State-of-the-Art Machine Learning Algorithms for Combinatorial Optimization: A Comprehensive Survey (2024–2025)

1. Introduction: The Renaissance of Heuristics via Machine Learning

The domain of Combinatorial Optimization (CO) is currently undergoing a structural transformation of a magnitude not seen since the development of metaheuristics in the late 20th century. For decades, the resolution of NP-hard problems such as the Traveling Salesman Problem (TSP) and the Vehicle Routing Problem (VRP) was the exclusive province of Operations Research (OR). Exact methods, exemplified by Branch-and-Bound and Branch-and-Cut algorithms, provided optimality guarantees but suffered from exponential time complexity, rendering them intractable for large-scale industrial applications. Conversely, heuristics and metaheuristics—ranging from simple greedy construction to sophisticated Genetic Algorithms (GA), Simulated Annealing (SA), and Tabu Search—offered computationally feasible solutions but relied heavily on "handcrafted" rules. These rules, distilled from human intuition and domain expertise, often proved brittle; a heuristic tuned for clustered urban deliveries might fail catastrophically when applied to long-haul rural logistics.[1]

Since approximately 2016, and accelerating dramatically through 2024 and 2025, a third paradigm has matured: **Neural Combinatorial Optimization (NCO)**. This data-driven approach posits that the optimal decision rules for combinatorial problems can be learned directly from problem instances using Deep Neural Networks (DNNs). By training on millions of generated or real-world examples, NCO models develop an intrinsic representation of problem topology, learning to map inputs (such as customer coordinates and demand profiles) to high-quality solutions with minimal inference latency.

The landscape of 2025 is defined by a shift from proof-of-concept architectures to robust, industrial-grade solvers. Early NCO research focused on simple, static Euclidean TSPs with 20 to 50 nodes. Today, the frontier has moved to **temporal, dynamic, and periodic** problems with thousands of nodes, reflecting the true complexity of global supply chains. This report provides an exhaustive analysis of these advancements, dissecting the architectures that have succeeded the seminal Attention Model (AM) and POMO, exploring the integration of Generative AI and Large Language Models (LLMs), and detailing the specialized frameworks designed for the Periodic Capacitated Vehicle Routing Problem (PVRP) and its dynamic

variants.

1.1 The Classification of Neural Solvers

To navigate the complex ecosystem of modern NCO, it is essential to categorize the methodologies based on their operational mechanics. The literature currently divides neural solvers into two primary families, each with distinct architectural requirements and use cases:

1. **Constructive Neural Heuristics:** These models generate a complete solution from scratch, typically in a single forward pass or a sequential autoregressive process. They begin with an empty set and iteratively select the next decision variable (e.g., the next city to visit) until a valid solution is constructed.
    - *Operational Analogy:* A delivery driver deciding which turn to take at each intersection, committing to a path step-by-step.
    - *Dominant Architectures:* Encoder-Decoder Transformers, Pointer Networks, Graph Neural Networks (GNNs).
    - *Key Advantages:* Extremely fast inference times (often milliseconds), making them ideal for real-time decision-making; ease of enforcing hard constraints during the construction process.
    - *Key Challenges:* The "greedy" nature of autoregression often leads to myopia, where suboptimal early decisions propagate errors that cannot be corrected later.
2. **Improvement Neural Heuristics:** These models begin with a complete, often suboptimal, solution and iteratively refine it to improve the objective function (e.g., reducing total tour length). They effectively learn to perform local search operations, replacing hand-designed operators like 2-opt or node swaps with learned policies.
    - *Operational Analogy:* A sculptor chipping away at a rough block to reveal the refined form within, or a logistics planner tweaking a schedule by swapping shifts.
    - *Dominant Architectures:* Neural Large Neighborhood Search (NLNS), Learning to Optimize (L2O) frameworks.
    - *Key Advantages:* An "anytime" property, allowing solution quality to improve given more computational budget; excellent scalability to large instances through decomposition.
    - *Key Challenges:* Slower than constructive methods due to the iterative nature; dependency on the quality of the initial solution.

The most recent and disruptive addition to this taxonomy is **Generative Neural Optimization**, utilizing Diffusion Models and Large Language Models. These approaches treat the solution not as a sequence of decisions, but as a holistic structure—a probability heatmap or a textual description—that is generated or "denoised" to reveal the optimal configuration.

1.2 The Imperative of Temporal and Dynamic Modeling

While static routing problems serve as the "fruit fly" of academic research—useful for benchmarking but limited in scope—real-world optimization is inherently temporal. The value of a solution in logistics is dictated not just by distance, but by time: the synchronization of

weekly schedules (Periodic VRP), the responsiveness to new orders arriving in real-time (Dynamic VRP), and the adaptation to fluctuating traffic conditions (Time-Dependent VRP). This report places a specific emphasis on how NCO architectures have evolved to encode time as a fundamental feature, moving beyond spatial embeddings to spatio-temporal representations capable of solving the PVRP and its variants.[3]

---

2. The Constructive Paradigm: Successors to POMO and AM

The foundation of modern constructive NCO lies in the Attention Model (AM) and its refinement, Policy Optimization with Multiple Optima (POMO). Understanding their mechanics is a prerequisite for appreciating the innovations of 2024–2025, which address their limitations regarding symmetry, asymmetry, and myopia.

2.1 The Ancestral Baseline: Attention Model (AM)

Introduced by Kool et al. (2019), the Attention Model represented the first successful application of the Transformer architecture to routing problems, replacing the Recurrent Neural Networks (RNNs) used in earlier Pointer Networks.

