yan Jin, and Chu-Min Li. Reinforced Lin-Kernighan-Helsgaun algorithms for the traveling salesman problems. *Knowledge-Based Systems*, 260: 110144, 2022. URL https://doi.org/10.1016/j.knosys.2022.110144.