A Unified Knowledge Embedded Reinforcement Learning-based Framework for Generalized Capacitated Vehicle Routing Problems

Wen Wang[1] , **Xiangchen Wu**[1] , **Liang Wang**[1*] , **Hao Hu**[1] and **Xianping Tao**[1]
[1] Nanjing University
{wangwen, ixc}@smail.nju.edu.cn, {wl, myou, txp}@nju.edu.cn

Abstract

The Capacitated Vehicle Routing Problem (CVRP) is a fundamental NP-hard problem with broad applications in logistics and transportation. Real-world CVRPs often involve diverse objectives and complex constraints, such as time windows or backhaul requirements, motivating the development of a unified solution framework. Recent reinforcement learning (RL) approaches have shown promise in combinatorial optimization, yet they rely on end-to-end learning and lack explicit problem-solving knowledge, limiting solution quality. In this paper, we propose a knowledge-embedded framework inspired by the Route-First Cluster-Second heuristics. It incorporates knowledge at two levels: (1) decomposing CVRPs into the route-first and cluster-second subproblems, and (2) leveraging dynamic programming to solve the second subproblem, whose results guide the RL-based constructive solver to solve the first problem. To mitigate partial observability caused by problem decomposition, we introduce a unified history-enhanced context processing module. Extensive experiments show that this framework achieves superior solution quality compared with state-of-the-art learning-based methods, with a smaller gap to classical heuristics, demonstrating strong generalization across diverse CVRP variants.

1 Introduction

The Capacitated Vehicle Routing Problem (CVRP) is a fundamental research topic in combinatorial optimization and serves as a canonical model for numerous real-world applications, including urban logistics [Jeong and Song, 2025], last-mile delivery [Tiwari and Sharma, 2023], postal distribution [Sbai *et al.*, 2022], and transportation planning [Leng and Li, 2021], where goods must be delivered from a central depot to a set of geographically distributed customers. In practical applications, CVRP instances often incorporate additional constraints to reflect real-world operational requirements. For example, time window constraints [Vidal *et al.*, 2014] enforce that customers must be served within specified intervals, while open route constraints [Li *et al.*, 2007] allow vehicles to finish their service without returning to the depot. Given the NP-hard nature [Lenstra and Kan, 1981] and the increased complexity introduced by these variants, developing a unified and efficient approximate solver has become an important and long-standing research objective.

With the development of learning-based methods in the field of combinatorial optimization, a number of foundation models for VRPs have recently been proposed [Berto *et al.*, 2024; Drakulic *et al.*, 2025; Zhou *et al.*, 2024]. These methods introduce advanced architectural designs inspired by Large Language Models (LLMs), incorporating components such as multi-head mixed-attention blocks, transformer-based encoders, and mixture-of-experts (MoE) structures to construct a unified network architecture capable of addressing diverse VRP variants. From a learning perspective, these models typically adopt POMO-style [Kwon *et al.*, 2020] reinforcement learning or imitation learning paradigms to train neural networks that directly generate high-quality routing solutions in an end-to-end manner. In this solving framework, the incorporation of domain knowledge is manifested in two ways: (1) the design of network architectures guided by problem-specific insights; and (2) solution samples generated by classical solvers. Although promising results have been achieved, relying solely on model- or data-driven domain knowledge overlooks the importance of algorithmic design in solving classical problems, thereby limiting further improvements in solution quality.

To directly leverage domain knowledge in solving CVRPs, we exploit the decomposable nature of the problem. Problem decomposition has a long-standing history in the study of CVRPs, particularly within the field of evolutionary computation [Zhang *et al.*, 2022]. Among these approaches, the Route-first Cluster-second (RFCS) heuristic [Prins *et al.*, 2014] represents one of the most influential strategies, and numerous variants have been developed following this principle. Specifically, the RFCS heuristic first constructs a global route covering all customer nodes (excluding the depot) using heuristic methods such as TSP solvers. It then applies a polynomial-time optimal algorithm to divide the global route into feasible vehicle tours, thereby producing a valid solution to original CVRP variants. Manually designed algorithms can naturally incorporate various operational con-

*Corresponding author: wl@nju.edu.cn

straints. More importantly, once the global route is generated, the subsequent process typically runs in polynomial time, ensuring high computational efficiency. For example, [Vidal, 2015] give a $O(n)$ algorithm given a global route for minsum CVRP, and [Wang *et al.*, 2025] gives a binary search-based splitting algorithm for minmax VRPs. Nevertheless, such methods generally rely on TSP-based heuristics to construct the global route, and the optimal TSP tour does not necessarily yield an optimal solution to the original CVRP variants after a cluster-second operation. In this paper, we introduce reinforcement learning to automatically learn the route first part, avoiding the limitations of handcrafted TSP-based heuristics.

However, the problem decomposition mechanism introduces a partial observability challenge: the context of a partial route cannot be fully assessed until the entire route is generated and the subsequent splitting stage is completed. To address this issue, we introduce a history-enhanced module as the core of our context processor. This module use the LSTM [Hochreiter and Schmidhuber, 1997] architecture to maintain a temporal hidden state that continuously aggregates historical information throughout the route construction process, allowing the model to infer the latent information of the problem even when the final evaluation signal is postponed. Beyond mitigating partial observability, the history-enhanced context representation also provides a unified neural architecture across different VRP variants. By encoding the evolving decision context into a learned hidden state, the framework eliminates the need for manually designed problem-specific contexts. As a result, the same network architecture can generalize seamlessly across multiple problem types—such as open-route, backhaul-request, and time-window CVRPs—without additional adaptation.

The main contributions can be summarized as follows.

- We propose a unified **knowledge-embedded RFCS framework** that integrates RL with the route-first cluster-second (RFCS) heuristic to address a wide range of CVRP variants. This framework combines general learning-based methods and algorithmic design principles, enabling flexible adaptation to various types of problem-specific constraints.
- We introduce a **history-enhanced context processing** module that preserves a unified neural network architecture across general CVRP variants. This module simplifies the design of context features by exploiting a common representation provided by the route-first component, enabling effective handling of complex constraints.
- We perform extensive experiments to evaluate the solution quality and transfer capabilities of the framework across diverse CVRP settings. The results demonstrate that the proposed knowledge-embedded RFCS framework consistently outperforms existing learning-based methods and exhibits strong zero-shot generalization to unseen problem variants.

