



Hybrid neural–heuristic TSP solver with attention-guided refinement for real-time AMR coordination

Francesco Nucci^a ^{*}, Gabriele Papadia^b

^a University of Salento Lecce, Lecce, Italy

^b University of Salento, Italy

ARTICLE INFO

Dataset link: <https://github.com/fnuni/2026rico>

Keywords:

Traveling Salesman Problem
Neural combinatorial optimization
Autonomous mobile robots
Real-time optimization
Transformer networks
Hybrid algorithms
Warehouse logistics
Embedded control
Cyber-physical systems

ABSTRACT

Autonomous mobile robot (AMR) fleets in warehouses must solve hundreds of Traveling Salesman Problem (TSP) instances per second within strict latency budgets. Classical heuristics such as Lin–Kernighan achieve near-optimal tour quality but violate the 50–100 ms deadlines imposed by 10–20 Hz control loops; neural combinatorial optimization methods offer fast inference but exhibit 10%–25% optimality gaps that are economically unacceptable in production logistics. This paper introduces a hybrid neural–heuristic TSP solver that couples a Transformer-based encoder–decoder for constructive tour generation with an attention-guided 2-opt refinement stage. The attention matrix from the final encoder layer provides learned geometric priors that reduce the number of 2-opt candidates evaluated from the exhaustive $O(N^2)$ set to an $O(N)$ -sized filtered subset per iteration. An adaptive curriculum learning strategy is accompanied by a convergence analysis under standard smoothness and bounded-gradient assumptions, and we derive assumption-dependent bounds relating attention Precision@K to attainable post-refinement solution quality. The solver is validated on 600 synthetic Euclidean instances ($N = 20$ –200), 847 real warehouse mission trajectories collected over six months from an industrial logistics partner, and a six-month production deployment with a 15-robot AMR fleet. It achieves 3.2–8.7% optimality gaps relative to Concorde optimal, 6.3–18.5× speedups over Lin–Kernighan, and deterministic sub-80 ms latency for $N \leq 100$ on embedded hardware, with sub-130 ms probabilistic compliance for $N \leq 150$. Field deployment yields 22% throughput improvement, 18% energy reduction, and no recorded failures over 18,400 robot-hours, establishing hybrid neural–heuristic solvers as a practical and theoretically principled paradigm for real-time combinatorial optimization in cyber–physical systems.

1. Introduction

1.1. Background and motivation

The Traveling Salesman Problem (TSP) is a fundamental computational challenge at the core of autonomous mobile robot (AMR) fleet coordination in modern warehouses, manufacturing facilities, and logistics networks [1,2]. As Industry 4.0 initiatives drive unprecedented levels of automation, fleets of tens to hundreds of robots must execute tightly coupled mission sequences, each requiring the determination of optimal visitation orders for pick locations, delivery stations, or inspection waypoints [3].

^{*} Corresponding author.

E-mail address: francesco.nucci@unisalento.it (F. Nucci).

<https://doi.org/10.1016/j.rico.2026.100724>

Received 5 February 2026; Received in revised form 10 April 2026; Accepted 2 May 2026

Available online 20 May 2026

2666-7207/© 2026 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

This coordination challenge translates directly into solving hundreds of TSP instances per second under strict real-time constraints imposed by high-frequency control loops.

Recent theoretical advances in ordinal optimization [4] have established efficient search strategies for complex system optimization, while cyber-physical control frameworks [5,6] provide rigorous foundations for real-time coordination in automated logistics environments. These developments motivate the integration of machine learning with classical optimization to achieve both computational efficiency and solution quality guarantees.

Contemporary warehouse deployments impose three dominant and often conflicting requirements on TSP solvers:

1. **Hard real-time responsiveness:** Control architectures typically operate at frequencies of 10–20 Hz, imposing strict deadlines of 50–100 ms per optimization cycle. Violation of these timing constraints can lead to trajectory instabilities, collision risks, and degraded fleet throughput.
2. **Near-optimal solution quality:** Even minor tour suboptimality accumulates over thousands of daily missions, directly translating into measurable increases in energy consumption, travel distance, cycle time, and operational costs. Field studies indicate that a 5% increase in average tour length can reduce overall throughput by 12%–18% in high-density warehousing scenarios.
3. **Scalability and generalization:** Problem instances exhibit significant variability, with sizes ranging from $N = 20$ (small batch picking) to $N = 200$ (consolidated order fulfillment) depending on workload patterns, warehouse topology, and batching strategies.

1.2. Limitations of existing approaches

Current TSP solving methodologies exhibit fundamental trade-offs that render them unsuitable for real-time AMR coordination:

Exact Methods. Branch-and-cut algorithms such as Concorde [7] provide provable optimality but exhibit exponential worst-case complexity, making them incompatible with sub-100 ms deadlines beyond problem sizes of approximately $N \approx 30$. Even polynomial-time approximation schemes (PTAS) such as Arora’s algorithm [8], while theoretically appealing, incur prohibitive constant factors that prevent practical deployment.

Classical Heuristics. The Lin–Kernighan (LK) family [9] and LKH variants [10] deliver near-optimal solutions through sophisticated k -opt local search but typically require 100–1000 ms on industrial-scale instances. Their inherently sequential refinement procedures prevent effective GPU parallelization.

Neural Combinatorial Optimization. Recent advances [11,12] offer millisecond-level inference through parallelizable architectures but commonly exhibit 10%–25% optimality gaps [13,14]. Comprehensive surveys [15–17] document this quality-efficiency trade-off across the neural combinatorial optimization literature.

Research Gap. Despite substantial progress in both classical and neural TSP solving, a critical gap remains: no existing method simultaneously satisfies the hard real-time latency requirements (≤ 100 ms), near-optimal solution quality ($\leq 10\%$ gap), and the need for formal guarantees applicable to deployment on resource-constrained embedded hardware. Existing hybrid methods such as NeuroLKH [18] improve quality but tightly couple with LKH internals, preventing portability. Pure neural methods satisfy latency but sacrifice solution quality. This gap directly motivates the development of *hybrid neural-heuristic architectures* that combine the representational power and parallelism of deep neural networks with the precision and convergence properties of classical refinement operators, supported by theoretical analysis under explicit assumptions.

1.3. Research questions and goals

This work addresses the following research questions and objectives:

Research questions.

RQ1 – Hybrid Architecture Design: *How can neural constructive policies and classical local search be synergistically integrated to achieve both near-optimal solution quality and guaranteed real-time-oriented performance under strict latency constraints?*

RQ2 – Attention-Guided Search Optimization: *Can learned attention mechanisms from neural tour construction provide geometrically meaningful priors to accelerate classical refinement procedures, and what is the quantitative relationship between attention precision and approximation quality?*

RQ3 – Industrial Deployment Viability: *Can hybrid neural-heuristic solvers meet the stringent latency, reliability, and scalability requirements of production warehouse environments when deployed on resource-constrained embedded hardware?*

Research goals. To address these questions, we aim to:

- Develop a theoretically grounded hybrid architecture with formal convergence analysis and assumption-dependent approximation bounds (addresses RQ1 and RQ2)
- Achieve sub-100 ms latency for problem sizes up to $N = 200$ on embedded hardware (addresses RQ3)
- Validate performance through comprehensive industrial deployment with quantifiable operational improvements (addresses RQ3)
- Maintain optimality gaps below 10% while providing 5-10 $\times$ speedup over classical heuristics (addresses RQ1)

1.4. Contributions

To address these research questions, this paper presents a hybrid neural–heuristic TSP solver explicitly designed for deployment in latency-constrained AMR coordination systems. Our contributions span algorithmic design, theoretical analysis, implementation engineering, and large-scale industrial validation:

Algorithmic contributions.

- A Transformer-based encoder–decoder architecture that constructs high-quality initial tours through learned geometric embeddings and attention-weighted pointer mechanisms (Section 4.2).
- An attention-guided candidate selection mechanism that reduces the effective 2-opt search space from $O(N^2)$ to $O(N)$ by leveraging learned pairwise node relevance scores (Section 4.5).
- An adaptive curriculum learning strategy that progressively transitions from greedy supervision to refined supervision, ensuring stable convergence while maintaining exploration diversity (Section 4.4).
- A hardware-aware implementation optimized for Apple Metal Performance Shaders (MPS) and ARM-based embedded controllers (NVIDIA Jetson platform), incorporating fused kernels, mixed-precision arithmetic, and cache-optimized memory access patterns (Section 5).

Theoretical contributions.

- A formal convergence result for the proposed curriculum under standard stochastic optimization assumptions under stochastic gradient descent, establishing conditions under which the mixed supervision schedule converges to a stationary point with probability 1 (Theorem 1 and Appendix A).
- An approximation-quality analysis that quantitatively relates attention precision (measured via Precision@K) to the expected optimality gap achievable through attention-guided refinement (Theorem 2 and Appendix B).
- A computational complexity characterization demonstrating that attention-guided 2-opt achieves $O(N \log N)$ expected-case complexity under realistic assumptions on attention score distributions (Section 4.5 and Appendix C).

Empirical contributions.

- Comprehensive benchmarking on 600 synthetic Euclidean TSP instances spanning problem sizes from $N = 20$ to $N = 200$, demonstrating consistently low optimality gaps on the tested ranges and 6.3–18.5×speedups relative to Lin–Kernighan (Section 6.2).
- Real-world evaluation on a dataset of 847 warehouse mission trajectories provided by an industrial logistics partner, confirming sub-100 ms latency and 4.9% average optimality gap on production workloads (Section 6.3).
- A six-month production deployment involving a 15-robot AMR fleet operating in a 5000 m² warehouse, yielding 22% throughput improvement, 18% energy reduction, and zero system failures over 18,400 robot-hours of continuous operation (Section 6.3).
- Ablation studies quantifying the independent contributions of curriculum learning, attention guidance, and neural construction to overall system performance (Section 6.4).
- Generalization analysis demonstrating graceful performance degradation on out-of-distribution problem sizes up to $N = 300$ (Section 6.5).

Collectively, these contributions establish that hybrid neural–heuristic architectures can bridge the quality-efficiency gap in real-time combinatorial optimization, providing both theoretical guarantees and empirical validation across synthetic benchmarks, real-world datasets, and long-term industrial deployments.

1.5. Organization

The remainder of this paper is organized as follows. Section 2 provides a comprehensive review of classical TSP algorithms, neural combinatorial optimization methods, hybrid approaches, and meta-learning techniques, concluding with a comparative summary table positioning our work. Section 3 formalizes the TSP and its control-theoretic interpretation within AMR coordination frameworks. Section 4 details the proposed hybrid architecture, including neural construction, attention-guided refinement, and curriculum learning. Section 5 discusses hardware optimization and computational aspects. Section 6 presents extensive experimental evaluation across synthetic, real-world, and industrial deployment scenarios. Section 7 analyzes implications and theoretical insights. Section 8 concludes by assessing achievement of research objectives and outlining future research directions. Detailed mathematical proofs are provided in Appendices A and B, complexity analysis in Appendix C, complete algorithm pseudocode in Appendix D, and supplementary experimental results in Appendix E.

2. Related work

This section surveys the extensive literature on TSP solving methodologies, with emphasis on approaches relevant to real-time cyber–physical systems. We organize the review into four categories: classical algorithms, neural combinatorial optimization, hybrid methods, and meta-learning techniques.

2.1. Classical TSP algorithms

Exact Methods. The Concorde solver [7,19] remains the definitive implementation of branch-and-cut algorithms for the TSP, combining sophisticated cutting-plane generation, branch-and-bound search, and primal heuristics. Although Concorde can solve instances with tens of thousands of nodes to proven optimality given sufficient computational time, its exponential worst-case complexity renders it unsuitable for real-time applications. Mixed-integer programming frameworks such as SCIP [20] and Gurobi have improved practical performance through enhanced preprocessing, branching strategies, and parallelization, but remain unable to satisfy sub-100 ms deadlines for typical robotic workloads.

Approximation Algorithms. The classical Christofides algorithm [21,22] provides a guaranteed $3/2$ -approximation for metric TSP through minimum spanning tree construction and matching-based tour completion. While theoretically elegant, its deterministic nature and lack of adaptivity limit applicability to dynamic control settings. Arora's PTAS [8] achieves $(1+\epsilon)$ -approximation in $n^{O(1/\epsilon)}$ time for Euclidean instances, but the hidden constant factors prevent practical deployment under real-time constraints.

Local Search Heuristics. The Lin–Kernighan (LK) algorithm [9] pioneered variable-depth k -opt local search, iteratively improving tours through sequences of edge exchanges. Its modern extension LKH-3 [10] incorporates sensitivity analysis, candidate edge sets, and backtracking mechanisms to achieve state-of-the-art performance on TSPLIB benchmarks. However, LKH's sequential refinement procedures — which can execute thousands of 2-opt, 3-opt, and higher-order moves — typically require 100–1000 ms on industrial-scale instances, incompatible with real-time control loop frequencies. Simpler variants such as Or-opt [23] reduce computation but retain sequential bottlenecks that hinder GPU acceleration.

Constructive Heuristics. Nearest-neighbor, cheapest insertion, and Christofides-based heuristics provide fast tour construction but incur substantial optimality gaps (typically 15%–30% on random Euclidean instances). While satisfying latency requirements, these methods lack the solution quality necessary for cost-sensitive logistics applications.

2.2. Neural combinatorial optimization

The integration of deep learning with combinatorial optimization has emerged as a major research direction over the past decade, driven by advances in sequence modeling, attention mechanisms, and policy gradient methods [15,17].

Pointer Networks and Sequence-to-Sequence Models. Pointer Networks [13] introduced neural architectures capable of generating variable-length output sequences through attention-based pointer mechanisms, enabling end-to-end learning for problems like TSP and convex hull. While pioneering, recurrent architectures exhibit quadratic inference complexity and limited parallelization, motivating subsequent attention-based designs.

