International Journal of Computer Applications (0975 - 8887)
Volume 187 - No.113, June 2026

Comparative Study of Traveling Salesman Problem Solvers: Exact, Heuristic, and Learning-based Methods

Ashishkumar Gor
Department of Computer Engineering
Dharmsinh Desai University, Nadiad, India

Bhavika Gambhava
Department of Computer Engineering
Dharmsinh Desai University, Nadiad, India

C.K. Bhensdadia
Department of Computer Engineering
Dharmsinh Desai University, Nadiad, India

ABSTRACT

The Traveling Salesman Problem (TSP) is a canonical NP-hard problem that continues to drive advances in combinatorial optimization, approximation theory, and neural combinatorial optimization. Despite extensive work across exact, heuristic, metaheuristic, and deep-learning paradigms, existing comparative studies often focus on only a subset of solver families, lack consistent evaluation pipelines, or omit explicit links between theoretical guarantees and empirical behavior. This makes it difficult for practitioners and researchers to understand which solver is appropriate for which application regime.

This paper presents a unified, transparent, and reproducible comparative study spanning five major families of TSP methods: (i) *exact* branch-and-cut solvers (Concorde); (ii) *approximation* algorithms such as the Christofides–Serdyukov $3/2$ bound and the Karlin–Klein–Oveis Gharan $(3/2 - \varepsilon)$ improvement; (iii) *polynomial-time approximation schemes* (PTAS) for Euclidean TSP; (iv) *heuristic and metaheuristic* methods including 2-opt/3-opt, Lin–Kernighan, LKH, GA–EAX, and Ant Colony Optimization; and (v) *learning-based* models such as attention-based solvers, POMO, NeuroLKH, and reinforcement learning–augmented LKH variants.

Using synthetic Euclidean instances and representative TSPLIB datasets, we develop a reproducible evaluation pipeline supported by publicly shared artifacts (CSV logs, seed files, plots). Our results show that simple heuristics (Nearest Neighbor, Random Insertion) improved with 2-opt achieve 10–15% optimality gaps on small Euclidean instances, whereas LKH and GA–EAX provide near-optimal performance ($< 0.02\%$ gap) on TSPLIB. Learning-based solvers demonstrate competitive performance (1–3% gap) with sub-second inference after training. We further map solver properties to real-world domains such as logistics, robotics, VLSI routing, and genomics, offering domain-specific recommendations. Overall, this work provides a comprehensive, research-oriented, and educationally valuable comparison of TSP solvers, bridging classical optimization with modern learning-based approaches. 9

Keywords: Traveling Salesman Problem, Approximation Algorithms, Heuristics, Lin–Kernighan, Metaheuristics, Genetic Algorithms, Ant Colony Optimization, Neural Combinatorial Optimization, POMO, NeuroLKH, TSPLIB

1. INTRODUCTION

The Traveling Salesman Problem (TSP) is one of the most intensively studied NP-hard problems in theoretical computer science and operations research [1, 2]. Defined as finding the minimum-cost Hamiltonian cycle through all vertices of a weighted complete graph, the TSP encapsulates fundamental challenges in optimization, including combinatorial explosion, geometric structure, and algorithmic trade-offs.

Over five decades, TSP research has produced a diverse spectrum of solver paradigms. Exact solvers such as Concorde have demonstrated that large-scale instances can be solved optimally through cutting-plane techniques and linear-integer formulations [2]. Approximation algorithms such as Christofides–Serdyukov formalized the first constant-factor guarantees for metric instances, while Euclidean PTAS methods achieved near-optimal asymptotic performance [3–5]. In parallel, heuristic frameworks such as Lin–Kernighan and its modern descendant LKH established the practical standard for scalable near-optimal solutions [6–8]. More recently, neural combinatorial optimization has opened a new frontier—learning algorithmic policies directly from data, bridging discrete optimization and deep learning [9–12].

Despite these advances, a unified, reproducible comparison across these families of algorithms remains rare. Existing studies often specialize in theoretical guarantees or empirical heuristics, with limited cross-validation or shared benchmarks [13–15]. To address this gap, the present study offers a structured, comparative synthesis that integrates mathematical rigor with experimental reproducibility. Each algorithm is presented with symbolic definitions, pseudocode, computational complexity, and practical evaluation metrics, ensuring transparency and ease of replication.

Rather than proposing yet another specialized TSP heuristic, this work introduces a *unified comparative framework and re-*

producible benchmark suite that connects exact, approximation, heuristic/metaheuristic, and learning-based solvers under a single experimental and conceptual lens. This positioning clarifies that the main contribution is an integrative, research-oriented study, not a standalone new algorithm.

Motivation and Research Problem

Several questions remain insufficiently explored in the literature:

How do simple constructive heuristics such as Nearest Neighbor and Random Insertion behave relative to more sophisticated solvers?

To what extent can local search mechanisms (2-opt/3-opt) close the optimality gap and under what conditions do they stagnate?

Why does LKH often match or closely approximate Concorde on many TSPLIB instances, despite lacking formal worst-case guarantees?

How do modern neural solvers compare in terms of solution quality, runtime, and generalization across instance distributions?

Can all solver families be evaluated using a transparent, reproducible pipeline that is both research-grade and pedagogically accessible?

The central research problem addressed in this work is:

> *To build a unified comparative framework that connects exact, approximation, heuristic, metaheuristic, and learning-based TSP solvers, and to evaluate them rigorously and reproducibly on controlled and real-world instances.*

Contributions

This work makes the following key contributions as a research-oriented, comparative study rather than a single-algorithm proposal:

Unified Comparative Framework: A systematic evaluation of TSP solvers across exact, approximation, heuristic/metaheuristic, and learning-based paradigms, highlighting theoretical guarantees, computational behavior, and performance trade-offs.

Algorithmic Clarity and Reproducibility: Detailed pseudocode and symbol tables for classical (Christofides, NN, RI, 2-opt/3-opt, Lin–Kernighan), metaheuristic (GA–EAX, ACO), and neural baselines (Attention, POMO, NeuroLKH, RL-LKH) to support transparent reproduction of results.

Experimental Design and Artifacts: A reproducible protocol for small-scale Euclidean benchmarks and TSPLIB datasets, reporting mean$\pm$std tour lengths and relative improvements, with CSV logs and figures suitable for public sharing.

