Transportation Research Part E 209 (2026) 104742

Contents lists available at ScienceDirect

Transportation Research Part E

journal homepage: www.elsevier.com/locate/tre

Deep reinforcement learning approach to solving clustered vehicle routing problems

Yaoxin Wu [a], Yue Yu [b], Lingxiao Wu [c], Tao Feng [d], Lu Zhang [b,*], Zhenkun Wang [e], Jie Gao [f,*]

[a] Department of Industrial Engineering and Innovation Sciences, Eindhoven University of Technology, Eindhoven, 5600 MB, The Netherlands
[b] School of Cybersecurity, Chengdu University of Information Technology, Chengdu, 178902, China
[c] Department of Aeronautical and Aviation Engineering, The Hong Kong Polytechnic University, Hong Kong, China
[d] School of Transportation and Logistics, Southwest Jiaotong University, Chengdu, China
[e] School of System Design and Intelligent Manufacturing, Southern University of Science and Technology, Shenzhen, China
[f] Department of Transport and Planning, Delft University of Technology, Delft, The Netherlands

ARTICLE INFO

Keywords:
Deep reinforcement learning
Clustered vehicle routing
Cluster-aware attention

ABSTRACT

Clustered vehicle routing problems (CluVRPs) represent a complex class of combinatorial optimization problems with significant real-world relevance. They extend classic VRPs by introducing pre-specified customer clusters and requiring effective routing both between clusters and within each cluster. While numerous deep learning approaches have been developed to address the standard VRP, research on CluVRPs remains relatively limited, presenting opportunities and challenges for advancing solutions to more practical VRPs with cluster-related constraints. This paper offers a deep reinforcement learning (DRL) approach to solving CluVRPs. We propose a cluster-aware attention module in the encoder, along with inter-cluster and intra-cluster decoders to specialize the constructive policies within and between clusters. Symmetrical data augmentation is adopted in the training to improve the performance. Empirical results in different CluVRP variants manifest that the DRL method outperforms existing approaches, consistently offering advantages for various instances.

1. Introduction

Clustered Vehicle Routing Problems (CluVRPs) are a class of vehicle routing problems that involve routing vehicles to serve customers grouped into different clusters (i.e., the community of customers). CluVRPs belong to the NP-hard class since they are generally more difficult than standard VRPs without the cluster-related constraints, such as traveling salesman problem (TSP) and capacitated VRP (CVRP). The cluster-related constraints introduce more complex solution space that respects both inter-cluster sequencing decisions and intra-cluster routing optimization. In addition to the standard CVRP constraints, such as vehicle capacity limits and subtour elimination, CluVRPs are characterized by cluster-related constraints. For example, all customers within a given cluster must be served continuously by the same vehicle. Consequently, in CluVRPs, the sequencing of clusters, the identification of inbound and outbound customers (i.e., the first and last customers served within a cluster), and the routing of customers within each cluster must all be jointly optimized. Despite their high complexity, CluVRPs closely resemble practical vehicle routing settings in many scenarios, such

* Corresponding authors.
E-mail addresses: y.wu2@tue.nl (Y. Wu), yyu012199@gmail.com (Y. Yu), lingxiao-leo.wu@polyu.edu.hk (L. Wu), tao.feng@swjtu.edu.cn (T. Feng), zhanglu@cuit.edu.cn (L. Zhang), wangzhenkun90@gmail.com (Z. Wang), J.Gao-1@tudelft.nl (J. Gao).

https://doi.org/10.1016/j.tre.2026.104742
Received 22 June 2025; Received in revised form 4 February 2026; Accepted 5 February 2026
Available online 12 February 2026

as waste collection, parcel delivery, and school bus routing (Kim et al., 2025; Sevaux et al., 2008; Baldacci et al., 2010). Traditional approaches to CluVRPs focus on exact and heuristic algorithms (Battarra et al., 2014; Vidal et al., 2015). According to the literature, exact algorithms, such as branch and price (Battarra et al., 2014), are typically limited to relatively small problem instances and are unable to produce solutions for practical scenarios within reasonable computation times (Heßler and Irnich, 2021). Consequently, heuristic algorithms are commonly employed to efficiently explore the solution space and obtain high-quality solutions. Therefore, heuristics are more commonly researched in the literature due to their better efficiency (Defryn and Sörensen, 2017; Hintsch and Irnich, 2018; Pop et al., 2018). Different metaheuristics, such as large multiple neighborhood search (Hintsch and Irnich, 2018) and variable neighborhood search (Defryn and Sörensen, 2017), have been proposed to address the CluVRP. Although more efficient than exact methods, state-of-the-art metaheuristics typically rely on a two-level search framework, applying different heuristics at the cluster and customer levels. It requires substantial human effort for heuristic design, limits generalizability to other CluVRP variants, e.g., the dynamic CluVRP, due to their distinct problem structure.

Based on the continuity of customer service within a cluster by the same vehicle, CluVRPs are primarily categorized into the hard-clustered VRP and the soft-clustered VRP (SoftCluVRP). The SoftCluVRP is a relaxed variant of the hard-clustered VRP, which does not require that all customers within a cluster be visited consecutively by the same vehicle (Hintsch and Irnich, 2020; Zhou et al., 2023). In other words, the same vehicle can leave and re-enter a cluster during service. In this work, we primarily focus on hard-clustered VRPs and their variants (such as the generalized VRP and dynamic CluVRP). Hence, the term CluVRP in the remainder of this paper refers specifically to the hard-clustered variant.

In the past few years, deep reinforcement learning (DRL) has been extensively studied to solve different VRPs, such as pickup and delivery problem (Kong et al., 2024), fair collaborative VRP (Mak et al., 2023), truck-drone routing problems (Peng et al., 2025; Li et al., 2025), automated guided VRP (Ho et al., 2025). While the learned solutions have demonstrated advantage over some traditional heuristic algorithms, the studied problems generally overlook the cluster-related constraints. In practical VRPs involving clustered customers (e.g., those living in different communities), both dimensions of inter-cluster sequencing and intra-cluster routing determine overall solution efficiency. Building on the above discussion, we aim to propose a DRL approach to automatically learn construction policies for the CluVRP and its variants. The learned policies can generate solutions more efficiently than exact or metaheuristic methods, without relying on extensive search procedures. Moreover, automatic policy learning eliminates the need for manual algorithm design and can be quickly adapted to different problem variants, including generalized and dynamic CluVRPs, as considered in this work.

In this paper, we propose a deep reinforcement learning (DRL) approach to solving CluVRPs, which integrates decisions on inter-cluster sequencing and intra-cluster routing in one single policy network. Specifically, we propose interactive Markov decision processes (MDPs) to describe the sequential decision making problems for both inter-cluster sequencing and intra-cluster routing. The MDPs of two agents are inextricably intertwined to influence the trajectories of each agent. Given the MDPs, we propose an encoder-decoder-based neural network to conduct two different types of policies by agents. First, we propose the cluster-aware attention module in the encoder, which employs self-attention for customer-level interactions and simultaneously adopts cross-attention to involve cluster-level interaction. After the shared encoder, the inter-cluster and intra-cluster decoders diverge to differentiate the policies within and between clusters, for inter-cluster sequencing and intra-cluster routing, respectively. We also apply symmetrical data augmentation to enhance the training for improving the performance. The main contributions of this paper are threefold:

- We propose a DRL approach to solving CluVRPs that synergizes the optimization for both inter-cluster sequencing and intra-cluster routing by using a single encoder-decoder-based neural network.
- Modeling CluVRPs by interactive MDPs, we structure the neural network to differentiate the policies by two agents. A cluster-aware, attention-based encoder integrates both customer-level and cluster-level interactions, while two decoders tailor policies for inter-cluster sequencing and intra-cluster routing.
- We evaluate our approach in different CluVRP variants including the classic CluVRP, dynamic CluVRP, and generalized VRP, which empirically verifies its effectiveness for various instances configured with distinct problem sizes and numbers of clusters.

