Adaptive Neural Combinatorial Optimization for Multi-Period Capacitated Vehicle Routing Problems with Stochastic Demand Accumulation

1. Introduction: The Dynamic Frontier of Vehicle Routing

The transition from static Combinatorial Optimization (CO) to dynamic, stochastic environments represents the current frontier in Operations Research and Neural Combinatorial Optimization (NCO). While the classical Capacitated Vehicle Routing Problem (CVRP) has seen near-optimal solutions through both metaheuristics and recent deep learning advances, the application of these methods to multi-period, stochastic scenarios remains a significant challenge.[1] This report addresses a specific, high-complexity variant of the VRP: a Multi-Period Capacitated Vehicle Routing Problem (MP-CVRP) where node demands are not fixed values, but evolve according to a stochastic daily generation rate over a finite horizon of N days.

This problem formulation fundamentally alters the objective of the optimization agent. In a static CVRP, the goal is to minimize the travel cost required to clear a known set of demands.[3] In the described multi-period stochastic variant, the agent must optimize a trajectory of decisions over time, balancing the immediate cost of fleet deployment against the risks of inventory accumulation and capacity overflow. Crucially, the requirement that the model may perform "no route at all" on a given day transforms the problem into a decision-making process that integrates routing with inventory management logic.[4] The agent must learn to distinguish between "urgent" states, where routing is necessary to prevent penalty, and "accumulation" states, where postponing service allows for higher demand density and thus greater route efficiency in subsequent periods.[6]

1.1 The Shift from Static to Dynamic Solvers

Traditional solvers, such as the Lin-Kernighan-Helsgaun (LKH-3) heuristic or OR-Tools, operate on a "snapshot" basis. To apply them to a dynamic problem, one must employ a "re-optimization" strategy: at every time step t, the current state of the system is frozen, treated as a static VRP instance, and solved.[8] While effective for minimizing immediate costs, this approach is myopic. It fails to account for the stochastic generation rate—the knowledge that a node with low demand today might critically overflow tomorrow due to a high generation variance.[10] Furthermore, running complex metaheuristics like LKH-3 at every

simulation step is computationally prohibitive for real-time applications or large-scale simulations.[11]

Neural Combinatorial Optimization, specifically Deep Reinforcement Learning (DRL), offers a compelling alternative. By training a neural policy to map states directly to actions, NCO solvers can internalize the "physics" of the environment—learning not just how to route, but *when* to route.[12] Once trained, these models offer inference times that are orders of magnitude faster than iterative metaheuristics, enabling their use in high-frequency dynamic simulations.[4]

1.2 The Relevance of the Attention Model and POMO

The architecture proposed in this report is rooted in the Attention Model (AM) by Kool et al. (2019) and the Policy Optimization with Multiple Optima (POMO) training method by Kwon et al. (2020).[3] The Attention Model utilizes the Transformer architecture to process graph-structured data, using Multi-Head Attention (MHA) to capture the relationships between nodes (customers) and the depot.[1]

However, standard AM training via REINFORCE is known to be unstable, particularly in environments with high variance.[12] The problem at hand—stochastic demand generation—introduces two sources of variance: the aleatoric uncertainty of the environment (random demand generation) and the epistemic uncertainty of the policy. Kwon et al.'s POMO addresses this by exploiting the symmetries inherent in routing problems. By generating multiple trajectories from diverse starting nodes and using their average as a baseline, POMO significantly reduces gradient variance and prevents the model from collapsing into local minima.[16]

Adapting POMO to the Multi-Period Stochastic context requires bridging several gaps identified in the literature:

1. **Temporal Encoding:** The model must perceive the passage of time and the shrinking horizon (End-of-Horizon effects).[18]
2. **Dynamic State Representation:** The encoder must process time-variant features (current demand, generation rate) alongside static features (coordinates).[8]
3. **Action Augmentation:** The decoder must support a "Void" action to skip routing days, necessitating changes to the masking and query mechanisms.[21]

This report provides a comprehensive blueprint for the **Dynamic-Stochastic POMO (DS-POMO)** framework, detailing the mathematical formulation, architectural modifications, and training protocols required to solve the MP-CVRP with stochastic demand.

