USPR: Learning a Unified Solver for Profiled Routing

Chuanbo Hua[1*], **Federico Berto**[1, 2*], **Zhikai Zhao**[1],
Jiwoo Son[3], **Changhyun Kwon**[1, 3], **Jinkyoo Park**[1, 3]

[1]KAIST, Daejeon, South Korea
[2]Radical Numerics, Palo Alto, CA, USA
[3]OMELET, Daejeon, South Korea
{cbhua, fberto, zzk020202}@kaist.ac.kr; jiwoo.son@omelet.ai; {chkwon, jinkyoo.park}@kaist.ac.kr

Abstract

The Profiled Vehicle Routing Problem (PVRP) extends the classical VRP by incorporating vehicle–client-specific preferences and constraints, reflecting real-world requirements such as zone restrictions and service-level preferences. While recent reinforcement-learning solvers have shown promising performance, they require retraining for each new profile distribution, suffer from poor representation ability, and struggle to generalize to out-of-distribution instances. In this paper, we address these limitations by introducing **U**nified **S**olver for **P**rofiled **R**outing (USPR), a novel framework that natively handles arbitrary profile types. USPR introduces on three key innovations: (i) Profile Embeddings (PE) to encode any combination of profile types; (ii) Multi-Head Profiled Attention (MHPA), an attention mechanism that models rich interactions between vehicles and clients; (iii) Profile-aware Score Reshaping (PSR), which dynamically adjusts decoder logits using profile scores to improve generalization. Empirical results on diverse PVRP benchmarks demonstrate that USPR achieves state-of-the-art results among learning-based methods while offering significant gains in flexibility and computational efficiency. We make our source code publicly available to foster future research.

Code — https://github.com/ai4co/uspr
Extended version — https://arxiv.org/abs/2505.05119

Introduction

The Vehicle Routing Problem (VRP) is an important combinatorial optimization problem that focuses on determining optimal delivery routes for a fleet of vehicles serving a set of clients. In real-world logistics operations, vehicles often have distinct characteristics that affect their suitability for serving specific clients, leading to the Profiled Vehicle Routing Problem (PVRP). This variant extends traditional VRP constraints by incorporating vehicle-client-specific preferences and operational requirements (Cordeau and Laporte 2001; Braekers, Ramaekers, and Van Nieuwenhuyse 2016; Zhong, Hall, and Dessouky 2007; Aiko, Thaithatukl, and Asakura 2018). These profiles can represent various practical considerations: specialized vehicle access permissions in urban areas, client-specific service level agreements, regulatory restrictions, or historical performance metrics that influence routing decisions (Team Locus 2020; Li et al. 2023). For instance, in last-mile delivery scenarios, certain vehicles might be preferred for specific neighborhoods based on size restrictions or noise regulations, while in B2B logistics, particular vehicle-driver combinations might maintain stronger relationships with certain clients. Fig. 1 provides an example illustration for the PVRP problem. The PVRP's inherent complexity stems from its NP-hard nature, as it generalizes the classical VRP while adding profile-specific constraints that exponentially increase the solution space (Papadimitriou and Steiglitz 1998; Golden et al. 1984). This complexity becomes particularly challenging in modern logistics operations, where organizations must optimize routes for large fleets while considering numerous client-specific requirements and dynamically changing preferences. Traditional approaches to solving PVRP typically rely on exact methods like Branch and Bound for small instances or metaheuristic algorithms such as genetic algorithms and simulated annealing for larger problems (Johnson and McGeoch 1997; Lozano, Molina, and Herrera 2011). While these methods can provide near-optimal solutions, they often require significant computational resources and extensive parameter tuning. Moreover, these approaches generally need to be redesigned or substantially modified when problem specifications change, such as when new types of prefer-

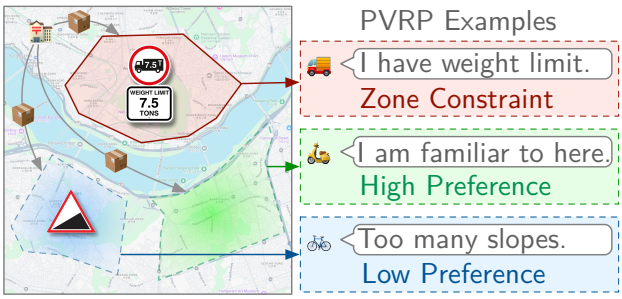

Figure 1: An illustrative example of the PVRP. The profiles are derived from real-world settings. Intuitively, zone constraints represent impassable regions; preference values indicate the "desirability" of an area for agents to visit.

*These authors contributed equally.

ences or constraints are introduced. This lack of adaptability poses a significant challenge in dynamic business environments where routing requirements frequently evolve.

Recent advances in neural combinatorial optimization (NCO), particularly through reinforcement learning (RL), have shown promising results for various VRP variants (Bello et al. 2016; Sun et al. 2019). These approaches leverage neural architectures, especially the pointer network paradigm (Vinyals, Fortunato, and Jaitly 2015; Kool, Van Hoof, and Welling 2018; Duan et al. 2020), to learn solution strategies through interaction with simulated environments. While initial work focused on basic VRP variants (Kwon et al. 2020; Kim, Park, and Park 2022b; Zheng et al. 2024), recent studies have extended these methods to handle more complex scenarios, including multi-agent routing (Zong et al. 2022), rich constrained variants (Liu et al. 2024a; Zhou et al. 2024; Bi et al. 2024), and heterogeneous fleet problems extensible to the PVRP (Li et al. 2022; Liu et al. 2024c; Hua et al. 2025).

However, existing learning-based PVRP solvers exhibit three major shortcomings: (i) *un-unified*, they must be retrained from scratch whenever the profile distribution or preference weights change, incurring prohibitive computational overhead; (ii) *context-agnostic representation*, they lack the representational capacity to capture complex and diverse vehicle–client interactions, leading to suboptimal embeddings and degraded solution quality; and (iii) *poor generalization*, they generalize poorly to out-of-distribution instances, making them fragile in dynamic or unseen settings. Together, these issues undermine the flexibility and efficiency required for real-world deployment.