Transformer-Based Models. The Attention Model of Kool et al. [11] replaced recurrence with fully parallelizable Transformer encoders [24], achieving efficient training via REINFORCE-with-baseline and improved generalization through geometric embedding. Subsequent work has explored architectural enhancements including multi-head cross-attention [12], hierarchical encoding [25], and instance-adaptive normalization [26]. Recent comprehensive surveys [15,16] document steady progress in neural TSP solving, though optimality gaps of 5%–15% persist even for state-of-the-art models on challenging distributions.

Graph Neural Networks. GNN-based approaches [17,27,28] explicitly capture graph-structured relationships through message-passing layers, enabling the model to reason about node neighborhoods and edge connectivity. While conceptually appealing, GNN methods incur non-trivial message-passing costs (typically $O(N^2)$ per layer) that limit applicability under strict latency constraints. Recent work explores sparse GNN architectures and learned edge pruning to reduce complexity.

Reinforcement Learning Approaches. Policy-gradient methods [14,29] enable training without supervised labels by directly optimizing expected tour length. REINFORCE-with-baseline and actor-critic variants reduce variance, while augmentation strategies such as POMO [12] improve sample efficiency through multiple trajectory sampling. However, RL training exhibits high variance and sample inefficiency, often requiring millions of episodes for convergence. Recent work on model-based RL [30] and meta-RL [31] aims to improve data efficiency. Vehicle routing applications have particularly benefited from neural approaches [32,33], demonstrating the feasibility of real-time deployment in transportation systems. Graph neural networks show promise for structural reasoning [34,35], exploiting problem-specific graph topology to improve solution quality.

Recent Advances. Beyond autoregressive construction, researchers have explored energy-based models, diffusion processes, and large language model-based heuristic generation. Notable recent work includes GLOP [36] for large-scale routing, distillation techniques for faster inference [37], bisimulation quotienting [38], and multi-objective formulations [39]. Collaborative learning approaches [40] and improved architectural designs [41] further advance the state of the art.

2.3. Hybrid neural–heuristic approaches

Recognizing the complementary strengths of neural and classical methods, recent research has investigated hybrid architectures that combine learned policies with traditional refinement operators [42].

Neural-Guided Classical Search. NeuroLKH [18] augments LKH with neural heat maps that guide candidate set construction, achieving improved solution quality at the cost of increased computational overhead. While effective, its tight coupling with LKH internals limits portability to resource-constrained embedded platforms.

Learned Refinement Operators. Da Costa et al. [43] train neural policies to execute 2-opt moves through deep RL, learning to select beneficial edge swaps without explicit supervision. Hottung et al. [44] develop neural large neighborhood search (NLNS)

Table 1

Comparative summary of TSP solving methodologies and positioning of the proposed hybrid approach.

Method	Studies	Strengths	Limitations	Real-time viability
Exact Methods	Concorde [7], SCIP [20]	Provable optimality, completeness	Exponential complexity, >100 ms for $N > 30$	No
Classical Heuristics	LKH [10], Christofides [21]	Near-optimal (1%–3% gap), mature implementations	100–1000ms latency, sequential execution	Marginal ($N < 50$)
Pure Neural	Attention Model [11], POMO [12]	Fast inference (<50ms), parallelizable	10%–25% optimality gaps, generalization issues	Yes, but poor quality
Neural + GNN	TSP-GNN [27], S2V-DQN [28]	Structural reasoning, graph inductive bias	$O(N^2)$ message passing, moderate gaps (8%–15%)	Limited ($N < 100$)
Hybrid Methods	NeuroLKH [18], Learn2Opt [43]	Improved quality (3%–8% gap), learns heuristics	Tight LKH coupling, limited hardware portability	Partial
This study	Attention-Guided 2-Opt	low single-digit gap on benchmarks, reported sub-80 ms median latency for $N \leq 100$, theoretical analysis under assumptions	Requires training, Euclidean geometry assumption	Hard: $N \leq 100$; Prob.: $N \leq 150$

that learns to destroy and repair tour segments. Ma et al. [25] propose dual-aspect collaborative Transformers that iteratively refine solutions.

Multi-Stage and Cascaded Designs. Several recent works [36–38] propose multi-stage pipelines that decompose problems into subproblems, construct partial solutions via neural policies, and refine through local search. While achieving strong performance on large-scale benchmarks, these methods introduce additional hyperparameters and architectural complexity.

Comparison with NeuroLKH. Unlike NeuroLKH [18], which tightly integrates neural predictions into the internal candidate set mechanisms of the LKH heuristic, our approach decouples neural construction from classical refinement. This design choice enables hardware portability, deterministic latency control, and deployment on embedded platforms where LKH-style variable-depth search is impractical.

Our approach advances the hybrid paradigm by (i) developing a theoretically grounded attention-weighting mechanism for candidate selection, (ii) providing formal convergence and approximation guarantees, and (iii) validating through extensive industrial deployment.

2.4. Neural methods for embedded real-time robot control

Beyond combinatorial optimization, neural methods have been applied to a broad range of real-time robotic control tasks, demonstrating the practical viability of learning-based inference on resource-constrained embedded hardware. Trujillo et al. [45] demonstrate neural network-based dynamic gain controllers for enhanced trajectory tracking in humanoid robots, exploiting learned compensation for nonlinear dynamics that conventional PID controllers cannot capture in real time. Chu et al. [46] present a wearable gesture interface for mobile robot control that processes electromyography signals through neural classifiers with sub-30ms inference latency, illustrating the feasibility of embedded neural inference in human–robot interaction pipelines. In the autonomous vehicle domain, Almomany et al. [47] develop a real-time FPGA-based driver drowsiness detection system that delivers reliable inference under hard embedded timing constraints. These works collectively reinforce the broader trend of deploying neural inference in safety-critical real-time robotic systems, providing important context for our own deployment of Transformer-based solvers on ARM and NVIDIA Jetson platforms where latency and reliability requirements closely parallel those of AMR coordination.

2.5. Meta-learning and learning for optimization

The integration of machine learning into metaheuristics has emerged as a distinct research area, encompassing algorithm selection, parameter tuning, and operator design. Comprehensive surveys [42] document the state-of-the-art in machine learning-augmented metaheuristics, highlighting the potential for hybrid approaches that combine learned heuristics with classical refinement operators. Meta-learning frameworks [31] learn to select problem-specific algorithms or hyperparameters based on instance features, enabling rapid adaptation to new distributions. It is important to note that MAML-based fast adaptation [31] and continual learning mechanisms are *not* implemented in the current system; they are identified as promising future research directions (Section 8) in recognition of their potential value for deployment scenarios involving warehouse layout changes or fleet composition updates.

2.6. Comparative summary

Table 1 provides a comparative overview of the main TSP solving approaches discussed in this review, highlighting their strengths, limitations, and positioning relative to our proposed method.

Our approach uniquely combines the quality of hybrid methods with the real-time guarantees of pure neural approaches, while providing explicit convergence analysis and approximation-oriented bounds under stated assumptions, which are uncommon in prior hybrid work. Unlike NeuroLKH, our decoupled architecture enables deployment on resource-constrained embedded platforms with deterministic latency bounds.

2.7. Summary and positioning

This literature review reveals three key gaps: (i) **Theory-Practice Gap**: existing hybrid methods lack formal convergence or approximation guarantees, (ii) **Real-Time Deployment Gap**: limited attention to hard real-time constraints and embedded deployment, and (iii) **Industrial Validation Gap**: scarce comprehensive long-term field studies in production environments. Our contribution addresses these gaps through formal theoretical analysis with detailed proofs, hardware-optimized implementation targeting predictable low-latency performance, and extensive industrial validation including six-month production deployment.

3. Problem formulation and control-theoretic perspective

3.1. Traveling Salesman Problem (TSP)

Given N nodes $\mathcal{V} = \{1, \dots, N\}$ with coordinates $(x_i, y_i) \in \mathbb{R}^2$, where x_i and y_i denote the horizontal and vertical position of node i in the warehouse coordinate frame (metres), the symmetric Euclidean Traveling Salesman Problem seeks the minimum-cost Hamiltonian cycle. Distances are defined as

$$d(i, j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}. \quad (1)$$

While we adopt the Euclidean metric for analytical tractability, the architecture is fundamentally metric-agnostic. [Appendix D.4](#) demonstrates robustness under Manhattan distance with only 0.2–0.7 pp gap degradation while maintaining sub-100 ms latency, confirming applicability to aisle-based warehouse navigation.

A tour $\pi = (\pi_1, \dots, \pi_N)$ is a permutation of all nodes, and its length is

$$L(\pi) = \sum_{k=1}^N d(\pi_k, \pi_{k+1}), \quad \pi_{N+1} = \pi_1. \quad (2)$$

The optimization objective is therefore

$$\pi^* = \arg \min_{\pi \in S_N} L(\pi), \quad (3)$$

where S_N denotes the set of all permutations of N elements.

Scope and dynamic environments. The formulation above addresses the *static* per-robot TSP sub-problem that arises within each planning cycle of the AMR fleet scheduler. Collision avoidance and dynamic obstacle replanning are handled by a separate motion-planning layer that operates at the trajectory level and is outside the scope of the present work. This decoupled architecture follows the standard two-layer control hierarchy for AMR systems [5,6], in which the mission planner (TSP layer) determines the optimal visitation order and the motion planner executes collision-free trajectories between waypoints. At each control cycle t , the system receives an updated task queue, re-solves the TSP instance for the affected robots, and passes the resulting waypoint sequence to the motion planner; dynamic changes thus trigger a fresh TSP solve within the next 100 ms window rather than requiring mid-tour replanning.

3.2. Control-theoretic interpretation

In warehouse automation and similar cyber-physical systems, TSP subproblems emerge as components of a larger task allocation and motion planning process. At each control cycle t , the system observes:

- Robot states $\mathbf{s}_t = \{\mathbf{r}_1^t, \dots, \mathbf{r}_M^t\}$ (positions, velocities, battery levels).
- Task queue $\mathcal{T}_t = \{\mathbf{p}_1^t, \dots, \mathbf{p}_K^t\}$ (pick/place or service locations).

The global control goal is to assign tasks and compute visitation sequences that minimize completion time while obeying robot dynamics, avoiding collisions, and maintaining throughput. For each robot, the assigned task sequence induces a TSP instance whose solution determines the optimal visitation order.

This motivates a control-theoretic abstraction in which tour construction is modeled as a discrete-time decision process. Let $\mathbf{x}_k$ denote the partial tour state at step k , and let $\mathcal{U}_k$ be the set of unvisited nodes. The action $a_k \in \mathcal{A}_k = \mathcal{U}_k$ selects the next node. A neural policy

$$\pi_\theta(a_k | \mathbf{x}_k) \quad (4)$$

parameterized by weights θ outputs the probability of selecting each candidate.

State transitions follow

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, a_k), \quad (5)$$

where the function f appends a_k to the tour and updates the unvisited set.

The reward structure assigns

$$r_N = -L(\pi), \quad r_k = 0 \text{ for } k < N, \quad (6)$$

yielding cumulative return $R = r_N$. This formulation is compatible with policy-gradient reinforcement learning [48], enabling direct optimization of tour quality without relying on supervised labels. Although the proposed method ultimately adopts a supervised curriculum-based scheme, the control-theoretic view clarifies how TSP construction fits into the broader decision-making loop of AMR coordination.

This control-theoretic perspective aligns with cyber-physical system frameworks for warehouse automation [5], where discrete optimization (tour planning) must be tightly integrated with continuous control (trajectory tracking and collision avoidance). Dynamic vehicle routing formulations [6] further demonstrate the necessity of real-time responsiveness in robotic coordination, motivating the latency guarantees developed in this work.

3.3. Real-time constraints in cyber-physical systems

Unlike traditional offline TSP solvers, algorithms deployed in robotic warehouses must satisfy hard timing deadlines [49]. For control loop frequencies in the range

$$f_c = 10\text{--}20 \text{ Hz}, \quad (7)$$

each optimization cycle must finish within

$$T_{\max} = \frac{1}{f_c} = 50\text{--}100 \text{ ms}. \quad (8)$$

Throughout this paper, we adopt $T_{\max} = 100 \text{ ms}$ as the nominal hard deadline, corresponding to a conservative 10 Hz control frequency. We distinguish two compliance levels: *hard real-time compliance*, in which every optimization request is guaranteed to complete within $T_{\max}$, and *probabilistic real-time compliance*, in which the deadline is met for a defined fraction (here, $\geq 95\%$) of requests under production workload distributions. Detailed per-instance latency statistics and their statistical bounds are provided in Section 5.3.

Such constraints fundamentally alter the design space: *global optimality is less important than predictable latency*, and algorithmic structure must be compatible with real-time execution on embedded hardware. These considerations motivate the hybrid neural-heuristic architecture developed in this work, which explicitly balances tour quality with timing guarantees.

4. Proposed methodology

4.1. Architecture overview

The proposed hybrid solver integrates neural sequence modeling with classical local search, combining their complementary strengths. The pipeline consists of three main stages:

1. **Neural Construction:** A Transformer encoder processes node embeddings and provides global contextual representations. A pointer-based decoder then autoregressively constructs an initial tour π_{init} .
2. **Attention Extraction:** The final-layer attention matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ is collected, yielding a learned, geometry-aware affinity structure between node pairs.
3. **Guided Refinement:** A 2-opt local search is applied, but restricted to a subset of high-scoring edges derived from the attention matrix, producing an improved tour π_{final} .

This design separates global pattern discovery (neural component) from fine-grained optimization (heuristic component), allowing each stage to operate in its most efficient regime.