Mechanism:
The AM treats the VRP as a sequence-to-sequence translation task. The Encoder processes the set of input nodes (customers and depot) using Multi-Head Attention (MHA) layers to produce a high-dimensional embedding for each node. These embeddings capture the global context of the graph—a node's representation is informed by its spatial relationship to all other nodes. The Decoder then generates the route autoregressively. At each step t, the decoder utilizes the embedding of the previously visited node (the "query") to compute attention weights over the embeddings of all unvisited nodes (the "keys"). These weights are interpreted as a probability distribution, from which the next node is sampled.
Limitations:
While AM outperformed classical heuristics like nearest-neighbor and insertion algorithms, it struggled to compete with specialized metaheuristics like LKH-3. A primary inefficiency was its handling of rotational variance. A VRP solution is valid regardless of the starting node or the orientation of the map, yet AM treated different permutations or rotations as distinct problems, requiring extensive data augmentation or beam search to achieve stability.6

2.2 The Pivot Point: POMO (Policy Optimization with Multiple Optima)

Kwon et al. (2020) revolutionized NCO training with **POMO**, which explicitly leverages the symmetries inherent in routing problems to stabilize Reinforcement Learning (RL).

Symmetry Exploitation:
In a TSP with N nodes, there are N distinct sequences that represent the exact same

cycle (e.g., $1 \to 2 \to 3 \to 1$ is identical to $2 \to 3 \to 1 \to 2$). AM ignored this, often learning different policies for different start nodes. POMO enforces consistency by generating trajectories starting from every possible node $i \in \{1, \dots, N\}$ during training.
The Self-Baseline Mechanism:
Standard REINFORCE algorithms require a baseline to estimate the advantage of an action (i.e., "is this action better than expected?"). Traditional baselines were often separate neural networks (Critics) or greedy rollouts, which introduced high variance or computational overhead. POMO utilizes the batch average of the N trajectories as the baseline for each trajectory.

- If the tour starting at Node 5 is shorter than the average tour length of all start nodes, it receives a positive reward.
- This "self-baseline" drastically reduces gradient variance, allowing for faster convergence and deeper training, establishing POMO as the robust backbone for most subsequent research.[6]

2.3 Sym-NCO: Mastering Geometric Invariance

While POMO addressed *solution* symmetry (invariance to the starting node), it did not address *problem* symmetry (invariance to geometric transformations). A square rotated by 45 degrees is still a square, and the optimal path around its vertices remains unchanged relative to the nodes. However, a standard neural network sees new coordinates and treats it as a novel problem.

Sym-NCO (Symmetric Neural Combinatorial Optimization) [8] extends the POMO philosophy to the geometric domain.

Architectural Innovations:

1. **Multi-View Training:** For every problem instance in a training batch, Sym-NCO generates multiple geometric views via rotation (e.g., $0^\circ, 90^\circ, 180^\circ, 270^\circ$) and reflection.
2. **Invariance Regularization Loss:** The model includes a specialized loss term that penalizes discrepancies in the hidden representations of these views. If the input graph is rotated, the internal embeddings should ideally rotate primarily in the feature space or remain invariant in their relational distances.
3. **Resulting Generalization:** By forcing the network to learn features that are invariant to the coordinate frame, Sym-NCO achieves superior generalization. It performs remarkably well on cross-distribution tasks—for instance, training on uniform random data and testing on clustered "real-world" data—where standard POMO often fails due to overfitting to the coordinate distribution of the training set.

2.4 MatNet: Conquering Asymmetry and Matrix Inputs

A critical limitation of AM, POMO, and Sym-NCO is their reliance on node coordinates (x, y). Many industrial optimization problems are defined not by geometry, but by arbitrary cost

matrices. In the **Asymmetric TSP (ATSP)**, the cost to travel from $A \to B$ may differ from $B \to A$ (e.g., due to one-way streets, wind, or elevation changes). In the **Flexible Flow Shop Problem (FFSP)**, the setup times between jobs depend on the specific sequence, defined by a tabular matrix.

MatNet (Matrix Encoding Network) [11] was introduced to bridge this gap, marking a shift from coordinate-centric to relation-centric NCO.

Matrix-Centric Architecture:
MatNet abandons the Euclidean inductive bias. Instead, it formulates the optimization problem as a complete bipartite graph.

- **Dual Embeddings:** The model maintains two sets of embeddings: row embeddings (representing source nodes) and column embeddings (representing destination nodes).
- **Matrix-Augmented Attention:** In a standard Transformer, attention scores A_{ij} are derived solely from the dot product of node embeddings ($Q_i \cdot K_j$). In MatNet, the attention score explicitly incorporates the cost matrix D_{ij}. The attention mechanism learns to weigh the "cost" of an edge alongside the semantic compatibility of the nodes.
- **Performance:** MatNet has achieved State-of-the-Art (SOTA) results on ATSP and FFSP benchmarks, domains where coordinate-based models are inapplicable. It demonstrates that NCO can effectively learn logic on tabular data structures, opening the door for broader applications in scheduling and roster management.[11]

2.5 LCH-Regret: Learning to Correct Mistakes

Autoregressive constructive solvers are inherently myopic. They greedily select the most promising node at step t based on the current context. However, a locally optimal choice at step t might force the solver into a disastrously expensive trajectory at step $t+10$. Once a node is added, it cannot be removed.

LCH-Regret (Learning Constructive Heuristics with Regret) [7] introduces a mechanism to mitigate this "greedy" behavior without sacrificing the speed of constructive methods.

The Regret Mechanism:
LCH-Regret augments the standard decoder with a "Regret Module" and a "Backtracking Policy."