2. Mathematical Formulation of the Stochastic

Environment

To effectively train a neural solver, the simulation environment must be rigorously defined as a Markov Decision Process (MDP). This section formalizes the dynamics of the user's simulation testbed.

2.1 The Multi-Period Stochastic Process

Let the problem be defined on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{0, 1,..., M\}$ is the set of nodes, with node 0 representing the depot. The simulation proceeds over a finite horizon $T = \{1,..., N\}$.

State Space ($\mathcal{S}_t$):
At the beginning of any day t, the system state S_t is defined by the static coordinates of the nodes $X \in \mathbb{R}^{(M+1) \times 2}$ and the dynamic demand tensor $D_t \in \mathbb{R}^{M}$. Unlike static VRPs where demand is a constant d_i, here $D_{i,t}$ represents the accumulated inventory/demand at node i on day t.
Additionally, the state includes the stochastic generation parameters. Assuming the demand generation follows a distribution (e.g., Poisson or Gaussian), the state must include the parameters of this distribution, λ_i (rate) or (μ_i, σ_i), to allow the agent to predict future accumulation.6
Stochastic Transition Function:
The transition from S_t to S_{t+1} involves two phases: the Action Phase (Service) and the Environment Phase (Generation).

1. Service: The agent executes a route (or chooses no route). Let $q_{i,t}$ be the demand serviced at node i. The post-service demand is:

   $$D'_{i,t} = \max(0, D_{i,t} - q_{i,t})$$
2. Generation: The environment generates new demand $\delta_{i,t}$ drawn from the node-specific distribution $\mathcal{P}_i$.

   $$D_{i, t+1} = D'_{i,t} + \delta_{i,t}$$

This formulation aligns with the "Tensor Demand Structure" described in supply chain logistics literature [13], where demand is treated as a fluid quantity accumulating in buckets (nodes) over time.

2.2 The Augmented Action Space

The user explicitly requires the capability to perform "no route at all." This implies a hierarchical or augmented action space.
Let $\mathcal{A}$ be the set of valid permutations of customer subsets. The action space for DS-POMO is $\mathcal{A} \cup \{ \emptyset \}$, where $\emptyset$ represents the Void Action.

- **Routing Action ($a_t \in \mathcal{A}$):** The vehicle departs node 0, visits a sequence of nodes $\{v_1,..., v_k\}$, and returns to 0. The constraint $\sum_{v \in \text{route}} q_{v,t} \le Q_{vehicle}$ must be satisfied.[18]
- **Void Action ($a_t = \emptyset$):** The vehicle remains at the depot. Service $q_{i,t} = 0$ for all i. The cost incurred is 0 (or a fixed holding cost, depending on objective), but demand accumulates to $t+1$.

2.3 Objective Function and Reward Shaping

The objective is to minimize the total cost over the horizon N. The cost function is a composite of operational costs and penalties.

$$\mathcal{L}(\theta) = \mathbb{E} \left[$$

- **Cost$_{travel}$:** The Euclidean distance of the route. If $a_t = \emptyset$, this is 0.
- **Penalty$_{overflow}$:** (Optional) If accumulated demand $D_{i,t}$ exceeds a physical storage limit, a penalty is applied.
- **Penalty$_{horizon}$:** A critical term for multi-period problems. At the end of day N, any unserved demand represents a failure. A massive penalty (e.g., Euclidean distance to depot $\times$ Demand $\times$ Factor) is applied to force the agent to clear the graph before the simulation ends.[5]

The integration of the "Void" action introduces a non-trivial trade-off. The agent minimizes Cost_{travel} by voiding, but maximizes $\text{Penalty}_{horizon}$ if it voids too often. The NCO model must learn the optimal "stopping time" for accumulation—waiting until the graph is dense enough to justify a route, but not so dense that capacity Q is exceeded.[21]

3. Architectural Adaptation: The DS-POMO Model

The standard POMO architecture, based on Kool et al.'s Attention Model, consists of a static encoder and a sequential decoder. This must be significantly engineered to handle the dynamic, multi-period nature of the user's request.