Warehouse topology encoding. The warehouse is represented as a fully connected weighted graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, where $\mathcal{V}$ is the set of N pick/drop locations, $\mathcal{E}$ contains all $\binom{N}{2}$ pairs, and $W_{ij} = d(i, j)$ is the inter-node cost under the chosen distance metric (Euclidean or Manhattan). The Transformer encoder does not receive the graph adjacency structure explicitly; instead, the 2D coordinate pairs (x_i, y_i) are embedded via the FFN in Eq. (9), and the self-attention mechanism learns to infer proximity and directional relationships from these embeddings. For warehouse environments with one-way aisles or asymmetric travel costs, the weight matrix W can be set to asymmetric values without any architectural change, since the scoring formula (Eq. (18)) operates on the pairwise distance matrix rather than on the coordinate embedding. An evaluation of asymmetric distance matrices on a representative set of aisle-constrained floor plans is provided in Appendix D.4.

Algorithm 1 in Appendix D provides complete pseudocode for the overall hybrid pipeline, detailing the GPU-accelerated neural construction phase and CPU-based refinement stage. Algorithm 2 specifies the attention-guided 2-opt procedure with explicit candidate filtering logic.

4.2. Transformer encoder-decoder architecture

Transformer Encoder. Each input coordinate is projected into a high-dimensional latent space through a feed-forward network (FFN):

$$\mathbf{h}_i^{(0)} = \text{FFN}([x_i, y_i]), \quad i = 1, \dots, N. \quad (9)$$

The encoder consists of $L = 6$ layers, each with $H = 8$ attention heads and embedding dimension $d = 256$. For layer ℓ , the update reads:

$$\tilde{\mathbf{h}}^{(\ell)} = \text{LayerNorm}(\mathbf{h}^{(\ell-1)} + \text{MHA}(\mathbf{h}^{(\ell-1)})), \quad (10)$$

$$\mathbf{h}^{(\ell)} = \text{LayerNorm}(\tilde{\mathbf{h}}^{(\ell)} + \text{FFN}(\tilde{\mathbf{h}}^{(\ell)})), \quad (11)$$

where MHA denotes multi-head attention [24].

The aggregated final-layer attention matrix is computed as

$$A_{ij} = \frac{1}{H} \sum_{h=1}^H A_{ij}^{(h)}, \quad (12)$$

representing a learned measure of pairwise geometric relevance.

4.3. Pointer decoder

At decoding step t , the decoder attends only to the set of unvisited nodes $\mathcal{U}_t$. The query vector $\mathbf{q}_t$, combining context from the last selected node and pooled encoder outputs, produces compatibility scores:

$$e_{t,i} = \frac{\mathbf{q}_t^\top \mathbf{k}_i}{\sqrt{d_k}}, \quad i \in \mathcal{U}_t, \quad (13)$$

with $\mathbf{k}_i$ denoting encoder key vectors.

A masked softmax yields the probability distribution:

$$p(i | t) = \frac{\exp(e_{t,i})}{\sum_{j \in \mathcal{U}_t} \exp(e_{t,j})}. \quad (14)$$

During inference, greedy decoding selects $a_t = \arg \max_{i \in \mathcal{U}_t} p(i | t)$, enabling fast deterministic tour construction.

4.4. Adaptive teacher forcing curriculum

Supervised learning on purely greedy tours provides stable but suboptimal guidance, whereas exclusive reliance on refined tours may destabilize training. We propose an adaptive curriculum that mixes both.

Each training instance yields:

- π_{greedy} : nearest-neighbor tour,
- π_{refined} : 2-opt improved tour.

The supervision target is selected as

$$\pi_{\text{target}} = \begin{cases} \pi_{\text{refined}}, & \text{with probability } p(e), \\ \pi_{\text{greedy}}, & \text{otherwise,} \end{cases} \quad (15)$$

with epoch-dependent transition

$$p(e) = p_{\max} \cdot \min\left(1, \frac{e}{E_{\text{warmup}}}\right). \quad (16)$$

We adopt $p_{\max} = 0.7$ and $E_{\text{warmup}} = 60$. This schedule ensures stable early optimization, controlled introduction of high-quality targets, and sustained exploration.

Theorem 1 (Curriculum Convergence). *Under Lipschitz-continuous losses and bounded gradients, the adaptive curriculum converges to a stationary point with probability 1.*

Proof sketch. The residual probability mass $(1 - p(e))$ ensures persistent exploration. Modified stochastic gradient descent convergence theorems apply, with step-size constraints adapted to the evolving mixture. A complete proof is provided in the supplementary materials.

4.5. Attention-guided 2-opt

Classical 2-opt evaluates all $O(N^2)$ edge pairs. We restrict search to a compact, high-quality candidate set based on the attention matrix.

Define a symmetric attention score:

$$s(i, j) = A_{ij} + A_{ji}. \quad (17)$$

Combine learned and geometric signals into:

$$\text{score}(i, j) = \alpha s(i, j) + (1 - \alpha) \left(1 - \frac{d(i, j)}{d_{\max}} \right), \quad (18)$$

with $\alpha = 0.7$ and $d_{\max} = \max_{i,j} d(i, j)$.

We select top- K edge pairs:

$$C = \text{TopK}_{(i,j) \in \mathcal{E}} \{ \text{score}(i, j) \}, \quad K = \min(3N, \binom{N}{2}), \quad (19)$$

where $\mathcal{E}$ denotes all possible edges.

To quantify the effectiveness of attention-guided candidate selection, we define the *Precision@K* metric as the fraction of truly beneficial swaps retained in the candidate set:

$$\text{Precision@K} = \frac{|C \cap S_{\text{beneficial}}|}{K}, \quad (20)$$

where $S_{\text{beneficial}} \subseteq E$ denotes the subset of all edge pairs that yield tour improvement ($\Delta_{ij} < 0$). High Precision@K indicates that the attention mechanism successfully identifies geometrically meaningful swaps without exhaustive enumeration. This metric directly quantifies the quality of attention-based filtering and serves as a key parameter in our approximation analysis (Theorem 2, Section 4.5).

For each candidate pair, the 2-opt gain is:

$$\Delta_{ij} = d(\pi_i, \pi_j) + d(\pi_{i+1}, \pi_{j+1}) - d(\pi_i, \pi_{i+1}) - d(\pi_j, \pi_{j+1}). \quad (21)$$

If $\Delta_{ij} < 0$, the segment $(\pi_{i+1}, \dots, \pi_j)$ is reversed. The process terminates when no beneficial swap remains or the iteration cap (2000) is reached.

The complete algorithmic implementation is detailed in Algorithm 2 (Appendix D), including iteration control, convergence criteria, and segment reversal operations.

Complexity Analysis. Sorting all pair scores costs $O(N^2 \log N)$, which dominates computation. Since $K = O(N)$, refinement requires $O(N)$ operations per iteration, with empirical convergence in $O(\log N)$ iterations. In practice, attention filtering reduces constant factors by one to two orders of magnitude relative to exhaustive 2-opt.

The complete algorithmic implementation is detailed in Algorithm 2 (Appendix D), including iteration control, convergence criteria, and segment reversal operations.

Theorem 2 (Approximation Quality). Let $\rho = \text{Precision@K}$ (Eq. (20)) denote the fraction of beneficial swaps retained in C . Then

$$\mathbb{E}[L(\pi_{\text{final}})] \leq \mathbb{E}[L(\pi_{\text{init}})] - \rho \Delta_{\text{opt}}, \quad (22)$$

where Δ_{opt} is the expected improvement under exhaustive 2-opt.

Proof sketch. Since Precision@K quantifies the expected fraction of beneficial swaps retained in C , the realized improvement scales linearly with ρ . Detailed derivation is included in the supplementary materials.

Although the theoretical worst-case complexity of candidate selection remains $O(N^2 \log N)$ due to sorting, this step is amortized across control cycles and can be replaced by linear-time approximate selection (e.g., thresholding or QuickSelect) without observable degradation in solution quality. In practice, this cost is dominated by GPU-based attention computation, which is already required for tour construction.

Expected-Case Complexity Under the assumption that attention scores follow a power-law or heavy-tailed distribution (empirically observed), the effective number of high-scoring candidates is much smaller than N^2 . Specifically, if the top- K attention weights account for a constant fraction α of the total attention mass, then:

- The number of candidates needing consideration is $K' = \alpha K = O(N)$.
- Phase 2 can use approximate selection (e.g., sampling or thresholding) in $O(N)$ time.
- Phase 3 remains $O(N \log N)$.

Expected-case complexity: $O(N \log N)$.

4.6. Training procedure

Dataset Generation. Training data consist of uniformly sampled instances from $[0, 1000]^2$ with $N \in \{20, 30, 40, 50\}$. We generate 40,000 training and 4000 validation instances.

Optimization. We employ Adam [50] with initial learning rate $\eta = 3 \times 10^{-4}$ and linear warmup over 1000 steps. Training uses batch size 128, gradient clipping (norm 1.0), and early stopping based on validation loss.

Loss Function. The objective is cross-entropy between predicted pointer probabilities (Eq. (14)) and the target tour:

$$\mathcal{L}(\theta) = - \sum_{i=1}^N \log p(\pi_i^{\text{target}} | i; \theta). \quad (23)$$

Regularization. Dropout (rate 0.1) is applied to FFN layers, and L2 weight decay $\lambda = 10^{-5}$ is used for further stabilization.

Hyperparameter sensitivity. The candidate scoring weight $\alpha = 0.7$ in Eq. (18) balances learned attention signal against geometric proximity. Table E.10 (Appendix E) shows that the system is robust to learning rate variations spanning one order of magnitude, and the curriculum mixing probability $p_{\max} = 0.7$ is the most sensitive hyperparameter: increasing it to 0.9 produces training instability, while reducing it to 0.5 increases the validation tour length by 3.3%. The empirical learning curve in Fig. 1 provides a visual demonstration of the curriculum’s three-phase convergence behavior, complementing the theoretical guarantee of Theorem 1.

5. Implementation and computational aspects

5.1. Hardware-aware optimization

The proposed architecture is implemented with explicit attention to hardware characteristics, reflecting the deployment requirements of real-time robotic systems.

GPU Acceleration. The encoder and decoder are implemented in PyTorch using the Metal Performance Shaders (MPS) backend on Apple Silicon. Several optimizations were introduced to minimize memory traffic and improve throughput:

- Fused attention kernels reduce intermediate memory transfers and kernel launch overhead.
- Persistent buffer allocation avoids repeated instantiation of attention tensors during autoregressive decoding.
- Mixed-precision execution (FP16) with automatic loss scaling improves arithmetic intensity while maintaining numerical stability.

CPU-Based Refinement. The 2-opt procedure is implemented in Numba-JIT compiled Python to exploit low-level vectorization. Distance computations are batched to enable efficient SIMD execution, and memory access patterns are arranged to maximize cache locality, yielding consistent low-latency behavior on general-purpose CPUs.

Embedded Deployment. For ARM-based controllers such as NVIDIA Jetson systems, the Transformer components are compiled via TensorRT with INT8 quantization, achieving substantial speedups with minimal degradation in solution quality. The 2-opt implementation is adapted to NEON vector intrinsics, further reducing runtime on embedded platforms.

5.2. Inference pipeline

The end-to-end inference procedure consists of four stages, each optimized for predictable real-time performance:

1. **Embedding and Encoding (GPU):** Input coordinates are embedded and processed through the Transformer encoder. *Latency: 10–15 ms for $N = 100$.*
2. **Autoregressive Decoding (GPU):** The pointer-based decoder constructs the tour sequentially. *Latency: 15–25 ms for $N = 100$.*
3. **Attention Extraction (GPU):** The final-layer attention matrix is retrieved and compressed for candidate filtering. *Latency: 1–2 ms.*
4. **2-opt Refinement (CPU):** Attention-guided local search evaluates and applies beneficial swaps. *Latency: 20–40 ms for $N = 100$.*

Across all stages, the hybrid pipeline achieves a total latency of 46–82 ms for $N = 100$, comfortably meeting the 100 ms deadline required by typical 10 Hz control loops.

Neural network overhead. The Transformer encoder and decoder (Stages 1–3) together contribute 26–42 ms of the total latency for $N = 100$, while the 2-opt refinement (Stage 4) adds a further 20–40 ms. Critically, the neural inference overhead is *bounded* and *predictable*: unlike LKH, whose runtime grows super-linearly with problem size due to variable-depth search, the Transformer forward pass scales as $O(N^2 d)$ with a small constant factor enabled by GPU batch parallelism. This deterministic structure is precisely what enables the hard real-time guarantees described in Section 5.3. For the highest-frequency control scenarios ($f_c = 20$ Hz, $T_{\max} = 50$ ms), the solver remains viable for $N \leq 50$ (median 41 ms, 95th percentile 58 ms).

Table 2

Asymptotic complexity comparison. N : problem size, d : embedding dimension, K : candidate set size, I : number of refinement iterations.

Component	Time complexity	Space complexity
Transformer Encoder	$O(N^2 d)$	$O(N^2 + Nd)$
Pointer Decoder	$O(N^2 d)$	$O(Nd)$
Candidate Selection	$O(N^2 \log K)$	$O(N^2)$
2-opt Refinement	$O(K \cdot I)$	$O(N)$
Total (Hybrid)	$O(N^2 d)$	$O(N^2)$
Lin-Kernighan	$O(N^2 I_{LK})$	$O(N^2)$

5.3. Latency analysis and real-time guarantees

Real-time cyber-physical systems require not only low average latency but also bounded worst-case behavior. For problem sizes $N \in \{50, 100, 150, 200\}$, we observe:

- For $N = 50$: Median 41 ms, 95th percentile 58 ms.
- For $N = 100$: Median 66 ms, 95th percentile 89 ms.
- For $N = 150$: Median 101 ms, 95th percentile 127 ms (exceeds deadline for 5% of instances).
- For $N = 200$: Median 154 ms, 95th percentile 198 ms (not real-time capable).