1. **Look-Ahead Evaluation:** During the construction process, the model estimates the uncertainty or entropy of the future trajectory.
2. **Rollback Capability:** If the model detects that the current partial solution has led to a state with low value (high predicted future cost), it triggers a "regret" signal. The solver reverts to a previous state $t-k$ and masks the previously selected node, forcing the exploration of an alternative branch.
3. **Differentiable Regret:** Crucially, this regret logic is differentiable and learned end-to-end. The model learns *when* to regret—identifying the specific patterns (like entering a cul-de-sac of nodes) that typically precede poor solutions.

4. **Integration:** LCH-Regret is designed as a plug-and-play module. It has been successfully integrated into AM, POMO, and MatNet backbones, consistently reducing the optimality gap by allowing the solver to self-correct during inference.[15]

2.6 Table: Comparative Analysis of Constructive Solvers

Model	Input Modality	Core Innovation	Symmetry Handling	Best Application Case
AM (2019)	Coordinates	Transformer for Routing	None	Baseline Euclidean VRP/TSP
POMO (2020)	Coordinates	Multi-start Training, Self-Baseline	Solution (Start Node)	High-speed, robust Euclidean VRP
Sym-NCO (2022)	Coordinates	Invariance Regularization	Problem (Geometric)	Cross-distribution generalization
MatNet (2021)	Cost Matrix	Matrix-Augmented Attention	None (Structure-based)	Asymmetric TSP, Flow Shop, Scheduling
LCH-Regret (2024)	Any	Backtracking/Rollback Module	Temporal (Decision)	Improving convergence of any LCH
GNARKD (2024)	Coordinates	Knowledge Distillation	None	Edge deployment, low-latency inference

---

3. Generative AI in Optimization: Diffusion and Heatmaps

Parallel to the refinement of constructive solvers, a radically different approach has emerged from the field of Generative AI. Inspired by the capability of diffusion models to generate coherent images from noise, researchers have adapted these probabilistic models to generate **solution heatmaps** for combinatorial problems. This paradigm shifts the question from "What is the next node?" to "What does the global structure of the optimal solution look like?"

3.1 DIFUSCO: Graph-Based Diffusion Solvers

DIFUSCO (Diffusion Solvers for Combinatorial Optimization) [16] represents the current state-of-the-art in non-autoregressive NCO. It frames the TSP/VRP not as a sequence generation task, but as a **discrete image generation** task on a graph.

The Formulation:
The solution to a TSP on a graph with N nodes is represented as a binary adjacency matrix $A \in \{0, 1\}^{N \times N}$, where $A_{ij} = 1$ if the edge (i, j) is part of the tour, and 0 otherwise.

- **Forward Diffusion (Corruption):** The training process takes a ground-truth optimal matrix A_{opt} and progressively corrupts it by flipping bits (in the discrete case) or adding Gaussian noise (in the continuous case) over T steps, until the matrix resembles random noise.
- **Reverse Diffusion (Denoising):** A neural network is trained to reverse this process. Given a noisy matrix at step t and the graph features (node coordinates), the network predicts the "cleaner" matrix at step $t-1$. By iteratively applying this denoising from pure noise, the model generates a high-probability adjacency matrix.[20]

Anisotropic Graph Neural Networks:
Unlike image diffusion models that utilize Convolutional Neural Networks (CNNs) over a fixed grid, DIFUSCO must respect the permutation invariance of graphs. It employs an Anisotropic Graph Neural Network (GNN) as its backbone. In this architecture, edge features are updated based on the embeddings of their connected nodes, but the message passing is modulated by the current "pixel value" (noise level) of the edge. This allows the diffusion process to respect the topological structure of the problem, enhancing flow along edges that are structurally promising.21

3.2 Discrete vs. Continuous Diffusion Modalities

DIFUSCO investigates two distinct noise modalities, with significant implications for performance:

1. **Gaussian (Continuous) Diffusion:** This approach relaxes the binary constraints, treating the adjacency matrix as a continuous field. Standard Gaussian noise is added. While this allows for smooth gradients and leverages established diffusion schedulers, the intermediate states are non-binary and difficult to interpret as graphs. It requires a final rounding or thresholding step.

2. **Bernoulli (Discrete) Diffusion:** This approach, often utilizing the D3PM (Discrete Denoising Diffusion Probabilistic Models) framework, models the edge existence explicitly as a Bernoulli variable. The noise process involves probabilistic bit-flipping. Research indicates that **Discrete Diffusion** yields superior results for combinatorial problems because it strictly adheres to the discrete nature of the optimization variables, producing sharper and more structurally valid heatmaps.[19]

3.3 Decoding the Heatmap

The output of the diffusion model is a probabilistic **heatmap**—a matrix P where P_{ij} represents the probability that edge (i, j) is in the optimal tour. This heatmap must be converted into a valid, connected tour. DIFUSCO employs two primary strategies:

- **Greedy Decoding:** A fast heuristic that greedily selects the highest-probability edges that do not violate tour constraints (e.g., degree constraints).
- **Monte Carlo Tree Search (MCTS):** A more computationally intensive but accurate method. The heatmap probabilities are used to bias the node selection policy within an MCTS. The diffusion model acts as a strong "prior," focusing the search on the most promising structural backbones while the MCTS ensures feasibility and closes the tour.[20]

3.4 Advancements: Blackout Diffusion and T2T

The diffusion paradigm is rapidly evolving to address its primary bottleneck: **Inference Speed**. A standard diffusion process might require 50 to 100 sequential denoising steps, making it significantly slower than constructive solvers like POMO.