2 Related Work

Since the introduction of neural networks into combinatorial optimization by the Pointer Network [Vinyals *et al.*, 2015], a series of learning-based combinatorial optimization solvers have subsequently emerged, collectively referred to as Neural Combinatorial Optimization (NCO). From the perspective of solution paradigms, learning-based approaches can be broadly classified into three categories [Wu *et al.*, 2024]: (1) Learning to Construct (L2C), which focuses on directly generating solutions; (2) Learning to Improve (L2I), which aims to refine or enhance existing solutions; and (3) Learning to Predict (L2P), which leverages neural networks to support and augment traditional operations research methods. In the L2C paradigm, the Attention Model (AM) [Kool *et al.*, 2018] proposed a Transformer-based encoder-decoder architecture to directly generate solutions. Subsequently, POMO [Kwon *et al.*, 2020] and Sym-NCO [Kim *et al.*, 2022a] introduced multi-start and symmetry-based [Shi *et al.*, 2025; Yu *et al.*, 2024] training strategies, respectively, during the solution generation process, further enhancing solution quality. More recent studies have focused on improving generalization capability [Gao *et al.*, 2024], addressing diverse objectives and constraints [Son *et al.*, 2024; Zheng *et al.*, 2024; Bi *et al.*, 2024], and scaling up to larger problem instances [Luo *et al.*, 2023; Kim *et al.*, 2022b]. This paradigm achieves a favorable balance between solution quality and inference efficiency, and it is also adopted in this work. The L2I paradigm originated from learning local rewriting operations [Chen and Tian, 2019], and was later extended to the learning of heuristic operators such as 2-OPT and cross-exchange [d O Costa *et al.*, 2020; Hottung and Tierney, 2020; Sui *et al.*, 2021; Wu *et al.*, 2021]. In recent years, research in this direction has increasingly focused on large-scale problem solving [Cheng *et al.*, 2023; Li *et al.*, 2025]. Although this paradigm is more suitable for handling large-scale problems, it is relatively difficult to employ a unified framework to address diverse types of constraints. The L2P paradigm integrates traditional algorithmic approaches with learning-based methods, such as combining learning with the LKH algorithm [Xin *et al.*, 2021; Zheng *et al.*, 2021], integrating learning with dynamic programming [Kool *et al.*, 2022; Xu *et al.*, 2020], and leveraging large language models (LLMs) and meta-learning for automated algorithm design [Liu *et al.*, 2024b; Dernedde *et al.*, 2025]. Methods of this type, by incorporating principles of algorithm design, have the potential to improve solution quality. In this work, we adopt a similar approach within the L2C framework, introducing knowledge as a reward signal, which facilitates the handling of general forms of constraints.

3 Preliminaries

In this section, we first introduce the general form CVRPs. Then, we summarize the RL formulation for CVRPs, describing how solutions can be modeled as sequential actions and optimized under the RL paradigm.

3.1 General Form CVRPs

The CVRPs can be defined on a complete graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_0, v_1, \cdots, v_n\}$ is the set of nodes, with v_0 representing the depot and $\{v_1, \cdots, v_n\}$ representing the customer nodes. Each edge $(i, j) \in \mathcal{E}$ is associated with a non-negative cost c_{ij}. As a common practice, each node is

assigned a two-dimensional coordinate, and the Euclidean distance is used as the cost. Vehicles with identical capacity Q are stationed at the depot. Each customer node $i \in \{1, \cdots, n\}$ has a nonzero demand $q_i \leq Q$ that must be satisfied by exactly one vehicle. Each vehicle travels from the depot node, serves customers and returns to the depot. The total demand of the served nodes should not exceed the capacity.

Following [Berto *et al.*, 2024], we consider a generalized formulation of the CVRP that incorporates multiple additional constraints: (1) Open routes — vehicles are not required to return to the depot; (2) Backhaul demands — customer requests may involve both delivery and pickup operations; (3) Distance limits — the travel distance of each route must not exceed a predefined threshold; and (4) Time windows — each customer must be served within a specified time interval. Each additional constraint can be independently activated or deactivated, resulting in sixteen problem variants, denoted as CVRP, OVRP, VRPB, VRPL, VRPTW, etc.

We aim to minimize the total travel cost over all vehicle routes. Let $\{\mathcal{R}_k\}_{k=1}^{K}$ denote a feasible routing plan, where $\mathcal{R}_k$ is the sequence of nodes served by vehicle k. The cost of route $\mathcal{R}_k$ is given by

$$C_k = \sum_{(i,j)\in\mathcal{R}_k} c_{i,j}, \tag{1}$$

with $c_{i,j}$ representing the travel cost between nodes i and j. The min-sum objective is then formulated as

$$\min_{\{\mathcal{R}_k\}_{k=1}^{K}} \sum_{k=1}^{K} C_k, \tag{2}$$

subject to the following feasibility conditions: 1) each customer is visited exactly once; 2) the total demand on each route does not exceed vehicle capacity; and 3) all additional constraints are satisfied. Here, K denotes the number of active vehicles in the solution.

3.2 Reinforcement Learning for CVRPs

In the RL framework for CVRPs, a constructive solver sequentially builds a solution by selecting one customer node at a time. Formally, at step t, the solver selects an action $a_t \in V_t$ based on a context c_t, which corresponds to visiting a customer node next, according to a stochastic policy $a_t \sim \pi_\theta(a_t \mid c_t)$, parameterized by neural network parameters θ. After visiting node a_t, the state is updated to c_{t+1}, and the set of available actions is reduced: $V_{t+1} = V_t \setminus \{a_t\}$.