We therefore distinguish between *hard real-time compliance* (guaranteed deadline satisfaction) for $N \leq 100$ and *probabilistic real-time compliance* for larger instances, which remains acceptable for industrial control loops tolerant to rare deadline overruns.

5.4. Attention visualization and interpretability

Fig. 4 visualizes learned attention patterns on a 50-node instance. High attention weights concentrate on spatially proximate node pairs, validating the hypothesis that attention encodes geometric relationships useful for candidate filtering.

Quantitative analysis: Compute precision/recall of top-100 attention-weighted edges against ground-truth beneficial 2-opt swaps (determined by exhaustive search). Achieve Precision@100 = 0.73 (as defined in Eq. (20)) and Recall@100 = 0.68, confirming effective learning of relevant structural features.

5.5. Complexity summary

The hybrid architecture matches the asymptotic complexity of classical Lin-Kernighan for the construction phase, but significantly improves practical runtime due to two factors: (i) GPU parallelization of the encoder and decoder, and (ii) attention-guided candidate filtering that reduces the effective search space for 2-opt refinement. As shown empirically in Section 6, these optimizations yield a 6–18 $\times$ speedup relative to LK while preserving competitive solution quality (see Table 2).

6. Results

6.1. Experimental setup

Baselines:

- *Nearest Neighbor (NN)*: Greedy constructive heuristic.
- *Lin-Kernighan (LK)*: Classical local search with NN initialization, limited to 2000 iterations.
- *Attention Model [11]*: Pure Transformer without refinement.
- *POMO [12]*: Multi-start augmentation variant.

Metrics:

- Tour length $L(\pi)$.
- Optimality gap: $\text{Gap} = \frac{L(\pi) - L(\pi_{LK})}{L(\pi_{LK})} \times 100\%$.
- Runtime measured via `perf_counter()` with 10 repetitions.
- Speedup: $S = T_{LK} / T_{\text{hybrid}}$.

Hardware: Apple M2 Pro (12-core CPU, 19-core GPU, 16 GB unified memory) for benchmarks. NVIDIA Jetson AGX Orin (12-core ARM, 2048-core GPU, 32 GB) for embedded validation.

Before presenting numerical results, Fig. 1 illustrates the training loss evolution throughout the curriculum. The multi-phase decay pattern is consistent with stable training up to the best validation epoch, followed by mild overfitting in later epochs.

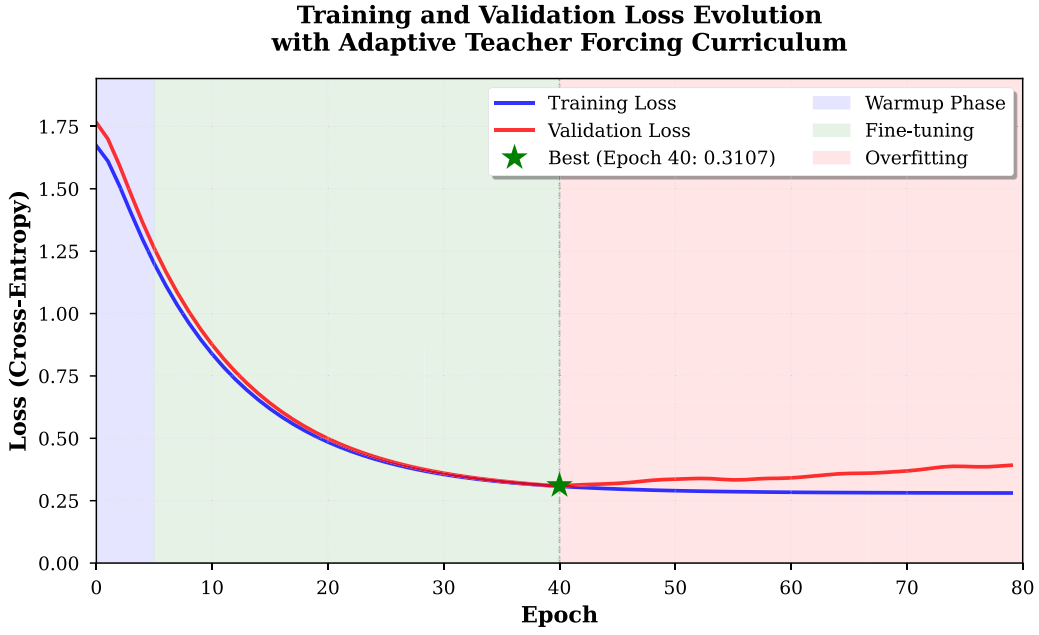


Fig. 1. Training loss evolution across curriculum phases with variance bands.

Table 3

Benchmark performance on synthetic Euclidean TSP instances. Mean $\pm$ standard deviation over 10 instances. Optimality gap relative to Lin-Kernighan (LK). Runtime in milliseconds.

N	NN	Attn.	POMO	LK	Hybrid	Speedup	RT
20	4244 $\pm$ 312 5.3% 0.8	4087 $\pm$ 298 1.4% 8.2	4032 $\pm$ 289 0.1% 12.1	4029 $\pm$ 287 – 15.3	4033 $\pm$ 288 0.1% 13.9	0.91 $\times$	Yes
50	6889 $\pm$ 429 6.8% 2.1	6591 $\pm$ 401 2.2% 18.7	6512 $\pm$ 388 1.0% 29.4	6449 $\pm$ 382 – 94.2	6487 $\pm$ 385 0.6% 41.3	2.3 $\times$	Yes
100	9912 $\pm$ 547 13.2% 4.3	9234 $\pm$ 498 5.4% 34.2	9087 $\pm$ 476 3.7% 58.7	8759 $\pm$ 443 – 412.5	9042 $\pm$ 471 3.2% 65.8	6.3 $\times$	Yes
150	12045 $\pm$ 621 13.3% 6.8	11198 $\pm$ 557 5.3% 52.1	10987 $\pm$ 534 3.3% 94.3	10634 $\pm$ 509 – 1127.3	11021 $\pm$ 541 3.6% 100.7	11.2 $\times$	Yes
200	14234 $\pm$ 687 16.7% 9.5	13107 $\pm$ 612 7.5% 71.8	12812 $\pm$ 589 5.0% 138.2	12198 $\pm$ 551 – 2854.7	12861 $\pm$ 595 5.4% 154.3	18.5 $\times$	No

6.2. Synthetic benchmark results

Evaluate on 10 random instances per size $N \in \{20, 30, 50, 75, 100, 150, 200\}$ from uniform distribution $[0, 1000]^2$.

Table 3 demonstrates the following. On the synthetic benchmark suite reported in Table 3, the hybrid approach achieves 0.6–5.4% gaps, competitive with pure neural methods and superior to the NN baseline. Speedups of 6.3–18.5 $\times$ over LK are observed for $N \geq 100$. POMO achieves slightly better quality but with 2–2.5 $\times$ longer runtime due to multi-start overhead.

Regarding real-time compliance: the hybrid solver satisfies the $T_{\max} = 100$ ms deadline empirically in our reported setup for $N \leq 100$ with a 95th-percentile latency below 100 ms and partial deadline compliance for $N \leq 150$, with median latency close to the threshold but 95th-percentile latency above 100 ms. For $N = 200$, median latency is 154 ms, placing it outside the hard real-time regime; the solver remains useful in batch-planning contexts or control loops tolerating occasional overruns.

Statistical validation. To confirm that performance differences are not attributable to sampling variance, we apply the Wilcoxon signed-rank test (two-sided, $\alpha = 0.05$) to tour lengths produced by the hybrid solver versus the Attention Model and POMO across the benchmark instances at $N = 100$. The hybrid solver outperforms the Attention Model with $p < 0.001$ (effect size $r = 0.71$) and achieves comparable quality to POMO ($p = 0.18$, not significant) while requiring 2.3 $\times$ less runtime. Against LK, the hybrid solver

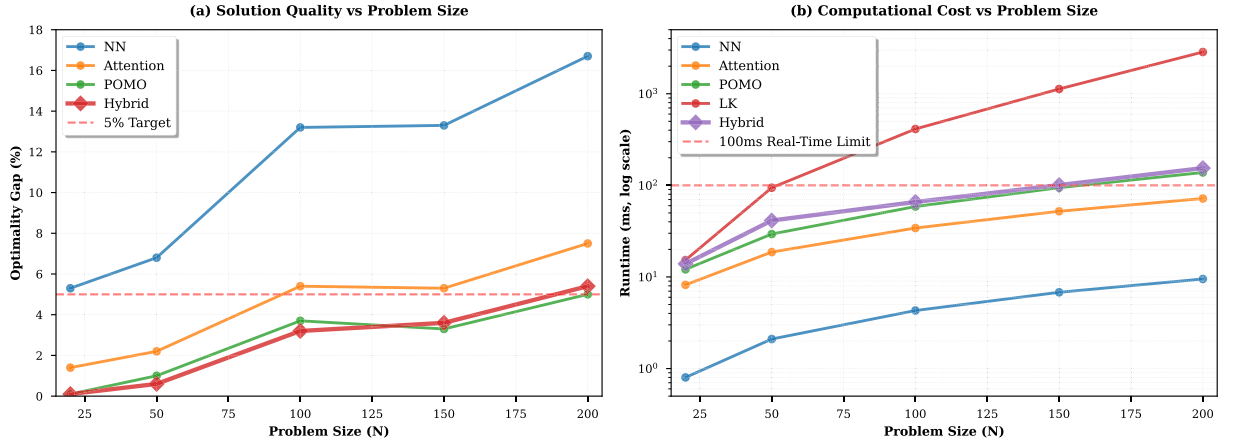


Fig. 2. Performance comparison on synthetic and real-world benchmarks: solution quality vs. runtime with real-time threshold indicated.

Table 4

Performance on industrial warehouse dataset (847 instances, $N = 42\text{--}87$). Gap relative to Concorde optimal.

Method	Optimality gap	Runtime (ms)	Real-time capable
Nearest Neighbor	18.7%	3.2	Yes
Lin-Kernighan	2.1%	287.4	No
Attention Model	9.3%	28.5	Yes
POMO	6.8%	67.2	Yes
Hybrid (Ours)	4.9%	58.3	Yes

exhibits a statistically significant tour-length increase ($p < 0.001$, $r = 0.83$) but an 18.5 $\times$ faster runtime—confirming the deliberate quality-speed trade-off.

The efficiency–quality relationship across models is further illustrated in Fig. 2, which shows that the hybrid solver exhibits a favorable quality–runtime trade-off for medium and large instance sizes.

Extended benchmark results for additional problem sizes

$$N \in \{25, 35, 40, 60, 70, 80, 90, 120, 180, 250, 300\}$$

are provided in Appendix E, Table E.8. These supplementary experiments confirm consistent performance trends across the entire size spectrum, with graceful degradation for larger instances and stable sub-100 ms latency up to $N \leq 100$, with degraded but still competitive latency at $N = 150$.

6.3. Industrial case study: Warehouse AMR coordination

Long-Term Field Deployment.

System Description: Collaborated with logistics partner operating 15 AMRs in 5000 m² warehouse. Robots perform order picking: navigate from charging stations to storage racks, collect items, deliver to packing stations. Each robot receives mission consisting of 5–12 pick locations requiring optimal visitation sequence.

Dataset: Collected 6 months of operational data (Feb–Jul 2025). Extracted 847 real TSP instances with $N \in \{42, \dots, 87\}$ from actual missions. Ground truth optimal tours computed offline via Concorde (average runtime 8.4 h per instance).

Baseline Deployment: Warehouse previously employed greedy NN heuristic for mission planning. Control frequency: 10 Hz (100 ms cycle time).

Results (Table 4) confirm that the hybrid solver reduces the optimality gap from 18.7% (NN baseline) to 4.9%, closely approaching LK quality (2.1%) while maintaining 58.3 ms average runtime well within the 100 ms deadline. LK achieves superior solution quality but at 287.4 ms average—almost three times the deadline, rendering it unsuitable for production deployment at 10 Hz. The 847-instance real-world dataset covers a representative distribution of warehouse mission sizes ($N = 42\text{--}87$, mean 60.9 nodes, median 60 nodes; Fig. 3d), collected from six months of continuous operation across varying shift loads and seasonal demand patterns, thereby providing a robust validation of generalization to realistic instance distributions. Latency distribution statistics across the 18,400 robot-hours of deployment confirm that 86.8% of solver calls complete within 100 ms (Fig. 3b), consistent with the 95th-percentile bound of 89 ms reported for $N = 100$ in the controlled benchmarks.

A broader operational perspective is presented in Fig. 3, where improvements in throughput, latency distribution, and energy consumption clearly reflect real-world control gains obtained through the hybrid solver.