To bridge these limitations, we introduce **U**nified **S**olver for **P**rofiled **R**outing (USPR), a transformer-based policy that addresses the three core shortcomings of existing methods: (i) *Profile Embeddings* (PE) encode arbitrary combinations of profile attributes and global weight parameters, removing the need to retrain for each new profile distribution; (ii) *Multi-Head Profiled Attention* (MHPA) overcomes weak representation by capturing rich, bidirectional vehicle–client interactions; and (iii) *Profile-aware Score Reshaping* (PSR) adaptively reweights decoder logits with profile scores and spatial penalties, yielding robust generalization to out-of-distribution instances, especially to large-scale instances. We summarize our main contributions as follows:

- We propose USPR, the first unified neural solver for PVRP. A single USPR model can generalize to diverse profile distributions without requiring retraining.
- We design the novel architecture built on three key components: PE for zero-shot adaptation to new profile distributions, MHPA to enhance representational capacity, and PAR to ensure robust generalization.
- We demonstrate through extensive experiments that a single USPR model significantly outperforms existing state-of-the-art methods in solution quality and improvement on out-of-distribution real-world large-scale instances, while reducing both model size and training time by an order of magnitude.

Related Work

Neural Combinatorial Optimization NCO has emerged as a powerful paradigm for solving VRPs, offering end-to-end solutions that reduce the need for manual algorithm design (Bengio, Lodi, and Prouvost 2021; Mazyavkina et al. 2021). The field was pioneered by Vinyals, Fortunato, and Jaitly (2015); Bello et al. (2016) with Pointer Networks, and later significantly advanced by Kool, Van Hoof, and Welling (2018), whose transformer-based RL framework remains the foundation for most modern neural VRP solvers. Recent developments can be broadly categorized into *construction* and *improvement* methods. Construction approaches generate solutions from scratch (Kim, Park, and Park 2022b; Grinsztajn et al. 2023; Pirnay and Grimm 2024), while improvement/search methods iteratively refine solutions (Hottung and Tierney 2020; Li, Yan, and Wu 2021; Ma, Cao, and Chee 2024). Innovations include non-autoregressive models (Kool et al. 2022; Sun and Yang 2024), population-based construction (Grinsztajn et al. 2024), and improved training strategies such as problem re-encoding (Bdeir, Falkner, and Schmidt-Thieme 2022) and test-time adaptation/search (Hottung, Kwon, and Tierney 2022; Choo et al. 2022). We focus on construction approaches due to their adaptability across VRP variants and favorable solution quality–speed tradeoffs.

NCO for Practical VRPs As NCO methods mature, increasing attention has turned to realistic "in-the-wild" VRPs with complex constraints. Early works examine the gap between synthetic and real-world settings (Duan et al. 2020). Extensions to multiple constraints and multi-task learning include (Bi et al. 2024; Drakulic, Michel, and Andreoli 2025; Liu et al. 2024a). Modeling multiple vehicles with limited fleet size has been explored via multi-agent RL (Zong et al. 2022) and autoregressive vehicle-by-vehicle reformulations (Son et al. 2024). More recent efforts address heterogeneous fleets (Li et al. 2022; Berto et al. 2024), and the more practical Profile VRPs (PRVPs) that model both vehicle heterogeneity and vehicle–node interaction profiles (Hua et al. 2025). Despite progress, neural VRP solvers still struggle with weak profile modeling, require retraining for each new preference or constraint, and generalize poorly to unseen scenarios-issues directly addressed in this work.

Preliminaries

We introduce the problem formulation of the PVRP in a *unified* manner in this section, including its Markov Decision Process (MDP) equivalent and the policy parametrization.

Problem Formulation

We consider a directed graph $G = (V, E)$, where $V = \{0, \dots, N\}$ is the set of nodes, including $\{0\}$ as the depot and $\{1, \dots, N\}$ as clients, and vehicle set $K = \{1, \dots, M\}$ is the set of vehicles. Each client $i \in V$ has demand d_i, and each vehicle $k \in K$ has capacity Q_k, speed v_k. Between each client i and vehicle k, there is a profile score $p_{ik} \in \mathbb{R} \cup \{\pm\infty\}$. Intuitively, a higher profile score means that vehicle k is encouraged to serve client i; symmetrically, this could also be understood as client i preferring vehicle

k to serve them. Particularly, if the profile score is $\pm\infty$, it means a hard constraint, i.e., $-\infty$ means that the vehicle k can not serve the client i, while $+\infty$ means that the vehicle k has to serve the client i. The edges E connect pairs of nodes, and each edge between node i and node j, $(i, j) \in E$, has a travel distance c_{ij}. We introduce the decision variables $x_{ij}^k = 1$ if vehicle k travels from i to j (and 0 otherwise) and $y_i^k = 1$ if vehicle k serves client i (and 0 otherwise). The objective is to maximize total profile reward minus travel time, balanced via a *profile weight* $\alpha \in [0, 1]$:

$$\max_{x,y} \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} \left(\alpha\, p_{ik} - \tfrac{c_{ij}}{v_k}\right) x_{ij}^k. \tag{1}$$

under the constraints that each client is served exactly once, vehicle capacities are not exceeded, each route is a continuous tour beginning and ending at the depot, and all decision variables are binary. By setting all $p_{ik} = 0$, we recover the classical VRP[1]. A more detailed problem formulation of PVRP is provided in the Appendix.

MDP Formulation

The PVRP can be naturally framed as a MDP, which enables the application of RL techniques for scalable and adaptive solution generation. We define the formulation as follows:

State Space ($\mathcal{S}$) A state $s^t \in \mathcal{S}$ at time step t captures the partial route constructed up to that point.

Action Space ($\mathcal{A}$) An action $a \in \mathcal{A}$ consists of selecting the client to visit next or returning to the depot.

Transition Dynamics ($\mathcal{T}$) The system evolves according to a deterministic transition function $s_{t+1} = \mathcal{T}(s_t, a_t)$, which updates vehicle locations, remaining capacities, and the set of visited clients based on the selected action.

Reward Function ($\mathcal{R}$) To align with the *bi-objective* nature of PVRP, we define the reward for each action as $r_t = \alpha p_{jk} - c_{ij}/v_k$ as per Eq. (1), where p_{jk} represents the client-vehicle preference score, c_{ij}/v_k accounts for the travel cost, and α controls the trade-off between preference satisfaction and transportation efficiency.

Policy (π) A policy $\pi(a_t|s_t)$ specifies the probability distribution over possible actions given the current state. Our objective is to learn an optimal policy π^* that maximizes the expected cumulative reward.