Once all customers are visited and feasible routes are constructed, the solver receives a reward R based on the objective. The RL training objective is to maximize the expected reward over all possible construction sequences:

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[R(\tau)\right], \tag{3}$$

where $\tau = (a_0, a_1, \cdots, a_T)$ denotes a trajectory generated by the policy. Gradient-based optimization can be performed using policy gradient methods, e.g., REINFORCE [Williams, 1992], optionally enhanced with baselines to reduce variance:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\Big[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid c_t)\left(R(\tau) - b\right)\Big], \tag{4}$$

where b is a baseline value.

4 The Proposed Method

Unlike a fully end-to-end learning paradigm, we propose a knowledge-embedded RL framework for solving generalized CVRPs. The framework integrates problem-solving principles with algorithmic design insights into the training process to improve solution quality. It also simplifies transition modeling in the training environment, reduces the complexity of incorporating new constraints, and enables a unified network architecture applicable across multiple CVRP variants.

4.1 The Overall Framework

The overall framework is illustrated in Figure 1. We adopt a **route-first, cluster-second** heuristic structure. The RL component focuses on the route-first stage, where the objective is to generate a sequence that visits all customer nodes exactly once, resulting in a permutation of customer indices. All complex constraints are handled by a dynamic programming-based cluster-second module. Given any customer permutation, this module determines the optimal splitting of the sequence into feasible vehicle routes while satisfying additional constraints such as time windows and duration limits.

Although the framework consists of two components, the reward signal is computed based on the solution produced by the cluster-second module. This reward is then used to optimize the route-first policy through RL, thereby aligning the optimization objective of the RL model with that of the original problem formulation. In this way, the framework eliminates the dependence on heuristic TSP solvers and achieves a direct, learning-based formulation of the routing problem.

4.2 The Unified Environment

The framework allows problems with heterogeneous constraints or objectives to be reduced to an identical learning structure, thus we adopt a unified formulation for constructing the RL environments under different constraint settings, as described below.

States: At step t, the environment maintains a state or context s_t that summarizes all relevant information needed for decision-making:

$$s_t = \{\mathbf{X}, \mathbf{q}, \mathbf{TW}, l, n_0, n_1, v_t, \mathbf{m}, \mathbf{f}\}, \tag{5}$$

where:

- $\mathbf{X} \in \mathbb{R}^{n\times 2}$ denotes the coordinates of depot and customer nodes. We use x_i, y_i for the i-th node;
- $\mathbf{q} \in \mathbb{R}^{n\times 1}$ represents the customer demands, including negative values for pickup nodes. We use q_i for the i-th node;
- $\mathbf{TW} \in \mathbb{R}^{n\times 3}$ encodes time windows and service durations. We use r_i, s_i, d_i to denote the start time, the service duration, and the deadline for node i.
- $l \in \mathbb{R}$ represents distance limits;
- $n_0, n_1 \in \mathbb{R}$ track the cumulative delivered and picked-up quantities for each route;
- $v_t \in \{0, \ldots, n\}$ is the current node being visited;

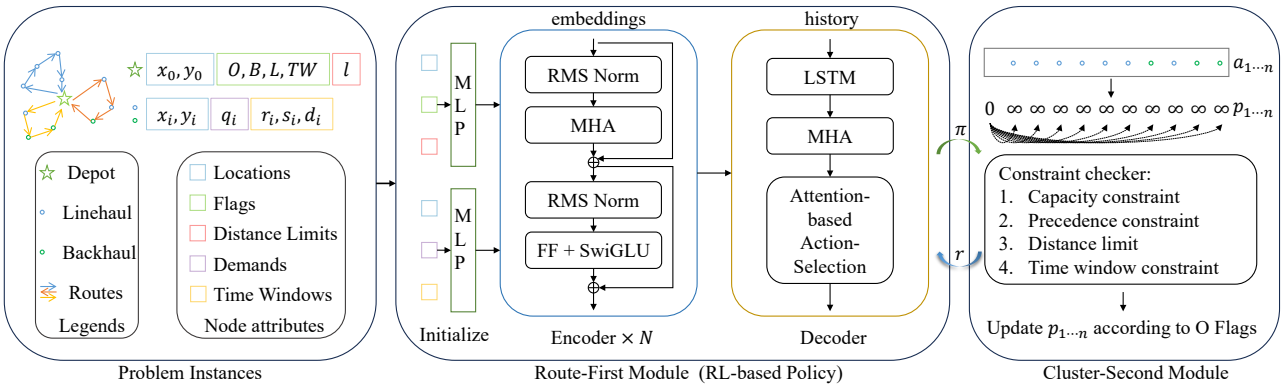

Figure 1: The Overall Framework of the Proposed Knowledge-embedded RFCS method.

- $\mathbf{m} \in \{0,1\}^n$ is the action mask, indicating feasible next visited nodes;
- $\mathbf{f} \in \{0,1\}^4$ encodes the variant features (O, B, L, TW) of the current instance, where each element represents a binary flag indicating whether the corresponding constraint is active.

When certain attributes are absent, default values are used as placeholders to maintain a unified data structure.

Actions: The action at step t is selecting the next node $a_t \in \{1, \ldots, n+1\}$ to visit.

Transitions: Benefiting from the proposed design, state updates can be performed through a simple and efficient procedure. The variables n_0, n_1, v_t are updated directly according to their definitions, while $\mathbf{m}$ is refreshed by setting the value of each visited node to zero. The responsibility for checking action feasibility is fully delegated to the cluster-second module.

Rewards: We employ a dynamic programming-based solver (Section 4.3) to optimally convert the solutions to the original problem. The reward is defined as the negative of the total cost computed by this solver.

We hypothesize that this structural consistency contributes substantially to the effectiveness and generality of the proposed framework.