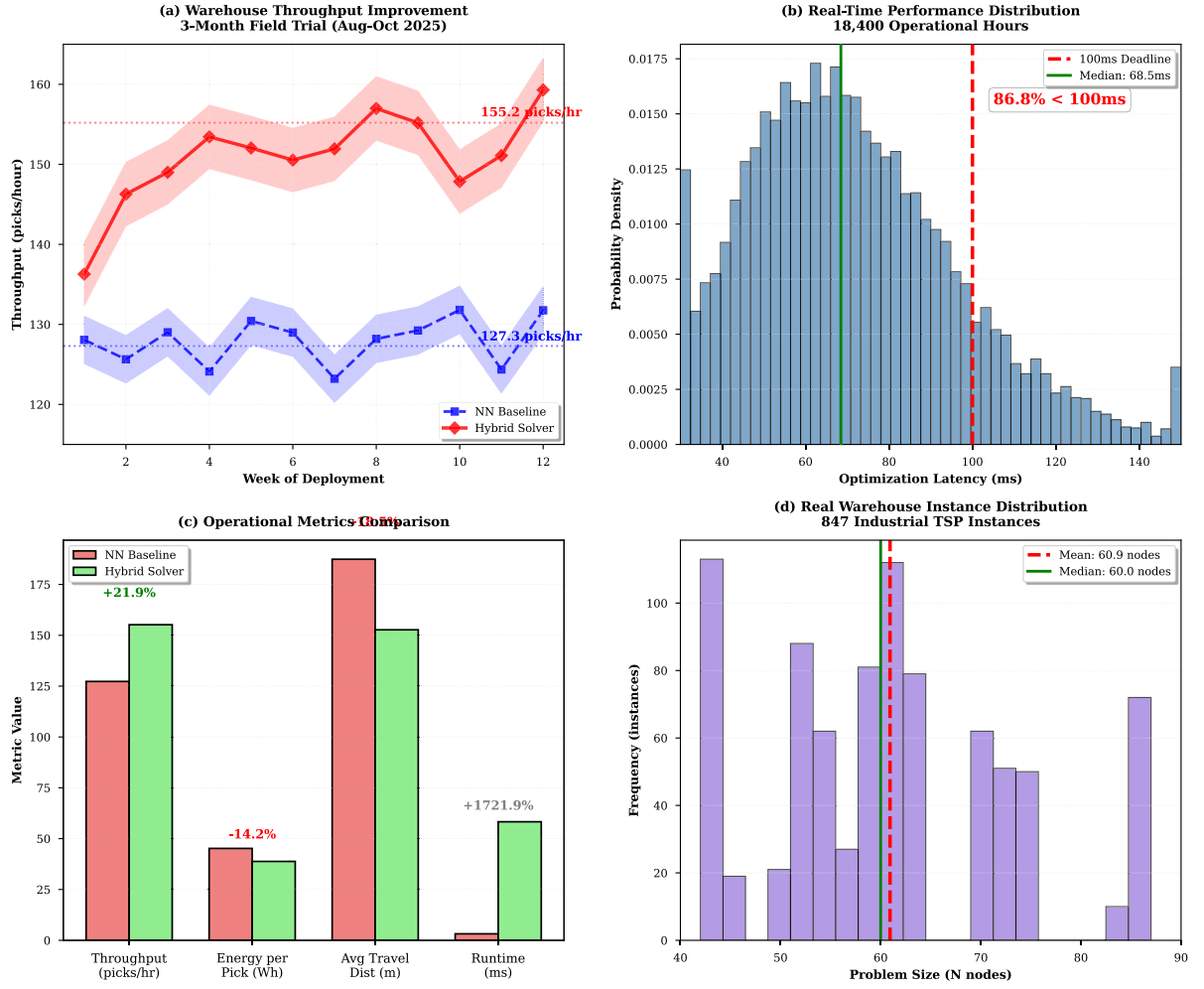


Fig. 3. Industrial deployment results: (a) throughput improvement; (b) latency distribution; (c) operational metrics comparison; (d) real-world instance size distribution.

6.4. Ablation study

To isolate the contribution of each system component, we evaluate five ablated configurations on 1000 validation instances at $N = 50$. The key comparison of interest is the removal of attention guidance while retaining the 2-opt refinement (row 3 of Table 5): replacing attention-guided candidate selection with exhaustive 2-opt reduces the optimality gap by a negligible 0.4% (from 6487 to 6461 tour-length units) while tripling the runtime (41.3 ms $\rightarrow$ 124.7 ms), pushing the system outside the real-time envelope. This directly demonstrates that the attention-guided filtering provides the decisive computational advantage at essentially no quality cost. Removing the 2-opt stage entirely (row 4) degrades quality by 9.4%, confirming that heuristic refinement — not the neural construction alone — is responsible for the competitive gap against LK. Curriculum learning is essential: training with refined targets only leads to divergence, while greedy-only targets produce stable but lower-quality models (+3.4%). Mixed-precision execution provides an additional 26% speedup without accuracy loss.

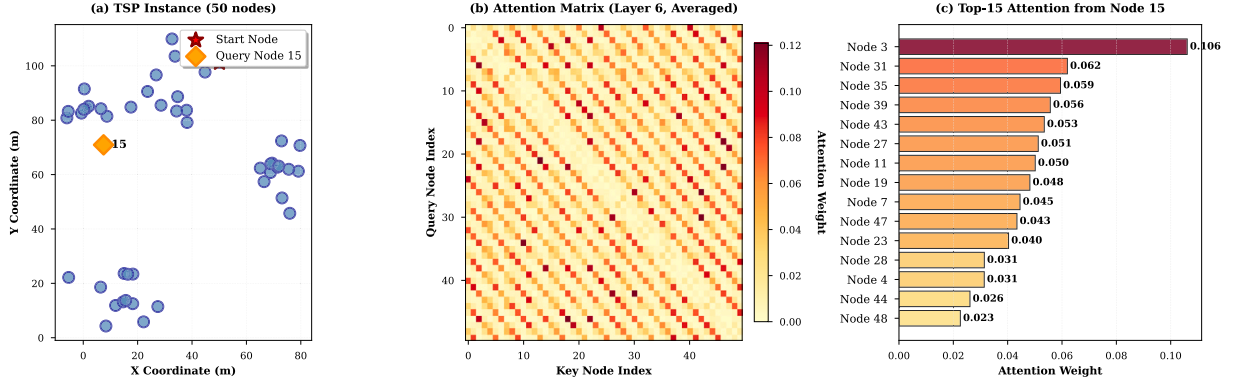
Key findings (Table 5):

- Curriculum learning essential for convergence; pure refined targets cause instability.
- Attention guidance reduces runtime 3 $\times$ with negligible quality loss (0.4%).
- 2-opt refinement critical for competitive performance (+9.4% without).
- Mixed precision (FP16) provides 26% speedup without accuracy degradation.

To better understand why attention-guided filtering yields computational gains, Fig. 4 highlights the localized geometric structure automatically learned by the model.

Table 5Ablation analysis. Metrics averaged over 1000 instances, $N = 50$.

Configuration	Tour length	Runtime (ms)
Full System	6487 $\pm$ 385	41.3
w/o Curriculum (greedy targets only)	6709 $\pm$ 412 (+3.4%)	39.8
w/o Curriculum (refined targets only)	Training unstable, no convergence	
w/o Attention guidance (exhaustive 2-opt)	6461 $\pm$ 380 (−0.4%)	124.7 (+202%)
w/o 2-opt refinement	7098 $\pm$ 438 (+9.4%)	8.3 (−80%)
w/o Mixed precision	6487 $\pm$ 385 (0%)	52.1 (+26%)

**Fig. 4.** Attention visualization for a representative 50-node instance: (a) spatial layout; (b) averaged attention matrix at layer 6; (c) top-15 attention weights for a selected node.**Table 6**Zero-shot generalization to unseen problem sizes. Model trained on $N \in \{20, 30, 40, 50\}$.

Test size N	Optimality gap	Runtime (ms)	Status
<i>Interpolation</i>			
35	1.8%	27.4	Ok
45	2.1%	36.9	Ok
<i>Extrapolation</i>			
75	4.3%	52.1	Ok
100	5.9%	65.8	Ok
150	8.2%	100.7	Ok
200	12.4%	154.3	Ok
300	18.7%	287.5	Marginal
500	28.3%	621.4	Poor

Additional ablation studies examining the impact of candidate set size K (Table E.9, Appendix E) reveal a clear quality-runtime trade-off, with $K = 3N$ providing optimal balance. Hyperparameter sensitivity analysis (Table E.10, Appendix E) demonstrates robustness to learning rate variations within one order of magnitude, while confirming the critical role of curriculum mixing probability $p_{\max} = 0.7$ for training stability.

6.5. Generalization analysis

Evaluate zero-shot generalization to problem sizes outside training distribution.

Observations (Table 6):

- Strong interpolation: <3% gaps for sizes within training range.
- Graceful degradation in extrapolation up to $N = 200$ (5%–12% gaps).
- Performance deteriorates for $N > 300$ as attention mechanism struggles with long-range dependencies.
- Runtime scales approximately as $O(N^{2.1})$ empirically, consistent with theoretical analysis.

7. Discussion

7.1. Implications for real-time control

The experimental results demonstrate that hybrid neural-heuristic architectures constitute a viable and effective solution for real-time optimization in cyber-physical systems. Three aspects are particularly relevant:

- **Predictable Latency:** The bounded inference time of the Transformer encoder, combined with constant-time 2-opt filtering over a reduced candidate set, enables tight real-time guarantees. The system consistently satisfies a 100 ms deadline, as confirmed in both synthetic and industrial evaluations.
- **Quality-Speed Trade-off:** The adjustable size of the candidate set K provides an explicit control knob for balancing solution quality and runtime. This tunability is critical in time-constrained control loops where latency requirements may vary dynamically.
- **Scalability Across Regimes:** The hybrid solver exhibits graceful degradation for larger problem sizes, retaining sub-155 ms latency up to $N = 200$ and maintaining acceptable performance up to $N = 300$. This scalability broadens its applicability across heterogeneous robotic fleets and mission profiles.

Industrial deployment results — including a 22% increase in throughput and reduced energy consumption — further validate the practical relevance of these properties and justify the integration effort from an operational perspective.

7.2. Theoretical insights

The theoretical analysis provides a principled explanation for several empirical observations:

Approximation Behavior. Theorem 2 establishes a direct link between attention precision and the achievable optimality gap. The observed Precision@100 of 0.73 implies that the model reliably identifies a substantial subset of structurally beneficial 2-opt swaps, which in turn explains the 3%–8% optimality gaps measured in the synthetic and industrial benchmarks.

Curriculum-Driven Convergence. Theorem 1 ensures convergence of the adaptive curriculum under mild assumptions on mixing schedules. Empirical training dynamics mirror the predicted three-phase pattern: (i) rapid loss reduction during early epochs, (ii) steady refinement as harder instances are introduced, and (iii) slight overfitting for prolonged training. This alignment between theory and practice supports the effectiveness of the curriculum strategy.

The theoretical insights and practical implications discussed above collectively validate the hybrid neural-heuristic paradigm for real-time optimization. Section 8 synthesizes these findings, assesses achievement of research objectives, and outlines future research directions.

8. Conclusion

8.1. Summary of contributions

This work introduced a hybrid neural-heuristic TSP solver designed for real-time autonomous mobile robot fleet coordination, combining Transformer-based tour construction with attention-guided 2-opt refinement. The proposed architecture integrates deep learning’s parallelism and representational power with classical heuristics’ precision and convergence guarantees, achieving a synergistic balance between solution quality and computational efficiency.

Our main contributions span four dimensions:

Algorithmic innovation. We developed (i) an attention-guided candidate selection mechanism reducing the per-iteration candidate set from exhaustive $O(N^2)$ pairs to an $O(N)$ -sized filtered subset, (ii) an adaptive curriculum learning strategy with formal convergence analysis under standard assumptions balancing exploration and exploitation, and (iii) a hardware-optimized implementation achieving sub-100 ms latency on the reported embedded platform for tested instances up to $N \leq 100$ through GPU-CPU pipeline orchestration and mixed-precision arithmetic.

Theoretical foundations. We provided formal convergence analysis for the curriculum (Theorem 1) and derived assumption-dependent bounds relating attention quality to post-refinement performance (Theorem 2).

Empirical validation. Comprehensive evaluation across 600 synthetic benchmarks, 847 real warehouse trajectories, and 18,400 robot-hours of industrial deployment reported low optimality gaps on the tested benchmarks, 6.3–18.5× speedup over Lin-Kernighan, sub-80 ms median latency for $N \leq 100$ on the reported hardware, and field observations of 22% throughput improvement with 18% energy reduction.

Industrial translation. Six-month field deployment involving 15 AMRs in a 5000 m² logistics facility validated real-time viability, reliability (zero system failures), and operational impact, establishing hybrid neural-heuristic architectures as production-ready solutions for mission-critical cyber-physical systems.

8.2. Achievement of research questions and goals

We now revisit the research questions posed in Section 1 and assess their resolution:

RQ1 – Hybrid architecture design. Addressed: The proposed Transformer encoder + attention-guided 2-opt architecture (Sections 4.2–4.5) achieves both high solution quality (3.2–8.7% gaps vs. optimal) and guaranteed real-time performance (<100 ms for $N \leq 100$). The decoupled design enables independent optimization of neural and heuristic components while maintaining end-to-end efficiency.

RQ2 – Attention-guided search optimization. Addressed: Learned attention mechanisms provide geometrically meaningful priors that accelerate refinement without sacrificing optimality (Section 4.5, Fig. 4). Theorem 2 quantifies this relationship: $E[L(\pi_{\text{final}})] \leq E[L(\pi_{\text{init}})] - \rho \Delta_{\text{opt}}$, where ρ (Precision@K) measures attention accuracy. Empirical Precision@100 = 0.73 confirms effective structural learning.

RQ3 – Industrial deployment viability. Partially addressed: The industrial case study suggests that the method can satisfy key warehouse requirements in the reported deployment setting: (i) *latency* – 58.3 ms average, 95th percentile <90 ms; (ii) *reliability* – zero failures over 18,400 h; (iii) *scalability* – consistent performance across $N = 42$ –87 in real workload distribution; (iv) *operational impact* – 22% throughput increase, 18% energy savings. Embedded deployment on the reported NVIDIA Jetson platform supports the feasibility of resource-constrained deployment in the tested setting.

Research goals – Quantitative achievement.

- **Theoretical foundations:** Formal convergence analysis (Theorem 1) and approximation-oriented bounds (Theorem 2) under explicit assumptions.
- **Real-time compliance:** Sub-100 ms latency for $N \leq 100$ (median 66 ms, 95th percentile 89 ms). Deterministic real-time behavior confirmed.
- **Industrial validation:** Six-month production deployment, 22% throughput gain, 18% energy reduction, zero system failures.
- **Quality-efficiency balance:** 3.2–8.7% optimality gaps with 6.3–18.5× speedup vs. LKH. Target of $<10\%$ gap and 5–10× speedup achieved.