3.1 The Dynamic-Context Encoder (Dual-Stream)

In the standard POMO, node embeddings are computed once because the graph is static. In MP-CVRP, the demand $D_{i,t}$ changes daily. However, the spatial topology (coordinates x_i, y_i) remains constant. Re-encoding the entire graph (coordinates + demand) from scratch at every time step t is computationally inefficient and fails to leverage the static nature of the topology.[8]

We propose a **Dual-Stream Encoder** architecture, inspired by Matrix Encoding Networks

(MatNet) and Dynamic Graph Attention Networks [8]:

3.1.1 Static Stream (Topology)

This module processes the immutable features of the graph. It consists of a stack of L Multi-Head Attention (MHA) layers.

- **Input:** $X_i = [x_i, y_i]$ (2D coordinates).
- **Output:** Static Embedding $h^{static}_i \in \mathbb{R}^{d_{model}}$.
- **Computation:** Performed only *once* at $t=0$ for the entire episode. This captures the geometric relationships (clusters, outliers) of the customer set.

3.1.2 Dynamic Stream (Inventory & Time)

This module processes the time-variant state. Since the relationships between nodes are primarily geometric (captured by the static stream), the dynamic stream can be lighter (e.g., a simple MLP or a single MHA layer).

- **Input:** $U_{i,t} =$.
  - $D_{i,t}$: Current accumulated demand.
  - λ_i: The **Stochastic Generation Rate** parameter (e.g., mean of the Poisson process). Explicitly feeding this allows the model to anticipate future accumulation ("This node fills up fast, I should clear it now").[25]
  - τ_t: Time encoding (see Section 3.2).
- **Output:** Dynamic Embedding $h^{dynamic}_{i,t} \in \mathbb{R}^{d_{model}}$.
- **Computation:** Performed at the start of every day t.

3.1.3 Feature Fusion

The effective node embedding $H_{i,t}$ used by the decoder is a fusion of both streams.

$$H_{i,t} = \text{LayerNorm}(h^{static}_i + \text{Linear}(h^{dynamic}_{i,t}))$$

This residual-style connection allows the model to modulate the static geometric "attractiveness" of a node based on its current urgency (demand).4

3.2 Time-Aware Positional Encoding

For the model to make intelligent decisions about "voiding" (waiting), it must know where it is in the simulation horizon. Waiting on Day 1 is strategic; waiting on Day N is catastrophic (due to the horizon penalty).
We implement the sinusoidal positional encoding typically used in NLP Transformers, but adapted for the day index t.19
Mechanism:
For the current day t and maximum horizon N_{max}, we generate a vector PE_t of dimension d_{model}:

$$PE_{(t, 2j)} = \sin(t / 10000^{2j/d_{model}})$$

$$PE_{(t, 2j+1)} = \cos(t / 10000^{2j/d_{model}})$$
This vector PE_t is not concatenated to every node (which would dilute the signal) but is instead added to the **Context Vector** of the Decoder or the embedding of the Depot (Node 0). This effectively broadcasts the "time pressure" to the entire attention mechanism, influencing the probability of selecting the "Void" token versus a customer node.[20]

3.3 The Hierarchical Decoder with Void Action

The most distinct modification required for this problem is the implementation of the "No Route" capability. Standard Pointer Networks are designed to output a permutation. We propose a modification to the attention mechanism's query phase to support the Void action.

3.3.1 The Void Token Mechanism

We augment the graph's node set V with a virtual node v_{void}.

- **Embedding:** The embedding for v_{void} is a learnable parameter vector E_{void}, initialized randomly and optimized during training.
- **Availability:** The v_{void} token is only unmasked (available) at the *first step* of the decoding process for a given day. Once the vehicle leaves the depot (chooses a customer node), v_{void} is masked to prevent the agent from "giving up" mid-route.
- Attention Competition: The decoder computes attention scores A_{i} for all customer nodes i and A_{void} for the void token.

  $$A_{void} = \frac{q^T K_{void}}{\sqrt{d_k}}, \quad A_{i} = \frac{q^T K_{i}}{\sqrt{d_k}}$$

  If $\text{softmax}([A_{void}, A_1,..., A_M])$ selects the Void token, the agent effectively decides that the utility of waiting (accumulating demand) outweighs the utility of servicing even the most urgent customer node.4