Blackout Diffusion (2025):
Standard discrete diffusion uses fixed, discrete timesteps. Blackout Diffusion 22 introduces a continuous-time framework for discrete variables. It models the forward process as a "pure death" process where information (edges) is lost ("blacked out") continuously over time, and the reverse process as a "birth" process. This formulation allows for flexible inference schedules and has been shown to improve the structural integrity of generated graphs on complex benchmarks by learning the "persistence" of edges—essentially identifying the critical "backbone" edges that define the optimal topology.23
T2T (Text-to-Tour) / Consistency Models:
Recent work 19 focuses on Consistency Distillation, a technique to distill the multi-step diffusion process into a model that can generate the solution in a single step (or very few steps). These "Fast Sampling" methods aim to combine the global structural awareness of diffusion with the speed of constructive heuristics.

---

4. The New Frontier: Temporal, Periodic, and Dynamic Routing

The shift from academic benchmarks to real-world applicability necessitates a focus on **Time**. Industrial logistics is rarely a static snapshot; it is a continuous flow of schedules, time windows, and dynamic events. NCO has responded with specialized architectures for temporal routing problems.

4.1 The Periodic Capacitated VRP (PVRP)

The **Periodic Capacitated VRP (PVRP)** extends the planning horizon from a single day to a period T (e.g., a week or month). A customer i has a service frequency f_i (e.g., 3 times/week) and a set of allowable visit patterns P_i (e.g., {Mon-Wed-Fri, Tue-Thu-Sat}). The solver must make two coupled decisions:

1. **Tactical Level:** Assign a visit pattern to each customer.
2. **Operational Level:** Solve the daily CVRPs for the resulting customer allocations.[3]

Hierarchical Neural Frameworks:
Neural approaches for PVRP typically adopt a Hierarchical Reinforcement Learning (HRL) strategy.25

- **The Upper-Level Agent (Manager):** This is a policy network that observes the global customer distribution and demand profiles. It outputs a pattern assignment for each customer. This action effectively decomposes the PVRP into T independent CVRP sub-problems.
- **The Lower-Level Agent (Worker):** This is a standard constructive neural solver (like POMO or AM) that solves the daily CVRP instances generated by the Upper-Level Agent.
- **End-to-End Reward Signal:** The total distance of the daily routes serves as the negative reward. Crucially, this reward is backpropagated to the Upper-Level Agent. The Manager learns to assign patterns not just to balance daily demand, but to maximize geographical clustering—learning, for example, to assign all northern customers to Monday and all southern customers to Tuesday, thereby minimizing travel cost.[28]

PVRP with Service Choice (PVRP-SC):
In this variant, the visit frequency itself is a decision variable. Visiting a customer more frequently might increase routing costs but generates higher revenue or service bonuses (or reduces inventory holding costs at the client site).

- **Service Head:** Neural models for PVRP-SC augment the Upper-Level encoder with a "Service Head." This module outputs a probability distribution over valid service frequencies (e.g., 1, 2, or 3 visits) alongside the pattern selection.
- **Mechanism:** The network learns the implicit trade-off: isolated rural customers are assigned minimum frequency to save travel time, while dense urban clusters are assigned maximum frequency to maximize service quality scores.[29]

4.2 Dynamic VRP (DVRP) and Deep Reinforcement Learning

In **Dynamic VRP (DVRP)**, the problem is not fully known at the start. New customer requests

arrive stochastically while the vehicle fleet is already executing the route. The goal is to maximize the number of served customers or minimize rejection rates/delay.[4]

MDP Formulation:
DVRP is modeled as a sequential Markov Decision Process (MDP).

- **State Space S_t:** Includes the current location and capacity of all vehicles, the status of current orders (pending, picked up, delivered), and the current time t.
- **Action Space A_t:** When a new event occurs (e.g., new order arrival or vehicle completes a job), the agent must decide to: (1) Assign the new order to a vehicle and insert it into the current route, (2) Reject the order, or (3) Re-optimize the remaining queue.
- **Reward R_t:** Immediate rewards for service completion minus penalties for travel distance and wait time.

Dynamic Attention and Snapshot Embeddings:
Standard static encoders fail here because the graph grows over time. Dynamic Attention Networks 4 utilize dynamic embedding updates.

- **Snapshot Encoding:** The problem timeline is treated as a sequence of "snapshots." At each decision point, the current graph (including new nodes) is encoded.
- **Temporal Features:** The encoder explicitly ingests temporal features: request arrival time t_{arr}, time window $[e_i, l_i]$, and current global time.
- **The "Wait" Action:** A critical innovation in RL for DVRP is the explicit "Wait" action. A naive greedy heuristic always moves. A trained RL agent, however, can learn to *wait* at a central location if the value function predicts a high probability of a new, high-priority order appearing nearby in the immediate future. This "anticipatory" behavior is a key advantage of Deep RL over classical reactive heuristics.[32]

4.3 Time-Dependent VRP (TDVRP)

In **TDVRP**, the cost (travel time) between two nodes is a function of the departure time: $c_{ij}(t)$. This models traffic congestion (rush hours).

- **Functional Approximation:** Instead of a static distance matrix, the network must learn the travel time profile. Some approaches use **Hypernetworks** to generate the weights of the edge encoder based on the current time slice.
- **Spatio-Temporal Transformers:** Recent architectures [34] use Transformer layers where the attention scores are modulated by sinusoidal Positional Encodings representing the time of day. This allows the model to "attend" differently to an edge depending on whether the planned traversal is at 8:00 AM (high cost) or 11:00 AM (low cost).