Policy Parameterization

To generate solutions efficiently, we employ parallel autoregressive models for policy learning with an encoder-decoder framework. The encoder network $f_\theta(\boldsymbol{x}, \alpha)$ processes problem instance $\boldsymbol{x}$ and *profile weight* $\alpha \in [0, 1]$ into a structured representation $\boldsymbol{h}$. At each step t, the decoder network g_θ generates a joint action vector $\boldsymbol{a}_t = (a_t^1, \ldots, a_t^M)$ for all M vehicles. The policy π_θ is formulated as:

$$\pi_\theta(\boldsymbol{a}|\boldsymbol{x}, \alpha) = \prod_{t=1}^{T} \psi \left(\prod_{k=1}^{M} g_\theta(a_t^k | \boldsymbol{a}_{t-1}, \boldsymbol{a}_{t-2}, \ldots, \boldsymbol{a}_1, \boldsymbol{h}) \right)$$

where ψ is a conflict resolution function that ensures solution feasibility by prioritizing assignments to the agent with the largest log-probability value.

Methodology

In this section, we present USPR as illustrated in Fig. 2. We introduce three key components for *unified* profile handling: *profile embeddings* (PE), *multi-head profiled attention* (MHPA), and *profile-aware score reshaping* (PSR). Together, these components enable flexible adaptation to different profile distributions and problem settings within a single model architecture. We then lay out the integration into the encoder-decoder framework and the training scheme.

Profile Embeddings

Previous approaches handle profiles by *retraining* separate models for each profile distribution and profile weight, which is computationally inefficient and lacks flexibility. PE overcome this limitation by learning a unified representation that can encode any combination of attributes and profile weight parameters. Given an instance $\boldsymbol{x}$ and a profile weight parameter α, we first embed these features into a latent space $\mathbf{h}^{(\cdot)}$ of size d_h by considering separated contributions for clients, vehicles, profiles, and profile weight with the dimension of $d_{(\cdot)}$ via parametrized linear layers.

Client Feature Embeddings project each client i client-specific information to the hidden space: $\mathbf{h}_i^c = \mathbf{W}_{\text{init}}^c \boldsymbol{x}_i^c + \mathbf{b}_{\text{init}}^c \in \mathbb{R}^{d_h}$, where $\boldsymbol{x}_i^c \in \mathbb{R}^{d_c}$ contains demands and locations, here $\mathbf{W}_{\text{init}}^c \in \mathbb{R}^{d_h \times d_c}$, $\mathbf{b}_{\text{init}}^c \in \mathbb{R}^{d_h}$.

Vehicle Feature Embeddings project each vehicle k vehicle-specific information to the hidden space: $\mathbf{h}_k^v = \mathbf{W}_{\text{init}}^v \boldsymbol{x}_k^v + \mathbf{b}_{\text{init}}^v \in \mathbb{R}^{d_h}$, where $\boldsymbol{x}_j^v \in \mathbb{R}^{d_v}$ contains capacity, speed, and starting location, which is practically the depot information, here $\mathbf{W}_{\text{init}}^v \in \mathbb{R}^{d_h \times d_v}$, $\mathbf{b}_{\text{init}}^v \in \mathbb{R}^{d_h}$.

Profiles Score Embeddings transform raw profile matrix into learnable embeddings: $\mathbf{h}_{ik}^p = \mathbf{W}_{\text{init}}^p p_{ik} + \mathbf{b}_{\text{init}}^p \in \mathbb{R}^{d_h}$, where $p_{ik} \in \mathbb{R}$ represents the profile score between vehicle i and client k, $\mathbf{W}_{\text{init}}^p \in \mathbb{R}^{d_h \times 1}$, and $\mathbf{b}_{\text{init}}^p \in \mathbb{R}^{d_h}$. For a hard constraint profile score, we have the embeddings by projecting with independent parametrized linear layers: $\mathbf{h}_{ik}^p = \mathbf{W}_{\text{init}}^{p,\pm\infty} \mathbf{1} + \mathbf{b}_{\text{init}}^{p,\pm\infty}$. With this design, our model could flexibly embed any type of mixed combination of soft-preferenced and hard-constrained profile matrix.

Profile Weight Embeddings encode the weight to enable flexible tradeoffs: $\mathbf{h}^\alpha = \mathbf{W}_{\text{init}}^\alpha \alpha + \mathbf{b}_{\text{init}}^\alpha \in \mathbb{R}^{d_h}$, where $\alpha \in \mathbb{R}$ represents the profile weights from Eq. (1) which are then broadcasted on other embeddings, $\mathbf{W}_{\text{init}}^\alpha \in \mathbb{R}^{d_h \times 1}$, and $\mathbf{b}_{\text{init}}^\alpha \in \mathbb{R}^{d_h}$. This simple yet effective design allows the model to dynamically adapt to various objective weights.

[1]Specifically, this case would be a Capacitated Vehicle Routing Problem (CVRP) where the objective is to minimize the total travel time (duration).