4.3 Dynamic Programming-based Solver

To compute the optimal cost of the route-first sequence, we develop a dynamic programming (DP)-based approach capable of naturally handling diverse constraint types. Let a given sequence of customer nodes (excluding the depot) be denoted as $\pi = (a_1, a_2, \ldots, a_n)$, and let $p[i]$ represent the minimum total cost to serve the first i nodes. The DP formulation recursively computes

$$p[i] = \min_{j<i} \{p[j] + \overline{C}(j+1, i)\}, \tag{6}$$

where $\overline{C}(j+1, i)$ is the cost of serving nodes $(\pi_{j+1}, \ldots, \pi_i)$ as a single vehicle route, subject to capacity, pickup-and-delivery precedence, distance, and time window constraints. To naturally handle these constraints, a simple yet effective strategy is adopted: if the sequence $(\pi_{j+1}, \ldots, \pi_i)$ cannot form a feasible single vehicle route, $\overline{C}(j+1, i)$ is defined as $+\infty$. Algorithm 1 details the procedure. Specifically, lines 5-6 manage the backhaul-related variants, and all other constraints, including capacity, time windows, and duration limits, are handled through their respective conditional checks. It is straightforward to extend this algorithm to incorporate additional constraints. For example, adding support for the mixed-backhaul constraint [Zhou *et al.*, 2024] can be achieved by simply adjusting the condition in line 6.

Algorithm 1 Dynamic Programming-based Solver

Require: Customer sequence $\pi = (a_1, a_2, \cdots, a_n)$, node attributes, variant flags
Ensure: Minimum total route cost $p[n]$

1: Initialize $p[0] \leftarrow 0$, $p[i] \leftarrow +\infty$ for all $i = 1, \ldots, n$
2: **for** $t = 0$ **to** $n-1$ **do**
3: Initialize $l_o \leftarrow 0$, $l_i \leftarrow 0$, $t_{now} \leftarrow 0$, $cost \leftarrow 0$
4: **for** $i = t+1$ **to** n **do**
5: $l_o \leftarrow l_o + \max(q_{a_i}, 0)$, $\quad l_i \leftarrow l_i + \max(-q_{a_i}, 0)$
6: **if** $l_o > Q$ **or** $l_i > Q$ **or** precedence violated **then**
7: **break**
8: **end if**
9: $cost \leftarrow cost + c_{a_{i-1}, a_i}$
10: **if** TW active **then**
11: $t_{now} \leftarrow \max(t_{now}, r_{a_i}) + s_{a_i}$
12: **if** $t_{now} > d_{a_i}$ **then**
13: **break**
14: **end if**
15: **end if**
16: **if** L active **and** $cost > l$ **then**
17: **break**
18: **end if**
19: **if** O active **then**
20: $p[i] \leftarrow \min(p[i],\ p[t] + cost)$
21: **else**
22: $p[i] \leftarrow \min(p[i],\ p[t] + cost + c_{a_i, 0})$
23: **end if**
24: **end for**
25: **end for**
26: **return** $p[n]$

4.4 Model Architecture

The route-first module is parameterized by a neural network consisting of an transformer-based encoder-decoder architecture augmented with history-enhanced context representation. The architecture is designed to handle various CVRP variants within a unified network architecture.

Encoder: The encoder maps the raw node features into a latent embedding space. Let the node attributes for the i-th customer be $\mathbf{x}_i = [x_i, y_i, q_i, r_i, s_i, d_i] \in \mathbb{R}^6$. For the depot node, the attribute is represented as $\mathbf{x}_0 = [x_0, y_0, \mathrm{O}, \mathrm{B}, \mathrm{L}, \mathrm{TW}, l] \in \mathbb{R}^7$.

The node features are projected via learnable linear layers:

$$\mathbf{h}_i^{(0)} = \begin{cases} \mathrm{MLP}_{\text{customer}}(\mathbf{x}_i), & i \neq 0 \\ \mathrm{MLP}_{\text{depot}}(\mathbf{x}_0), & i = 0 \end{cases}.$$

The sequence of initial embeddings $\mathbf{H}^{(0)} = [\mathbf{h}_0^{(0)}, \ldots, \mathbf{h}_n^{(0)}] \in \mathbb{R}^{(n+1)\times d}$ is then processed by a stack of multiple Routefinder Transformer [Berto *et al.*, 2024] blocks. Each block consists of:

1. Multi-head self-attention:
   $\overline{\mathbf{H}}^{(i)} = \mathbf{H}^{(i-1)} + \mathrm{MHA}(\mathrm{RMSNorm}(\mathbf{H}^{(i-1)}))$,
2. Gated feedforward network:
   $\mathbf{H}^{(i)} = \overline{\mathbf{H}}^{(i)} + \mathrm{GatedMLP}(\mathrm{RMSNorm}(\overline{\mathbf{H}}^{(i)}))$.

A final RMS normalization is applied to produce node embeddings $\mathbf{H}^{(N)} \in \mathbb{R}^{(n+1)\times d}$ (Hereafter, (N) is omitted for notational simplicity), and a graph-level embedding $\mathbf{g}$ is computed by mean pooling of $\mathbf{H}$.

Decoder: At each decoding step t, the policy selects the next node a_t based on a context embedding $\mathbf{c}_t \in \mathbb{R}^d$ computed from:

$$\mathbf{c}_t = \mathrm{LSTM}\big([\mathbf{g}, \mathbf{h}_0, \mathbf{h}_{\text{current}}, \mathbf{d}_t]\big),$$

where $\mathbf{h}_0$ and $\mathbf{h}_{\text{current}}$ are embeddings of the depot and current nodes, and $\mathbf{d}_t = \mathrm{Linear}([n_0, n_1])$ encodes the cumulative delivery/pickup loads. This history-enhanced context enables the agent to maintain a hidden state capturing historical information, mitigating partial observability induced by the knowledge-embedded RFCS framework.

A multi-head attention mechanism is applied between $\mathbf{c}_t$ and the node embeddings $\mathbf{H}$:

$$\mathbf{y}_t = \mathrm{MHA}(\mathbf{c}_t, \mathbf{H}, \mathbf{H}, \mathrm{mask} = \mathbf{m}_t),$$

where $\mathbf{m}_t$ is the action mask. The resulting attention output is projected back into the embedding space, producing a compatibility score for each candidate node:

$$\pi_\theta(a_t \mid s_t) \propto \exp(\alpha \cdot \tanh(\mathbf{y}_t^\top \mathbf{H})),$$

with infeasible actions masked by $-\infty$. Here, $\alpha = 10$ is a temperature scaling factor controlling exploration.