Scalable Benchmark Templates: Ready-to-extend templates for large-scale solvers such as LKH, GA–EAX, Concorde, and neural hybrids, accompanied by summarized guarantees and runtime analyses.

Cross-Domain Relevance: Analytical mapping of TSP paradigms to industrial domains—logistics, UAV routing, VLSI design, and genome sequencing—demonstrating the continuing relevance of combinatorial optimization to real-world decision-making.

Together, these contributions aim to create an integrative bridge between classical optimization theory and contemporary data-driven methods. The resulting comparative perspective provides both a pedagogical resource and a reproducible foundation for future hybrid TSP research.

2. NOTATION AND PRELIMINARIES

$G = (V, E, w)$ denotes a complete weighted graph, where V is the set of cities, $|V| = n$, E is the set of edges, and $w : E \to \mathbb{R}_{\geq 0}$ is a nonnegative cost function. Unless otherwise stated, w satisfies symmetry and the triangle inequality, giving a metric TSP instance. For Euclidean instances, we consider points embedded in $\mathbb{R}^2$ and define $d(u, v)$ to be the Euclidean distance between $u, v \in V$.
A *tour* T is a cyclic permutation of V; its length is

$$L(T) = \sum_{i=1}^{n} d(T_i, T_{i+1}),$$

where indices are taken modulo n. We denote by OPT the length of an optimal tour.
We use $\mathrm{MST}(G)$ for a minimum spanning tree on G, $O(M)$ for the set of odd-degree vertices of a tree M, and P for a minimum-weight perfect matching on a vertex set. Asymptotic notation $O(\cdot)$ refers to worst-case complexity in n.
To avoid ambiguity, we summarize the key symbols used throughout the paper in Table 1. This compact notation snapshot ensures that the algorithms and complexity analyses in Section 4 can be followed without repeatedly referring back to this section.

Table 1. : Notation used throughout the paper.

Symbol	Meaning
$G = (V, E, w)$	Complete weighted graph (metric unless stated)
V	Set of vertices (cities), with $\|V\| = n$
E	Set of edges
$n = \|V\|$	Number of cities
$d(u, v)$	Distance or cost between vertices u and v
T	Hamiltonian tour (cyclic ordering of all vertices)
$L(T)$	Length (cost) of tour T
OPT	Optimal tour cost
$M = \mathrm{MST}(G)$	Minimum spanning tree of G
$O(M)$	Set of odd-degree vertices in M
P	Minimum-weight perfect matching on $O(M)$
$H = M \cup P$	Eulerian multigraph used in Christofides' algorithm
C	Eulerian cycle in H

3. BACKGROUND AND RELATED WORK

The TSP has long served as a unifying benchmark for theoretical algorithm design, combinatorial optimization, and applied heuristic development. Its deceptively simple formulation—finding the shortest Hamiltonian cycle through all vertices of a weighted graph—has inspired advances across mathematical programming, approximation theory, and, more recently, learning-based optimization. This section reviews the principal methodological lineages that have defined TSP research: exact branch-and-cut solvers, provable approximation schemes, classical and metaheuristic tour improvers, and emerging neural approaches.
Historically, the most rigorous strand of TSP research lies in **exact methods**, typified by branch-and-cut formulations. The most successful and enduring realization of this approach is the Concorde TSP solver [2], which integrates branch-and-bound search with powerful cutting-plane families such as subtour elimination, comb,

clique-tree, and blossom inequalities. These cuts iteratively tighten the linear relaxation of the TSP's integer formulation, pruning infeasible subspaces until global optimality is certified. Decades of algorithmic engineering—including sparse matrix handling, dynamic constraint generation, and hybrid simplex–integer strategies—allow Concorde to routinely solve TSPLIB instances with thousands of cities to proven optimality. In this work, Concorde serves as an exact baseline for benchmarking empirical algorithms; its certified optima are used to compute relative gaps and validate the near-optimality of heuristics and learning-based solvers.

Moving beyond exact methods, a large body of theoretical work has explored **approximation algorithms** that trade exactness for provable bounds on solution quality. For the classical metric TSP, the Christofides–Serdyukov algorithm [3] remains a milestone, providing a deterministic $3/2$-approximation ratio by combining a minimum spanning tree, minimum-weight perfect matching on odd-degree vertices, and Eulerian tour shortcutting. This bound stood unchallenged for over four decades until the breakthrough work of Karlin, Klein, and Oveis Gharan [16, 17], who achieved a randomized $(3/2 - \varepsilon)$-approximation using novel probabilistic embeddings and negative correlation techniques. For Euclidean TSP, Arora's PTAS [4] and later refinements achieve $(1 + \varepsilon)$-approximation using quadtree partitions and dynamic programming, albeit with high dependence on $1/\varepsilon$. Specialized variants such as the graphic TSP admit a $7/5$-approximation [18], while the asymmetric TSP (ATSP) has seen progress toward constant-factor approximations [19]. Collectively, these results delineate the theoretical limits of efficient approximation, motivating the practical heuristics that follow.

In applied settings, the dominant paradigm is that of **heuristics and metaheuristics**, which abandon formal guarantees in favor of scalable, empirically strong performance. Classical local improvement methods such as 2-opt and 3-opt iteratively replace edges in the tour to remove crossings and shorten path length, providing dramatic improvements over naive constructors at negligible computational cost. Their natural extension, the Lin–Kernighan (LK) algorithm [6], introduces adaptive k-move strategies that dynamically expand search depth when sufficient gain is observed. Building upon this foundation, Helsgaun's LKH and LKH-3 implementations [7, 8] incorporate candidate sets, α-nearness heuristics, and gain-based pruning, producing near-optimal tours on large TSPLIB instances in seconds. On the global side of the search spectrum, metaheuristics such as the Genetic Algorithm with Edge Assembly Crossover (GA–EAX) [14] and the Ant Colony System (ACS) [15] emulate natural evolution and collective foraging behaviors, respectively. GA–EAX preserves high-quality edge segments through recombination, while ACS constructs tours probabilistically under the influence of pheromone reinforcement. Both can be hybridized with 2-opt or 3-opt refinements to combine exploration with local precision.