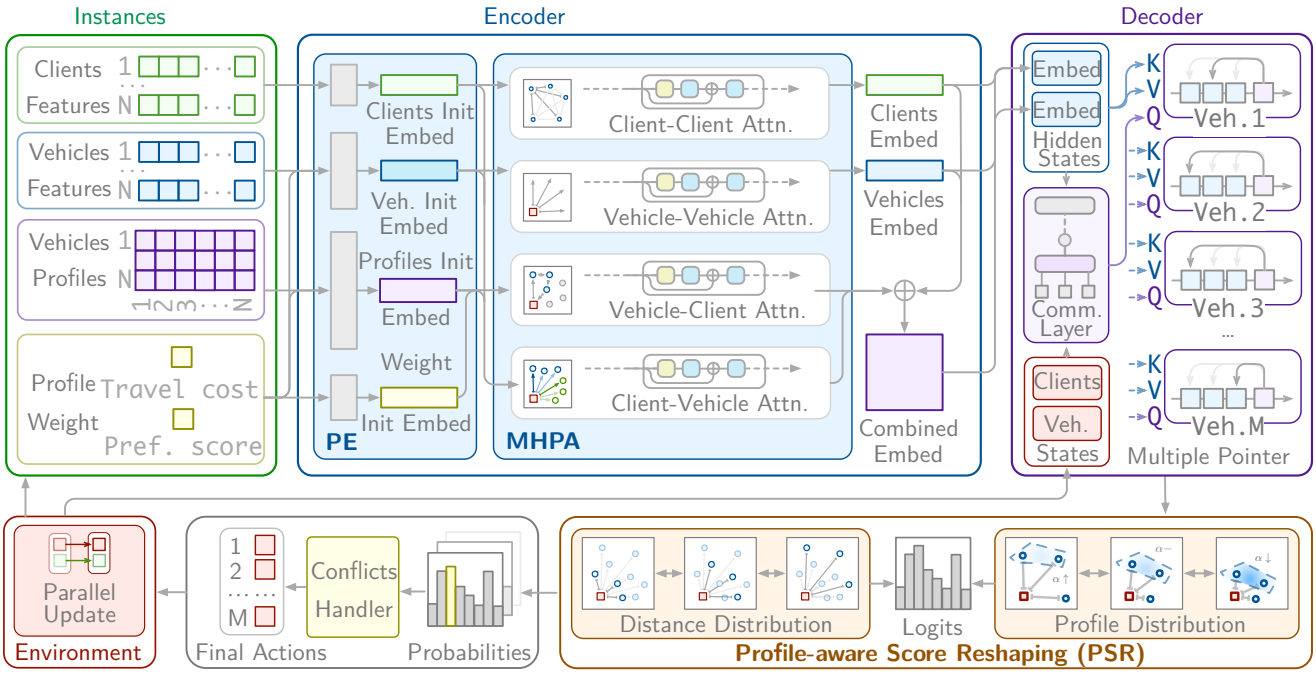

Figure 2: An overview of USPR. Our framework follows an encoder-decoder architecture and introduces three key components to handle diverse problem profiles within a single model. (i) **PE** firstly encode arbitrary instance inputs, including client-vehicle profiles and *profile weights*, into a unified latent representation. (ii) **MHPA** then captures rich, bidirectional interactions between all entities to produce powerful, context-aware embeddings. (iii) **PSR** adaptively integrates profile and distance distribution and scale information to adjust the policy output logits, ensuring the robust generalization ability.

PE create a shared latent space that enables the model to handle arbitrary profile types and profile weights without retraining. By projecting the client, vehicle, profiles, and profile weights, the model can simultaneously process profiles with varying magnitudes and values.

Multi-Head Profiled Attention

A fundamental limitation of existing approaches is their inability to capture rich bidirectional interactions between vehicles and clients with different profiles. To address this, we introduce MHPA to allow for improved information exchange in each encoder layer:

$$\mathbf{h} = \text{Norm}(\text{MHPA}(\mathbf{h})) \tag{2}$$

$$\mathbf{h} = \text{Norm}(\text{FFN}(\mathbf{h}) + \mathbf{h}) \tag{3}$$

where $\text{FFN}(\cdot)$ denotes a multi-layer perceptron. MHPA is based on the multi-head attention (MHA):

$$\text{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \left(\Big\Vert_{i=1}^{n_\text{h}} \text{Attn}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V) \right) \mathbf{W}^O$$

$$\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_h}} \right) \mathbf{V}$$

MHPA improves on MHA for modeling PVRP with four distinct types of information exchange between clients and vehicles bi-directionally, depending on the init embedding:

Client-Client (CC) Attention enables information sharing between clients about their spatial location, distance relationships, and demands: $\mathbf{h}^{c'} = \text{MHA}(\mathbf{h}^c, \mathbf{h}^c, \mathbf{h}^c)$.

Vehicle-Vehicle (VV) Attention facilitates communication between vehicles about their current locations, speed, distance relationships, maximum and available capacities, and service capabilities: $\mathbf{h}^{v'} = \text{MHA}(\mathbf{h}^v, \mathbf{h}^v, \mathbf{h}^v)$.

Vehicle-Client (VC) Attention enables vehicles to attend to relevant clients based on *profiles* about soft preferences or hard constraints: $\mathbf{h}_i^{pc'} = \text{MHA}(\mathbf{h}^v, \mathbf{h}^c, \mathbf{h}_i^{pv})$.

Client-Vehicle (CV) Attention allows clients to consider their suitability for different vehicles based on their *profiles* in a reversed way compared with the VC attention to enrich the relationship embeddings: $\mathbf{h}_k^{pv'} = \text{MHA}(\mathbf{h}^c, \mathbf{h}^v, \mathbf{h}_k^{pc})$.

Unlike previous approaches that only consider unidirectional interactions or simple aggregations, MHPA enables comprehensive information exchange through bidirectional attention mechanisms. The final MHPA output integrates all processed information:

$$\mathbf{h}_{ik}^{p'} = \text{concat}(\mathbf{h}_k^v, \mathbf{h}_i^c, \mathbf{h}_k^{pv'}, \mathbf{h}_i^{pc'}) + \mathbf{h}_{ik}^p \tag{4}$$

where each vehicle k is assigned its own hidden embeddings $\mathbf{h}$ processed latent representation.

Profile-Aware Score Reshaping

As we are using one unified model for various distributions of profiles and scales, to maintain robustness and out-of-distribution performance, we further introduce PAR, which dynamically adjusts decoder logits based on the distance and profile distributions. Building on recent advancements in distance-based heuristics (Wang et al. 2024; Huang et al. 2025), PSR further combines learned embeddings with explicit profile and distance information:

$$\mathbf{Z} = C \cdot \tanh\left(\frac{\mathbf{U}(\mathbf{W}_{\text{ptr}}\mathbf{h})^\top}{\sqrt{d_h}} - \log(\text{dist}_{ij} + p_{ij})\right) \quad (5)$$

where C is a scaling factor set to 10 following Bello et al. (2016), $\mathbf{U} \in \mathbb{R}^{M \times d_h}$ represents the decoder's query vectors, $\mathbf{W}_{\text{ptr}}$ projects node embeddings into the pointer space, dist_{ij} is the distance between nodes i and j, and p_{ij} is the preference score between vehicle i and client j[2].

PSR provides several key advantages: it balances the flexibility of neural approaches with the reliability of traditional heuristics; naturally penalizes nodes that are either spatially distant or have low preference scores; ensures consistent performance across varying profile distributions and out-of-distribution instances; and enables smooth adaptation without retraining when preference weights change. The final action probabilities are computed by masking infeasible actions and applying softmax:

$$P(\mathbf{a}_t|\mathbf{s}_t) = \text{softmax}(\mathbf{Z} + \mathbf{M}_t) \quad (6)$$

where $\mathbf{M}_t$ is the mask tensor for infeasible actions at step t.