2. Related work

This section presents the literature on traditional solutions for CluVRPs and existing deep learning based solutions for VRPs without cluster-related constraints.

2.1. CluVRPs

Traditional algorithms for solving CluVRPs encompass exact and heuristic methods. Exact solutions were generally developed in the branch and bound framework. The early exact algorithms for a CluVRP variant were proposed in Battarra et al. (2014), Hoogeboom et al. (2016), which simplified the problem by generating the intra-cluster routes in the preprocessing stage and solving the problem by branch and cut algorithm. Three different CluVRP models were discussed in Freitas et al. (2023) that were implemented and solved by the branch-cut-and-price algorithm in the VRPSolver package. Although exact algorithms are proposed to obtain optimal solutions, their efficiency is significantly limited due to the NP-hard nature of CluVRPs. Instead, heuristic methods were more often researched in the literature. An iterative local search and hybrid genetic search method is proposed in Vidal et al. (2015) to solve a CluVRP. Afterward, a two-level variable neighborhood search (VNS) heuristic method was developed for the CluVRP, which employed customer-level and cluster-level VNS to obtain significantly better solutions in less time (Defryn and Sörensen, 2017). Following

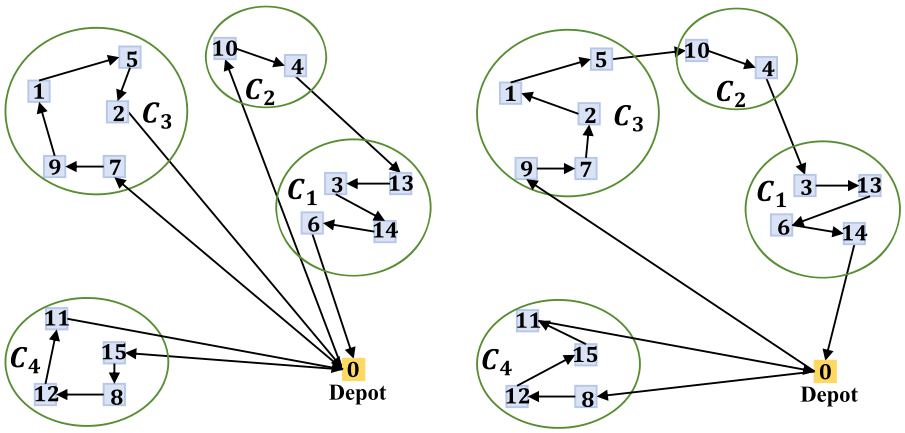

Fig. 1. An illustration of two solutions for a classic CluVRP instance consisting of 15 customers in 4 clusters.

the similar neighborhood search scheme, the multiple destruction and repair operators were proposed in accordance with variable neighborhood descent (Hintsch and Irnich, 2018). Islam et al. (2021) integrated VNS with the particle swarm optimization to keep solution diversity and intensity during search process. Kim et al. (2025) combined ant colony optimization with K-means clustering to solve a specific CluVRP for waste collection.

Note that there is a line of work on soft-clustered vehicle routing problem (SoftCluVRP), which is a relaxed version of the hard-clustered vehicle routing problem (CluVRP) (Hintsch and Irnich, 2020). Unlike CluVRP, which requires a vehicle to serve all customers within a cluster in a continuous sequence, SoftCluVRP relaxes this constraint by allowing the vehicle to leave and re-enter a cluster during service. Since exact algorithms are often limited to small instances (Heßler and Irnich, 2021), several heuristics attempted to search efficient solutions for SoftCluVRP. For example, Cosma et al. (2022) proposed a two-level genetic algorithm that addresses the clustering and routing problems separately at two levels. Similarly, Latorre (2025) tackled cluster-level sequence optimization and node-level route optimization by two tailored local search algorithms, respectively. Hintsch (2021) proposed a large multiple neighborhood search that integrates multiple cluster destruction and repair operators, combined with a variable neighborhood descent for post-optimization. Zhou et al. (2023) developed a two-level memetic search that integrates genetic operators, neighborhood search and tabu search to maintain a population of solutions. They further enhanced performance through a bi-population collaborative search that promotes both diversity and convergence in the search process (Zhou et al., 2025b). In this work, we focus on CluVRP and its variants (generalized VRP and dynamic CluVRP). However, we also adapt our approach to automatically learn a policy for solving SoftCluVRP, as demonstrated by the experiment in Section 5.7.

2.2. DRL for VRPs

DRL methods for VRPs can be categorized based on the types of learned policies. The majority of methods learned constructive policies to build up solutions in an autoregressive way. Different neural networks have been proposed to infer the probabilities of customers linking to a partial solution, including but not limited to pointer networks (Bello et al., 2017; Jin et al., 2023), Transformer-like networks (Kool et al., 2018; Kwon et al., 2020) and graph neural networks (Dai et al., 2017; Kwon et al., 2021; Qiu et al., 2022). The follow-up literature enhanced the solution performance and diverged by exploiting different properties in VRPs, such as symmetricity (Kim et al., 2022; Fang et al., 2024), transferable locality (Gao et al., 2023), and decomposability (Fu et al., 2021; Hou et al., 2023). The other DRL methods investigate improving policies to reform an initial solution, such as local search (Hudson et al., 2021), neighborhood search (Li et al., 2021b; Hottung and Tierney, 2022), beam search (Choo et al., 2022), and evolutionary algorithm (Ye et al., 2024; Kim et al., 2024). Alternatively, the (un)supervised learning for VRPs has been less explored (Luo et al., 2023; Sun and Yang, 2023; Min et al., 2023). More recently, studies have focused on solving large-scale VRPs (Hu et al., 2025), developing more effective neural policy alternatives (Pan et al., 2025), and managing complex constraints (Bi et al., 2024). For a comprehensive overview, we refer interested readers to the recent survey (Zhou et al., 2025a). In general, the advance of DRL methods is mostly centered on simple VRPs such as TSP and CVRP. Although some of them have been extended to other VRP types (Liu et al., 2024; Bogyrbayeva et al., 2023), cluster-related constraints have been underexplored by DRL methods, which are challenging to address in practical vehicle routing optimization. As a first attempt in this direction, this work introduces a DRL approach applicable to general CluVRPs (such as the generalized VRP and dynamic CluVRP), effectively addressing the cluster-related constraints in these problems.

3. Preliminaries

3.1. Definition of CluVRPs

A CluVRP problem involving n customers is typically delineated as a graph $G = (V, E)$ (Sevaux et al., 2008). The set $V = \{0, 1, \ldots, n\}$ represents the node set, where vertex 0 denotes the depot, and the remaining vertices represent customers. Each customer i is rep-

resented as $i = \{(x_i, y_i), C_i\}$, where (x_i, y_i) specifies the node's coordinate, and $C_i \in \{C_1, C_2, \ldots, C_K\}$ denotes the cluster of customer i. Each customer is assigned to exactly one cluster, and d_k is a positive value denoting the total demand of customers in a cluster. E represents the set of arcs, and each arc $e \in E$ has an associated travel cost c_{ij}, with $i, j \in V$. In this work, we use the Euclidean distance between nodes to calculate the cost, and define the objective value is the total travel cost of a route, which is to be minimized. In the CluVRP, the vehicles start at the depot and visit each customer exactly once. The vehicles must serve all customers continuously within a cluster before returning to the depot or proceeding to customers in other clusters. Consequently, a vehicle can enter and leave a given cluster only once, and each cluster is assigned to exactly one vehicle. The total demand on each subroute that starts and ends at the depot cannot exceed the vehicles' capacity Q. In other words, each vehicle undertakes a single trip: it departs the depot empty and returns to the depot upon completion, without subsequent departures. The total demand of all clusters visited on that trip should not exceed the vehicle load. Therefore, situations where the remaining vehicle capacity is insufficient to serve a cluster are prohibited, since revisiting a cluster is not allowed under the single-entry/single-exit constraint. In our DRL approach, we implement a masking scheme that assigns zero probability to customers within clusters whose demands exceed the vehicle's remaining load. There are different variants of the introduced classic CluVRP, such as generalized VRP and dynamic CluVRP, which are delineated in Appendix A. The detailed mathematical model of the classic CluVRP is shown in Appendix B.