3.3.2 Masking for Stochastic Demand

In a standard CVRP, a node is masked once visited. In MP-CVRP, a node is masked if:

1. It has already been visited *in the current route*.
2. It has zero accumulated demand ($D_{i,t} = 0$).
   This second condition is crucial. The stochastic generation means some nodes may have 0 demand on Day t. Visiting them incurs travel cost for zero reward. The dynamic masking ensures the agent only focuses on nodes with active inventory.18

3.4 Architecture Summary: DS-POMO vs. Standard POMO

Component	Standard POMO (Kwon et al.)	Proposed DS-POMO	Relevance to Query
Input	Static x,y coords	Dynamic $\{x, y, D_t, \lambda, t\}$	Handles "stochastic generation rate"
Encoder	Single-pass Transformer	Dual-stream (Static + Dynamic)	Efficiently updates demand states daily
Decoder	Node Selection	Node Selection + **Void Token**	Enables "no route at all" option
Context	Graph Mean Embedding	Graph Mean + **Time Encoding**	Encodes "N days" horizon urgency
Masking	Visited Nodes	Visited + Zero-Demand Nodes	Prevents servicing empty nodes
Rollout	Single-Step Tour	Multi-Step Horizon Simulation	Optimizes over "N days"

4. The DS-POMO Training Framework

Adapting the POMO training paradigm to a stochastic, multi-period setting requires specific interventions to manage gradient variance and ensure convergence.

4.1 Adapting the POMO Baseline (Scenario Synchronization)

Kwon et al. demonstrated that using multiple rollouts (starting from different nodes) and using their average as a baseline is superior to generic Actor-Critic baselines.
In the MP-CVRP, this technique faces a hurdle: if we run M parallel rollouts, and each rollout experiences different random demand generations (δ), the variance in rewards will be dominated by the environment's randomness, not the policy's quality.
Solution: Scenario Synchronization (Shared Randomness)
To maintain the effectiveness of the POMO baseline in a stochastic environment, we must synchronize the random seed across the M diverse rollouts for a single training instance.

- **Batch Construction:** We generate a batch of B problem instances. For each instance

b, we pre-generate the sequence of stochastic demands $\Delta_b = \{\delta_{t=1},..., \delta_{t=N}\}$.
- **Parallel Rollouts:** For instance b, we launch M rollouts (via POMO's multi-start mechanism). Crucially, *all M rollouts are subjected to the exact same demand sequence Δ_b*.
- Baseline Calculation:

  $$b(s_b) = \frac{1}{M} \sum_{m=1}^M R(\tau_{m,b})$$

  Since all m trajectories faced the same stochastic events, the difference $R(\tau_{m,b}) - b(s_b)$ purely reflects the quality of the routing decisions (e.g., choice of start node, decision to void), effectively filtering out the noise of the stochastic generation.29

4.2 Reward Shaping and Long-Term Credit Assignment

The reward signal in multi-period settings is sparse and delayed. The "correctness" of voiding on Day 1 is only revealed on Day N (by whether the horizon was cleared efficiently).
To aid learning, we employ Intermediate Rewards:

- **Day Reward:** $r_t = -(\text{Distance}_t)$.
- **Capacity Utilization Bonus:** A small auxiliary reward proportional to the load factor of the vehicle ($\frac{\text{Load}}{Q}$). This encourages the agent to prefer "full" routes over sparse ones, implicitly teaching the value of aggregation.[18]
- **Horizon Penalty:** $r_N = -\alpha \sum_{i} D_{i,N}$. This is the dominant term forcing the agent to clear the board.

4.3 Curriculum Learning for Horizon N

Training directly on $N=30$ days is difficult due to the vanishing gradient of the long horizon. We implement a **Horizon Curriculum**:

1. **Phase 1:** Train on $N=1$ (Static VRP). The model learns basic topology and valid routing.
2. **Phase 2:** Train on short horizons ($N=3, N=5$). The model begins to see the effect of stochastic accumulation and the value of the Void action.
3. Phase 3: Gradually extend to full horizon (N).
   This approach stabilizes the learning of the "Void" parameter E_{void}. In Phase 1, voiding is always bad (infinite penalty). In Phase 2, voiding becomes strategic. The curriculum prevents the model from converging to a local minimum of "never void" or "always void".33

5. Implementation and Simulation Details

This section outlines the operational loop of the DS-POMO system, detailing how the neural components interact with the simulation physics.