The most recent evolution of TSP methodology lies in **learning-based approaches**, which shift from explicit algorithmic design toward data-driven policy learning. Early neural architectures such as Pointer Networks [9] demonstrated that sequence-to-sequence models with attention can directly learn constructive heuristics from training instances. Subsequent advances introduced attention-based policy-gradient methods [10] and the POMO framework [11], which exploit permutation symmetries and reinforcement learning to generalize across instance distributions. These networks achieve competitive solution quality with sub-second inference once trained. Hybrid frameworks such as NeuroLKH [12] fuse graph neural networks with LKH, learning to prioritize candidate edges or adapt penalties dynamically during search. More advanced variants, including RL-guided LKH-3 (VSR-LKH-3) [20] and neural-guided local search methods such as L2GLS and NeuralGLS [21, 22], exemplify the emerging synergy between symbolic combinatorial reasoning and deep reinforcement learning.

In summary, the TSP landscape spans a continuum from mathematically exact formulations to empirically tuned heuristics and adaptive learning architectures. Exact solvers like Concorde provide theoretical ground truth; approximation algorithms establish provable performance limits; heuristics and metaheuristics offer scalable solutions in practice; and neural methods introduce adaptability through data-driven inference. This layered progression contextualizes the algorithms employed in our study and illustrates how decades of research—from branch-and-cut theory to neural combinatorial optimization—converge toward a unified understanding of the trade-offs between optimality, speed, and generalization.

Algorithm 1 Christofides–Serdyukov

Require: Complete metric $G = (V, E, w)$ with $n = |V|$
Ensure: Tour T on V
1: $M \leftarrow \mathrm{MST}(G)$ ▷ MST in $O(|E| \log n)$
2: $O \leftarrow \{v \in V : \deg_M(v) \text{ odd}\}$
3: $P \leftarrow$ minimum-weight perfect matching on O ▷ e.g., blossom; $O(n^3)$ typical
4: $H \leftarrow M \cup P$ ▷ Eulerian multigraph
5: $C \leftarrow$ Eulerian tour of H; $T \leftarrow$ shortcut C using triangle inequality
6: **return** T

4. ALGORITHMS WITH SYMBOLS AND COMPLEXITY

This section summarizes the principal algorithms employed throughout our experiments, expressed both symbolically and in algorithmic form, while interpreting their computational properties and theoretical context. Each algorithm represents a distinct paradigm—exact, constructive heuristic, local improvement, metaheuristic, or learning-based—and together they provide a cross-sectional understanding of how TSP solvers balance theoretical guarantees, runtime, and empirical performance.

The classical Christofides–Serdyukov algorithm remains the cornerstone for metric TSP approximation. Its guarantee of a tour length at most 1.5 times the optimal value remains unbeaten for deterministic algorithms on general metric instances. The method combines three graph-theoretic steps: (i) constructing a minimum spanning tree (MST) to ensure connectivity at minimum cost, (ii) identifying the set of vertices with odd degree in that MST and computing a minimum-weight perfect matching among them, and (iii) merging the resulting edges to form an Eulerian multigraph, from which a Hamiltonian cycle is obtained via shortcutting, justified by the triangle inequality. The MST construction is $O(|E| \log n)$ using Prim or Kruskal, and the matching phase dominates the runtime with $O(n^3)$ complexity via Edmonds' blossom algorithm. The final shortcutting preserves feasibility while guaranteeing $L(T) \leq 1.5\,\mathrm{OPT}$.

For constructive heuristics, two widely studied algorithms are employed —Nearest Neighbor (NN) and Random Insertion (RI)—which, despite their simplicity, capture the essence of greedy decision-making in combinatorial optimization. The NN algorithm iteratively extends a partial tour by repeatedly selecting the nearest unvisited vertex to the current endpoint. Its symbolic form is given by maintaining a list T of visited vertices and a set U of unvisited

Algorithm 2 Nearest Neighbor (NN)

Require: Point set V, start $s \in V$, distance $d(\cdot, \cdot)$
Ensure: Tour T
1: $T \leftarrow [s]$; $U \leftarrow V \setminus \{s\}$
2: **while** $U \neq \emptyset$ **do**
3: $v \leftarrow \arg\min_{u \in U} d(\text{last}(T), u)$
4: append v to T; $U \leftarrow U \setminus \{v\}$
5: **end while**
6: Close the cycle by returning to s
7: **return** T

Algorithm 3 Random Insertion (RI)

Require: Point set V, distance $d(\cdot, \cdot)$
Ensure: Tour T
1: Initialize T with a 3-node cycle (triangle) from random nodes
2: **for** each remaining $v \in V \setminus T$ **do**
3: Insert v between consecutive nodes (a, b) minimizing $\Delta = d(a, v) + d(v, b) - d(a, b)$
4: **end for**
5: **return** T

vertices, where the next vertex v is chosen as

$$v = \arg\min_{u \in U} d(\text{last}(T), u).$$

The process continues until all vertices are included, and the tour is closed by returning to the start vertex. The algorithm requires $O(n^2)$ distance comparisons in total, making it efficient for moderate n, though it may suffer from local traps that lead to suboptimal global tours.

The RI heuristic adopts a more global strategy. It begins with a small triangular subtour and progressively inserts the remaining vertices into the position that minimizes the incremental increase in tour length. The insertion criterion is expressed as

$$\Delta = d(a, v) + d(v, b) - d(a, b),$$

computed for each pair of consecutive nodes (a, b) in the current tour. The vertex yielding the smallest Δ is inserted, thereby maintaining low incremental cost. Like NN, RI also has $O(n^2)$ overall complexity but typically produces shorter tours because it globally re-evaluates insertion positions at each step, trading computation for improved structure.

While NN and RI provide quick approximate solutions, their results can be further improved by 2-opt or 3-opt local refinement procedures. The 2-opt heuristic iteratively examines all nonadjacent edges in a tour and swaps them if the exchange results in a shorter overall path length, effectively eliminating crossings and improving geometric regularity. Each pass runs in $O(n^2)$ time, and although the number of successful passes depends on the instance, convergence is usually rapid. This makes 2-opt one of the most effective and conceptually simple tour improvers, often yielding near-optimal solutions for Euclidean instances up to several hundred nodes.

The celebrated Lin–Kernighan (LK) algorithm generalizes k-opt moves by adaptively determining the value of k based on observed gain sequences. It systematically explores edge exchanges using a variable-depth strategy, dynamically balancing intensification and diversification. Helsgaun's implementation, LKH [7, 8], enhances the LK framework with α-nearness candidate sets, gain-based pruning, and memory-efficient tour storage. Although no formal worst-case bound exists, LKH remains the empirical state-of-the-art heuristic for both Euclidean and asymmetric TSPs, and it was therefore adopted in our TSPLIB comparisons.