This architecture ensures that the same network can process all CVRP variants without variant-specific modifications. The decoder outputs a probability distribution over feasible next nodes, from which the action is sampled or greedily selected. The network is trained via REINFORCE [Williams, 1992] accroding to equation (4).

5 Experiments

We empirically validate the effectiveness of the proposed knowledge-embedded framework in this section. Our experimental design examines three key aspects: solution quality, generalization ability, and ablation analysis—corresponding to the following research questions:

RQ1 Does the proposed knowledge-enhanced framework improve solution quality?

RQ2 How effectively does the proposed framework generalize across different problems?

RQ3 What is the impact of incorporating historical information into the learning process?

5.1 Experiment Settings

Following the data generation process in [Berto *et al.*, 2024], depot and customer coordinates are uniformly sampled within the unit square. For training instances with 50 and 100 customers, the vehicle capacities Q are set to 40 and 50, respectively, while customer demands are uniformly sampled from $\{1, 2, \cdots, 9\}$. In the backhaul scenario, 20% of the customers are randomly assigned negative demands to simulate pickup requests. For the time-window constraints, the service duration of each customer is uniformly sampled from $[0.15, 0.18]$. The length of the time window is drawn uniformly from $[0.18, 0.2]$, and both the starting and ending times are generated by applying a random offset uniformly sampled from the feasible range. The distance limit is uniformly drawn from $[2D_{max}, 3]$, where D_{max} denotes the maximum Euclidean distance between the depot and the farthest customer node.

During training, all problem attributes are randomly generated on the fly, and the constraint flags are uniformly sampled across different variants. For each training batch, we fix one particular constraint configuration so that all instances in the batch share the same problem setting. For each problem setting, the model is trained for 300 epochs, with 1,280,000 samples per epoch and a batch size of 512. To stabilize training and improve final convergence, the learning rate is progressively decreased toward the end of training. The framework demonstrates better performance without using POMO-style multi-start [Kwon *et al.*, 2020]. During training, we simply perform eight stochastic policy rollouts per instance and take their average cost as the baseline.

We evaluate our framework using the open-source dataset from [Berto *et al.*, 2024]. We do not use multi-start sampling; instead, for each instance, we perform 8-fold data augmentation and generate as many trajectories as there are nodes, selecting the best among them. This approach ensures consistent comparison data and produces an identical number of trajectories for all methods. The code can be found at https://github.com/wenwenla/ijcai26-cvrp-solver.

5.2 Baselines

We select the following methods as baselines for comparison, which are described in detail below. These solvers are open-source, allowing reproducible comparisons. We select two classical optimization algorithms, PyVRP [Wouda

n=50	CVRP	OVRP	VRPB	VRPL	VRPTW	OVRPB	OVRPL	OVRPTW
PyVRP	*10.372*	*6.507*	*9.687*	*10.587*	*16.031*	*6.898*	*6.507*	*10.510*
Or-Tools	10.572	6.553	9.802	10.776	16.089	6.928	6.552	10.519
MTPOMO	10.518	6.718	10.033	10.775	16.410	7.108	6.719	10.668
MvMOE	10.501	6.702	10.005	10.751	16.404	7.089	6.707	10.669
RouteFinder	10.499	6.684	9.977	**10.737**	16.364	7.071	6.686	10.652
RFCS (ours)	**10.484**	**6.600**	**9.873**	10.750	**16.228**	**6.986**	**6.601**	**10.571**
n=50	VRPBL	VRPBTW	VRPLTW	OVRPBL	OVRPBTW	OVRPLTW	VRPBLTW	OVRPBLTW
PyVRP	*10.186*	*18.292*	*16.356*	*6.899*	*11.669*	*10.510*	*18.589*	*11.668*
Or-Tools	10.331	18.366	16.441	6.927	11.682	10.519	18.694	11.681
MTPOMO	10.672	18.639	16.824	7.112	11.814	10.670	18.990	11.817
MvMOE	10.637	18.640	16.811	7.098	11.819	10.671	18.985	11.822
RouteFinder	10.575	18.600	16.750	7.074	11.805	10.653	18.937	11.805
RFCS (ours)	**10.475**	**18.464**	**16.650**	**6.987**	**11.724**	**10.571**	**18.849**	**11.723**
n=100	CVRP	OVRP	VRPB	VRPL	VRPTW	OVRPB	OVRPL	OVRPTW
PyVRP	*15.628*	*9.725*	*14.377*	*15.766*	*25.423*	*10.335*	*9.724*	*16.926*
Or-Tools	16.280	9.995	14.933	16.407	25.814	10.577	10.001	17.027
MTPOMO	15.934	10.210	15.082	16.149	26.412	10.878	10.214	17.420
MvMOE	15.888	10.177	15.023	16.099	26.389	10.840	10.184	17.416
RouteFinder	15.857	10.121	14.942	**16.051**	26.235	10.772	10.120	17.327
RFCS (ours)	**15.837**	**10.010**	**14.840**	16.064	**26.047**	**10.645**	**10.009**	**17.158**
n=100	VRPBL	VRPBTW	VRPLTW	OVRPBL	OVRPBTW	OVRPLTW	VRPBLTW	OVRPBLTW
PyVRP	*14.779*	*29.467*	*25.757*	*10.335*	*19.156*	*16.926*	*29.810*	*19.156*
Or-Tools	15.426	29.945	26.259	10.582	19.303	17.027	30.396	19.305
MTPOMO	15.712	30.437	26.891	10.884	19.635	17.420	30.898	19.637
MvMOE	15.640	30.436	26.868	10.847	19.638	17.419	30.892	19.641
RouteFinder	15.528	30.241	26.689	10.778	19.550	17.327	30.688	19.551
RFCS (ours)	**15.438**	**30.058**	**26.580**	**10.648**	**19.381**	**17.159**	**30.577**	**19.381**