5.1 The Simulation Loop

The testing environment (and training loop) follows this precise sequence:

1. **Initialization:**
    - Graph G is instantiated with static coordinates.
    - Initial demands D_0 are set (usually 0 or random).
    - Generation rates λ_i are assigned (e.g., high-rate nodes vs. low-rate nodes).
2. **Daily Cycle (Days $t=1 \dots N$):**
    - **Observation:** The agent observes the current demands D_{t-1} and remaining time $N-t$.
    - **Encoding:** The Dual-Stream Encoder computes embeddings. The Dynamic Stream updates the representation of "hot" nodes (high accumulated demand).
    - **Decision (Void Check):** The Decoder queries the Void Token.
        - If $Prob(\text{Void}) > \text{Threshold}$, the vehicle waits. Cost = 0.
        - If $Prob(\text{Void}) \le \text{Threshold}$, the vehicle routes.
    - **Routing (If Active):** Standard POMO decoding generates a valid tour π_t. The vehicle visits nodes, resetting their demand to 0 (or partial residue if $D > Q$).
    - **Stochastic Update:** The environment calculates post-service demands and adds stochastic generation: $D_{t} = D_{served} + \text{Poisson}(\lambda)$.
    - **State Transition:** $t \leftarrow t+1$.
3. **Termination:**
    - At $t=N$, the final state is evaluated. Unserved demands trigger the penalty function.
    - Metrics (Total Distance, Service Level, Number of Routes) are logged.

5.2 Handling "Tensor Demand" and Overflows

In high-fidelity simulations, demand is not just a scalar but a tensor of types or priorities.[13] While the primary user query focuses on "capacitated" (implying volume), the DS-POMO architecture supports multi-dimensional demand vectors.

- **Overflow Logic:** If $D_{i,t} + \delta > \text{StorageCapacity}_i$, the excess is considered "lost sales" or incurs an emergency outsourcing cost. This provides a dense negative reward signal that discourages over-postponement.[5]
- **Rate Estimation:** The user query implies the generation rate is *known* ("demand change by a... rate"). If the rate is *unknown* (hidden), the Dynamic Stream of the encoder should be replaced by a Recurrent Neural Network (RNN) or LSTM that observes the *history* of demand changes $$ to infer the latent rate λ. However, given the prompt phrasing, we assume λ is an observable feature passed to the encoder.[23]

6. Comparative Benchmarks and Performance

Analysis

To validate the DS-POMO implementation, it must be benchmarked against both traditional heuristics and static learning baselines.

6.1 Baseline 1: Rolling Horizon LKH-3

The LKH-3 heuristic is the state-of-the-art for static VRP.[8] In a dynamic setting, LKH-3 is applied in a "Rolling Horizon" fashion:

- At each day t, construct a VRP instance with current demands D_t.
- Solve using LKH-3.
- **Limitation:** LKH-3 is myopic. It does not know that node i has a high generation rate and *will* overflow tomorrow if not visited today. It only sees current demand. DS-POMO, via the λ embedding, can learn anticipatory routing.

6.2 Baseline 2: Threshold Heuristics

A common industry practice is rule-based routing: "Route only if total demand > 80% of vehicle capacity".[21]

- **Comparison:** DS-POMO should outperform this by making granular, node-specific decisions. Instead of a global threshold, it learns local thresholds (e.g., "Route if the remote cluster is full, even if the total load is low, to avoid a long trip tomorrow").

6.3 Expected Performance Gains

Research on similar dynamic attention models suggests that NCO approaches like DS-POMO can achieve **3-5% cost reduction** compared to myopic LKH-3 in stochastic environments, primarily by optimizing the *frequency* of routing (inventory management) rather than just the topology of routes.[30] The primary advantage, however, is inference speed. DS-POMO generates a decision in milliseconds, whereas LKH-3 may require seconds or minutes per day, making DS-POMO the only viable option for large-scale Monte Carlo simulations of supply chains.