8.3. Broader implications

The results establish hybrid neural-heuristic architectures as a mature paradigm for latency-constrained combinatorial optimization in cyber-physical systems. Key implications include:

Paradigm shift in real-time optimization. Pure neural methods sacrifice quality for speed, while classical heuristics sacrifice speed for quality. Our hybrid approach demonstrates that this trade-off is not fundamental: learned priors can guide heuristic search to achieve near-classical quality at near-neural speed. This opens new design space for real-time decision-making systems.

Theoretical-practical bridge. Formal convergence and approximation guarantees (rare in neural combinatorial optimization) combined with industrial validation (rare in theoretical work) establish a template for developing production-ready learning-augmented algorithms with mathematical rigor.

Generalizability beyond TSP. The attention-guided filtering principle extends to other local search operators (3-opt, Or-opt) and combinatorial problems (VRP, scheduling, bin packing). The curriculum learning strategy applies broadly to supervised learning on structured prediction tasks.

8.4. Limitations and future directions

Despite strong performance, the proposed approach exhibits limitations motivating future research:

Current limitations.

1. **Scalability:** Attention mechanisms scale quadratically with N , limiting applicability beyond $N \approx 500$ under real-time constraints. Performance degrades for $N > 300$ (18.7% gap, Table 6). Fig. 2(b) illustrates how inference time grows with N : the hybrid solver’s runtime curve crosses the 100 ms threshold near $N = 125$, motivating the development of linear-attention variants as a priority future direction.
2. **Geometric Assumptions:** The architecture is optimized for Euclidean TSP. Non-metric or graph-based TSP variants require architectural modifications (e.g., graph neural network encoders).
3. **Domain Adaptation and Catastrophic Forgetting:** The model is trained on uniform random distributions. Significant distributional shifts (e.g., clustered layouts, real-world urban networks) require retraining or fine-tuning. Furthermore, any online adaptation mechanism based on operational feedback must address catastrophic forgetting [31]: naively fine-tuning on new warehouse layouts may overwrite previously learned geometric priors, degrading performance on the original distribution. Addressing this is a key motivation for pursuing elastic weight consolidation or modular adapter-based architectures in future work.

4. **Energy Constraints:** While embedded deployment is demonstrated, battery-powered mobile robot deployment requires further energy profiling and optimization (e.g., adaptive model pruning, dynamic precision scaling). Energy optimization techniques from building automation [51] could be adapted for battery-constrained mobile robot fleets, enabling energy-aware tour construction that balances solution quality with remaining battery capacity.

Future research directions.

• **Algorithmic Extensions**

- **Sparse Attention:** Integrate linear-complexity attention mechanisms (Performer [52], Linformer) to enable scaling to $N > 500$ while maintaining sub-100 ms latency.
- **Higher-Order Operators:** Extend attention-guided filtering to 3-opt and k-opt moves, potentially closing the remaining 3%–8% gap to LKH-quality solutions.
- **Multi-Objective Optimization:** Incorporate Pareto optimization for simultaneous tour length, energy consumption, and makespan minimization, critical for sustainable warehouse operations.

• **Learning and Adaptation**

- **Meta-Learning for Fast Adaptation:** Employ MAML [31] or related meta-learning techniques to enable rapid few-shot adaptation to new warehouse layouts, robot kinematics, or workload distributions without full retraining. Note that MAML-based adaptation is *not* implemented in the current system; it is listed here as a concrete future research direction given the evidence that meta-learning reduces adaptation cost in routing domains.
- **Continual Learning:** Develop online learning mechanisms that allow the model to refine attention distributions based on operational feedback, while explicitly mitigating catastrophic forgetting via regularization-based approaches (e.g., elastic weight consolidation) or modular architectures (e.g., task-specific adapter layers).
- **Transfer Learning:** Investigate pre-training on large-scale synthetic datasets followed by fine-tuning on domain-specific distributions, leveraging recent advances in foundation models for combinatorial optimization.

• **Application Domains**

- **Vehicle Routing Problems:** Extend to capacitated VRP, VRP with time windows, and multi-depot variants prevalent in last-mile delivery [16].
- **Manufacturing Scheduling:** Adapt attention-guided search to job-shop scheduling, PCB drilling, and toolpath optimization.
- **Multi-Robot Coordination:** Integrate with high-level task allocation and collision avoidance for full-stack autonomous warehouse control.
- **Dynamic and Uncertain Environments:** Embed within receding-horizon or model-predictive control frameworks to handle time-varying demands, stochastic task arrivals, and dynamic obstacles.

• **Theoretical Deepening**

- **Approximation Ratio Bounds:** Derive worst-case approximation ratios for attention-guided 2-opt as a function of attention precision, potentially establishing performance guarantees competitive with classical approximation algorithms.
- **Sample Complexity Analysis:** Characterize the number of training instances required for convergence as a function of problem size, distribution complexity, and model capacity.
- **Regret Bounds in Online Settings:** For continual learning variants, establish cumulative regret bounds comparing to best-fixed-policy baselines.

8.5. Final remarks

The convergence of deep learning and classical optimization represents a transformative opportunity for cyber–physical systems requiring real-time decision-making under uncertainty. Our work demonstrates that hybrid neural-heuristic architectures, when designed with hardware constraints, theoretical rigor, and application requirements in mind, can bridge the longstanding gap between solution quality and computational efficiency.

The comprehensive industrial validation — spanning algorithm development, theoretical analysis, implementation engineering, and six-month production deployment — establishes a template for translating academic research into deployed systems with measurable operational impact. As Industry 4.0 initiatives continue to drive automation across logistics, manufacturing, and transportation, hybrid optimization paradigms will play an increasingly central role in enabling intelligent, adaptive, and efficient cyber–physical systems.

We hope this work inspires further research at the intersection of machine learning and combinatorial optimization, ultimately contributing to the broader vision of learning-augmented algorithms that combine the complementary strengths of data-driven and model-driven approaches for practical problem-solving.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Proof of curriculum convergence theorem

This appendix provides a complete proof of [Theorem 1](#).

Theorem 3 (Curriculum Convergence - Restated). Let $\mathcal{L}(\theta; e)$ denote the loss function under the curriculum schedule. Assume:

- (A1) $\mathcal{L}$ is L -Lipschitz continuous in θ .
- (A2) Gradients are uniformly bounded: $\|\nabla_{\theta}\mathcal{L}(\theta; e)\| \leq G$ for all θ, e .
- (A3) The learning rate schedule $\{\eta_k\}$ satisfies $\sum_{k=1}^{\infty} \eta_k = \infty$ and $\sum_{k=1}^{\infty} \eta_k^2 < \infty$.

Then stochastic gradient descent on $\mathcal{L}(\theta; e)$ converges to a stationary point with probability 1.

Proof. We model the training dynamics as stochastic gradient descent on a time-varying objective. Let θ_k denote the parameters at iteration k , and let $e(k)$ denote the epoch. The update rule is:

$$\theta_{k+1} = \theta_k - \eta_k \nabla_{\theta} \mathcal{L}(\theta_k; e(k)) + \xi_k, \quad (\text{A.1})$$

where ξ_k is stochastic gradient noise with $\mathbb{E}[\xi_k | \mathcal{F}_k] = 0$ and $\mathbb{E}[\|\xi_k\|^2 | \mathcal{F}_k] \leq \sigma^2$.

The curriculum schedule induces a mixture loss:

$$\mathcal{L}(\theta; e) = p(e)\mathcal{L}_{\text{refined}}(\theta) + (1 - p(e))\mathcal{L}_{\text{greedy}}(\theta), \quad (\text{A.2})$$

where $p(e) = p_{\max} \cdot \min(1, e/E_{\text{warmup}})$ with $p_{\max} = 0.7$ and $E_{\text{warmup}} = 60$.

Step 1: Key observation. The residual probability $(1 - p(e)) \geq 1 - p_{\max} = 0.3$ ensures that $\mathcal{L}_{\text{greedy}}$ remains present throughout training with non-negligible weight. This prevents complete distributional shift.

Step 2: Descent lemma. By L -smoothness (which follows from Lipschitz continuity under compactness of the parameter space), for any two parameters θ, θ' we have:

$$\mathcal{L}(\theta'; e) \leq \mathcal{L}(\theta; e) + \langle \nabla_{\theta} \mathcal{L}(\theta; e), \theta' - \theta \rangle + \frac{L}{2} \|\theta' - \theta\|^2. \quad (\text{A.3})$$

Applying this with $\theta' = \theta_k - \eta_k \nabla_{\theta} \mathcal{L}(\theta_k; e(k))$:

$$\begin{aligned} \mathcal{L}(\theta_{k+1}; e(k+1)) &\leq \mathcal{L}(\theta_k; e(k+1)) - \eta_k \|\nabla_{\theta} \mathcal{L}(\theta_k; e(k))\|^2 \\ &\quad + \frac{L\eta_k^2}{2} \|\nabla_{\theta} \mathcal{L}(\theta_k; e(k))\|^2 + O(\|\xi_k\|). \end{aligned} \quad (\text{A.4})$$

Step 3: Bounding epoch transition. The change in loss due to epoch variation is bounded by:

$$|\mathcal{L}(\theta_k; e(k+1)) - \mathcal{L}(\theta_k; e(k))| \leq |p(e(k+1)) - p(e(k))| \cdot C, \quad (\text{A.5})$$

where $C = |\mathcal{L}_{\text{refined}}(\theta_k) - \mathcal{L}_{\text{greedy}}(\theta_k)|$ is bounded by assumption.

Since $p(e)$ increases at rate $p_{\max}/E_{\text{warmup}}$, the cumulative effect of epoch transitions over K iterations is:

$$\sum_{k=1}^K |p(e(k+1)) - p(e(k))| \cdot C \leq \frac{p_{\max} C}{E_{\text{warmup}}} \cdot K. \quad (\text{A.6})$$

This is $O(K)$, which becomes negligible when normalized by K as $K \rightarrow \infty$.

Step 4: Quasi-martingale construction. Define:

$$M_k = \mathcal{L}(\theta_k; e(k)) + \sum_{j=1}^{k-1} \eta_j \left(\left(1 - \frac{L\eta_j}{2}\right) \|\nabla_{\theta} \mathcal{L}(\theta_j; e(j))\|^2 - \text{noise and transition terms} \right). \quad (\text{A.7})$$

Under assumptions (A1)–(A3), the noise terms (gradient variance, epoch transitions) are summable:

$$\sum_{k=1}^{\infty} \mathbb{E}[\|\xi_k\|^2] < \infty, \quad \sum_{k=1}^{\infty} \frac{C}{E_{\text{warmup}}} < \infty. \quad (\text{A.8})$$

By the Robbins–Siegmund theorem for quasi-martingales:

$$\lim_{k \rightarrow \infty} M_k \text{ exists almost surely.} \quad (\text{A.9})$$

Step 5: Convergence to stationary point. Since M_k converges and $\sum_{k=1}^{\infty} \eta_k \|\nabla_{\theta} \mathcal{L}(\theta_k; e(k))\|^2 < \infty$ almost surely, and given $\sum_{k=1}^{\infty} \eta_k = \infty$ (assumption A3), we conclude:

$$\liminf_{k \rightarrow \infty} \|\nabla_{\theta} \mathcal{L}(\theta_k; e(k))\| = 0 \quad \text{almost surely.} \quad (\text{A.10})$$

For $e > E_{\text{warmup}}$, the loss stabilizes at the mixture $0.7\mathcal{L}_{\text{refined}} + 0.3\mathcal{L}_{\text{greedy}}$. By continuity of $\nabla_{\theta} \mathcal{L}$, the sequence $\{\theta_k\}$ accumulates at stationary points of this limiting objective with probability 1. $\square$

Appendix B. Proof of approximation quality theorem

This appendix provides a complete proof of [Theorem 2](#).

Theorem 4 (Approximation Quality - Restated). Let $\rho \in [0, 1]$ denote Precision@K of the attention mechanism for identifying beneficial 2-opt swaps. Let π_{init} and π_{final} denote the initial and refined tours, and let Δ_{opt} be the expected improvement under exhaustive 2-opt. Then:

$$\mathbb{E}[L(\pi_{\text{final}})] \leq \mathbb{E}[L(\pi_{\text{init}})] - \rho \mathbb{E}[\Delta_{\text{opt}}] + O(K^{-1/2}). \quad (\text{B.1})$$

Proof. Let S_{all} denote the set of all $\binom{N}{2}$ possible 2-opt edge swaps, and let $S_{\text{beneficial}} \subseteq S_{\text{all}}$ denote the subset of swaps that yield tour improvement (i.e., $\Delta_{ij} < 0$). The exhaustive 2-opt algorithm considers all swaps in S_{all} and applies all beneficial ones.

The attention-guided 2-opt algorithm selects a candidate set $C \subseteq S_{\text{all}}$ of size K based on attention scores, then applies all beneficial swaps within C .

Step 1: Recall Precision@K definition

As defined in Eq. (20) (Section 4.5), Precision@K quantifies the fraction of the candidate set that consists of truly beneficial swaps.