Given the definition of CluVRP, there is a trade-off between optimizing inter-cluster sequencing and intra-cluster routing. The inbound and outbound customers (the first and last customers served in a cluster) play pivotal roles in influencing the travel cost between clusters. In the meantime, as the start and end points of the tours within each cluster, they also affect the travel costs within clusters. The inbound and outbound customers making the smallest travel cost between clusters are not necessarily optimizing intra-cluster routing. We illustrate two solutions for a CluVRP in Fig. 1, where the second solution has a smaller inter-cluster cost but a larger intra-cluster cost. It indicates the equal importance of inter-cluster sequencing and intra-cluster routing for solution improvement. In line with this, state-of-the-art CluVRP methods predominantly adopt a two-level framework (Defryn and Sörensen, 2017; Hintsch and Irnich, 2018; Islam et al., 2021) that simultaneously handles inter-cluster sequencing at the outer level and optimizes intra-cluster routes at the inner level. By contrast, traditional CVRP methods operate at the node level and lack mechanisms for inter-cluster sequencing, so that, traditional CVRP methods cannot be directly adapted after the inclusion of the cluster constraint.

3.2. Current DRL framework for VRPs

Transformer networks can be used to construct solutions to combinatorial optimization problems (COPs) (Kool et al., 2018; Kwon et al., 2020). Concretely, the numerical features of a COP instance G are projected by an embedding layer, and then updated by an encoder. The global representation learned by the encoder, along with the context representation (e.g., the embedding of the partial route in a constructive process), is taken as input to the decoder. The decoder iteratively outputs the probabilities of candidate nodes linking to the partial route. The decoding procedure ends when all nodes are selected and linked to the route, according to the probabilities at each iteration. The probabilistic chain rule for constructing a solution π is $p_\theta(\pi|G) = \prod_{t=1}^{T} p_\theta(\pi_t|G, \pi_{<t})$, where T refers to the number of total iterations; π_t and $\pi_{<t}$ denote the selected node and the current partial route at the iteration t. The Transformer is often trained by REINFORCE algorithm (Williams, 1992) with the gradients calculated below,

$$\nabla_\theta J(\theta|G) \approx \mathbb{E}_{p_\theta(\pi|G)}[(f(\pi) - b(G))\nabla_\theta \log p_\theta(\pi \mid G)] \tag{1}$$

where $\mathbb{E}_{p_{(\theta)}(\pi|G)} f(\pi)$ is the expected cost of solutions and $b(\cdot)$ is a baseline for reducing estimation variance. Typically, the baseline is empirically implemented as the average cost of solutions across a batch of instances.

The framework outlined above has shown promise in simple VRP variants, such as the TSP and CVRP. However, previous DRL-based methods assume homogeneous decisions at each constructive step, namely, selecting an (unserved) customer to serve. In the CluVRP, this homogeneity might lessen the importance of decisions for selecting inbound customers, which are pivotal in determining the sequence of clusters, and, as start points, affect the solution construction within clusters. Hence, it is paramount to differentiate the policy for selecting inbound customers, i.e., the inter-cluster sequencing policy, from the intra-cluster routing policy.

4. The method

We model the solution construction for CluVRPs by interactive MDPs, and delineate the encoder-decoder-based policy network featured by a cluster-aware attention module in the encoder. To emphasize actions for selecting inbound customers, we differentiate policies by employing inter-cluster and intra-cluster decoders. Finally, we present the training procedure.

4.1. Interactive MDPs

We propose interactive MDPs to model the constructive process of CluVRP solutions. The interactive MDPs consist of two agents: one responsible for intra-cluster sequencing and the other for inter-cluster routing. Formally, the interactive MDPs are defined by a six-tuple $\langle S, A, \tau, r, p, F_{clu} \rangle$, where S, A, τ, r denote state space, action space, transition and reward function, respectively. p represents the policy parameterized as a neural network p_θ. F_{clu} is a flag indicating which agent should execute the next action for either sequencing or routing.

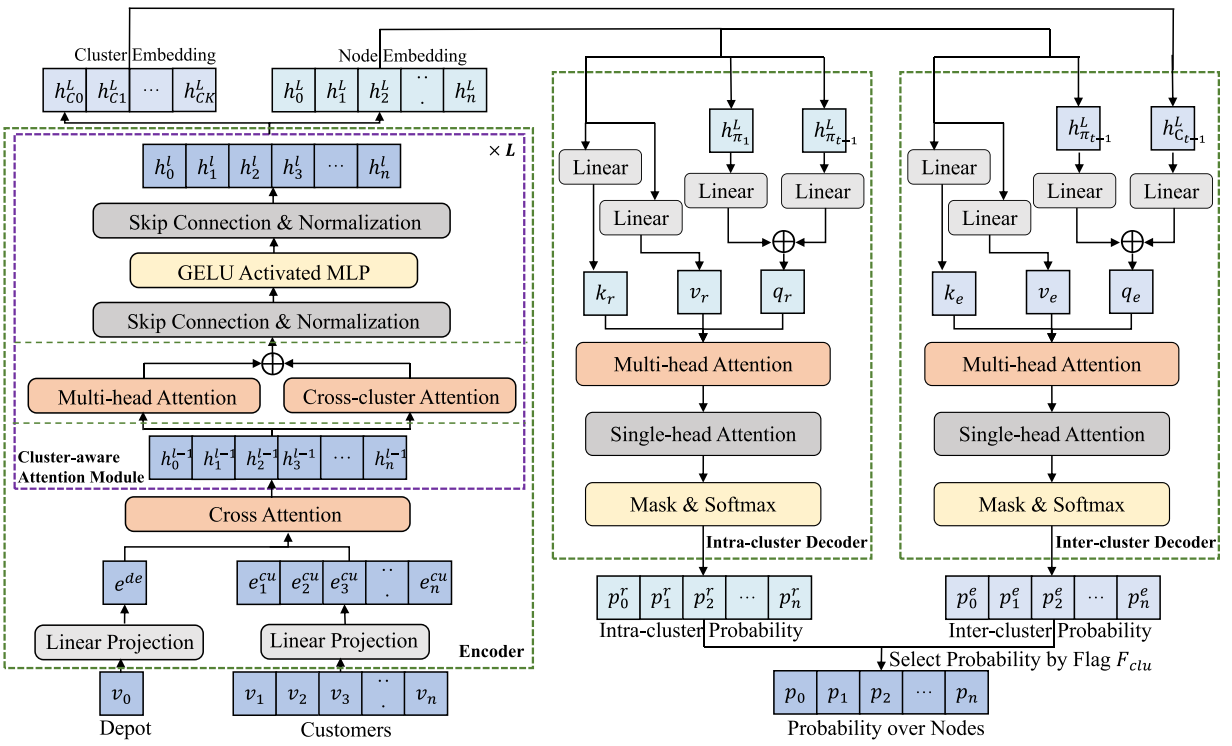

Fig. 2. An illustration of the policy network. The cluster-aware attention module in the encoder consists of a multi-head self-attention layer for customer-level interactions and a cross-attention layer for cluster-level interactions. The decoding works out probabilities over nodes by deriving probabilities from inter-cluster and intra-cluster decoders, and selecting either one for sequencing or routing decisions based on F_{clu}.

State. A state at tth constructive step is denoted by $s_t = \langle P_t, Q_t \rangle, s_t \in S$. P_t represents the partial solution, including all the visited nodes till step t. Q_t represents the load of the vehicle at step t. At step 0, the vehicle starts from the depot, where P_t only includes the depot. Every time the vehicle visits the depot, Q_t is reset to the maximum capacity Q.