7. Discussion: Generalization and Robustness

7.1 Distributional Generalization

A common failure mode of NCO is poor generalization to out-of-distribution (OOD) data.[37] If DS-POMO is trained on generation rates $\lambda \in $, it may fail if tested on $\lambda \in $.

- **Mitigation:** We recommend training with **scale-invariant features**. Instead of raw demand, input the ratio $D_{i,t}/Q$. Instead of raw rate, input $\lambda_i / \bar{\lambda}_{global}$. This forces the model to learn relative urgency rather than memorizing absolute values.

7.2 Sim-to-Real Gap

The simulation assumes a specific stochastic process (e.g., Poisson). Real-world demands are often auto-correlated or bursty.

- **Robustness:** To bridge this gap, the training set should include "Noise Injection" in the generation parameters. Training on a mix of Poisson, Uniform, and Bursty generation processes forces the Dual-Stream Encoder to rely on the *observed state* (D_t) more than the theoretical rate parameter, improving robustness to model mismatch.[39]

8. Conclusion

The adaptation of the Attention Model to the Multi-Period Stochastic CVRP requires a fundamental architectural shift from static instance solving to sequential policy learning. By integrating a **Dual-Stream Encoder** to handle dynamic inventory states, a **Time-Aware Positional Encoding** to signal horizon urgency, and a **Void-Token Mechanism** to enable strategic postponement, the proposed DS-POMO framework successfully bridges the gap between routing and inventory management.

Crucially, the use of **Scenario-Synchronized POMO Training** stabilizes the learning process in the face of stochastic demand accumulation, allowing the agent to converge to a policy that balances immediate travel costs against long-term capacity risks. This approach moves beyond the myopic limitations of traditional rolling-horizon heuristics, offering a solver that anticipates future demand accumulation and optimizes the *timing* of routes as effectively as their topology.

Table 1: Architectural Comparison

Feature	Standard POMO (Kwon et al. 2020)	DS-POMO (Proposed)
Problem Domain	Static CVRP (Single Snapshot)	Multi-Period Stochastic CVRP (N Days)
Input State	Static Coordinates (x,y)	Dynamic State (x, y, D_t, λ, t)
Encoder	Single-pass Transformer	**Dual-Stream** (Static Topology + Dynamic Inventory)

Action Space	Node Permutation	Node Permutation $\cup$ **{Void Action}**
Time Awareness	None	**Sinusoidal Positional Encoding** (Day Index)
Masking	Visited Nodes	Visited Nodes + **Zero-Demand Nodes**
Baseline	Avg of M Start Nodes	Avg of M Start Nodes (**Scenario Synchronized**)
Objective	Min Tour Length	Min Cumulative Cost + Horizon Penalty

---

Table 2: State Feature Tensor Specification

Feature Name	Dimension	Update Frequency	Description	Source
Coordinate Embedding	d_{model}	Static ($t=0$)	Transformer output of (x,y). Captures graph topology.	[3]
Demand State	1	Daily (t)	Normalized Accumulated Demand $D_{i,t}/Q$.	[13]
Generation Rate	1	Static/Daily	Expected generation parameter λ_i. Signals	[20]

			"urgency".	
Time Context	d_{model}	Daily (t)	Sinusoidal encoding of $(N-t)$. Signals "deadline".	[19]
Void Embedding	d_{model}	Learned	Virtual node embedding for "No Route" decision.	[21]