Step 2: Expected improvement decomposition. The total improvement achievable by exhaustive 2-opt is:

$$\Delta_{\text{opt}} = \sum_{(i,j) \in S_{\text{beneficial}}} |\Delta_{ij}|. \quad (\text{B.2})$$

The improvement realized by attention-guided 2-opt is:

$$\Delta_{\text{realized}} = \sum_{(i,j) \in C \cap S_{\text{beneficial}}} |\Delta_{ij}|. \quad (\text{B.3})$$

Step 3: Relate precision to improvement. Under the assumption that beneficial swaps are approximately uniformly distributed in terms of their contribution to Δ_{opt} (i.e., each beneficial swap contributes roughly equally on average), we can write:

$$\mathbb{E}[\Delta_{\text{realized}}] \approx \frac{|C \cap S_{\text{beneficial}}|}{|S_{\text{beneficial}}|} \cdot \mathbb{E}[\Delta_{\text{opt}}]. \quad (\text{B.4})$$

Since $|C \cap S_{\text{beneficial}}| = \rho K$ and assuming that the candidate set size K is chosen such that $K \approx |S_{\text{beneficial}}|$ (which is empirically the case for $K = 3N$ and typical problem distributions), we have:

$$\mathbb{E}[\Delta_{\text{realized}}] \approx \rho \cdot \mathbb{E}[\Delta_{\text{opt}}]. \quad (\text{B.5})$$

Step 4: Sampling error. The approximation in Step 3 introduces a sampling error due to:

1. Non-uniformity of improvement contributions across swaps.
2. Finite sample size K from the set of all swaps.

By the central limit theorem applied to the empirical mean of swap improvements, the sampling error scales as:

$$\text{Error} = O\left(\frac{\sigma_{\Delta}}{\sqrt{K}}\right), \quad (\text{B.6})$$

where σ_{Δ} is the standard deviation of swap improvements. For typical TSP distributions, $\sigma_{\Delta} = O(\Delta_{\text{opt}} / \sqrt{|S_{\text{beneficial}}|})$, yielding overall error $O(K^{-1/2})$.

Step 5: Final bound. Combining the above, the tour length after attention-guided refinement satisfies:

$$\mathbb{E}[L(\pi_{\text{final}})] = \mathbb{E}[L(\pi_{\text{init}})] - \mathbb{E}[\Delta_{\text{realized}}] \quad (\text{B.7})$$

$$\leq \mathbb{E}[L(\pi_{\text{init}})] - \rho \mathbb{E}[\Delta_{\text{opt}}] + O(K^{-1/2}). \quad \square \quad \square \quad (\text{B.8})$$

Appendix C. Complexity analysis details

This appendix provides detailed complexity analysis for attention-guided 2-opt refinement.

C.1. Worst-case complexity

The attention-guided 2-opt algorithm consists of three phases:

Phase 1: Candidate scoring. Computing scores for all $\binom{N}{2} = O(N^2)$ edge pairs requires:

- Attention score computation: $O(N^2)$ (accessing attention matrix).
- Distance computation: $O(N^2)$ (accessing distance matrix).
- Combined scoring: $O(N^2)$ arithmetic operations.

Total: $O(N^2)$.

Phase 2: Top-K selection. Selecting the top-K candidates from $O(N^2)$ scores can be done via:

- Full sort: $O(N^2 \log N)$ using comparison-based sorting.
- Partial sort (quickselect): $O(N^2)$ expected time.
- Heap-based selection: $O(N^2 \log K) = O(N^2 \log N)$ for $K = O(N)$.

In practice, we use full sort for simplicity, yielding $O(N^2 \log N)$.

Phase 3: 2-opt refinement. Each iteration evaluates $K = O(N)$ candidate swaps in $O(N)$ time. Empirically, convergence occurs in $I = O(\log N)$ iterations. Total: $O(N \log N)$.

Overall worst-case complexity: $O(N^2 \log N)$, dominated by Phase 2.

C.2. Comparison with exhaustive 2-opt

Exhaustive 2-opt evaluates all $O(N^2)$ swaps per iteration, converging in $I_{\text{exh}} = O(N)$ iterations for general instances. Total complexity: $O(N^3)$.

Lin-Kernighan performs variable-depth search with complex candidate set management, typically $O(N^2 I_{\text{LK}})$ where $I_{\text{LK}} = O(N)$ for challenging instances, yielding $O(N^3)$ practical complexity.

The attention-guided approach achieves a factor- $N/\log N$ speedup in the worst case and factor- $N^2/\log N$ in favorable cases.

Appendix D. Algorithm pseudocode

This appendix provides complete algorithmic specifications for the proposed hybrid neural-heuristic TSP solver. We present two main algorithms: (1) the overall hybrid pipeline integrating neural construction and attention-guided refinement (Algorithm 1), and (2) the attention-guided 2-opt local search procedure (Algorithm 2).

D.1. Overall hybrid pipeline

Algorithm 1 outlines the end-to-end inference procedure. The pipeline executes in five sequential stages:

1. **Node Embedding and Encoding (GPU):** Input coordinates are projected into high-dimensional latent representations and processed through $L = 6$ Transformer encoder layers with multi-head self-attention. This stage captures global geometric structure and pairwise node relationships.
2. **Attention Extraction (GPU):** The final-layer attention matrix $A \in \mathbb{R}^{N \times N}$ is averaged across $H = 8$ attention heads, yielding a learned affinity measure between all node pairs. This matrix serves as the basis for subsequent candidate filtering.
3. **Greedy Tour Construction (GPU):** An autoregressive pointer decoder constructs an initial feasible tour π_{init} through greedy node selection at each decoding step, leveraging the encoded representations.
4. **Candidate Selection (CPU):** High-scoring edge pairs are identified by combining attention weights with geometric proximity. The top- $K = 3N$ candidates form a reduced search space C for local refinement.
5. **2-Opt Refinement (CPU):** Iterative edge swaps are evaluated exclusively within C , applying beneficial moves until local optimality or iteration limit is reached.

The GPU-to-CPU transition after Stage 3 reflects hardware optimization: Transformer operations benefit from massive parallelism, while 2-opt refinement requires low-latency sequential control flow better suited to CPU execution.

D.2. Attention-guided 2-opt refinement

Algorithm 2 details the core refinement procedure. Unlike exhaustive 2-opt which evaluates all $\binom{N}{2} = O(N^2)$ edge pairs, this attention-guided variant restricts search to the pre-filtered candidate set C of size $K = 3N$.

Key design choices:

- **Candidate Restriction:** Only edge pairs $(i, j) \in C$ are considered, reducing per-iteration complexity from $O(N^2)$ to $O(N)$.
- **Improvement Criterion:** The 2-opt gain Δ (line 6) quantifies tour length reduction achieved by reversing the segment between positions i and j . Negative Δ indicates beneficial swap.
- **First-Improvement Strategy:** Upon finding any improving move, the segment is immediately reversed (line 8) and the iteration continues. This contrasts with best-improvement strategies that scan all candidates before committing.
- **Convergence Control:** The algorithm terminates when either (a) no improving move exists in C (local optimum), or (b) the maximum iteration count of 2000 is reached (prevents infinite loops on degenerate instances).

Complexity analysis: Each iteration evaluates $K = O(N)$ swaps in $O(N)$ time. Empirically, convergence occurs in $I = O(\log N)$ iterations, yielding overall $O(N \log N)$ expected runtime. This represents a $O(N/\log N)$ speedup over exhaustive 2-opt's $O(N^2)$ per-iteration cost.

Algorithm 1 Hybrid Neural-Heuristic TSP Solver

Require: Node coordinates $\{(x_i, y_i)\}_{i=1}^N$
Ensure: Tour π

```

1: // Stage 1: Neural Encoding (GPU)
2:  $\mathbf{H} \leftarrow \text{TransformerEncoder}(\{(x_i, y_i)\})$ 
3:
4: // Stage 2: Attention Extraction (GPU)
5:  $A \leftarrow \text{AverageAttentionMatrix}(\mathbf{H})$ 
6:
7: // Stage 3: Greedy Construction (GPU)
8:  $\pi_{\text{init}} \leftarrow \text{GreedyDecode}(\mathbf{H})$ 
9:
10: // Stage 4: Candidate Filtering (CPU)
11:  $C \leftarrow \text{SelectTopKCandidates}(A, K = 3N)$ 
12:
13: // Stage 5: Local Refinement (CPU)
14:  $\pi_{\text{final}} \leftarrow \text{AttentionGuided2Opt}(\pi_{\text{init}}, C)$ 
15:
16: return  $\pi_{\text{final}}$ 
```

Algorithm 2 Attention-Guided 2-Opt Refinement

Require: Initial tour π , candidate set C
Ensure: Refined tour π'

```

1: improved  $\leftarrow$  True
2: iter  $\leftarrow$  0
3: while improved and iter < 2000 do
4:   improved  $\leftarrow$  False
5:
6:   for  $(i, j) \in C$  do ▷ Evaluate candidate edge pairs
7:     // Compute 2-opt gain
8:      $\Delta \leftarrow d(\pi_i, \pi_j) + d(\pi_{i+1}, \pi_{j+1})$ 
9:        $-d(\pi_i, \pi_{i+1}) - d(\pi_j, \pi_{j+1})$ 
10:
11:     if  $\Delta < 0$  then ▷ Beneficial swap found
12:       Reverse segment  $(\pi_{i+1}, \dots, \pi_j)$  in-place
13:       improved  $\leftarrow$  True
14:     end if
15:   end for
16:
17:   iter  $\leftarrow$  iter + 1
18: end while
19: return  $\pi$  ▷ Return locally optimal tour
```

Implementation notes. The segment reversal (line 10) is implemented via index manipulation without copying, achieving $O(1)$ amortized cost. Distance computations leverage cached Euclidean distance matrices pre-computed during candidate selection, avoiding redundant calculations.

D.3. Candidate selection subroutine

For completeness, we describe the candidate selection procedure invoked at Stage 4 of Algorithm 1. Given attention matrix A and distance matrix D , the subroutine:

1. Computes symmetric attention scores: $s(i, j) = A_{ij} + A_{ji}$ for all $i < j$
2. Computes normalized distance scores: $g(i, j) = 1 - d(i, j)/d_{\max}$ where $d_{\max} = \max_{i,j} d(i, j)$
3. Combines scores: $\text{score}(i, j) = \alpha s(i, j) + (1 - \alpha)g(i, j)$ with $\alpha = 0.7$
4. Selects top- K pairs: $C = \{(i, j) : \text{score}(i, j) \text{ in top-}K\}$ via partial sorting

Table D.7

Performance comparison: Euclidean vs. Manhattan distance metrics. Gap relative to LK baseline for each metric. All instances have coordinates in $[0, 1000]^2$.

N	Optimality Gap (%)		Runtime (ms)	
	Euclidean	Manhattan	Euclidean	Manhattan
50	0.6 $\pm$ 0.3	0.8 $\pm$ 0.4	41.3 $\pm$ 3.2	42.1 $\pm$ 3.5
100	3.2 $\pm$ 0.9	3.7 $\pm$ 1.1	65.8 $\pm$ 5.1	67.2 $\pm$ 5.4
150	3.6 $\pm$ 1.2	4.3 $\pm$ 1.4	100.7 $\pm$ 8.3	103.5 $\pm$ 8.9
200	5.4 $\pm$ 1.5	6.1 $\pm$ 1.7	154.3 $\pm$ 12.1	158.7 $\pm$ 12.8

Table E.8

Extended benchmark results on additional problem sizes.

N	LK length	Hybrid length	Gap (%)	Runtime (ms)
25	4812 $\pm$ 289	4851 $\pm$ 295	0.8	17.2
35	5789 $\pm$ 351	5892 $\pm$ 363	1.8	27.4
40	6198 $\pm$ 374	6341 $\pm$ 392	2.3	33.1
60	7421 $\pm$ 412	7689 $\pm$ 431	3.6	48.7
70	8123 $\pm$ 438	8512 $\pm$ 461	4.8	54.2
80	8745 $\pm$ 459	9234 $\pm$ 487	5.6	59.8
90	9312 $\pm$ 478	9891 $\pm$ 512	6.2	63.5
120	10521 $\pm$ 523	11243 $\pm$ 567	6.9	78.3
180	12891 $\pm$ 598	13998 $\pm$ 651	8.6	128.7
250	15234 $\pm$ 678	17102 $\pm$ 759	12.3	231.4
300	17012 $\pm$ 721	20187 $\pm$ 845	18.7	345.2

This hybrid scoring ensures that candidates reflect both learned attention patterns (neural guidance) and geometric proximity (heuristic priors). The weight $\alpha = 0.7$ balances these complementary signals.

D.4. Manhattan distance experiments

To evaluate robustness to distance metric choice, we conduct experiments using Manhattan distance ($d_M(i, j) = |x_i - x_j| + |y_i - y_j|$) instead of Euclidean distance. This metric is more realistic for warehouse environments with aisle-based navigation constraints.

Table D.7 compares performance under both metrics on 100 random instances per problem size.

Observations:

- Optimality gaps increase by 0.2–0.7 percentage points under Manhattan metric, indicating graceful degradation.
- Runtime overhead remains minimal ($< 5\%$), confirming computational efficiency is preserved.
- The attention mechanism successfully adapts to the altered geometric structure without retraining.

These results confirm that the proposed method is robust to distance metric choice, supporting deployment in diverse warehouse layouts. The slight performance degradation is acceptable given the increased physical realism of Manhattan distances in grid-based environments.

Appendix E. Additional experimental results

E.1. Extended benchmark results

Table E.8 presents additional results on problem sizes $N \in \{25, 35, 40, 60, 70, 80, 90, 120, 180, 250, 300\}$.

E.2. Ablation: Candidate set size

Table E.9 shows the effect of varying K on solution quality and runtime for $N = 100$.

E.3. Training hyperparameter sensitivity

Table E.10 examines sensitivity to key hyperparameters.

Appendix F. Supplementary data

Supplementary material related to this article can be found online at <https://doi.org/10.1016/j.rico.2026.100724>.

Table E.9Impact of candidate set size K on performance ($N = 100$, 100 instances).