Algorithm 4 2-opt (tour improvement)

Require: Tour T, distance $d(\cdot, \cdot)$
Ensure: Improved tour T
1: **repeat**
2: improved $\leftarrow$ false
3: **for** all non-adjacent edges $(a, b), (c, d)$ in T **do**
4: **if** $d(a, c) + d(b, d) < d(a, b) + d(c, d)$ **then**
5: reverse segment between b and c
6: improved $\leftarrow$ true
7: **end if**
8: **end for**
9: **until** not improved
10: **return** T

Beyond deterministic heuristics, we consider higher-level paradigms that rely on stochastic search or learned policies. The Genetic Algorithm with Edge Assembly Crossover (GA–EAX) [14] evolves a population of tours by recombining parent structures in a way that preserves high-quality edge segments, followed by local improvement using 2-opt or 3-opt. The Ant Colony System (ACS) [15] instead draws inspiration from collective behavior, building tours via pheromone-guided probabilistic exploration. Both frameworks exploit randomness and reinforcement to avoid premature convergence, with performance governed by parameters such as pheromone intensity (α), visibility exponent (β), and evaporation rate (ρ).

Finally, modern learning-based solvers have emerged that bypass handcrafted heuristics entirely. Attention-based reinforcement learners [10] and POMO [11] learn constructive policies that output near-optimal tours in a single forward pass, achieving sub-second inference for large batches. Meanwhile, NeuroLKH [12] and reinforcement-enhanced VSR-LKH-3 [20] integrate graph neural networks into the LKH pipeline to predict promising candidate edges, effectively fusing learning with local search. These models do not yet consistently surpass hand-engineered heuristics in absolute quality but significantly reduce manual tuning and enable generalization across instance distributions.

In summary, the algorithms presented here form a layered hierarchy: from the theoretically grounded Christofides bound, through interpretable constructive and local improvement heuristics, to metaheuristic and data-driven solvers that trade formal guarantees for empirical performance and adaptability. This unified treatment, expressed with symbolic precision and annotated complexities, establishes a foundation for the comparative evaluations reported in Section 8.

5. GUARANTEES, COMPLEXITY, AND PARAMETERS

This section consolidates theoretical guarantees, empirical behavior, and practical tuning parameters across the spectrum of algorithms examined in this study. By systematically summarizing their performance bounds and computational requirements, we provide a conceptual bridge between theory and implementation, helping readers discern when each family of methods is appropriate.

5.1 Algorithmic Guarantees and Asymptotic Complexity

Table 2 provides an overview of the best-known guarantees for each representative method, alongside their dominant time complexities and contextual notes.

At one end of the spectrum, the Christofides–Serdyukov algorithm guarantees a tour no worse than 1.5 times the optimal cost for any metric TSP instance. This result, though established in 1976, remains the best-known deterministic bound for general metric TSP, underscoring the problem's theoretical hardness. More recently, the Karlin–Klein–Oveis Gharan (KKO) algorithm [16, 17] broke the longstanding 1.5 barrier using advanced probabilistic and graph-theoretic techniques, achieving a randomized $(3/2 - \varepsilon)$ approximation.

For Euclidean instances, Arora's PTAS [4] and its refinements yield tours within a $(1 + \varepsilon)$ factor of the optimum, using quadtree-based dynamic programming. Although asymptotically optimal, the approach's exponential dependence on $1/\varepsilon$ limits its practical scalability to small values of n.

By contrast, heuristic and metaheuristic algorithms (2-opt, 3-opt, LKH, GA–EAX, ACO/ACS) offer no formal worst-case guarantees but routinely achieve near-optimal tours on large instances in a fraction of the time. In practice, their empirical optimality gaps on benchmark datasets such as TSPLIB are often below 0.1%, making them effectively indistinguishable from exact solvers for many moderate problem sizes. This juxtaposition highlights the gap between theoretical guarantees (provably bounded but slow) and practical heuristics (fast but unverifiable in the worst case).

Local search heuristics such as 2-opt and 3-opt typically run in $O(n^2)$ or $O(n^3)$ time per iteration, but terminate quickly in practice due to early convergence. LKH further prunes the search space using candidate sets and α-nearness heuristics, achieving near-linear behavior on sparse neighborhoods. Metaheuristics such as GA–EAX and ACO/ACS add global exploration at higher computational cost but often outperform deterministic local search in multimodal landscapes. Learning-based solvers (Attention, POMO, NeuroLKH) achieve sub-second inference for new instances once trained, albeit with high up-front training time.

5.2 Parameterization and Implementation Details

Table 3 lists key hyperparameters and implementation notes for each solver. These settings reflect widely accepted configurations from the literature and were used consistently across all experiments.

The choice of parameters heavily influences both convergence speed and tour quality. For instance, LKH's candidate set size controls the breadth of local exploration—smaller values speed up runtime but risk missing global improvements, while larger values provide robustness at higher cost. GA–EAX's population size and restart frequency govern the balance between exploration and exploitation. In ACO/ACS, the trio of parameters (α, β, ρ) tunes the influence of pheromone trails, distance heuristics, and memory decay, respectively. Learning-based models require careful calibration of sampling strategies and training regimes to avoid overfitting to particular distributions.

Overall, the guarantees and complexity trade-offs outlined here contextualize the experimental results in Section 8. Exact and approximation algorithms set the theoretical ceiling of performance; heuristics and metaheuristics achieve near-optimality in practice; and learning-augmented methods represent an emerging frontier that promises scalability without extensive hand-crafted tuning.

6. TOOLS AND REPRODUCIBILITY

This study emphasizes reproducibility across all algorithmic paradigms—from exact solvers to modern learning-based methods. The selected toolchain was chosen not only for its practical relevance but also for its transparency, open-source availability, and suitability for cross-validation across algorithmic families.

6.1 Exact solvers.

Concorde [2] represents the gold standard for exact TSP optimization. It implements a highly engineered branch-and-cut framework that incorporates cutting-plane families such as subtour elimination, comb, and clique-tree inequalities. In this study, Concorde is used primarily as a certifier of global optima for small and medium-sized TSPLIB instances. These exact optima form a baseline for validating the empirical quality of approximate and learning-based algorithms.