In particular, when the sequencing agent selects an inbound node (to enter a cluster), the load is reduced by the total demand of the cluster. After that, the routing agent resumes intra-cluster routing within the cluster without any change in load.

Action. An action a_t ($a_t \in A$) at state s_t is selecting a node (customer) to visit. Given $F_{clu} = 0$, indicating all customers in the current cluster are visited, the sequencing agent selects an (inbound) customer from unserved clusters to determine the next cluster to visit. $F_{clu} = 1$ indicates that there are unvisited customers remaining within the current cluster, so the routing agent continues selecting customers to serve within the cluster. Node selection is sampled based on the output of policy p at state s_t. We employ masking to avoid infeasible nodes to be selected. For example, served clusters and those with demand exceeding the current load Q_t are masked before taking action at state s_t.

Transition. After an action, the transition appends the selected node to the current partial solution P_t. If no nodes remain in the current cluster after appending, F_{clu} is converted to 0, indicating the sequencing agent will select a customer (cluster) at state s_{t+1}. Conversely, $F_{clu} = 1$ remains, indicating that the routing continues within the cluster. Additionally, if the action is made by the sequencing agent, $Q_{t+1} = Q_t - d_k$ is updated to deduct the demand of the selected cluster from vehicle load, so that $s_{t+1} = \langle P_{t+1}, Q_{t+1} \rangle$.

Reward. After constructing a complete route, we calculate its total length $C_r(\pi)$ and define the reward function as $R(\pi) = -C_r(\pi)$. The reward encourages two agents to improve customer selections, resulting in shorter routes.

Policy. The probability of constructing a complete solution is represented by a chain rule $p(a|s) = \prod_{t=0}^{T-1} p(a_{t+1}|a_{0:t}, s_t)$ At each constructive step, the policy infers probabilities for selecting nodes, which are used to iteratively sample the route $a = \{a_0, a_1, \ldots, a_T\}$, where a_0 represents the depot node and T is the number of steps to complete the solution.

Individual perspective. Despite the coupled representation above, the two agents have peculiarities in their own MDPs, featuring different states, action spaces, and transitions. Sequencing agent selects nodes from unserved clusters at any outbound node, while routing agent is always focused on nodes in every single cluster. Their transitions are asynchronous, occurring at different step intervals. To account for this asynchrony, we design a policy network capable of distinguishing agents' policies at varying step intervals, aiming to improve solution performance.

To the best of our knowledge, such an interactive MDP formulation has not been previously proposed in the VRP domain. However, it is critical to provide a more informative and comprehensive framework for modeling the class of clustered VRPs. To embed to flag in the neural network, we propose dual decoders to act as the two agents executing two types of heterogeneous actions, i.e., intra-cluster sequencing and inter-cluster routing, which are elaborated in the following section.

4.2. Policy network

We outline the parameterization of policy π using an encoder-decoder-based neural network π_θ, which can solve different CluVRPs. To learn policies for both inter-cluster and intra-cluster decisions, we propose the cluster-aware attention module in the encoder to involve customer-level interactions, and come up with inter-cluster and intra-cluster decoders to customize policies for sequencing and routing, respectively. The entire architecture of policy network is illustrated in Fig. 2.

4.2.1. Cluster-aware encoder

The encoder consists of an input encoding module and L cluster-aware attention modules. The input encoding module is used to map raw features of the depot and customer nodes into d_h-dimension initial embeddings that preliminarily capture the relationship between the depot and customers. The L cluster-aware attention modules share an identical structure while maintaining independent parameters. In each cluster-aware attention module, we use a self-attention layer to capture information transmission between customer nodes. Further, we propose a cross-attention layer to parallelly capture the information transmission between customers in each cluster and those in the other clusters.

Input encoding module. The input encoding module takes features of nodes $\{v_0, v_1, \dots, v_n\}$ as input. As described in Section 3, the nodes $\{0, 1, \dots, n\}$ in a CluVRP instance represent the depot and customers. We define the feature by $v_0 = (x_0, y_0, z_0)$ for the depot, and $v_i = (x_i, y_i, z_i, x_C^i)$ for the remaining (customer) nodes with $i > 0$. In particular, z_i denotes the set of augmented coordinates of (x_i, y_i), which are derived from invariant transformations detailed in Section 4.3. x_C^i represents the center coordinate of the cluster, to which node i belongs. Given the node features, we first adopt two independent linear layers to map the depot and customers into embeddings $e^{de}, e_i^{cu} \in \mathbb{R}^{1\times d_h}$, respectively. We also linearly project the average demand of each customer, i.e., $d_k/|C_k|$, into demand embedding that is added to the node embeddings of customers. Afterward, we adopt a cross-attention layer to capture the relationship between the depot and customers. In particular, we use e^{de} and e_i^{cu} to generate the *query* $\mathbf{Q}_0^e$ and *key-value* pairs $(\mathbf{K}_i^e, \mathbf{V}_i^e)$, respectively, which are formulated as follows:

$$\mathbf{Q}_0^e = W_q^e e^{de},\ \mathbf{K}_i^e = W_k^e e_i^{cu},\ \mathbf{V}_i^e = W_v^e e_i^{cu} \tag{2}$$

where W_q^e, W_k^e, $W_v^e \in \mathbb{R}^{d_h\times d_h}$ are trainable matrices. Then, the cross-attention is executed by:

$$\alpha_i^e = \frac{\mathbf{Q}_0^e(\mathbf{K}_i^e)^\top}{\sqrt{d_h}},\quad \forall i \in \{1, \dots, n\} \tag{3}$$

$$h_i^0 = \alpha_i^e \mathbf{V}_i^e,\ \ \forall i \in \{1, \dots, n\} \tag{4}$$

In the implementation, we apply multi-head version of cross-attention to obtain the node embedding h_i^0 by $h_i^0 = [h_{i,1}^0, \dots, h_{i,\mathcal{H}}^0]$, where $[,]$ denotes concatenation, and $h_{i,j}^0 \in \mathbb{R}^{d_h/\mathcal{H}}$ represents the embedding from the ηth head parameterized by $W_q^{e,\eta}$, $W_k^{e,\eta}$, $W_v^{e,\eta} \in \mathbb{R}^{d_h\times d_h/\mathcal{H}}$

Cluster-aware attention module. After the input encoding module, we further evolve node embeddings h_i^0 by the attention module. To capture information transmission between all nodes (depot and customers), the self-attention layer is firstly applied following Vaswani et al. (2017), which is similar to the aforementioned cross-attention, but enables interactions between all pairs of nodes. Concretely, we calculate the dot product of *query* and *key* for every node pair, perform scaling and softmax operation over all dot products, and multiply them with *values* of nodes. The *query*, *key* and *value* are derived by trainable matrices in different heads.

Unlike other VRPs, CluVRPs are characterized by unique cluster-related constraints, which largely influence the inter-cluster sequencing optimization. In this light, we design a cluster-aware cross-attention layer to capture the relationships between customers in each cluster and those in the other clusters (here the depot is viewed as a cluster). In this cross-attention layer, the calculation of *query*, *key*, and *value* is similar to the cross-attention in the input encoding module, such that:

$$\alpha_{i,j}^{c,\eta} = \frac{1}{\sqrt{d_h}}(W_q^{c,\eta} h_i^{l-1})(W_k^{c,\eta} h_j^{l-1})^\top,\ \ \forall i, j \in \{0, 1, \dots, n\} \tag{5}$$

where $W_q^{c,\eta}, W_k^{c,\eta} \in \mathbb{R}^{d_h\times d_h/\mathcal{H}}$ are trainable *query* and *key* matrices for head η ($\eta \in \{1, \dots, \mathcal{H}\}$).