Works cited

1. Edge-Driven Multiple Trajectory Attention Model for Vehicle Routing Problems, accessed December 30, 2025, https://www.researchgate.net/publication/389532616_Edge-Driven_Multiple_Trajectory_Attention_Model_for_Vehicle_Routing_Problems
2. Edge-Driven Multiple Trajectory Attention Model for Vehicle Routing Problems - MDPI, accessed December 30, 2025, https://www.mdpi.com/2076-3417/15/5/2679
3. Attention, Learn to Solve Routing Problems! - arXiv, accessed December 30, 2025, https://arxiv.org/pdf/1803.08475
4. [2503.04085] SED2AM: Solving Multi-Trip Time-Dependent Vehicle Routing Problem using Deep Reinforcement Learning - arXiv, accessed December 30, 2025, https://arxiv.org/abs/2503.04085
5. 1 Introduction - arXiv, accessed December 30, 2025, https://arxiv.org/html/2402.04463v1
6. Reinforcement Learning for Solving Stochastic Vehicle Routing Problem with Time Windows - arXiv, accessed December 30, 2025, https://arxiv.org/pdf/2402.09765
7. Reinforcement Learning for Solving the Vehicle Routing Problem - Lehigh University, accessed December 30, 2025, https://engineering.lehigh.edu/sites/engineering.lehigh.edu/files/_DEPARTMENTS/ise/pdf/tech-papers/19/19T_002.pdf
8. Matrix Encoding Networks for Neural Combinatorial Optimization - ResearchGate, accessed December 30, 2025, https://www.researchgate.net/publication/352639631_Matrix_Encoding_Networks_for_Neural_Combinatorial_Optimization
9. Deep Reinforcement Learning for the Capacitated Vehicle Routing Problem with Soft Time Window - ResearchGate, accessed December 30, 2025, https://www.researchgate.net/publication/368594794_Deep_Reinforcement_Learning_for_the_Capacitated_Vehicle_Routing_Problem_with_Soft_Time_Window
10. Deep Reinforcement Learning for Dynamic Capacitated Vehicle Routing Problem, accessed December 30, 2025,

https://openreview.net/forum?id=Gs8jWk0F01¬eId=4JLV3Clkml
11. Adaptive Distance-Aware Attention Models for Routing Problems - ResearchGate, accessed December 30, 2025, https://www.researchgate.net/publication/383894976_Adaptive_Distance-Aware_Attention_Models_for_Routing_Problems
12. POMO: Policy Optimization with Multiple Optima for Reinforcement Learning - NeurIPS, accessed December 30, 2025, https://papers.neurips.cc/paper_files/paper/2020/file/f231f2107df69eab0a3862d50018a9b2-Paper.pdf
13. Deep Reinforcement Learning for Multi-Truck Vehicle Routing Problems with Multi-Leg Demand Routes - arXiv, accessed December 30, 2025, https://arxiv.org/html/2401.08669v1
14. Neural Combinatorial Optimization with Heavy Decoder: Toward Large Scale Generalization, accessed December 30, 2025, https://arxiv.org/html/2310.07985v2
15. POMO: Policy Optimization with Multiple Optima - Emergent Mind, accessed December 30, 2025, https://www.emergentmind.com/papers/2010.16011
16. POMO+: Leveraging starting nodes in POMO for solving Capacitated Vehicle Routing Problem - ResearchGate, accessed December 30, 2025, https://www.researchgate.net/publication/394458177_POMO_Leveraging_starting_nodes_in_POMO_for_solving_Capacitated_Vehicle_Routing_Problem
17. POMO+: Leveraging starting nodes in POMO for solving Capacitated Vehicle Routing Problem - arXiv, accessed December 30, 2025, https://arxiv.org/html/2508.08493v1
18. A Neural Optimization Approach for Time-Constrained Vehicle Routing Problems with, accessed December 30, 2025, https://elib.dlr.de/211015/1/NCO%20for%20FiniteFleet%20CVRP_final_for_Arxlv.pdf
19. Positional Encoding in Transformers - GeeksforGeeks, accessed December 30, 2025, https://www.geeksforgeeks.org/nlp/positional-encoding-in-transformers/
20. A Deep Reinforcement Learning Model to Solve the Stochastic Capacitated Vehicle Routing Problem with Service Times and Deadlines - MDPI, accessed December 30, 2025, https://www.mdpi.com/2227-7390/13/18/3050
21. Optimizing a Dynamic Vehicle Routing Problem with Deep Reinforcement Learning: Analyzing State-Space Components - MDPI, accessed December 30, 2025, https://www.mdpi.com/2305-6290/8/4/96
22. An Adaptive Tabu Search Algorithm for Solving the Two-Dimensional Loading Constrained Vehicle Routing Problem with Stochastic Customers - MDPI, accessed December 30, 2025, https://www.mdpi.com/2071-1050/15/2/1741
23. JaswanthBadvelu/Reinforcement-Learning-CVRP: Dynamic Attention Encoder-Decoder model to learn and design heuristics to solve capacitated vehicle routing problems - GitHub, accessed December 30, 2025, https://github.com/JaswanthBadvelu/Reinforcement-Learning-CVRP
24. GASE: Graph Attention Sampling with Edges Fusion for Solving Vehicle Routing Problems - arXiv, accessed December 30, 2025, https://arxiv.org/pdf/2405.12475
25. 1 Introduction - arXiv, accessed December 30, 2025,