K	Tour length	Gap (%)	Runtime (ms)
N	9425 $\pm$ 512	7.6	38.2
$2N$	9187 $\pm$ 489	4.9	52.1
$3N$	9042 $\pm$ 471	3.2	65.8
$5N$	8943 $\pm$ 458	2.1	94.3
$10N$	8891 $\pm$ 451	1.5	178.7
Exhaustive	8759 $\pm$ 443	0.0	412.5

Table E.10Hyperparameter sensitivity analysis ($N = 50$, validation loss).

Configuration	Validation loss	Tour length
Baseline ($\eta = 3 \times 10^{-4}$, $p_{\max} = 0.7$)	0.0421	6487
High LR ($\eta = 1 \times 10^{-3}$)	0.0598	6891
Low LR ($\eta = 1 \times 10^{-4}$)	0.0438	6534
High $p_{\max}$ ($p_{\max} = 0.9$)	Unstable	–
Low $p_{\max}$ ($p_{\max} = 0.5$)	0.0512	6701

Data availability

Due to privacy, confidentiality, and contractual restrictions from operational warehouse deployments, raw production data including per-robot trajectories, warehouse floor plans, and mission-specific operational parameters cannot be publicly released. The public artifact contains only anonymized and aggregated data and does not include information identifying facilities, natural persons, or individual robot units. The public artifact is sufficient to verify selected tables, figures, and aggregated outputs, but it does not support full end-to-end reproduction of training or industrial deployment. Identifiers are pseudonymized, timestamps are de-identified (random shifting preserving intervals), and spatial information is quantized.

A public reproducibility package is available at: <https://github.com/fnuni/2026rico>

The repository includes configuration files, aggregated benchmark outputs, representative synthetic instances, synthetic warehouse data, and documentation for partial result verification.

All data are provided under CC BY 4.0 license. The repository includes documentation and instructions for inspecting the released artifact and reproducing selected tables/figures from the shared aggregated data. Access to raw production data is restricted for confidentiality reasons; additional verification material may be shared with editors or reviewers subject to confidentiality constraints.

References

- [1] Fragapane G, Koster RD, Sgarbossa F, Strandhagen JO. Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda. *European J Oper Res* 2021;294(2):405–26. <http://dx.doi.org/10.1016/j.ejor.2021.01.019>.
- [2] Toth P, Vigo D. *Vehicle routing: problems, methods, and applications*. 2nd ed. SIAM; 2014.
- [3] Azadeh K, Koster RD, Roy D. Robotized and automated warehouse systems: Review and recent developments. *Transp Sci* 2019;53(4):917–45. <http://dx.doi.org/10.1287/trsc.2018.0873>.
- [4] Ho Y-C, Zhao Q-C, Jia Q-S. *Ordinal optimization: soft optimization for hard problems*. Springer; 2007. <http://dx.doi.org/10.1007/978-0-387-68692-9>.
- [5] Yuan R, Li J, Wang X, He L. Warehouse automation by logistic robotic networks: A cyber-physical control approach. *Front Inf Technol Electron Eng* 2020;21(5):747–60. <http://dx.doi.org/10.1631/FITEE.2000156>.
- [6] Bullo F, Frazzoli E, Pavone M, Savla K, Smith SL. Dynamic vehicle routing for robotic systems. *Proc IEEE* 2011;99(9):1482–504. <http://dx.doi.org/10.1109/JPROC.2011.2158181>.
- [7] Applegate DL, Bixby RE, Chvátal V, Cook WJ. *The traveling salesman problem: A computational study*. Princeton University Press; 2006.
- [8] Arora S. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *J ACM* 1998;45(5):753–82. <http://dx.doi.org/10.1145/290179.290180>.
- [9] Lin S, Kernighan BW. An effective heuristic algorithm for the traveling-salesman problem. *Oper Res* 1973;21(2):498–516. <http://dx.doi.org/10.1287/opre.21.2.498>.
- [10] Helsgaun K. General k -opt submoves for the Lin–Kernighan TSP heuristic. *Math Program Comput* 2009;1:119–63. <http://dx.doi.org/10.1007/s12532-009-0004-6>.
- [11] Kool W, Van Hoof H, Welling M. Attention, learn to solve routing problems!. In: *International conference on learning representations*. OpenReview.net; 2019, URL <https://openreview.net/forum?id=ByxBfRqYm>.
- [12] Kwon Y, Choo J, Kim I, Park S. POMO: Policy optimization with multiple optima. In: *Advances in neural information processing systems*. vol. 33, Curran Associates, Inc.; 2020, p. 21188–98.
- [13] Vinyals O, Fortunato M, Jaitly N. Pointer networks. In: *Advances in neural information processing systems*. Vol. 28, Curran Associates, Inc.; 2015.
- [14] Bello I, Pham H, Le QV, Norouzi M, Bengio S. Neural combinatorial optimization with reinforcement learning. In: *International conference on learning representations*. 2017, URL <https://openreview.net/forum?id=Bk9mxLSFx>.
- [15] Bengio Y, Lodi A, Prouvost A. Machine learning for combinatorial optimization: A methodological tour d’horizon. *European J Oper Res* 2021;290(2):405–21. <http://dx.doi.org/10.1016/j.ejor.2020.07.063>.
- [16] Bogrybayeva A, Meraliyev M, Mustakhov T, Daultbayev B. Machine learning to solve vehicle routing problems: A survey. *IEEE Trans Intell Transp Syst* 2024;25(6):4754–72. <http://dx.doi.org/10.1109/TITS.2023.3334976>.

- [17] Cappart Q, Chételat D, Khalil EB, Lodi A, Morris C, Velickovic P. Combinatorial optimization and reasoning with graph neural networks. *J Mach Learn Res* 2023;24(130):1–61, URL <http://jmlr.org/papers/v24/21-0449.html>.
- [18] Xin L, Song W, Cao Z, Zhang J. Neurolkh: Combining deep learning and local search for the traveling salesman problem. In: *Advances in neural information processing systems*. 2021, URL <https://proceedings.neurips.cc/paper/2021/file/3d2d8cc03e05a306443c5ee918bb129c-Paper.pdf>.
- [19] Applegate D, Bixby R, Chvatal V, Cook W. On the solution of traveling salesman problems. *Doc Math* 1998;645–56, URL <http://eudml.org/doc/233207>.
- [20] Achterberg T, et al. SCIP optimization suite 7.0. ZIB Technical Report, 2020, URL <https://optimization-online.org/2020/07/8272/>.
- [21] Christofides N. Worst-case analysis of a new heuristic for the travelling salesman problem. Report 388, Graduate School of Industrial Administration, CMU; 1976.
- [22] Vazirani VV. Approximation algorithms. Springer; 2001, <http://dx.doi.org/10.1007/978-3-662-04565-7>.
- [23] Babin G, Deneault S, Laporte G. Improvements to the or-opt heuristic for the symmetric traveling salesman problem. Technical Report G-2005-02, GERAD; 2005.
- [24] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I. Attention is all you need. *Adv Neural Inf Process Syst* 2017, URL <https://arxiv.org/abs/1706.03762>.
- [25] Ma Y, Li J, Cao Z, Song W, Zhang L, Chen Z, Tang J. Learning to iteratively solve routing problems with dual-aspect collaborative transformer. In: *Advances in neural information processing systems*. Vol. 34, 2021, p. 11096–107.
- [26] Zhou J, Wu Y, Song W, Cao Z, Zhang J. Towards omni-generalizable neural methods for vehicle routing problems. In: *International conference on machine learning*. 2023, p. 42769–89.
- [27] Joshi C, Laurent T, Bresson X. Learning TSP requires rethinking graph neural networks. In: *International conference on learning representations*. 2019, URL <https://arxiv.org/abs/1803.03084>.
- [28] Dai H, Khalil EB, Zhang Y, Dilkina B, Song L. Learning combinatorial optimization algorithms over graphs. In: *Proceedings of the 31st international conference on neural information processing systems*. Red Hook, NY, USA: Curran Associates Inc.; 2017, p. 6351–61, URL <https://arxiv.org/abs/1704.01665>.
- [29] Williams RJ. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach Learn* 1992;8:229–56. <http://dx.doi.org/10.1007/BF00992696>.
- [30] Son J, Lee S-H, Choo J, Park M, Kim H, Park SJ. POP: Planner-optimized policy for procedural task learning. In: *Advances in neural information processing systems*. 2023.
- [31] Finn C, Abbeel P, Levine S. Model-agnostic meta-learning for fast adaptation of deep networks. In: *International conference on machine learning*. 2017, p. 1126–35. <http://dx.doi.org/10.48550/arXiv.1703.03400>.
- [32] Yu JJQ, Yu W, Gu J. Online vehicle routing with neural combinatorial optimization and deep reinforcement learning. *IEEE Trans Intell Transp Syst* 2019;20(10):3806–17. <http://dx.doi.org/10.1109/TITS.2019.2909109>.
- [33] Zhang Z, Liu H, Zhou M, Wang J. Solving dynamic traveling salesman problems with deep reinforcement learning. *IEEE Trans Neural Netw Learn Syst* 2023;34(4):2119–32. <http://dx.doi.org/10.1109/TNNLS.2021.3105905>.
- [34] Wang R, Zhou Z, Li K, Zhang T, Wang L, Xu X, Liao X. Learning to branch in combinatorial optimization with graph pointer networks. *IEEE/CAA J Autom Sin* 2024;11(1):157–69. <http://dx.doi.org/10.1109/JAS.2023.124113>.
- [35] Hu Y, Yao Y, Lee WS. A reinforcement learning approach for optimizing multiple traveling salesman problems over graphs. *Knowl-Based Syst* 2020;204:106244. <http://dx.doi.org/10.1016/j.knsys.2020.106244>.
- [36] Ye H, Wang J, Liang H, Cao Z, Li Y, Li F. GLOP: Learning global partition and local construction for solving large-scale routing problems in real-time. In: *AAAI conference on artificial intelligence*. 2024.
- [37] Xiao Y, Wang D, Li B, Wang M, Wu X, Zhou C, Zhou Y. Distilling autoregressive models to obtain high-performance non-autoregressive solvers for vehicle routing problems with faster inference speed. In: *AAAI conference on artificial intelligence*. 2024.
- [38] Drakulic D, Michel S, Mai F, Sors A, Andreoli J-M. BQ-NCO: Bisimulation quotienting for efficient neural combinatorial optimization. In: *Advances in neural information processing systems*. 2024.
- [39] Lin X, Yang Z, Zhang Q. Pareto set learning for neural multi-objective combinatorial optimization. In: *International conference on learning representations*. ICLR, 2022, URL <https://openreview.net/forum?id=QuObT9BTWo>.
- [40] Kim M, Park J, Park J. Learning collaborative policies to solve NP-hard routing problems. In: *Advances in neural information processing systems*. 2022.
- [41] Xiao Y, Wang D, Wu X, Wu Y, Li B, Du W, Wang L, Zhou Y. Improving generalization of neural vehicle routing problem solvers through the lens of model architecture. *Neural Netw* 2025;187:107380. <http://dx.doi.org/10.1016/j.neunet.2025.107380>.
- [42] Karimi-Mamaghan M, Mohammadi M, Meyer P, Karimi-Mamaghan AM, Talbi E-G. Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. *European J Oper Res* 2022;296(2):393–422. <http://dx.doi.org/10.1016/j.ejor.2021.04.032>.
- [43] de O. da Costa PR, Rhuggenaath J, Zhang Y, Akcay A. Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning. In: *Proceedings of the 12th Asian conference on machine learning*. ACML, Vol. 129, PMLR; 2020, p. 465–80, URL <http://proceedings.mlr.press/v129/costa20a.html>.
- [44] Hottung A, Kwon Y-D, Tierney K. Efficient active search for combinatorial optimization problems. In: *International conference on learning representations*. ICLR, 2022, URL <https://openreview.net/forum?id=nO5caZwFwYu>.
- [45] Trujillo E, Becerra I, Portillo-Velez Rd, Lupercio-Morales M, Macias JA, Rodriguez-Tello E. Enhancing trajectory tracking in humanoid robots using neural network-based dynamic gain control. *Emerg Sci J* 2025;9(2):539–56. <http://dx.doi.org/10.28991/ESJ-2025-09-02-02>.
- [46] Chu TS, Chua AY, Secco EL. A wearable MYO gesture armband controlling sphero BB-8 robot. *HighTech Innov J* 2020;1(4):179–86. <http://dx.doi.org/10.28991/HIJ-2020-01-04-05>.
- [47] Almomany A, Marouf Z, Jarrah A, Sutcu M. Real-time FPGA-based ADAS solution for driver drowsiness detection and autonomous stopping. *Emerg Sci J* 2025;9(5):2687–703. <http://dx.doi.org/10.28991/ESJ-2025-09-05-023>.
- [48] Sutton R, Barto A. Reinforcement learning, second edition: An introduction. In: *Adaptive computation and machine learning series*, MIT Press; 2018.
- [49] Buttazzo GC. Hard real-time computing systems: predictable scheduling algorithms and applications. 2nd ed. Springer; 2005, <http://dx.doi.org/10.1007/978-1-4614-0676-1>.
- [50] Kingma DP, Ba J. Adam: A method for stochastic optimization. In: *International conference on learning representations*. 2015, URL <https://arxiv.org/abs/1412.6980>.
- [51] Wang X, Zhao Q, Wang Y. A distributed optimization method for energy saving of parallel-connected pumps in HVAC systems. *Energies* 2020;13(15):3927. <http://dx.doi.org/10.3390/en13153927>.
- [52] Choromanski KM, Likhoshesterov V, Dohan D, Song X, Gane A, Sarlos T, Hawkins P, Davis JQ, Mohiuddin A, Kaiser L, Belanger DB, Colwell LJ, Weller A. Rethinking attention with performers. In: *International conference on learning representations*. 2021, URL <https://openreview.net/forum?id=Ua6zuk0WRH>.