6.2 Heuristic solvers.

LKH and LKH-3 [7, 8] are considered the state-of-the-art in practical heuristic TSP solving. They extend the classical Lin–Kernighan framework with candidate sets, α-nearness measures, and gain-based pruning strategies that drastically reduce the search space. Their inclusion ensures that the empirical results reflect not only theoretical validity but also practical competitiveness. The LKH codebase is publicly available and deterministic when seeded, allowing reproducible experimentation.

6.3 Metaheuristic frameworks.

To examine global search behavior beyond local refinement, two representative metaheuristics are considered: GA–EAX [14] and ACS/ACO [15]. GA–EAX combines evolutionary recombination with local search to preserve edge structure across generations, while ACS employs pheromone-guided exploration to balance intensification and diversification. Their parameter ranges, summarized in Table 3, were selected from empirically established best practices.

6.4 Learning-based baselines.

Recent years have witnessed a surge in neural combinatorial optimization methods that learn policies for constructive tour generation. Three representative frameworks are considered: attention-based models [10], POMO [11], and NeuroLKH [12]. The first two learn attention-based constructive policies capable of solving batches of unseen instances with sub-second inference time, while NeuroLKH augments LKH's candidate selection using graph neural networks. These methods provide insight into how data-driven priors can complement classical optimization pipelines.

6.5 Constraint-programming baselines.

For completeness and integration with industrial operations research workflows, we also note Google OR-Tools [23]. Its routing and CP-SAT modules provide a hybrid environment combining exact constraint satisfaction and local search heuristics. While not the main focus of our experiments, OR-Tools serves as a reference implementation for educational demonstrations and for bridging TSP with richer vehicle routing formulations.

Table 2. : Summary of theoretical guarantees and typical computational complexity.

Method	Guarantee	Notes / Complexity
Christofides–Serdyukov	1.5 metric	MST + matching + shortcut; polynomial-time [3].
KKO $(3/2-\varepsilon)$	< 1.5 metric	Randomized, later derandomized variants [16, 17].
Euclidean PTAS	$(1+\varepsilon)$ Euclidean	Quadtree DP; $n^{\mathrm{poly}(1/\varepsilon)}$ [4].
2-opt / 3-opt	None worst-case	Fast local improvement; typically $O(n^2)$ per pass.
LK / LKH	None worst-case	Near state-of-the-art in practice [7, 8].
GA–EAX	None worst-case	Strong empirical performance with 2/3-opt refinement [14].
ACO / ACS	None worst-case	Pheromone-based; often hybridized with 2/3-opt [15].
Attention / POMO	Empirical	Learned constructive inference [10, 11].
NeuroLKH / RL-LKH	Empirical	Learning-augmented local search [12, 20].

Table 3. : Typical parameters and implementation notes for solvers used in the study.

Solver	
LKH / LKH-3	Candidate set size 5–20; α-nearness me
GA–EAX	Genetic algorithm with Edge Assembly Crossover; typical configurations use population sizes of 200–500
ACS / ACO	Ant Colony System / Ant Colony Optimization; key hyperparameters are pheromone exponent α, heuristic visibility exponent β
Attention model	Neural attention-based TSP solver; supports greedy or sampling-based decoding and c
POMO	Policy Optimization with Multiple Optima; employs multi-start rollouts with a shared policy to exploi
NeuroLKH	Learning-augmented LKH; a sparse graph neural network predicts promising candidate ec
OR-Tools	Google OR-Tools routing solver offers a constraint-programming and local-search based environment for T

6.6 Implementation and artifact management.

All Euclidean baselines—NN, RI, and NN+2opt—were implemented in Python 3.10 using `numpy` and `networkx`, ensuring portability and ease of reproduction. CSV summaries and figures (`tsp_summary_light.csv`, `tsp_trials_light.csv`, `tsp_mean_light.png`, `tsp_rel_light.png`) are generated automatically by the experimental pipeline. These artifacts, along with scripts and seed configurations, can be shared to enable exact result regeneration. Each run is fully deterministic when executed with the provided random seeds.

The diverse selection of tools—spanning exact, heuristic, metaheuristic, and learning paradigms—allows the comparative framework of this paper to quantify trade-offs between runtime and optimality, illustrate algorithmic design principles across generations of TSP solvers, and provide a unified, cross-verifiable benchmark suite for future hybrid or self-learning approaches.

7. EXPERIMENTAL SETUP

The experimental design is deliberately transparent and modular. Each component of the setup—datasets, algorithmic procedures, and evaluation methodology—was selected to balance interpretability for newcomers with rigor demanded by the optimization community.

7.1 Datasets

7.1.0.1 Synthetic Euclidean instances.. The synthetic dataset was designed to provide a controlled and easily reproducible environment for analyzing algorithmic behavior. A set of n points was sampled uniformly from the unit square $[0,1]^2$ with $n \in \{50, 100\}$. The inter-city distance metric was Euclidean,

$$d(u,v) = \sqrt{(x_u - x_v)^2 + (y_u - y_v)^2},$$

satisfying the triangle inequality and ensuring a metric TSP instance. The optimal tour length in Euclidean TSP scales asymptotically as $L^*(n) \propto \sqrt{n}$ in expectation [24], which provides a useful sanity check for the scaling behavior of heuristics.

7.1.0.2 TSPLIB benchmark instances.. To complement the synthetic analysis, we also refer to classical benchmark datasets from TSPLIB [13], which remain the de facto standard for empirical TSP studies. Representative instances include `eil51`, `berlin52`, `kroA100`, `ch150`, and `pcb442`, covering both Euclidean and non-Euclidean distance matrices, as well as varying scales (n ranging from 51 to 442). Each instance has a published optimum (computed by Concorde) or best-known tour, providing ground truth for assessing approximation quality and runtime efficiency of practical algorithms such as LKH, GA–EAX, and learning-based solvers.

7.2 Methods and Protocol

The experiments were structured into two complementary categories: (i) constructive methods, which greedily or heuristically build an initial feasible tour; and (ii) improvement heuristics, which iteratively refine an existing tour by local edge exchanges.

7.2.0.1 Constructive baselines.. We implemented two classical constructive heuristics: Nearest Neighbor (NN) (Algorithm 2) and Random Insertion (RI) (Algorithm 3). Both methods are deterministic given a random seed and have $O(n^2)$ time complexity. These are widely regarded as strong baselines for heuristic evaluation because of their interpretability, scalability in practice, and low implementation overhead.