Integration into Encoder-Decoder Framework

The three components described above are integrated into an encoder-decoder framework. The encoder first processes raw problem features using PE to create initial embeddings, then applies encoder layers with MHPA to capture complex interactions between clients and vehicles. The encoder outputs a set of embeddings $\mathbf{h} = [\mathbf{h}^1, \ldots, \mathbf{h}^M]$, where each $\mathbf{h}^k \in \mathbb{R}^{(M+N) \times d_h}$ represents the encoded graph information for vehicle k. The decoder generates vehicle-specific queries that capture both profile and current state info:

$$\mathbf{q}_k^t = \mathbf{W}_{\text{query}}[\mathbf{h}^k \| \mathbf{h}_{\text{cur}}^k \| \mathbf{W}_{\text{state}}\mathbf{s}_k^t] \quad (7)$$

where $\cdot\|\cdot$ denotes concatenation, $\mathbf{h}^k$ is the vehicle's profile embedding, $\mathbf{h}_{\text{cur}}^k$ represents the current node embedding, and $\mathbf{s}_k^t$ captures the state features at time t. We then process the multiple vehicle queries $\mathbf{q}_t = [\mathbf{q}_t^1, \ldots, \mathbf{q}_t^M]$ via a communication layer. The communication layer follows Berto et al. (2024) and is composed of standard MHA and FFN: this processes embeddings into a multiple pointer mechanism that generates the decoder output logits. Finally, PSR is applied to reshape the logits and compute action probabilities for each vehicle as the action.

[2]We note that while we manually design the attention reshaping mechanism in this paper, we could leverage automatic algorithm design (Liu et al. 2024b; Ye et al. 2024; Pham, Doan, and Huynh 2025; Tran et al. 2025; Zhao et al. 2025) to do it, which we leave as future work.

Training

We train our model using the REINFORCE algorithm with a shared baseline across all agents (Kim, Park, and Park 2022a). During training, we sample weights α_i from $[0, 1]$ and p_{ik} from $[p_{\min}, p_{\max}]$ with a predefined probability to be $\pm\infty$ for each instance in the batch. This allows the model to learn a unified policy across different preference-cost trade-offs and profile matrix distributions. The policy gradient is estimated as:

$$\nabla_\theta \mathcal{L} = \frac{1}{B \cdot L} \sum_{i=1}^{B} \sum_{j=1}^{L} (R(\boldsymbol{x}_i, \boldsymbol{a}_{ij}, \alpha_i) - b^{\text{shared}}(\boldsymbol{x}_i)) \cdot \nabla_\theta \log p_\theta(\boldsymbol{a}_{ij}|\boldsymbol{x}_i, \alpha_i) \quad (8)$$

where B is the batch size, L is the number of solutions per instance, and b^{shared} is the shared baseline value obtained through symmetric augmentation sampling.

Experiments

We evaluate the effectiveness of our proposed approach with comprehensive experiments in various settings, including in-distribution performance analysis, large-scale (out-of-distribution) generalization analysis, real-world application, model components ablation study, and further analyses. We compare against both classical and learning-based baselines.

Experimental Setup

Data Generation For basic features of VRP, including the clients and depot coordinates, demands, capacity, and speed, we follow the widely used settings from (Kool, Van Hoof, and Welling 2018). Profile scores are drawn as $\sim \text{Uniform}(0, 1)$ independently for all client–vehicle pairs. Without losing the generalization, we sample two probabilities $\sim \text{Uniform}(0, 0.1)$ for each instance to randomly set part of the profile matrix to $\pm\infty$ as the hard constraints.

Classical Baselines We employ two state-of-the-art classical solvers: Google OR-Tools (Perron and Furnon 2023), a versatile framework that combines exact and heuristic methods via constraint programming, and HGS-PyVRP (Wouda, Lan, and Kool 2024), an open-source implementation of the Hybrid Genetic Search for the CVRP (Vidal 2022) that supports the PVRP. We handle vehicle-specific profiles by modifying the cost matrices for each vehicle according to the objective function in Eq. (1) and masking for hard constraints.

Neural Baselines We compare against several recent neural VRP solvers: ET (Son et al. 2024), which specializes in sequential multi-agent routing with equitable workload distribution; DPN (Zheng et al. 2024), which enhances ET with an improved encoder for route partitioning; 2D-Ptr (Liu et al. 2024c), which uses dual encoding for dynamic adaptation in heterogeneous routing; PARCO (Berto et al. 2024), which employs parallel decoding with inter-agent communication; and CAMP (Hua et al. 2025), which was specifically designed for PVRP. We follow CAMP to adapt ET, DPN, 2D-Ptr, and PARCO to PVRP for a fair comparison.

N	60				80				100				Gap(%)
M	3	5	7	Time	3	5	7	Time	3	5	7	Time	avg.
OR-Tools	7.98	8.25	8.45	10m	9.33	9.86	10.02	12m	11.25	11.41	11.67	15m	10.24
HGS-PyVRP	7.07	7.42	7.66	10m	8.51	8.97	9.23	12m	9.99	10.51	10.78	15m	0.00
ET (*g.*)	8.31	8.77	9.02	0.17s	9.98	10.50	10.79	0.23s	11.69	12.28	12.70	0.29s	17.38
DPN (*g.*)	8.23	8.65	8.88	0.18s	9.87	10.51	10.88	0.23s	11.68	12.21	12.51	0.29s	16.59
2D-Ptr (*g.*)	8.01	8.38	8.62	0.15s	9.61	10.14	10.41	0.20s	11.25	11.89	12.21	0.25s	12.97
PARCO (*g.*)	7.98	8.36	8.66	0.15s	9.59	10.04	10.37	0.22s	11.26	11.85	12.12	0.25s	12.62
CAMP (*g.*)	7.79	8.26	8.46	0.18s	9.38	9.87	10.22	0.25s	10.98	11.65	11.91	0.33s	10.50
USPR (*g.*)	**7.73**	**8.11**	**8.38**	0.11s	**9.30**	**9.78**	**10.08**	0.14s	**10.90**	**11.44**	**11.76**	0.20s	**9.20**
ET (*s.*)	7.82	8.23	8.45	0.25s	9.40	9.89	10.20	0.36s	11.06	11.64	11.90	0.45s	10.59
DPN (*s.*)	7.79	8.18	8.46	0.26s	9.36	9.88	10.18	0.36s	10.99	11.58	11.92	0.44s	10.25
2D-Ptr (*s.*)	7.55	7.93	8.17	0.17s	9.12	9.55	9.87	0.21s	10.69	11.19	11.49	0.26s	6.79
PARCO (*s.*)	7.53	7.86	8.13	0.22s	9.03	9.52	9.82	0.34s	10.56	11.18	11.51	0.41s	6.25
CAMP (*s.*)	7.39	7.77	8.01	0.34s	8.90	9.37	9.67	0.42s	10.44	10.98	11.27	0.53s	4.58
USPR (*s.*)	**7.35**	**7.71**	**7.96**	0.22s	**8.85**	**9.32**	**9.60**	0.33s	**10.39**	**10.93**	**11.22**	0.42s	**4.02**

Table 1: Benchmarks and results for PVRP at varying sizes and agent numbers. Highlighting cost (↓) and average (avg.) gaps (↓) to the HGS-PyVRP solver. The average inference time for a single instance of each size is shown in the Time columns.