Before applying softmax operation to $\alpha_{i,j}^{c,\eta}$, we employ a cluster masking scheme to filter out the nodes belonging to the same cluster as each node, thus differentiating each cluster from the other clusters. The masking is formulated as:

$$\hat{\alpha}_{i,j}^{c,\eta} = \begin{cases} \alpha_{i,j}^{c,\eta}, & \forall i, j \in \{0, \dots, n\} \text{ with } C_i \neq C_j \\ -\infty, & \forall i, j \in \{0, \dots, n\} \text{ with } C_i = C_j \end{cases} \tag{6}$$

by which we can interact nodes in a cluster with the other clusters in parallel, without introducing any attention computation overhead. Given the masked attention scores $\{\hat{\alpha}_{i,j}^{c,\eta}\}_{j=0}^{n}$, they are normalized to $\{\tilde{\alpha}_{i,j}^{c,\eta}\}_{j=0}^{n}$ over the index j using the softmax function, which are then used to aggregate *values* in every head:

$$h_{i,\eta}^{c} = \sum_{j=0}^{n} \tilde{\alpha}_{i,j}^{c,\eta}(W_{v}^{c,\eta} h_{j}^{l-1}), \quad \forall i, j \in \{0, 1, \ldots, n\} \tag{7}$$

where $W_{v}^{c,\eta} \in \mathbb{R}^{d_h \times d_h/\mathcal{H}}$ are trainable *value* matrices. Then, we concatenate $h_{i,\eta}^{c}$ in different heads to obtain the updated d_h-dimension node embeddings $h_i^c = \left[h_{i,1}^{c}, \ldots, h_{i,\mathcal{H}}^{c}\right]$. While some prior work has addressed VRPs with two types of nodes (Li et al., 2021a), we did not find any neural architectures designed for multiple clusters of nodes in the current VRP literature. Accordingly, the cluster-aware attention we propose is broadly applicable to VRPs with clustered structures.

Next, we concatenate the node embeddings from the self-attention h_i^s and cross-attention layers h_i^c, and pass them through skip connection, instance normalization (Norm), multilayer perceptron (MLP) to obtain the output of the lth attention module:

$$h_i = \left[h_i^c, h_i^s\right] \tag{8}$$

$$h_i^{l'} = \mathrm{Norm}(h_i^{l-1} + h_i) \tag{9}$$

$$h_i^{l} = \mathrm{Norm}(\mathrm{MLP}(h_i^{l'}) + h_i^{l'}) \tag{10}$$

where Norm() represents instance normalization and The MLP has one GELU-activating hidden layer with $2d_h$ hidden dimensions and d_h output dimensions.

We iterate the cluster-aware attention module L times. After obtaining the node embeddings $\{h_i^L\}_{i=0}^{n}$ from the final attention module, we average embeddings of nodes within each cluster to derive cluster embeddings $\{h_{C_k}^L\}_{k=0}^{K}$.

4.2.2. Inter- and intra-cluster decoders

To distinguish policies between the two agents in the interactive MDPs, we design two decoders for calculating probabilities of selecting customers for sequencing and routing decisions, respectively. In particular, we use different context embeddings to customize the agents' policies. The context embedding of intra-cluster decoder is represented as:

$$h_t^{(c_1)} = [h_{\pi_1}^L, h_{\pi_{t-1}}^L] \tag{11}$$

where $h_{\pi_1}^L$ is the embedding of the first node visited after the sequencing agent enters a cluster, and $h_{\pi_{t-1}}^L$ is the embedding of the previous node visited. The context embedding of the inter-cluster decoder is represented as:

$$h_t^{(c_2)} = \left[h_{C_{t-1}}^L, h_{\pi_{t-1}}^L\right] \tag{12}$$

where $h_{C_{t-1}}^L$ represents the embedding of the recently visited cluster.

After the context embedding, the two decoders, sharing identical structures but taking independent parameters, are used to infer the probabilities. We first use multi-head attention to aggregate information from multiple nodes and single-head attention to generate the selection probabilities for each node. Notably, we pass every state through two decoders simultaneously to generate two distinct probability distributions. We use the aforementioned flag F_{clu} in the interactive MDPs to determine which probability distribution should be selected for constructing the solution at the current step. This approach facilitates batch processing, enabling both agents to simultaneously interact with the environment and use different decoders for node selection, across a batch of instances that may be under different states. In the policy network, we apply the masking scheme to enforce the hard constraints imposed by cluster demands, e.g., masking clusters whose demands exceed the remaining vehicle capacity, to prevent infeasible decisions. In addition, we employ cluster-aware cross-attention to capture demand interactions across all clusters. Together, demand-based masking and demand-aware encoding enable the model to effectively represent inter-cluster demand relationships.

4.3. Training procedure

We employ REINFORCE algorithm to train the policy network by using the objective values of the constructed solutions. For maximizing the expected reward (the negative objective value), the training loss is defined as below,

$$\mathcal{L}^{\theta} = \mathbb{E}_{G\sim\mathcal{G}, \pi\sim p_\theta(\pi|G)}[C_r(\pi)], \tag{13}$$

where $C_r(\pi)$ denotes the objective value (i.e., the total length) of the route π. The policy network is updated to optimize the loss $\mathcal{L}^{\theta}$ via gradient descent (Sutton, 2018), as below,

$$\nabla\mathcal{L}^{\theta} = \mathbb{E}_{G\sim\mathcal{G}, \pi\sim p_\theta(\pi|G)}[C_r(\pi) - \tilde{C}_r(G)]\nabla_\theta p_\theta(\pi|G), \tag{14}$$

where $\tilde{C}_r(G)$ is used to reduce the variance of the estimated gradients. Consequently, we empirically calculate the approximate gradients over batches of M instances, as below,

$$\nabla\mathcal{L}^{\theta} \approx \frac{1}{Mk}\sum_{i=1}^{M}\sum_{j=1}^{k}[C_r(\pi_i^j) - \tilde{C}_r(G)]\nabla_\theta \log p_\theta(\pi_i^j|G_i) \tag{15}$$

Table 1
Results for four scenarios with different numbers of customers and clusters.

Method	N20C3			N50C6			N100C12			N150C18		
	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
CPLEX	2.744	0.20%	14d	–	–	–	–	–	–	–	–	–
LKH3	2.753	0.52%	11h	5.366	1.41%	31h	10.540	9.11%	46h	14.193	3.8%	134h
Two-level VNS	**2.739**	**0.00%**	33m	**5.291**	**0.00%**	3h	9.669	0.10%	10h	13.695	0.17%	14h
HGA	2.788	1.81%	28h	5.574	5.35%	51h	11.035	14.24%	9d	16.276	19.04%	14d
Nearest Neighbor	2.985	9.02%	9s	6.027	13.90%	17s	11.213	16.09%	2.0m	16.037	17.29%	4.5m
Nearest Insertion	3.017	10.16%	13s	6.113	15.53%	43s	11.423	18.25%	2.0m	16.377	19.78%	4.8m
Farthest Insertion	3.159	15.36%	13s	5.762	8.91%	43s	11.926	23.46%	2.2m	17.041	24.63%	2.5m
AM	2.765	0.97%	0.1m	5.343	0.98%	0.4m	9.754	0.98%	2.5m	13.809	0.99%	5.0m
POMO	2.744	0.18%	0.2m	5.303	0.22%	0.5m	9.690	0.32%	2.7m	13.716	0.32%	5.4m
Our approach	**2.739**	**0.00%**	0.2m	**5.291**	**0.00%**	1.0m	**9.659**	**0.00%**	3.2m	**13.672**	**0.00%**	9.8m

where $\tilde{C}_r(G) = \frac{1}{Mk}\sum_{i=1}^{M}\sum_{j=1}^{k} C_r(\pi_i^j)$ represents the average objective value over Mk instances (k sampled solutions for each of the M instances). The parameters of policy network comprise parameters of the encoder θ^{en}, and parameters of two decoders θ^{inter}, θ^{intra}. While the encoder parameters θ^{en} are used and updated at all steps in a route construction, the parameters θ^{inter} and θ^{intra} are updated at specific steps in accordance with the interactive MDPs.