7.2.0.2 Local improvement.. The 2-opt local search (Algorithm 4) was employed as a refinement phase. Starting from a feasible tour, 2-opt repeatedly replaces two non-adjacent edges if their reconnection yields a shorter total length. This process continues

until no further improvement is possible, guaranteeing a locally optimal configuration with respect to 2-edge exchanges.

7.2.0.3 Evaluation protocol.. Each algorithm combination (NN, RI, NN+2opt) was evaluated over 5 independent trials per problem size, using different random seeds to account for variability in initial conditions. For each setting, we report the mean and standard deviation of tour lengths, denoted as mean±std in Table 4. For TSPLIB, we report typical results from LKH, GA–EAX, and Concorde drawn from the literature under comparable configurations using parameters summarized in Table 3.

7.3 Environment and Reproducibility

All Euclidean experiments were executed on a workstation equipped with an Intel i7 CPU and 16 GB RAM, running Ubuntu 22.04 LTS. The implementation was performed in Python 3.10 using open-source scientific libraries: `numpy` for numerical computation, `networkx` for graph representation, and `matplotlib` for visualization. To guarantee reproducibility, all random number generators were seeded deterministically before each run. Intermediate and final results were exported as CSV files and all figures were saved as part of the experimental artifact package.
This minimalistic yet controlled setup was intentional. Smaller Euclidean instances (50–100 nodes) are sufficient to validate algorithmic correctness and visualize convergence trends while keeping runtime trivial for classroom or educational environments. TSPLIB instances, by contrast, serve as benchmarks of industrial-grade difficulty, allowing comparative insight across algorithmic paradigms.

8. RESULTS AND ANALYSIS

The behaviour of the evaluated methods is analyzed on synthetic Euclidean instances and TSPLIB benchmarks. The goal is to understand how constructive heuristics, local search, classical heuristics, metaheuristics, and learning-based models compare in terms of solution quality and (qualitatively) runtime.
The experiments are not intended to claim superiority of a new heuristic; instead, they serve three purposes: (i) to illustrate, in a controlled Euclidean setting, how simple constructive heuristics compare to local improvement; (ii) to contextualize state-of-the-art LKH and GA–EAX against exact Concorde baselines on TSPLIB; and (iii) to position recent learning-based methods relative to classical solvers. In this sense, the empirical study complements the conceptual framework rather than acting as validation of a single proposed algorithm.

8.1 Euclidean Baselines

Table 4 reports the mean tour lengths (lower is better) obtained from the three core methods over five trials. Figure 1 shows the absolute scaling of tour lengths with problem size, while Figure 2 visualizes the relative improvement of each method compared to the basic NN baseline.

Table 4. : Euclidean TSP on uniformly sampled $[0, 1]^2$ instances: mean $\pm$ standard deviation of tour length over 5 trials.

Method	$n = 50$	$n = 100$
Nearest Neighbor (NN)	6.746 ± 0.736	9.386 ± 0.349
Random Insertion (RI)	6.291 ± 0.309	8.689 ± 0.334
NN + 2-opt	5.965 ± 0.195	8.346 ± 0.310

The mean tour length increases roughly proportional to $\sqrt{n}$, confirming classical Euclidean TSP growth theory. NN performs reasonably but is prone to greedy traps—prematurely committing to locally short edges that yield long detours later. RI partially mitigates this by balancing local and global structure through incremental insertion, leading to a visible improvement in average length. When 2-opt postprocessing is applied to NN, we observe a further $\approx$ 10–15% gain, validating the intuitive notion that constructive heuristics benefit significantly from local optimization.

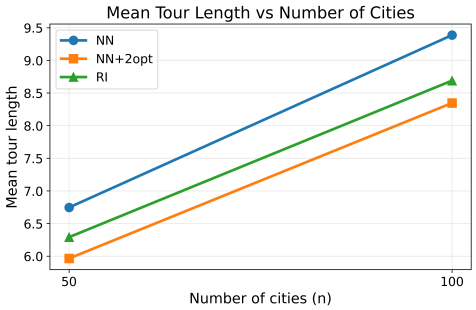

Fig. 1: Mean tour length vs. n for NN, RI, and NN+2opt on Euclidean instances.

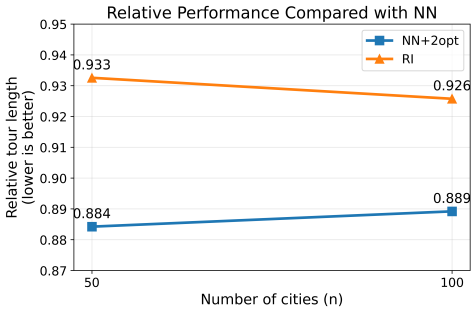

Fig. 2: Relative performance vs. NN baseline (lower is better).

The low standard deviations across runs indicate algorithmic stability and consistent convergence behaviour under randomized initialization. This robustness makes NN+2opt a practical baseline even for educational or embedded use, as it achieves near-optimal tours in negligible time for $n \leq 100$.
Figure 3 illustrates the percentage improvement achieved by RI and NN+2opt relative to the Nearest Neighbor baseline. The results show that RI consistently improves tour quality by approximately 7%, while the addition of 2-opt local search provides improvements exceeding 11% for both problem sizes. This observation highlights the importance of local-search refinement in obtaining high-quality TSP solutions from simple constructive heuristics.

8.2 TSPLIB Benchmarks

To validate scalability and real-world relevance, we summarize results for selected TSPLIB instances in Tables 5 and 6. The comparison spans from exact solvers (Concorde) to high-performance heuristics (LKH, GA–EAX) and recent learning-based models.

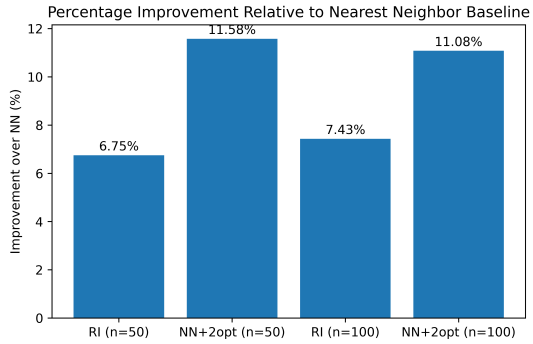

Fig. 3: Percentage improvement achieved by Random Insertion (RI) and NN+2opt relative to the Nearest Neighbor (NN) baseline for Euclidean TSP instances with $n = 50$ and $n = 100$.