Training Configuration We optimize using Adam (Kingma and Ba 2014) with an initial learning rate of 10^{-4}, decaying by a factor of 0.1 at epochs 80 and 95. Training runs for 100 epochs with 10^5 samples per epoch, using a batch size of 32 and 8 augmented rollouts via Sym-NCO (Kim, Park, and Park 2022b) per instance for baseline estimation. Architecturally, each model employs $d_h = 128$ hidden-dimensional embeddings, 8 attention heads, and 512-dimensional feedforward layers across 3 encoder layers. We set the scale of the number of agents and vehicles during training from 60 to 100, from 3 to 7, respectively. For baseline models, following the setting from CAMP, we train separate models for each fixed α taken from $\alpha \in \{0.00, 0.025, \ldots, 0.20\}$, where the final reported performance is the average of results across all α values. As a unified model, USPR is trained with the α is randomly sampled from 0 to 0.2 for each instance, also including a sampling probability for the mixed hard-constraints from 0 to 10%. All training and testing are run on an AMD Ryzen Threadripper 3960X (24-core) CPU with an NVIDIA RTX 4090 GPU. For more detailed training hyperparameters and device information, please refer to the Appendix.

Testing Protocol We consider three main settings for evaluating our approach. Firstly, we consider in-distribution results, where we evaluate each model on 1,280 randomly generated instances for each size, following the same generating rule as training. Secondly, we test 128 instances in out-of-distribution generalization, with vehicle numbers $M \in \{15, 25, 35\}$ for $N = 500$. We finally introduce a variation of the real-world data of CVRPLib[3], which we coin *PVRPLib*, which is based on the number of vehicles, capacity values, and coordinates of the original CVRPLib but with profiles generated as described in the data generation paragraph. For each instance, we measure both greedy performance ($g.$), i.e., taking the $\arg\max$ over decoder log-probabilities, and sampling 1,280 solutions per instance ($s.$). Final performance metrics are reported as averages over all profile distributions and α settings.

[3]http://vrp.atd-lab.inf.puc-rio.br/index.php/en/

Main Results and Analysis

We address the three limitations of prior works in the following paragraphs: (i) the lack of a *unified model*, (ii) *context-agnostic representations*, and (iii) *poor generalization*.

Unified Model A primary advantage of our unified model is its exceptional efficiency. As demonstrated in Table 2, our approach significantly reduces total training time compared to training multiple single-task models like CAMP. By using a single, shared architecture, we substantially lower both memory and computational costs, cutting the total number of required parameters. This streamlined process not only accelerates the training cycle but also maintains competitive performance. Consequently, our method eliminates the need to develop and manage separate models for various problem types, offering a scalable and adaptable solution for PVRP that ensures high-quality optimization across diverse vehicle-client profiles.

	# Models	# Total Param.	# Total Epochs	Train Time
CAMP	10	17.6M	1000	4.6 days
USPR	**1**	**1.5M**	**100**	**11 hours**

Table 2: Our unified model enables substantial memory and training time savings compared to single-task CAMP.

In-distribution Performance Table 1 shows a comparison between our method and the baseline models on the in-distribution scale. The results demonstrate that our USPR achieves state-of-the-art performance for all problem settings, consistently outperforming all existing neural solvers in both solution quality and computational efficiency. Note

that, unlike previous approaches that require training separate models for different preference weights, USPR is trained as a *single* model, effectively handling varying client-vehicle constraints and preference distributions within a unified framework.

M	30	50	70	Time	Gap(%)
OR-Tools	102.86	112.65	115.96	15m	58.98
HGS-PyVRP	64.70	70.86	72.94	15m	0.00
ET (*g.*)	123.73	135.51	139.48	2.34s	91.23 (+71.88)
DPN (*g.*)	109.58	120.01	123.53	2.34s	69.36 (+50.01)
2D-Ptr (*g.*)	91.49	100.20	103.14	1.07s	41.41 (+22.06)
PARCO (*g.*)	81.24	88.97	91.58	1.24s	25.56 (+ 6.21)
CAMP (*g.*)	80.05	87.67	90.24	1.33s	23.72 (+ 4.37)
USPR (*g.*)	**77.22**	**84.57**	**87.05**	**1.07s**	**19.35**
ET (*s.*)	117.90	129.13	132.92	2.59s	82.23 (+68.08)
DPN (*s.*)	100.69	110.28	113.52	2.58s	55.63 (+41.48)
2D-Ptr (*s.*)	85.27	93.39	96.13	1.18s	31.80 (+17.65)
PARCO (*s.*)	80.82	88.52	91.12	1.37s	24.92 (+10.77)
CAMP (*s.*)	79.87	87.47	90.04	1.43s	23.44 (+ 9.29)
USPR (*s.*)	**73.86**	**80.89**	**83.26**	**1.18s**	**14.15**

Table 3: Benchmarks and results for large-scale PVRP instances ($N = 1000$). We report the solution cost (↓) and average gap (↓) to the HGS-PyVRP solver. Average inference time is shown in the Time column.

Out-of-Distribution Performance Table 3 shows the out-of-distribution performance, particularly in large scales up to $10\times$ the number of agents M and $10\times$ the number of nodes N.

	Size	BKS	CAMP		USPR	
			Cost	Gap	Cost	Gap (%)
Set A	31-79	9.24	9.87	6.82(+1.63)	9.72	**5.19**
Set B	30-77	10.18	10.80	6.09(+0.88)	10.71	**5.21**
Set F	44-134	12.68	13.58	7.10(+0.48)	13.52	**6.62**
Set M	100-199	45.63	54.39	19.20(+5.74)	51.77	**13.46**
Set P	15-100	9.09	9.67	6.38(+1.32)	9.55	**5.06**
Set X	100-300	15.56	17.43	12.01(+3.29)	16.92	**8.72**
	300-500	38.08	44.79	17.62(+5.96)	42.52	**11.66**
	500-700	66.04	78.42	18.74(+5.32)	74.90	**13.42**
	700-1K	99.08	124.61	25.77(+10.56)	114.15	**15.21**

Table 4: Results of CAMP and USPR about cost (↓) and average gap (↓) across PVRPLib instances. The best-known solutions (BKS) are collected by the HGS-PyVRP solver.