4.4 Inventory Routing Problem (IRP)

Closely related to PVRP is the **Inventory Routing Problem (IRP)**, where the distributor manages the customer's inventory levels and decides when to replenish.

- **Constrained RL (CRL):** IRP introduces hard constraints on inventory (stockouts are forbidden). Standard RL struggles with such hard constraints. **Constrained Reinforcement Learning (CRL)** frameworks [36] utilize Lagrangian relaxation techniques. They learn Lagrange multipliers that dynamically weight the penalty for stockouts in the reward function, guiding the policy toward the feasible region without destabilizing training.
- **MatNet for IRP:** MatNet-like architectures are increasingly used to model the "Customer-Day" compatibility matrix in IRP, effectively learning a replenishment policy that synchronizes with routing efficiencies.[36]

---

5. Neural Improvement Heuristics: The Search for Perfection

While constructive solvers prioritize speed, **Improvement Heuristics** prioritize quality. They mimic classical metaheuristics (Local Search, Large Neighborhood Search) but replace random operators with learned policies.

5.1 Neural Large Neighborhood Search (NLNS)

Classical Large Neighborhood Search (LNS) works by repeatedly destroying part of a solution (removing K customers) and repairing it (re-inserting them). The "destroy" operator is usually random or heuristic (e.g., "remove customers with high removal cost").

Neural LNS (NLNS) [38] learns the "Destroy" operator.

- **The Learned Ruin Operator:** A CNN or GNN analyzes the current complete tour. It outputs a probability score for each node, indicating the likelihood that removing and re-inserting this node will lead to a better global solution. The model learns to identify "structural defects" in the tour—nodes that cause detours or edge crossings—that are not obvious to simple heuristics.
- **The Neural Repair:** The repair step uses a constructive neural solver (like a mini-AM) to re-insert the removed nodes optimally. Because it only deals with a small subset of nodes ($K \approx 20-50$), the neural repair is extremely fast and precise.
- **Scalability:** NLNS is one of the few neural methods that scales effectively to instances with thousands of nodes. By breaking the large problem into a sequence of small local repairs, it bypasses the quadratic complexity of global attention.[40]

5.2 NeuLNS: Learning to Design Heuristics

NeuLNS [41] takes a meta-learning approach. Instead of learning *where* to apply an operator, it learns *which* operator to apply.

- **Operator Portfolio:** The system has access to a library of classical operators: 2-opt,

3-opt, Node Swap, Or-opt, Shift.

- **The Policy:** An RL agent observes the state of the search (current cost, improvement rate, stagnation counter, graph topology). It selects the next operator to apply.
- **Hyper-Heuristic Behavior:** The model effectively learns an algorithm. On clustered instances, it might learn to favor "Swaps" to fix intra-cluster routing. On random instances, it might rely on 2-opt. This adaptability allows NeuLNS to outperform static metaheuristics that use a fixed sequence of operators.[43]

5.3 Learning 2-opt and Local Search

Recent work [44] focuses specifically on the **2-opt** operator, the workhorse of TSP solvers.

- **GFlowNet-Guided 2-opt:** Instead of exhaustively checking all $O(N^2)$ possible 2-opt moves (which is slow for large N), a Graph Neural Network (GNN) trained with GFlowNet principles predicts a "heatmap" of promising 2-opt moves. The local search only evaluates moves indicated by the heatmap.
- **Efficiency:** This reduces the complexity of the local search step from quadratic to near-linear, enabling the application of sophisticated local search to massive graphs.

---

6. The Rise of Large Language Models (LLMs) in Optimization

The years 2024 and 2025 mark the entry of **Large Language Models (LLMs)** into the optimization arena. Their utility is not in number crunching—at which they are inefficient—but in reasoning, code generation, and constraint handling.

6.1 LLMs as End-to-End Optimizers (FOARL)

Can an LLM solve a TSP by reading the coordinates as text and outputting the tour as text?

- **The Challenge:** Zero-shot LLMs (like GPT-4) perform poorly on geometric reasoning. They hallucinate coordinates and fail to construct valid cycles.
- **The Solution: FOARL (Feasibility-and-Optimality-Aware RL):** This framework [45] fine-tunes an LLM (e.g., Llama-3 7B) specifically for optimization.
  - *Stage 1: Supervised Fine-Tuning (SFT):* The model is trained on textual representations of VRP instances paired with optimal solution strings generated by a solver. This teaches the model the *syntax* of a valid solution.
  - *Stage 2: RL with Feasibility Rewards:* The model is further trained using PPO. The reward function includes severe penalties for infeasibility (e.g., visiting a node twice, exceeding capacity) and positive rewards for solution quality (short distance).
- **Performance:** While not yet beating HGS, FOARL-tuned models achieve optimality gaps of 1–8% on small instances. Their key value is **flexibility**: they can handle novel, text-described constraints (e.g., "Do not visit Node A before Node B") without retraining

the architecture, simply by including the constraint in the prompt.[47]

6.2 Evolution of Heuristics (EoH) and Code Generation

A more robust application is using LLMs to **write the solver code** rather than *be* the solver.

- **EoH Framework: Evolution of Heuristics (EoH)** [48] treats the LLM as a mutation operator in a genetic algorithm.
  - *Prompt:* "Write a Python function to swap two nodes in a TSP tour to minimize distance."
  - *Evaluation:* The generated code is executed on a benchmark set.
  - *Evolution:* The best-performing code snippets are fed back into the LLM with the prompt: "Here is a good heuristic. Improve it to handle edge cases better."
- **Discovery:** Through this iterative process, LLMs have successfully rediscovered classical highly-optimized operators (like Lin-Kernighan logic) and, more interestingly, generated novel heuristic variants tailored to specific hardware constraints (e.g., GPU-efficient parallel implementations).[49]

6.3 Neural Solver Selection

LLMs also serve as high-level **dispatchers** in ensemble systems.