Table 1: Evaluation Results for RQ1. The results are averaged over 1000 instances per variant. Lower values indicate better results. Bold numbers denote the best results among learning-based methods, while italic numbers represent the best-known results.

et al., 2024] and OR-Tools [Furnon and Perron, 2024], as non-learning baselines for comparison. We also compare with three recent learning-based VRP solvers: (1) MTPOMO [Liu *et al.*, 2024a], a multi-task VRP solver based on POMO [Kwon *et al.*, 2020]; (2) MVMoE [Zhou *et al.*, 2024], a multi-task VRP solver leveraging mixture-of-experts networks for unified handling of different VRP variants; and (3) RouteFinder [Berto *et al.*, 2024], a transformer-based foundation model designed for general VRPs.

5.3 Results for RQ1

Table 1 presents the average costs over 1,000 unseen instances [Berto *et al.*, 2024] for sixteen CVRP variants, including the atomic problems—CVRP, OVRP, VRPB, VRPL, and VRPTW—as well as their various combinations. The knowledge-embedded framework outperforms all learning-based methods across all variants, except for the VRPL variant, where it performs slightly below RouteFinder. The average optimality gap are shown in Figure 2. RFCS achieves a marked improvement by reducing the average optimality gap from 2.60% to 1.82% on average, highlighting the stronger solution quality achieved by our framework. Since the proposed framework shares the same neural network forward-propagation cost as other learning-based methods, and the dynamic programming-based splitting algorithm has a time complexity of $O(n^2)$, its computational overhead is negligible compared with neural network inference. Therefore, the overall solving time of our framework is comparable to that of the baseline learning-based methods. While classic approaches often yield better solution quality, they usually demand tens of minutes to complete a computation, in contrast to learning-based methods, which require only a few seconds. For more discussions regarding runtime and scalability, please refer to the appendix of the arXiv version.

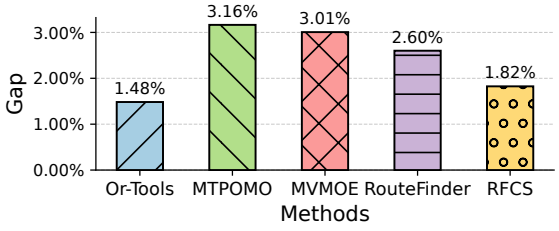

Figure 2: The Average Optimality Gap.

5.4 Results for RQ2

We evaluate the zero-shot transfer capability of RFCS across different capacity distributions encountered during training. During training, for scenarios with $n = 50$, the vehicle capacity was set to 40. We directly evaluate the model obtained in RQ1 on instances with varying vehicle capacities to examine its performance changes. The target distribution of vehicle capacities ranges from $[50, 90]$, and the experimental results of zero-shot transfer evaluation are presented in Table 2. The results indicate that when the capacity distributions differ between training and evaluation, the knowledge-embedded framework exhibits superior zero-shot transfer performance compared to existing learning-based methods.

Capacity	50	60	70	80	90
MTPOMO	9.255	8.482	7.950	7.553	7.269
MvMOE	9.237	8.456	7.915	7.511	7.206
RouteFinder	9.238	8.453	7.908	7.487	7.180
RFCS (ours)	**9.214**	**8.433**	**7.882**	**7.469**	**7.154**

Table 2: Evaluation of the Knowledge-embedded Framework Zero-Shot Transfer to Unseen Vehicle Capacity Distributions.

A commonly studied CVRP variant introduces mixed linehaul and backhaul demands [Zhou *et al.*, 2024], allowing vehicles to alternate between delivery and pickup operations instead of strictly finishing deliveries before pickups. In this case, the key constraint is that the vehicle load must not violate the capacity limit at any time. As described in section 4.3, we only need to modify the Algorithm 1 to adapt the solver to this scenario. The zero-shot transfer results to mixed demands are shown in Table 3. The results show that our method outperforms the baselines when transferring from VRPB and OVRPB to the mixed constraint scenario, achieves comparable performance for VRPBL, and exhibits a notable performance drop for VRPBTW. While the framework does not surpass the baselines across every test case, the findings highlight its inherent flexibility and adaptability.

Source	VRPB	OVRPB	VRPBL	VRPBTW
MTPOMO	9.900	7.002	10.284	17.257
MvMOE	9.857	6.990	10.241	**17.103**
RouteFinder	9.879	6.948	**10.239**	17.298
RFCS (ours)	**9.534**	**6.721**	10.301	18.194

Table 3: Evaluation of the Knowledge-embedded Framework Zero-Shot Transfer to Unseen Mixed Backhaul Constraints.

5.5 Results for RQ3

We conduct an ablation study to assess the contribution of the history-enhanced context in the knowledge-embedded framework. The results are summarized in Table 4. For the variant without historical information, we adopt an identical training setup, except that the LSTM module in the decoder responsible for handling historical information is replaced with a fully connected layer. The results of this ablation study highlight the necessity of incorporating historical information. While the LSTM is not the most advanced architecture for modeling history, it was chosen for its ability to incrementally maintain past information. Further improvements in processing historical information by introduce more advanced network represent an interesting direction for future research.

	CVRP	OVRP	VRPB	VRPL	VRPTW
RFCS	**10.484**	**6.600**	**9.873**	**10.750**	**16.228**
Gap	1.08%	1.43%	1.92%	1.54%	1.23%
w/o H	10.720	6.667	10.035	10.990	16.271
Gap	3.36%	2.46%	3.59%	3.81%	1.50%

Table 4: Ablation Results for 50-node CVRP Variants. "Gap" denotes the optimality gap relative to the best-known solutions.