Real-world Settings We further analyze the performance of USPR against the SOTA neural method CAMP on the newly proposed PVRPLib, containing real-world location distributions, demands, number of vehicles, and the added preferences in Table 4. Our method remarkably improves on CAMP by more than 10% in large-scale instances.

Ablation Study We perform an ablation study to evaluate the contribution of model components in Table 5. We remove the PSR, SR (distance only), and MHPA (same as baselines). Removing each component will cause a drop in performance, showing the importance of each design. The most significant drop occurs when eliminating the MHPA, highlighting its critical role in improving inter-agent communication and capturing profile-specific interactions.

Model	USPR	-PSR	-PSR & SR	-PSR, SR & MHPA
Avg. Gap (%)	**4.42**	4.72	5.12	6.72

Table 5: Ablation study results on the size of $N = 100$.

Qualitative Analysis Figure 3 visualizes example solutions for a PVRP instance. With increasing the *profile weight* α, our model shifts focus towards profile adherence while maintaining strong duration optimization, effectively capturing vehicle-client interactions. A single unified model is used to solve for any value of α. It demonstrates how increasing its value makes the model trade off duration for overall preferences. This makes our model highly scalable and adaptable for real-world applications.

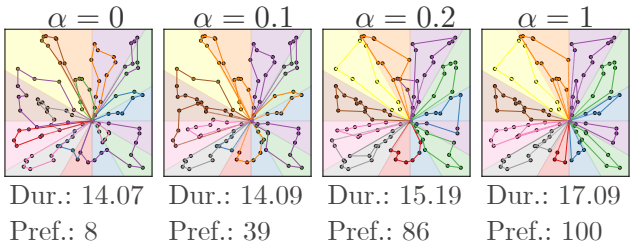

Figure 3: USPR's PVRP solutions to the same instance with different α processed through the profile embeddings. A higher α favors adherence to profile values, while $\alpha = 0$ converts the problem to a classical VRP.

Conclusion

In this work, we introduced USPR, a learning-based framework that addresses the key limitations of existing PVRP neural solvers through three novel components: PE for encoding arbitrary profile distributions, MHPA for modeling rich vehicle-client interactions, and PSR for robust generalization. Our comprehensive experiments demonstrate that USPR consistently outperforms state-of-the-art neural methods across both preference-based and zone-constrained routing problems, while matching or exceeding classical solvers at significantly lower computational cost. Notably, a single USPR model effectively handles varying profile weights and generalizes to out-of-distribution instances up to $10\times$ larger than training data. By providing this unified approach to profiled routing optimization and making our implementation publicly available, we aim to advance NCO research and enable more flexible, efficient solutions for complex routing problems in real-world logistics operations. A limitation to address in future works is reducing the memory usage of models.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. RS-2024-00410082), by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (RS-2025-00563763), and by the InnoCORE program of the Ministry of Science and ICT(N10250154).

References

Aiko, S.; Thaithatukl, P.; and Asakura, Y. 2018. Incorporating user preference into optimal vehicle routing problem of integrated sharing transport system. *Asian Transport Studies*, 5(1): 98–116.

Bdeir, A.; Falkner, J. K.; and Schmidt-Thieme, L. 2022. Attention, filling in the gaps for generalization in routing problems. In *ECML PKDD*, 505–520. Springer.

Bello, I.; Pham, H.; Le, Q. V.; Norouzi, M.; and Bengio, S. 2016. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*.

Bengio, Y.; Lodi, A.; and Prouvost, A. 2021. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2): 405–421.

Berto, F.; Hua, C.; Luttmann, L.; Son, J.; Park, J.; Ahn, K.; Kwon, C.; Xie, L.; and Park, J. 2024. PARCO: Learning Parallel Autoregressive Policies for Efficient Multi-Agent Combinatorial Optimization. *arXiv preprint arXiv:2409.03811*.

Bi, J.; Ma, Y.; Zhou, J.; Song, W.; Cao, Z.; Wu, Y.; and Zhang, J. 2024. Learning to Handle Complex Constraints for Vehicle Routing Problems. *arXiv preprint arXiv:2410.21066*.

Braekers, K.; Ramaekers, K.; and Van Nieuwenhuyse, I. 2016. The vehicle routing problem: State of the art classification and review. *Computers & industrial engineering*, 99: 300–313.

Choo, J.; Kwon, Y.-D.; Kim, J.; Jae, J.; Hottung, A.; Tierney, K.; and Gwon, Y. 2022. Simulation-guided beam search for neural combinatorial optimization. *NeurIPS*, 35: 8760–8772.

Cordeau, J.-F.; and Laporte, G. 2001. A tabu search heuristic for the site dependent vehicle routing problem with time windows. *INFOR*, 39(4): 292–298.

Drakulic, D.; Michel, S.; and Andreoli, J.-M. 2025. GOAL: A Generalist Combinatorial Optimization Agent Learning. In *ICLR*.

Duan, L.; Zhan, Y.; Hu, H.; Gong, Y.; Wei, J.; Zhang, X.; and Xu, Y. 2020. Efficiently Solving the Practical Vehicle Routing Problem: A Novel Joint Learning Approach. In *KDD*. ACM.

Golden, B.; Assad, A.; Levy, L.; and Gheysens, F. 1984. The fleet size and mix vehicle routing problem. *Computers & Operations Research*, 11(1): 49–66.

Grinsztajn, N.; Furelos-Blanco, D.; Surana, S.; Bonnet, C.; and Barrett, T. 2023. Winner takes it all: Training performant RL populations for combinatorial optimization. *NeurIPS*, 36: 48485–48509.

Grinsztajn, N.; Furelos-Blanco, D.; Surana, S.; Bonnet, C.; and Barrett, T. 2024. Winner Takes It All: Training Performant RL Populations for Combinatorial Optimization. *NeurIPS*, 36.

Hottung, A.; Kwon, Y.-D.; and Tierney, K. 2022. Efficient active search for combinatorial optimization problems. In *ICLR*.

Hottung, A.; and Tierney, K. 2020. Neural large neighborhood search for the capacitated vehicle routing problem. In *ECAI 2020*. IOS Press.

Hua, C.; Berto, F.; Son, J.; Kang, S.; Kwon, C.; and Park, J. 2025. CAMP: Collaborative Attention Model with Profiles for Vehicle Routing Problems. In *AAMAS*.

Huang, Z.; Zhou, J.; Cao, Z.; and Xu, Y. 2025. Rethinking Light Decoder-based Solvers for Vehicle Routing Problems. *arXiv preprint arXiv:2503.00753*.