- **Contextual Understanding:** An LLM analyzes the metadata of a problem instance (e.g., "This is a VRP with highly clustered customers and tight time windows").
- **Solver Routing:** Based on this description, it routes the problem to the most appropriate neural solver (e.g., "Use POMO for clustered data, use MatNet if the matrix is asymmetric"). This meta-learning approach [1] significantly reduces the average optimality gap by matching the problem type to the solver's inductive bias.

---

7. Scalability: Conquering the 10,000-Node Barrier

Scaling NCO to "industrial" sizes (1,000 to 100,000 nodes) remains the primary technical hurdle. The attention mechanism in Transformers has $O(N^2)$ memory and compute complexity, which becomes prohibitive beyond $N \approx 500$.

7.1 GLOP: Global Partition, Local Construction

GLOP (Global and Local Optimization Policies) [51] represents the definitive solution to the scalability problem in 2024/2025. It abandons the idea of a single monolithic network in favor of a **Divide-and-Conquer** strategy.

The Hierarchical Workflow:

1. **Global Partitioning (The "Manager"):** A high-level policy partitions the large graph (e.g., 10,000 nodes) into K smaller sub-regions or clusters (e.g., 100 clusters of 100

nodes).
    - *Objective:* This is not a random grid. The partitioner minimizes the "cut weight"—the cost of edges connecting different clusters—ensuring that the resulting sub-problems are as independent as possible. It often uses a non-autoregressive GNN heatmap to identify natural clusters.
2. **Local Construction (The "Worker"):** Each sub-problem (cluster) is solved independently and in parallel by a high-precision neural solver (like POMO or Sym-NCO). Because $N_{sub} \approx 100$, these solvers operate in their optimal performance regime (high accuracy, low latency).
3. **Boundary Merging:** The sub-tours are stitched together using a specialized boundary-aware policy that optimizes the connections between clusters.

Performance: GLOP achieves state-of-the-art results on 10,000-node instances. Crucially, it scales **linearly** ($O(N)$) rather than quadratically, because the size of the sub-problems is constant regardless of the total problem size. It effectively hybridizes the speed of constructive solvers with the decomposition logic of exact OR methods.[53]

7.2 Generalization Gaps

A persistent issue in NCO is **Zero-Shot Generalization**. A model trained on 50 nodes often fails when tested on 100 nodes, and a model trained on uniform data fails on clustered data.

- **Size Generalization:** GLOP solves the size generalization problem by reducing all problems to small sub-problems. **LCH-Regret** helps by allowing the model to recover from errors induced by distribution shifts.
- **Distribution Generalization: Sym-NCO** remains the most effective technique here. By forcing the model to be invariant to geometric transformations, it prevents overfitting to the specific coordinate range or density of the training set. New benchmarks like **FrontierCO** [54] and **RoutBench** [55] are now standardizing the evaluation of this robustness, punishing models that cannot adapt to novel distributions.

---

8. Conclusion and Future Outlook

The state of Machine Learning for Combinatorial Optimization in 2025 is defined by **specialization, hybridization, and temporal awareness**. We have moved past the era of the "generic TSP solver."

- **Architectures** have specialized: MatNet handles matrix inputs; Dynamic Attention handles time; Anisotropic GNNs handle diffusion.
- **Methods** have hybridized: GLOP integrates decomposition with learning; LCH-Regret integrates search with construction; FOARL integrates language reasoning with reinforcement learning.
- **Scope** has expanded: The focus has shifted from static geometric problems to the

messy, dynamic, temporal reality of PVRP, DVRP, and IRP.

The future of the field lies in **Grey Box Optimization**. We are moving away from treating the solver as a black box end-to-end neural network. Instead, the solvers of the future will be modular systems where Deep Learning provides the intuition (heatmaps, learned heuristics, partition strategies) and classical Operations Research provides the structure (constraints, tree search, decomposition), all orchestrated perhaps by a reasoning agent derived from Large Language Models. This synthesis promises to finally bridge the gap between the speed of AI and the reliability of rigorous optimization.

Works cited

1. Neural Solver Selection for Combinatorial Optimization | OpenReview, accessed December 17, 2025, https://openreview.net/forum?id=CFLEIeX7iK
2. (PDF) Neural combinatorial optimization with reinforcement learning in industrial engineering: a survey - ResearchGate, accessed December 17, 2025, https://www.researchgate.net/publication/389008680_Neural_combinatorial_optimization_with_reinforcement_learning_in_industrial_engineering_a_survey
3. A set-covering based heuristic algorithm for the periodic vehicle routing problem - NIH, accessed December 17, 2025, https://pmc.ncbi.nlm.nih.gov/articles/3990422/
4. Deep Reinforcement Learning for Dynamic Capacitated Vehicle Routing Problem, accessed December 17, 2025, https://openreview.net/forum?id=Gs8jWk0F01¬eId=4JLV3CIkml
5. DEEP REINFORCEMENT LEARNING FOR DYNAMIC CAPACITATED VEHICLE ROUTING PROBLEM - OpenReview, accessed December 17, 2025, https://openreview.net/pdf?id=Gs8jWk0F01
6. POMO: Policy Optimization with Multiple Optima for Reinforcement Learning, accessed December 17, 2025, https://papers.neurips.cc/paper_files/paper/2020/file/f231f2107df69eab0a3862d50018a9b2-Paper.pdf
7. Learning Encodings for Constructive Neural Combinatorial Optimization Needs to Regret, accessed December 17, 2025, https://ojs.aaai.org/index.php/AAAI/article/view/30069/31882
8. alstn12088/Sym-NCO - GitHub, accessed December 17, 2025, https://github.com/alstn12088/Sym-NCO
9. Sym-NCO: Leveraging Symmetricity for Neural Combinatorial ..., accessed December 17, 2025, https://arxiv.org/pdf/2205.13209
10. Sym-NCO: Leveraging Symmetricity for Neural Combinatorial Optimization - OpenReview, accessed December 17, 2025, https://openreview.net/pdf?id=kHrE2vi5Rvs
11. Matrix Encoding Networks for Neural Combinatorial Optimization, accessed December 17, 2025, https://papers.neurips.cc/paper_files/paper/2021/file/29539ed932d32f1c56324cded92c07c2-Paper.pdf