Data augmentation. We apply similar instance augmentation as in Kwon et al. (2020). Since a Euclidean CluVRP instance can be represented by different graphs sharing the same optimal solution, we augment instances in the training process. Given coordinates (x_i, y_i) in a CluVRP instance, we transform them by: $(x', y') = (x, y), (y, x), (x, 1-y), (y, 1-x), (1-x, y), (1-y, x), (1-x, 1-y), (1-y, 1-x)$. These transformations retain the invariant optimal solutions for an instance. We apply the transformations to each instance, so that for each node, we input all transformed coordinates, along with the other features described in Section 4.2, to the policy network. Through the augmentation, better solutions for instances can be explored to encourage training progress toward achieving a better policy. Unlike (Kwon et al., 2020), which uses augmented instances during inference, we found that augmentation during the training process delivers greater performance improvements and does not introduce additional runtime overhead for testing multiple transformations on each instance.

5. Experiments

We evaluate the method with both synthetic and benchmark datasets by comparing with state-of-the-art algorithms. The effects of methodological components are empirically verified by ablation study. We also consider a dynamic CluVRP and generalized VRP in our evaluation, where our method can be effectively applied to the different problem variants.

5.1. Experimental setup

Following the synthetic data generation in DRL research (Kool et al., 2018; Kwon et al., 2020), we randomly generate coordinates of customers and demands of clusters in a planar region $[0, 1]^2$, and nodes are assigned to clusters using the K-Means algorithm. We predefined four training scenarios with instances configured with 20 customers in 3 clusters (N20C3), 50 customers in 6 clusters (N50C6), 100 customers in 12 clusters (N100C12), and 150 customers in 18 clusters (N150C18). We train the policy network in each scenario with 3000 epochs. 10,000 instances are generated in each epoch. We set the batch size to $M = 1000$. We apply Adam optimizer with the initial learning rate 1e-4. Our policy network is configured by $d_h = 128$, $L = 6$ and $\mathcal{H} = 8$.

We evaluate the trained policy network by using 10,000 synthetic instances in each scenario, which are generated in the same manner as the training instances. We also conduct the evaluation with the benchmark dataset provided in Battarra et al. (2014). In each scenario, we record objective values and runtimes for 10,000 instances to calculate the average objective value, average gap and total runtime. Given an instance, the gap for a method is calculated by comparing its objective value with the best found objective value among all methods. We conduct all experiments with an Intel Xeon Platinum 8358 at 2.60 GHz and an NVIDIA A40 GPU.

We select the baseline methods primarily based on their methodological advances and the availability of accessible code implementations. Specifically, we compare our method with **CPLEX** (v22.1.1), a widely used commercial solver that provides exact solutions to combinatorial optimization problems (Cplex, 2009); **LKH3**, a widely recognized and high-performing heuristic solver for vehicle routing problems (Helsgaun, 2017), which among the few solvers has included an implementation for the CluVRP; **two-level VNS**, the state-of-the-art metaheuristic for the CluVRP that combines multiple destruction and repair operators within a two-level variable neighborhood search framework, and was implemented by C++ (Hintsch and Irnich, 2018); hybrid genetic algorithm (**HGA**) in Vidal et al. (2015) that is enhanced by efficient large neighborhood search procedures based on re-optimization techniques. While HGA has no released code, we reproduced the algorithm following the descriptions in the original paper using the hybrid genetic search-based solver package.[1] In addition, our DRL approach learns constructive policies, and therefore the traditional constructive

[1] https://github.com/PyVRP/PyVRP/tree/main

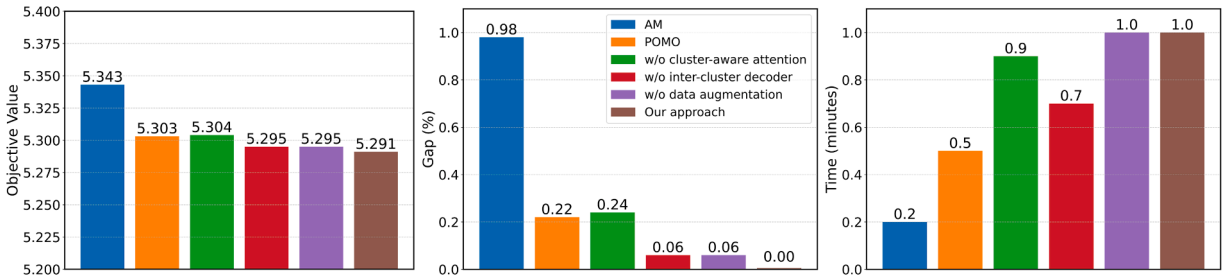

Fig. 3. Ablation effects of methodological components vs. DRL-based constructive heuristics.

Table 2
Results for benchmark instances with different methods.

Instance	LKH3		Two-level VNS		HGA		Nearest Neighbor		Nearest Insertion		Farthest Insertion		POMO		Ours	
	Obj.	Time(s)	Obj.	Time(s)	Obj.	Time(s)	Obj.	Time(s)	Obj.	Time(s)	Obj.	Time(s)	Obj.	Time(s)	Obj.	Time(s)
B-N78-C10	**8468**	8.24	8681	1.46	9001	24.16	9689	0.01	8780	0.01	10,956	0.01	8726	0.15	8675	0.27
X-N101-C13	11,628	14.26	11,639	0.33	12,721	36.64	13221	0.01	13,474	0.01	14,581	0.01	11,559	0.21	**11451**	0.32
X-N106-C14	18,199	22.66	18,354	3.89	19,104	33.72	20,289	0.01	20,646	0.01	18,570	0.01	18,113	0.21	**17573**	0.38
X-N134-C17	**18868**	33.55	19,758	11.90	22,808	40.54	22,656	0.01	23,674	0.01	24579	0.01	19,370	0.37	19,293	0.62
X-N181-C23	**14595**	36.77	15,019	0.03	19,317	64.22	18,099	0.03	18,965	0.02	21,995	0.03	15,037	0.37	15,002	0.68
X-N228-C29	**16529**	95.34	16,769	1.92	23,696	82.06	20,654	0.03	22,465	0.03	22,019	0.03	17,093	0.41	16,763	0.74
X-N233-C30	**18386**	69.38	18,994	27.33	24,867	68.87	22,864	0.03	21,971	0.03	24,503	0.04	18,740	0.46	18,673	0.75
X-N359-C45	48,385	311.93	47,121	22.65	59,042	125.04	59,072	0.02	58455	0.02	58,711	0.10	46,613	0.96	**46665**	1.68
X-N384-C49	48,384	471.71	48,578	44.41	55,880	120.56	55,702	0.07	56,758	0.08	58,216	0.10	48,045	1.11	**48289**	1.88

heuristics and DRL-based constructive heuristics are also compared, including **nearest neighbor, farthest insertion, nearest insertion** (Gendreau et al., 2008), the attention model (**AM**) (Kool et al., 2018) and its enhanced version by using policy optimization with multiple optima (**POMO**) (Kwon et al., 2020).

5.2. Comparative results

We tested all methods on instance sets generated in the same manner as training sets. From the results in Table 1, we observe an obvious advantage of our approach over the other methods. Given the relatively short runtime, our approach consistently attains the best objective values and gaps for different instance sets. While the two-level VNS is competitive on the N20C3 instance set, achieving the same results as our approach, its performance gradually declines as the problem complexity increases with more customers and clusters. In general, the metaheuristics HGA and two-level VNS surpass the constructive heuristics, but consume much more runtime. AM and POMO substantially improve constructive solution performance, achieving small gaps with short runtime. However, they still bare clear gaps to two-level VNS and our approach. With a bit more runtime than AM and POMO, our approach significantly outperforms two-level VNS, the state-of-the-art algorithm for the CluVRP. Regarding the exact solver CPLEX, we observe it struggles to cope with the smallest instances, and cannot gain the optimal solutions in reasonable time. Therefore, we only apply it on the N20C3 instance set, and configure a time limit 2m to prevent prohibitively long solving time for all 10,000 instances. The specialized VRP solver LKH3 outperforms HGA and traditional constructive heuristics, yet it still falls short of our approach. We summarize that our approach learns effective constructive policies to significantly improve the performance over the traditional exact and heuristic solutions, as well as the current DRL-based constructive heuristics.