Johnson, D. S.; and McGeoch, L. A. 1997. The traveling salesman problem: a case study. *Local search in combinatorial optimization*.

Kim, M.; Park, J.; and Park, J. 2022a. Neuro CROSS exchange: Learning to CROSS exchange to solve realistic vehicle routing problems. *arXiv preprint arXiv:2206.02771*.

Kim, M.; Park, J.; and Park, J. 2022b. Sym-nco: Leveraging symmetricity for neural combinatorial optimization. *NeurIPS*, 35: 1936–1949.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kool, W.; van Hoof, H.; Gromicho, J.; and Welling, M. 2022. Deep policy dynamic programming for vehicle routing problems. In *CPAIOR*, 190–213. Springer.

Kool, W.; Van Hoof, H.; and Welling, M. 2018. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*.

Kwon, Y.-D.; Choo, J.; Kim, B.; Yoon, I.; Gwon, Y.; and Min, S. 2020. Pomo: Policy optimization with multiple optima for reinforcement learning. *NeurIPS*, 33: 21188–21198.

Li, J.; Ma, Y.; Gao, R.; Cao, Z.; Lim, A.; Song, W.; and Zhang, J. 2022. Deep reinforcement learning for solving the heterogeneous capacitated vehicle routing problem. *IEEE Transactions on Cybernetics*.

Li, S.; Yan, Z.; and Wu, C. 2021. Learning to delegate for large-scale vehicle routing. *NeurIPS*, 34: 26198–26211.

Li, Y.; Zhou, C.; Yuan, P.; and Ngo, T. T. A. 2023. Experience-based territory planning and driver assignment with predicted demand and driver present condition. *Transportation research part E: logistics and transportation review*, 171: 103036.

Liu, F.; Lin, X.; Wang, Z.; Zhang, Q.; Xialiang, T.; and Yuan, M. 2024a. Multi-task learning for routing problem with cross-problem zero-shot generalization. In *KDD*.

Liu, F.; Xialiang, T.; Yuan, M.; Lin, X.; Luo, F.; Wang, Z.; Lu, Z.; and Zhang, Q. 2024b. Evolution of Heuristics: Towards Efficient Automatic Algorithm Design Using Large Language Model. In *ICML*.

Liu, Q.; Liu, C.; Niu, S.; Long, C.; Zhang, J.; and Xu, M. 2024c. 2D-Ptr: 2D Array Pointer Network for Solving the Heterogeneous Capacitated Vehicle Routing Problem. In *AAMAS*, 1238–1246.

Lozano, M.; Molina, D.; and Herrera, F. 2011. Editorial scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems. *Soft computing*, 15: 2085–2087.

Ma, Y.; Cao, Z.; and Chee, Y. M. 2024. Learning to search feasible and infeasible regions of routing problems with flexible neural k-opt. *NeurIPS*, 36.

Mazyavkina, N.; Sviridov, S.; Ivanov, S.; and Burnaev, E. 2021. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134: 105400.

Papadimitriou, C. H.; and Steiglitz, K. 1998. *Combinatorial optimization: algorithms and complexity*. Courier Corporation.

Perron, L.; and Furnon, V. 2023. OR-Tools. Google.

Pham, V. T. D.; Doan, L.; and Huynh, T. T. B. 2025. HSEvo: Elevating Automatic Heuristic Design with Diversity-Driven Harmony Search and Genetic Algorithm Using LLMs. In *AAAI*. Association for the Advancement of Artificial Intelligence (AAAI).

Pirnay, J.; and Grimm, D. G. 2024. Take a step and reconsider: Sequence decoding for self-improved neural combinatorial optimization. In *ECAI*.

Son, J.; Kim, M.; Choi, S.; Kim, H.; and Park, J. 2024. Equity-Transformer: Solving NP-Hard Min-Max Routing Problems as Sequential Generation with Equity Context. In *AAAI*.

Sun, Z.; Li, Z.; Wang, H.; He, D.; Lin, Z.; and Deng, Z. 2019. Fast structured decoding for sequence models. *NeurIPS*, 32.

Sun, Z.; and Yang, Y. 2024. Difusco: Graph-based diffusion solvers for combinatorial optimization. *NeurIPS*, 36.

Team Locus. 2020. Zone-Based Routing is the Need of the Hour. *Locus Blog*. Access: 2024-10-17.

Tran, C. D.; Nguyen-Tri, Q.; Binh, H. T. T.; and Thanh-Tung, H. 2025. Large Language Models powered Neural Solvers for Generalized Vehicle Routing Problems. In *Towards Agentic AI for Science: Hypothesis Generation, Comprehension, Quantification, and Validation*.

Vidal, T. 2022. Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood. *Computers & Operations Research*, 140: 105643.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer networks. *NeurIPS*, 28.

Wang, Y.; Jia, Y.-H.; Chen, W.-N.; and Mei, Y. 2024. Distance-aware Attention Reshaping: Enhance Generalization of Neural Solver for Large-scale Vehicle Routing Problems. *arXiv preprint arXiv:2401.06979*.

Wouda, N. A.; Lan, L.; and Kool, W. 2024. PyVRP: A high-performance VRP solver package. *INFORMS Journal on Computing*.

Ye, H.; Wang, J.; Cao, Z.; Berto, F.; Hua, C.; Kim, H.; Park, J.; and Song, G. 2024. ReEvo: Large Language Models as Hyper-Heuristics with Reflective Evolution. In *NeurIPS*.

Zhao, Z.; Hua, C.; Berto, F.; Lee, K.; Ma, Z.; Li, J.; and Park, J. 2025. TrajEvo: Designing Trajectory Prediction Heuristics via LLM-driven Evolution. *arXiv preprint arXiv:2505.04480*.

Zheng, Z.; Yao, S.; Wang, Z.; Xialiang, T.; Yuan, M.; and Tang, K. 2024. DPN: Decoupling Partition and Navigation for Neural Solvers of Min-max Vehicle Routing Problems. In *ICML*.

Zhong, H.; Hall, R. W.; and Dessouky, M. 2007. Territory planning and vehicle dispatching with driver learning. *Transportation Science*.

Zhou, J.; Cao, Z.; Wu, Y.; Song, W.; Ma, Y.; Zhang, J.; and Xu, C. 2024. MVMoE: Multi-Task Vehicle Routing Solver with Mixture-of-Experts. In *ICML*.

Zong, Z.; Zheng, M.; Li, Y.; and Jin, D. 2022. Mapdp: Cooperative multi-agent reinforcement learning to solve pickup and delivery problems. In *AAAI*.