12. [PDF] Neural Improvement Heuristics for Graph Combinatorial Optimization Problems | Semantic Scholar, accessed December 17, 2025, https://www.semanticscholar.org/paper/Neural-Improvement-Heuristics-for-Graph-Problems.-Garmendia-Ceberio/e6f6997eeab46c96aa82a13570e6a6c685c69b35
13. Matrix Encoding Networks for Neural Combinatorial Optimization - arXiv, accessed December 17, 2025, https://arxiv.org/pdf/2106.11113
14. DET: LEARN TO SOLVE THE TUNNEL TRAVELING SALESMAN PROBLEM USING DOUBLE-ENCODER TRANSFORMER - OpenReview, accessed December 17, 2025, https://openreview.net/pdf?id=2YzeOOjvOi
15. Learning Encodings for Constructive Neural Combinatorial Optimization Needs to Regret, accessed December 17, 2025, https://www.researchgate.net/publication/379290275_Learning_Encodings_for_Constructive_Neural_Combinatorial_Optimization_Needs_to_Regret
16. DIFUSCO: Graph-based Diffusion Solvers for Combinatorial Optimization - arXiv, accessed December 17, 2025, https://arxiv.org/html/2302.08224v2
17. [PDF] DIFUSCO: Graph-based Diffusion Solvers for Combinatorial Optimization, accessed December 17, 2025, https://www.semanticscholar.org/paper/DIFUSCO%3A-Graph-based-Diffusion-Solvers-for-Sun-Yang/5f90d43e6ece5c6ee6e8186e4b57d46c85377713
18. [2302.08224] DIFUSCO: Graph-based Diffusion Solvers for Combinatorial Optimization - arXiv, accessed December 17, 2025, https://arxiv.org/abs/2302.08224
19. Blackout DIFUSCO: Exploring Continuous-Time Dynamics in Discrete Optimization - arXiv, accessed December 17, 2025, https://arxiv.org/pdf/2502.05221?
20. DIFUSCO: Graph-based Diffusion Solvers for Combinatorial ..., accessed December 17, 2025, https://proceedings.neurips.cc/paper_files/paper/2023/file/0ba520d93c3df592c83a611961314c98-Paper-Conference.pdf
21. [Quick Review] DIFUSCO: Graph-based Diffusion Solvers for Combinatorial Optimization, accessed December 17, 2025, https://liner.com/review/difusco-graphbased-diffusion-solvers-for-combinatorial-optimization
22. Blackout DIFUSCO: Exploring Continuous-Time Dynamics in Discrete Optimization - arXiv, accessed December 17, 2025, https://arxiv.org/html/2502.05221v1
23. [Literature Review] Blackout DIFUSCO - Moonlight, accessed December 17, 2025, https://www.themoonlight.io/en/review/blackout-difusco
24. A Genetic Algorithm for Solving Periodic Heterogeneous Vehicle Routing Problem - e-journal umm, accessed December 17, 2025, https://ejournal.umm.ac.id/index.php/industri/article/download/18453/12121/84715
25. Hierarchical reinforcement learning in network routing optimization - DiVA portal, accessed December 17, 2025, http://www.diva-portal.org/smash/get/diva2:1955666/FULLTEXT01.pdf