5.3. Ablation study

For verifying the effects of components in our approach, we conducted the ablation study on the N20C3 instance set as used in Section 5.2. We removed the cluster-aware cross-attention, inter-cluster decoder, and data augmentation from our approach, respectively, and trained three corresponding policy networks. We test the policy networks and gather results in Fig. 3. It is clear that without any of three components, the performance declines to some extent. Among them, removing the cluster-aware cross-attention yields the worst results, even causing larger objective value and gap than POMO. The effects of the inter-cluster decoder and data augmentation are on par with each other, leading to a weaker performance decline than that caused by the cluster-aware cross-attention. Removing a component from the encoder or decoder has little influence on the runtime, with only a negligible reduction. The data augmentation used during training does not introduce additional runtime during testing.

5.4. Benchmarking results

We evaluate our approach on benchmark instances by applying the policy network trained with the N150C18 instance set. Specifically, following prior studies (Freitas et al., 2023; Hintsch and Irnich, 2018), we draw benchmark instances from CVRPlib (Uchoa

Table 3
Results for four generalized VRP scenarios, each of which includes 10,000 test instances.

Method	N20C7			N50C15			N100C30			N150C50		
	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
CPLEX	4.145	0.18%	14d	5.252	0.23%	14d	7.62	7.50%	14d	12.195	16.84%	14d
HGA	4.189	1.24%	29h	7.421	41.60%	57h	15.116	113.15%	6d	25.926	148.40%	15d
Nearest Neighbor	4.428	7.02%	5s	6.047	15.39%	19s	8.628	21.66%	1.0m	12.689	21.57%	2.5m
Nearest Insertion	4.357	5.30%	5s	5.865	11.91%	22s	8.402	18.47%	1.3m	12.503	19.79%	3.4m
Farthest Insertion	4.449	7.52%	1.3m	6.064	15.70%	1.6m	8.534	20.33%	2.6m	12.537	20.12%	5.7m
AM	4.152	0.33%	0.1m	5.281	0.76%	0.2m	7.210	1.67%	0.7m	10.504	0.64%	2.0m
POMO	**4.138**	**0.00%**	0.2m	**5.241**	**0.00%**	0.3m	7.100	0.11%	0.7m	10.451	0.13%	2.1m
Our approach	4.140	0.05%	0.2m	5.242	0.02%	0.4m	**7.092**	**0.00%**	0.9m	**10.438**	**0.00%**	2.7m

Table 4
Generalization results on instances with different numbers of clusters. Each set contains 10,000 instances.

Method	N100C20			N100C25			N100C30			N100C35			N100C40		
	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
HGA	9.804	84.6%	8h	12.469	99.57%	8h	15.116	113.15%	8h	17.867	129.38%	8h	20.494	132.13%	8h
Nearest Neighbor	6.363	19.79%	1.1m	7.571	21.17%	1.4m	8.628	21.66%	1.4m	9.713	24.69%	1.5m	10.725	21.48%	1.9m
Nearest Insertion	6.157	15.92%	1.3m	7.348	17.60%	1.4m	8.402	18.47%	1.6m	9.512	22.11%	1.9m	10.533	19.30%	1.9m
Farthest Insertion	6.354	19.62%	1.2m	7.516	20.29%	1.3m	8.534	20.33%	1.8m	9.589	23.11%	1.8m	10.543	19.41%	1.9m
AM	5.411	1.86%	1.5m	6.355	1.70%	1.6m	7.210	1.67%	1.9m	8.139	4.48%	2.1m	8.999	1.92%	2.0m
POMO	5.321	0.17%	1.9m	6.254	0.08%	2.0m	7.100	0.11%	2.9m	8.004	2.76%	2.3m	8.845	0.18%	2.5m
Our approach	**5.312**	**0.00%**	2.4m	**6.248**	**0.00%**	3.4m	**7.092**	**0.00%**	2.7m	**7.789**	**0.00%**	2.7m	**8.829**	**0.00%**	2.8m

et al., 2017) and extend them to CluVRP instances. To incorporate a wider variation in the number of nodes, we primarily adopt instances from the Set-X dataset.[2] The CVRPlib instances are transformed into CluVRP instances following the method proposed in Bektaş et al. (2011). We select the instances with distinct numbers of customers (ranging from 78 to 384) and clusters to cover different CluVRP scenarios. The results in Table 2 showcase the advantage of our approach in the instances substantially different from the training set. The objective values derived from our approach are consistently smaller than the other methods. We observe that POMO and two level VNS generally outperform HGA and constructive heuristic methods. Compared to POMO, two-level VNS retains its superiority on the instances with fewer customers and clusters. For the complex instances X-N233-C30, X-N359-C45, and X-N384-C49, POMO achieves better solutions with a very short runtime. It indicates the potential of DRL-based methods in efficiently solving CluVRP. However, POMO, which follows the existing DRL framework for VRPs, cannot effectively incorporate cluster-related information into the policy learning process. In contrast, our approach synergizes the optimization of both inter-cluster sequencing and intra-cluster routing in CluVRP, leading to more effective solutions.

5.5. Generalized VRP

To demonstrate the versatility of our approach, we also evaluate it on the Generalized VRP, a variant of the CluVRP, with the definition provided in Appendix A.1. The experiments and analysis are presented in the following subsections.

5.5.1. Comparison with baselines

Our approach can be easily adapted to generalized VRP. We remain most parts of our approach and only revise the masking scheme in the training and test. That is, after a customer in a cluster is selected, all customers in the cluster are masked and cannot be selected any more. The customers in the unserved clusters are available for selection at each step of the solution construction process. We train the proposed policy network with the instances generated in the same manner as the instances of the classic CluVRP. The training setting of generalized VRP also remains the same as classic CluVRP, as described in Section 5.1. We generate 10,000 instances for testing and gather the results of our approach and baselines in Table 3. Note that the two-level VNS is a specialized state-of-the-art algorithm for the classic CluVRP, but it is not applicable to the generalized VRP. Therefore, we exclude it from our comparison. As observed from the table, although our approach slightly underperforms POMO in the instance sets N20C7 and N50C15, our approach obtains the best results for the instance sets N100C30 and N150C50 with more customers and clusters. Compared to the other methods, our approach is significantly superior in terms of solution quality and computational efficiency.

5.5.2. Generalization with different numbers of clusters

For verifying the generalization of the training policy network, we employ the policy trained for N100C30 to solve another five instance sets with four having different numbers of clusters, which are represented as N100C20, N100C25, N100C30, N100C35, and N100C40. From the results in Table 4, our approach is still viewed as the best for all instance sets, which indicates the superior

[2] http://vrp.galgos.inf.puc-rio.br/index.php/en/

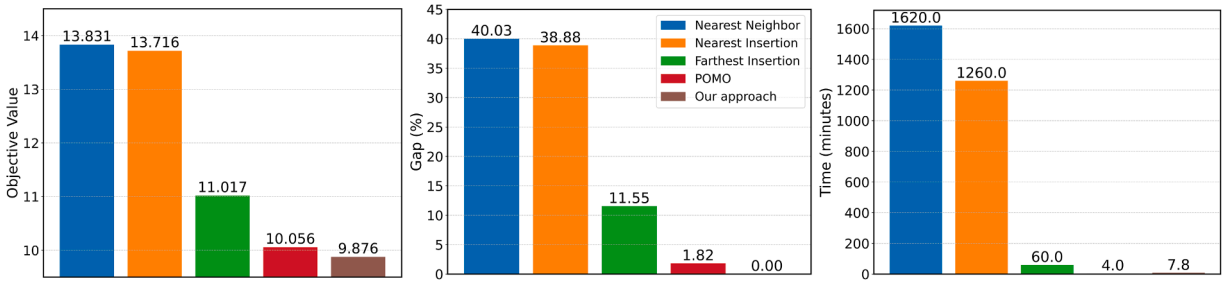

Fig. 4. Results of constructive heuristics for dynamic CluVRP.