Another interesting ablation concerns whether incorporating the dynamic programming-based reward during training is necessary. To address this question, we examine the results obtained by directly applying the dynamic programming module to optimize the solutions for CVRP generated by baselines. The corresponding results are shown in Table 5. From the table, we observe that applying the cluster-second optimization improves the solutions generated by the learning-based methods. However, the improvement is marginal, and the resulting performance remains inferior to that of RFCS trained with the dynamic programming-based reward. This not only highlights the optimality improvement provided by the DP procedure, but also demonstrates that incorporating the DP-based reward during training is necessary.

	Original	AfterDP	Reduced
MTPOMO	10.518	10.516	0.02 %
MvMOE	10.501	10.499	0.02 %
RouteFinder	10.499	10.498	0.01 %

Table 5: Ablation Study on the Necessity of Dynamic Programming-Based Rewards in RFCS Training.

6 Conclusion

In this paper, we propose a unified framework for a broad range of CVRP variants. Inspired by the route-first cluster-second heuristic, the framework integrates RL with dynamic programming-based algorithmic design knowledge to solve CVRPs with various constraints. Specifically, the route-first component leverages RL to generate visitation sequences. The cluster-second component employs dynamic programming to optimally convert a given sequence into a feasible solution for the original problem, and provides the cost as a reward to guide RL. The proposed framework not only achieves strong performance in terms of solution quality and transferability but also offers a flexible and unified approach to handling diverse constraints. In future work, we plan to incorporate more advanced network architectures with this framework to achieve even better results. We also plan to extend this approach to other domains beyond routing problems.

Acknowledgments

This work was supported by the NSFC Major Research Plan Key Program under Grant No. 92582204, and the Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

[Berto *et al.*, 2024] Federico Berto, Chuanbo Hua, Nayeli Gast Zepeda, André Hottung, Niels Wouda, Leon Lan, Kevin Tierney, and Jinkyoo Park. Routefinder: Towards foundation models for vehicle routing problems. In *ICML 2024 Workshop on Foundation Models in the Wild*, 2024.

[Bi *et al.*, 2024] Jieyi Bi, Yining Ma, Jianan Zhou, Wen Song, Zhiguang Cao, Yaoxin Wu, and Jie Zhang. Learning to handle complex constraints for vehicle routing problems. *Advances in Neural Information Processing Systems*, 37:93479–93509, 2024.

[Chen and Tian, 2019] Xinyun Chen and Yuandong Tian. Learning to perform local rewriting for combinatorial optimization. *Advances in neural information processing systems*, 32, 2019.

[Cheng *et al.*, 2023] Hanni Cheng, Haosi Zheng, Ya Cong, Weihao Jiang, and Shiliang Pu. Select and optimize: Learning to solve large-scale tsp instances. In *International conference on artificial intelligence and statistics*, pages 1219–1231. PMLR, 2023.

[d O Costa *et al.*, 2020] Paulo R d O Costa, Jason Rhuggenaath, Yingqian Zhang, and Alp Akcay. Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning. In *Asian conference on machine learning*, pages 465–480. PMLR, 2020.

[Dernedde *et al.*, 2025] Tim Dernedde, Daniela Thyssens, Sören Dittrich, Maximilian Stubbemann, and Lars Schmidt-Thieme. Moco: A learnable meta optimizer for combinatorial optimization. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 236–248. Springer, 2025.

[Drakulic *et al.*, 2025] Darko Drakulic, Sofia Michel, and Jean-Marc Andreoli. Goal: A generalist combinatorial optimization agent learner. In *Proceedings of the International Conference on Learning Representations (ICLR) 2025*, 2025. Poster.

[Furnon and Perron, 2024] Vincent Furnon and Laurent Perron. Or-tools routing library, 2024.

[Gao *et al.*, 2024] Chengrui Gao, Haopu Shang, Ke Xue, Dong Li, and Chao Qian. Towards generalizable neural solvers for vehicle routing problems via ensemble with transferrable local policy. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 6914–6922, 2024.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Hottung and Tierney, 2020] André Hottung and Kevin Tierney. Neural large neighborhood search for the capacitated vehicle routing problem. In *ECAI 2020*, pages 443–450. IOS Press, 2020.

[Jeong and Song, 2025] Ho Young Jeong and Byung Duk Song. Optimizing urban logistics: Vehicle routing problem with underground transportation. *IEEE Transactions on Intelligent Transportation Systems*, 2025.

[Kim *et al.*, 2022a] Minsu Kim, Junyoung Park, and Jinkyoo Park. Sym-nco: Leveraging symmetricity for neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 35:1936–1949, 2022.

[Kim *et al.*, 2022b] Minsu Kim, Jiwoo Son, Hyeonah Kim, and Jinkyoo Park. Scale-conditioned adaptation for large scale combinatorial optimization. In *NeurIPS 2022 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2022.

[Kool *et al.*, 2018] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2018.

[Kool *et al.*, 2022] Wouter Kool, Herke van Hoof, Joaquim Gromicho, and Max Welling. Deep policy dynamic programming for vehicle routing problems. In *International conference on integration of constraint programming, artificial intelligence, and operations research*, pages 190–213. Springer, 2022.

[Kwon *et al.*, 2020] Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. Pomo: Policy optimization with multiple optima for reinforcement learning. *Advances in Neural Information Processing Systems*, 33:21188–21198, 2020.

[Leng and Li, 2021] Kaijun Leng and Shanghong Li. Distribution path optimization for intelligent logistics vehicles of urban rail transportation using vrp optimization model. *IEEE Transactions on Intelligent Transportation Systems*, 23(2):1661–1669, 2021.

[Lenstra and Kan, 1981] Jan Karel Lenstra and AHG Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981.

[Li *et al.*, 2007] Feiyue Li, Bruce Golden, and Edward Wasil. The open vehicle routing problem: Algorithms, large-scale test problems, and computational results. *Computers & operations research*, 34(10):2918–2930, 2007.

[Li *et al.*, 2025] Ke Li, Fei Liu, Zhenkun Wang, and Qingfu Zhang. Destroy and repair using hyper-graphs for routing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 18341–18349, 2025.