26. [2502.08340] Hierarchical Learning-based Graph Partition for Large-scale Vehicle Routing Problems - arXiv, accessed December 17, 2025, https://arxiv.org/abs/2502.08340
27. Hierarchical Deep Reinforcement Learning for Vehicle Routing Problem - OpenReview, accessed December 17, 2025, https://openreview.net/forum?id=6G7cF9RNzP
28. Hierarchical Learning-based Graph Partition for Large-scale Vehicle Routing Problems | Request PDF - ResearchGate, accessed December 17, 2025, https://www.researchgate.net/publication/388954586_Hierarchical_Learning-based_Graph_Partition_for_Large-scale_Vehicle_Routing_Problems
29. The Period Vehicle Routing Problem with Service Choice - ResearchGate, accessed December 17, 2025, https://www.researchgate.net/publication/220413114_The_Period_Vehicle_Routing_Problem_with_Service_Choice
30. The Period Vehicle Routing Problem with Service Choice - PubsOnLine - INFORMS.org, accessed December 17, 2025, https://pubsonline.informs.org/doi/10.1287/trsc.1050.0140
31. Optimizing a Dynamic Vehicle Routing Problem with Deep Reinforcement Learning: Analyzing State-Space Components - MDPI, accessed December 17, 2025, https://www.mdpi.com/2305-6290/8/4/96
32. A comparison of reinforcement learning policies for dynamic vehicle routing problems with stochastic customer requests - R Discovery, accessed December 17, 2025, https://discovery.researcher.life/article/a-comparison-of-reinforcement-learning-policies-for-dynamic-vehicle-routing-problems-with-stochastic-customer-requests/c55bf9d42ce7321dacc4861e78de2ca1
33. A Combined Diffusion Model and Reinforcement Learning Approach for Solving the Vehicle Routing Problem With Multiple Soft Time Windows - IEEE Xplore, accessed December 17, 2025, https://ieeexplore.ieee.org/iel8/6287639/10820123/11053837.pdf
34. Vehicle Routing Optimization with Graph Embedding Based on Deep Reinforcement Learning, accessed December 17, 2025, https://waseda.repo.nii.ac.jp/record/2004406/files/Honbun-9635.pdf
35. Machine learning multi-objective optimization for time-dependent green vehicle routing problem | Request PDF - ResearchGate, accessed December 17, 2025, https://www.researchgate.net/publication/392951691_Machine_learning_multi-objective_optimization_for_time-dependent_green_vehicle_routing_problem
36. Enhanced multi-task deep reinforcement learning for the integrated inventory-routing problem under VMI mode - ResearchGate, accessed December 17, 2025, https://www.researchgate.net/publication/396080568_Enhanced_multi-task_deep_reinforcement_learning_for_the_integrated_inventory-routing_problem_under_VMI_mode
37. [2503.05276] Constrained Reinforcement Learning for the Dynamic Inventory Routing Problem under Stochastic Supply and Demand - arXiv, accessed

December 17, 2025, https://arxiv.org/abs/2503.05276
38. Neural Large Neighborhood Search for Routing Problems | Request PDF - ResearchGate, accessed December 17, 2025, https://www.researchgate.net/publication/363713160_Neural_Large_Neighborhood_Search_for_Routing_Problems
39. ahottung/NLNS: Neural Large Neighborhood Search for the Capacitated Vehicle Routing Problem - GitHub, accessed December 17, 2025, https://github.com/ahottung/NLNS
40. [PDF] Neural Large Neighborhood Search for the Capacitated Vehicle Routing Problem, accessed December 17, 2025, https://www.semanticscholar.org/paper/Neural-Large-Neighborhood-Search-for-the-Vehicle-Hottung-Tierney/04636d0edc2c6822af5119bef83efdc63b65dc14
41. water-mirror/NeuLNS: Neural Large Neighborhood Search: Learn to Design Heuristics for Vehicle Routing Problem (VRP), by Deep Learning and Reinforcement Learning. - GitHub, accessed December 17, 2025, https://github.com/water-mirror/NeuLNS
42. Learn to Design the Heuristics for Vehicle Routing Problem - ResearchGate, accessed December 17, 2025, https://www.researchgate.net/publication/339399371_Learn_to_Design_the_Heuristics_for_Vehicle_Routing_Problem
43. learn to design the heuristics for vehicle routing problem - arXiv, accessed December 17, 2025, https://arxiv.org/pdf/2002.08539
44. AGOF: A GFlowNet-Guided 2-Opt Framework for Vehicle Routing Problems | OpenReview, accessed December 17, 2025, https://openreview.net/forum?id=kivcvgV52Z
45. Large Language Models as End-to-end Combinatorial Optimization Solvers - NeurIPS 2025, accessed December 17, 2025, https://neurips.cc/virtual/2025/poster/115835
46. Large Language Models as End-to-end Combinatorial Optimization Solvers - OpenReview, accessed December 17, 2025, https://openreview.net/pdf/4296c5041f16629e2699894c324d0916fc60f7a3.pdf
47. Large Language Models as End-to-end Combinatorial Optimization Solvers - OpenReview, accessed December 17, 2025, https://openreview.net/forum?id=qr5uMEs6iR
48. dongjinkun/COPZoo: Neural Combinatorial Optimization - GitHub, accessed December 17, 2025, https://github.com/dongjinkun/COPZoo
49. A Systematic Survey on Large Language Models for Evolutionary Optimization: From Modeling to Solving - arXiv, accessed December 17, 2025, https://arxiv.org/html/2509.08269v1
50. Neural Solver Selection for Combinatorial Optimization - arXiv, accessed December 17, 2025, https://arxiv.org/html/2410.09693v1
51. GLOP: Learning Global Partition and Local Construction for Solving Large-scale Routing Problems in Real-time | Request PDF - ResearchGate, accessed December 17, 2025, https://www.researchgate.net/publication/376357887_GLOP_Learning_Global_Par

tition_and_Local_Construction_for_Solving_Large-scale_Routing_Problems_in_Real-time
52. GLOP: Learning Global Partition and Local Construction for Solving Large-Scale Routing Problems in Real-Time | Request PDF - ResearchGate, accessed December 17, 2025, https://www.researchgate.net/publication/379283035_GLOP_Learning_Global_Partition_and_Local_Construction_for_Solving_Large-Scale_Routing_Problems_in_Real-Time
53. Improving Generalization of Neural Combinatorial Optimization for Vehicle Routing Problems via Test-Time Projection Learning - arXiv, accessed December 17, 2025, https://arxiv.org/html/2506.02392v2
54. A Comprehensive Evaluation of Contemporary ML-Based Solvers for Combinatorial Optimization - arXiv, accessed December 17, 2025, https://arxiv.org/pdf/2505.16952
55. ARS: Automatic Routing Solver with Large Language Models - OpenReview, accessed December 17, 2025, https://openreview.net/forum?id=Tw1kyA5akb