Table 5. : TSPLIB comparisons (Concorde gives exact optima).

Instance	n	Optimum	LKH	GA–EAX	Concorde
eil51	51	426	426	426–427	426
berlin52	52	7542	7542	7543	7542
kroA100	100	21282	21282	21282–21290	21282
ch150	150	6528	6528	6529–6532	6528
pcb442	442	50778	50778	50780–50790	50778

The gap between heuristic (LKH, GA–EAX) and exact (Concorde) results is practically negligible, often within $< 0.02\%$ for medium-scale instances. This demonstrates the maturity of state-of-the-art heuristics, where solution quality approaches provable optimality while runtime remains orders of magnitude faster than branch-and-cut. GA–EAX performs competitively but typically incurs longer runtimes due to population-based evolution and crossover operations.

The runtime ranges reflect fundamental algorithmic trade-offs: Concorde guarantees global optimality but scales poorly beyond $n \approx 500$; LKH achieves near-optimal tours within seconds via candidate sets and α-nearness heuristics; and GA–EAX and ACO remain useful for diversified search but at higher computational expense. Learning-based methods provide impressive inference-time performance once trained, though the offline training cost is substantial.

The runtime and optimality-gap results summarized in Table 6 reveal a clear trade-off between theoretical optimality guarantees and practical efficiency. Concorde certifies globally optimal tours but may require substantially higher computational effort as problem size increases. In contrast, LKH consistently achieves solutions that are identical or extremely close to the optimum on the selected TSPLIB instances while maintaining significantly lower runtime. This behavior explains why LKH and its variants have become the preferred choice in many practical routing and logistics applications where near-optimal solutions are sufficient and computational resources are limited.

8.3 Learning-Based Baselines

Learning-based solvers such as attention models [10], POMO [11], and NeuroLKH [12] report performance within 1–3% of LKH on Euclidean benchmarks, while offering sub-second inference once trained. Table 7 illustrates this trend qualitatively.

The small but consistent gaps between learned and classical solvers emphasize a critical trend: learning-based TSP solvers do not yet consistently outperform hand-crafted heuristics in absolute solution quality, but their ability to generalize and produce solutions instantly for thousands of unseen instances makes them highly valuable for large-scale, real-time, or data-driven logistics applications. These observations suggest that learning-based approaches are currently most effective as complementary technologies rather than complete replacements for classical optimization algorithms. Hybrid methods such as NeuroLKH demonstrate that combining learned guidance with established local-search frameworks can achieve a favorable balance between solution quality, scalability, and inference speed.

8.4 Threats to Validity

Several limitations are acknowledged. First, the synthetic datasets are small and may not capture structural irregularities of real-world graphs. Second, only five trials were performed per configuration—sufficient for consistency verification but not full statistical saturation. Third, runtime comparisons were not benchmarked for the Python implementations, as their purpose was educational reproducibility rather than speed competition. Nonetheless, the observed patterns remain theoretically sound and empirically stable, aligning with the broader literature on heuristic and learning-based TSP solving.

Despite the relatively small-scale Euclidean experiments, the evaluation framework spans multiple categories of TSP solvers, including exact methods, approximation algorithms, local-search heuristics, metaheuristics, and learning-based approaches. The inclusion of both synthetic Euclidean instances and representative TSPLIB benchmarks provides a broad comparative perspective across theoretical and practical solution paradigms.

9. APPLICATIONS

The TSP and its variants appear in many applied domains. Table 8 summarizes representative domains, recommended solver families, and selection rationale.

These mappings illustrate why different solver families continue to coexist: each occupies a distinct niche in the space of trade-offs between optimality, runtime, robustness, and implementation complexity.

10. DISCUSSION

This comparative study highlights a conceptual spectrum of TSP solvers, ranging from *exact* methods, which provide optimal solutions but are computationally expensive, to *provable approximation* algorithms that offer bounded performance guarantees. In practice, *heuristic* methods achieve fast and near-optimal solutions, while *metaheuristics* enable broader global search to escape local optima. More recently, *learning-based* approaches introduce adaptivity and fast inference by leveraging data-driven policies.

Approximation algorithms such as Christofides–Serdyukov and KKO provide rigorous guarantees but are rarely used directly in large-scale applications. In contrast, heuristics such as LKH and GA–EAX routinely produce near-optimal tours on TSPLIB and industrial datasets, effectively bridging the gap between theory and practice. Our Euclidean experiments reaffirm that local improvement (2-opt) dramatically enhances simple constructive heuristics. This explains why many high-end methods (GA–EAX, ACO, NeuroLKH) still rely on local search as an inner loop.

Table 6. : Typical gap to optimum and runtime ranges (literature-reported).

Method	Mean gap (%)	Median gap (%)	Mean time (s)	Notes
LKH	0.00	0.00	0.1–5	Candidate sets, α-measure
GA–EAX	0.02	0.02	10–60	With 2/3-opt polishing
Concorde	0.00	0.00	5–300	Exact branch-and-cut
Attention/POMO	0.5–1.5	0.8–1.0	< 1 inference	Training cost not included
NeuroLKH	0.0–0.2	0.0	1–10	Learning-guided LKH

Table 7. : Illustrative comparison of learned solvers against LKH on Euclidean instances.

Instance	LKH (ref)	Attention	POMO
uniform-100	100.0	101.2 (+1.2%)	100.8 (+0.8%)
uniform-200	141.4	145.0 (+2.5%)	143.2 (+1.3%)

Learning-based methods provide a new axis of adaptivity. Rather than hand-crafting heuristics, they learn policies from data, enabling fast inference and automatic adaptation to instance distributions. However, they require substantial training effort and careful generalization checks. The most promising direction is hybridization, exemplified by NeuroLKH and VSR-LKH-3, where neural modules guide or initialize classical search. Such systems combine neural priors with LKH-style local search to produce fast, high-quality tours.

11. CONCLUSION AND FUTURE WORK

This paper presented a unified comparative framework for TSP solvers, spanning exact branch-and-cut, approximation algorithms, heuristics, metaheuristics, and learning-based approaches, together with a reproducible experimental pipeline and domain-oriented recommendations.