[Liu *et al.*, 2024a] Fei Liu, Xi Lin, Qingfu Zhang, Xialiang Tong, and Mingxuan Yuan. Multi-task learning for routing problem with cross-problem zero-shot generalization. *International Conference on Knowledge Discovery and Data Mining (KDD)*, 2024.

[Liu *et al.*, 2024b] Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu

Zhang. Evolution of heuristics: towards efficient automatic algorithm design using large language model. In *Proceedings of the 41st International Conference on Machine Learning*, pages 32201–32223, 2024.

[Luo *et al.*, 2023] Fu Luo, Xi Lin, Fei Liu, Qingfu Zhang, and Zhenkun Wang. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. *Advances in Neural Information Processing Systems*, 36:8845–8864, 2023.

[Prins *et al.*, 2014] Christian Prins, Philippe Lacomme, and Caroline Prodhon. Order-first split-second methods for vehicle routing problems: A review. *Transportation Research Part C-emerging Technologies*, 40:179–200, 2014.

[Sbai *et al.*, 2022] Ines Sbai, Saoussen Krichen, and Olfa Limam. Two meta-heuristics for solving the capacitated vehicle routing problem: the case of the tunisian post office. *Operational Research*, 22(1):507–549, 2022.

[Shi *et al.*, 2025] Rongye Shi, Xin Yu, Yandong Wang, Yongkai Tian, Zhenyu Liu, Wenjun Wu, Xiao-Ping Zhang, and Manuela M Veloso. Symmetry-informed marl: A decentralized and cooperative uav swarm control approach for communication coverage. *IEEE Transactions on Mobile Computing*, 24(01):1–18, 2025.

[Son *et al.*, 2024] Jiwoo Son, Minsu Kim, Sanghyeok Choi, Hyeonah Kim, and Jinkyoo Park. Equity-transformer: Solving np-hard min-max routing problems as sequential generation with equity context. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 20265–20273, 2024.

[Sui *et al.*, 2021] Jingyan Sui, Shizhe Ding, Ruizhi Liu, Liming Xu, and Dongbo Bu. Learning 3-opt heuristics for traveling salesman problem via deep reinforcement learning. In *Asian conference on machine learning*, pages 1301–1316. PMLR, 2021.

[Tiwari and Sharma, 2023] Krishna Veer Tiwari and Satyendra Kumar Sharma. An optimization model for vehicle routing problem in last-mile delivery. *Expert Systems with Applications*, 222:119789, 2023.

[Vidal *et al.*, 2014] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, 234(3):658–673, 2014.

[Vidal, 2015] Thibaut Vidal. Technical note: Split algorithm in o(n) for the capacitated vehicle routing problem. *Comput. Oper. Res.*, 69:40–47, 2015.

[Vinyals *et al.*, 2015] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *Advances in neural information processing systems*, 28, 2015.

[Wang *et al.*, 2025] Wen Wang, Xiangchen Wu, Liang Wang, Hao Hu, Xianping Tao, and Linghao Zhang. Solving the min-max multiple traveling salesmen problem via learning-based path generation and optimal splitting. *arXiv preprint arXiv:2508.17087*, 2025.

[Williams, 1992] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256, May 1992.

[Wouda *et al.*, 2024] Niels A. Wouda, Leon Lan, and Wouter Kool. PyVRP: a high-performance VRP solver package. *INFORMS Journal on Computing*, 36(4):943–955, 2024.

[Wu *et al.*, 2021] Yaoxin Wu, Wen Song, Zhiguang Cao, Jie Zhang, and Andrew Lim. Learning improvement heuristics for solving routing problems. *IEEE transactions on neural networks and learning systems*, 33(9):5057–5069, 2021.

[Wu *et al.*, 2024] Xuan Wu, Di Wang, Lijie Wen, Yubin Xiao, Chunguo Wu, Yuesong Wu, Chaoyu Yu, Douglas L Maskell, and You Zhou. Neural combinatorial optimization algorithms for solving vehicle routing problems: A comprehensive survey with perspectives. *arXiv preprint arXiv:2406.00415*, 2024.

[Xin *et al.*, 2021] Liang Xin, Wen Song, Zhiguang Cao, and Jie Zhang. Neurolkh: Combining deep learning model with lin-kernighan-helsgaun heuristic for solving the traveling salesman problem. *Advances in Neural Information Processing Systems*, 34:7472–7483, 2021.

[Xu *et al.*, 2020] Shenghe Xu, Shivendra S Panwar, Murali Kodialam, and TV Lakshman. Deep neural network approximated dynamic programming for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1684–1691, 2020.

[Yu *et al.*, 2024] Xin Yu, Rongye Shi, Pu Feng, Yongkai Tian, Simin Li, Shuhao Liao, and Wenjun Wu. Leveraging partial symmetry for multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17583–17590, 2024.

[Zhang *et al.*, 2022] Haifei Zhang, Hongwei Ge, Jinlong Yang, and Yubing Tong. Review of vehicle routing problems: Models, classification and solving algorithms. *Archives of Computational Methods in Engineering*, 29(1):195–221, 2022.

[Zheng *et al.*, 2021] Jiongzhi Zheng, Kun He, Jianrong Zhou, Yan Jin, and Chu-Min Li. Combining reinforcement learning with lin-kernighan-helsgaun algorithm for the traveling salesman problem. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 12445–12452, 2021.

[Zheng *et al.*, 2024] Zhi Zheng, Shunyu Yao, Zhenkun Wang, Tong Xialiang, Mingxuan Yuan, and Ke Tang. Dpn: Decoupling partition and navigation for neural solvers of min-max vehicle routing problems. In *International Conference on Machine Learning*, pages 61559–61592. PMLR, 2024.

[Zhou *et al.*, 2024] Jianan Zhou, Zhiguang Cao, Yaoxin Wu, Wen Song, Yining Ma, Jie Zhang, and Xu Chi. Mvmoe: Multi-task vehicle routing solver with mixture-of-experts. In *International Conference on Machine Learning*, pages 61804–61824. PMLR, 2024.