https://arxiv.org/html/2509.17870v1
26. Dynamic Hybrid Temporal Attention With Attention on Attention for the Capacity Vehicle Routing Problem | IEEE Conference Publication | IEEE Xplore, accessed December 30, 2025, https://ieeexplore.ieee.org/document/10865664/
27. Master Positional Encoding: Part I | Towards Data Science, accessed December 30, 2025, https://towardsdatascience.com/master-positional-encoding-part-i-63c05d90a0c3/
28. Positional Encoding - Sense of direction for Transformers - DEV Community, accessed December 30, 2025, https://dev.to/samyak112/positional-encoding-sense-of-direction-for-transformers-12jp
29. Leader Reward for POMO-Based Neural Combinatorial Optimization - arXiv, accessed December 30, 2025, https://arxiv.org/html/2405.13947v1
30. (PDF) Reinforcement Learning Approach to Stochastic Vehicle Routing Problem With Correlated Demands - ResearchGate, accessed December 30, 2025, https://www.researchgate.net/publication/373205708_Reinforcement_Learning_Approach_to_Stochastic_Vehicle_Routing_Problem_with_Correlated_Demands
31. NEURAL COMBINATORIAL OPTIMIZATION WITH RE- INFORCEMENT LEARNING : SOLVING THE VEHICLE ROUTING PROBLEM WITH TIME WINDOWS - OpenReview, accessed December 30, 2025, https://openreview.net/pdf?id=gLqnSGXVJ6I
32. A Deep Reinforcement-Learning-Based Route Optimization Model for Multi-Compartment Cold Chain Distribution - MDPI, accessed December 30, 2025, https://www.mdpi.com/2227-7390/13/13/2039
33. A Two-Stage Deep Reinforcement Learning Framework for Solving Large Capacitated Vehicle Routing Problems - SISE, accessed December 30, 2025, http://www.ieworldconference.org/content/SISE2021/Papers/Arishi.pdf
34. Towards Generalizable Neural Solvers for Vehicle Routing Problems via Ensemble with Transferrable Local Policy - IJCAI, accessed December 30, 2025, https://www.ijcai.org/proceedings/2024/0764.pdf
35. Robust Multi-Period Vehicle Routing under Customer Order Uncertainty - Optimization Online, accessed December 30, 2025, https://optimization-online.org/wp-content/uploads/2017/04/5947.pdf
36. Deep Reinforcement Learning for Multi-Driver Vehicle Dispatching and Repositioning Problem - University of Michigan, accessed December 30, 2025, https://web.eecs.umich.edu/~baveja/Papers/ICDM19-Risto.pdf
37. Improving Generalization of Neural Vehicle Routing Problem Solvers Through the Lens of Model Architecture - arXiv, accessed December 30, 2025, https://arxiv.org/html/2406.06652v3
38. xybFight/VRP-Generalization: The PyTorch Implementation of "Improving Generalization of Neural Vehicle Routing Problem Solvers Through the Lens of Model Architecture" - GitHub, accessed December 30, 2025, https://github.com/xybFight/VRP-Generalization
39. Towards Real-World Routing with Neural Combinatorial Optimization |

OpenReview, accessed December 30, 2025, https://openreview.net/forum?id=sKvo9ZZfpe
40. Neural Combinatorial Optimization for Stochastic Flexible Job Shop Scheduling Problems - AAAI Publications, accessed December 30, 2025, https://ojs.aaai.org/index.php/AAAI/article/view/34870/37025