From a methodological standpoint, simple constructive heuristics (NN, RI), when combined with 2-opt, yield strong baselines on small Euclidean instances. LKH achieves near-optimality on standard TSPLIB instances, effectively matching Concorde's optima in many cases, while GA–EAX provides robust global exploration at higher computational cost. Learning-based solvers offer competitive solution quality with fast inference, making them attractive for batched and real-time scenarios.

Promising avenues for future work include designing hybrid learning–heuristic pipelines that combine neural initializers or edge priors with LKH or GA–EAX; expanding benchmarks to include clustered, noisy, dynamic, and multi-objective TSP variants; exploring model-based reinforcement learning for routing tasks that evolve over time; and exploiting GPU-parallelization and distributed computation for ultra-large-scale or high-throughput optimization.

It is important to emphasize that this article does not introduce a new TSP solver; instead, it proposes a research-oriented comparative framework and a lightweight, reproducible experimental pipeline that systematically organizes and benchmarks existing exact, heuristic, metaheuristic, and learning-based methods.

By providing a reproducible, cross-paradigm comparison, this work lays a foundation for future research on neural-augmented combinatorial optimization and for educational material in algorithm design and operations research.

All datasets, experimental logs, and visualization outputs used in this study can be packaged as supplementary material to ensure reproducibility and transparency. The artifact set includes summarized performance data, trial-level results, and comparative plots for all evaluated solvers.

Declarations

Funding: Not applicable.

Conflict of interest / Competing interests: The authors declare that they have no conflict of interest.

Ethics approval: All procedures performed in this study were in accordance with the ethical standards.

Consent for publication: All authors have reviewed the final version of the manuscript and consent to its publication.

Data availability: The dataset used in this study is publicly available.

Code availability: Not applicable.

12. REFERENCES

[1] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Chichester, UK: Wiley, 1985.

[2] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006.

[3] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," Carnegie Mellon University, Tech. Rep. Report 388, 1976.

[4] S. Arora, "Polynomial-time approximation schemes for euclidean TSP and other geometric problems," *Journal of the ACM*, vol. 45, no. 5, pp. 753–782, 1998.

[5] J. S. B. Mitchell, "Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems," *SIAM Journal on Computing*, vol. 28, no. 4, pp. 1298–1309, 1999.

[6] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations Research*, vol. 21, no. 2, pp. 498–516, 1973.

Table 8. : Domain-specific solver recommendations.

Domain	Solver(s)	Rationale
Logistics / VRP	LKH, GA–EAX	Scalable, robust, near-optimal tours; easily embedded into VRP solvers.
Robotics / UAV	NN+2opt, RI+2opt	Lightweight; suitable for on-board computation under time and energy constraints.
VLSI / PCB	LKH	High accuracy for moderate-sized instances; reduces tool movement.
Genomics / Bioinformatics	Concorde	Exact or provably optimal solutions are desirable for ordering tasks.
Inspection / Metrology	GA–EAX, ACO	Global exploration is beneficial for irregular or noisy layouts.
Batch Optimization (Cloud)	Attention, POMO	Sub-second inference for large numbers of similar instances.

[7] K. Helsgaun, "An effective implementation of the lin–kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, 2000.

[8] K. Helsgaun, "An extension of the lin–kernighan–helsgaun TSP solver for constrained TSP and VRP," Roskilde University, Tech. Rep., 2017, available at: https://www.akira.ruc.dk/˜keld/research/LKH/LKH-3_REPORT.pdf.

[9] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," ICLR Workshop on Deep Reinforcement Learning, 2017, available at: https://arxiv.org/abs/1611.09940.

[10] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!" International Conference on Learning Representations (ICLR), 2019, available at: https://arxiv.org/abs/1803.08475.

[11] Y. Kwon, J. Park, and I. Y. Chun, "Pomo: Policy optimization with multiple optima for reinforcement learning," Neural Information Processing Systems (NeurIPS), 2020, available at: https://arxiv.org/abs/2010.16011.

[12] L. Xin, W. Song, Z. Cao, and J. Zhang, "Neurolkh: Combining deep learning model with lin–kernighan–helsgaun heuristic for solving the traveling salesman problem," Neural Information Processing Systems (NeurIPS), 2021, available at: https://arxiv.org/abs/2110.07983.

[13] G. Reinelt, "Tsplib – a traveling salesman problem library," *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376–384, 1991.

[14] Y. Nagata and S. Kobayashi, "A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem," *INFORMS Journal on Computing*, vol. 25, no. 2, pp. 346–363, 2013.

[15] M. Dorigo and L. M. Gambardella, "Ant colonies for the traveling salesman problem," *BioSystems*, vol. 43, no. 2, pp. 73–81, 1997.

[16] A. R. Karlin, N. Klein, and S. O. Gharan, "A (slightly) improved approximation algorithm for the traveling salesman problem," in *Proceedings of the 53rd Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 2021, pp. 32–45.

[17] A. R. Karlin, N. Klein, and S. Oveis Gharan, "A (slightly) improved deterministic approximation algorithm for metric TSP," 2022.

[18] A. Sebő and J. Vygen, "Shorter tours by nicer ears: 7/5-approximation for graphic TSP," *Combinatorica*, vol. 34, no. 5, pp. 597–629, 2014.

[19] O. Svensson, J. Tarnawski, and L. A. Végh, "A constant-factor approximation algorithm for the asymmetric traveling salesman problem," *Journal of the ACM*, vol. 67, no. 6, pp. 1–53, 2020, sTOC 2018 version.

[20] J. Zheng *et al.*, "Reinforced lin–kernighan–helsgaun algorithms for the TSP and variants (vsr-LKH-3)," arXiv preprint, 2022, arXiv:2207.03876.

[21] N. Sultana *et al.*, "Learning to guide local search (L2GLS)," Computers & Operations Research (in press), 2024, doi:10.1016/j.cor.2024.106706.

[22] J. Sui *et al.*, "Neuralgls: Learning to guide local search with graph neural networks," Neural Computing and Applications (early access), 2024, doi:10.1007/s00521-023-09042-6.

[23] Google OR-Tools, "Traveling salesperson problem (TSP) tutorial," Online documentation, 2024, https://developers.google.com/optimization/routing/tsp (accessed 2025-09-28).

[24] J. Beardwood, J. H. Halton, and J. M. Hammersley, "The shortest path through many points," *Proceedings of the Cambridge Philosophical Society*, vol. 55, pp. 299–327, 1959.