Table 5
Results for three SoftCluVRP scenarios, each of which includes 10,000 test instances.

Method	N20C3			N50C6			N100C12		
	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
CPLEX	**2.029**	**0.00%**	56h	–	–	–	–	–	–
Two-level GA	2.077	2.39%	14h	3.658	0.16%	14h	6.709	0.03%	28h
Memetic Search	2.036	0.35%	3.6h	3.701	1.34%	8.9h	6.804	1.45%	14h
Nearest Neighbor	2.103	3.68%	27s	3.997	9.44%	80s	7.767	15.80%	3.3m
Nearest Insertion	2.096	3.29%	23s	3.977	8.89%	70s	7.739	15.39%	3.3m
Farthest Insertion	2.126	4.79%	24s	4.097	12.18%	3.3m	8.156	21.60%	6.7m
Our approach	2.033	0.20%	0.2m	**3.652**	**0.00%**	0.9m	**6.707**	**0.00%**	3.5m

generalization performance of our policy network when applied to generalized VRP with different numbers of clusters. We summarize from Tables 3 and 4 that our approach performs well for the generalized VRP, showcasing advantageous solutions for various instances.

5.6. Dynamic CluVRP

Our approach learns the constructive policy for solving the CluVRP. One advantage of (learned or traditional) constructive policies is that they can progressively select customers step by step to build the solution, without requiring heavy computation when dynamics are updated. Therefore, one natural extension of our DRL approach is its application to the dynamic CluVRP, in which customers in clusters appear in real-time rather than being known beforehand. The definition of dynamic CluVRP is detailed in Appendix A.2. To validate the applicability, we assume that customers randomly appear at the constructive steps of solutions, following the Poisson process. Without loss of generality, we assume the process has rate 1. For the other settings, including customer coordinates and demands, they are the same as those used in Section 5.1. Since metaheuristics are not applicable in this dynamic setting, we only compare our approach with traditional and DRL-based constructive heuristics. As seen in Fig. 4, our approach attains the lowest objective value and gap among all constructive solutions with relatively short runtime.

5.7. Soft-clustered VRP

In this work, our policy network is designed according to CluVRPs. Specifically, the inter-cluster and intra-cluster decoders are used to differentiate the sequencing decisions between clusters and routing decisions in each cluster. However, we attempt to adapt our model for solving soft-clustered VRP (SoftCluVRP), with the definition provided in Appendix A.3. In this problem, the routing is allowed between clusters, thus making the sequencing of clusters less important. To adapt our model, we remove the inter-cluster decoder and modify the intra-cluster decoder such that, when computing the query vector, the embedding $h^L_{\pi_1}$ corresponds to the first customer in the current route (i.e., the node immediately following the most recent depot visit).

We generate the instances in the same way as explained in Section 5.1. We compare the trained model with the two-level genetic search algorithm in Latorre (2025) and the memetic search approach in Zhou et al. (2023). As shown in Table 5, while the two metaheuristics perform better than construction heuristics, our approach outperforms all baselines on N50C6 and N100C12 instances. The exact solver CPLEX achieves the optimal solutions for N20C3 instances but fails to obtain a feasible solution within 3600s on most instances in N50C6 and N100C12 scenarios. Since our approach learns a construction heuristic without relying on complex heuristic searches, it is significantly more time-efficient than exact and metaheuristic methods. We also compare the methods on some benchmark instances used in Latorre (2025), and report the results in Table 6. We directly use the released C++ implementation of the two-level GA and maintain a runtime comparable to that of our approach. For the memetic search, we reimplement the algorithm in Python based on the description in the paper, as no official code is available. We set the number of genetic operations to 80 and perform 200 iterations of neighborhood search per run. It is clear that our model is generally superior to the baseline methods across all instances, achieving the best solutions for most instances. Note that we directly test the model trained on synthetic data on the benchmark instances. To further enhance performance, we plan to integrate the learned policy with other metaheuristics, using the solutions of our approach as efficient starting points for additional improvement. We will explore this direction in future work.

Table 6
Results for SoftCluVRP benchmark instances with different methods.

Instance	Two-level GA		Memetic Search		Nearest Neighbor		Nearest Insertion		Farthest Insertion		Our approach	
	Obj.	Time(s)	Obj.	Time(s)	Obj.	Time(s)	Obj.	Time(s)	Obj.	Time(s)	Obj.	Time(s)
Golden_1-C35-N241	**4703**	10	4848	48	5517	0.02	6393	0.02	5983	0.01	4809	10.70
Golden_2-C54-N321	7524	10	7672	122	8941	0.05	9281	0.04	9263	0.02	**7496**	10.60
Golden_2-C65-N321	7740	10	7760	114	9171	0.05	9037	0.05	9453	0.02	**7669**	10.60
Golden_3-C58-N401	**10419**	10	10,488	345	13,068	0.09	13,178	0.08	12,391	0.03	10,517	10.70
Golden_4-C97-N481	14,173	10	**13556**	795	15,537	0.15	16,796	0.13	16,807	0.04	14,517	10.80
Golden_5-C19-N201	6904	10	7148	13	8399	0.01	9194	0.01	8966	0.01	**6886**	10.40
Golden_5-C21-N201	6794	10	7256	10	8265	0.01	8822	0.01	9621	0.01	**6774**	10.40
Golden_5-C29-N201	6725	10	7392	17	9212	0.01	8686	0.01	8811	0.01	**6705**	10.40
Golden_7-C61-N361	9984	10	10,204	283	12,685	0.08	12,276	0.07	11,511	0.02	**9808**	10.70
Golden_14-C65-N321	699	10	712	110	860	0.05	857	0.05	891	0.02	**689**	10.60
Golden_15-C34-N397	**837**	10	892	179	1068	0.08	1011	0.07	1047	0.02	**837**	10.70
Golden_15-C50-N397	856	10	896	82	1137	0.08	1056	0.07	1083	0.02	**832**	10.70
Golden_16-C97-N481	1060	10	1088	243	1378	0.12	1308	0.12	1445	0.04	**1050**	10.90
Golden_17-C41-N241	**392**	10	418	99	598	0.03	515	0.03	483	0.01	415	10.50
Golden_18-C61-N301	**578**	10	623	201	776	0.04	712	0.04	748	0.01	592	10.50
Golden_19-C73-N361	**778**	10	830	259	1089	0.06	961	0.06	1005	0.02	789	10.70
Golden_20-C53-N421	1060	10	1061	186	1470	0.07	1270	0.08	1309	0.03	**1047**	10.80

6. Conclusion

We present a DRL approach to solving CluVRPs that synergizes inter-cluster sequencing and intra-cluster routing with a single policy network. By modeling CluVRPs through interactive MDPs, we effectively capture the intertwined decision-making dynamics of two agents and employ a novel encoder-decoder-based neural network to customize their respective policies. Our approach is validated across CluVRP variants, showing its effectiveness for diverse instances with varying configurations. Our contributions underscore the potential of DRL-based methods in addressing cluster-related constraints in vehicle routing. In the future, we will extend the approach to solving large-scale problems and other CluVRP variants. We will also study the generalization behavior of the policy network along various dimensions of CluVRPs, including the distribution of demands and the number of clusters, and aim to develop explainable policies grounded in these dimensions.

CRediT authorship contribution statement

Yaoxin Wu: Supervision, Methodology, Conceptualization; **Yue Yu:** Validation, Methodology, Investigation; **Lingxiao Wu:** Writing – original draft, Supervision, Methodology; **Tao Feng:** Writing – review & editing, Resources, Data curation; **Lu Zhang:** Supervision, Project administration, Data curation; **Zhenkun Wang:** Writing – original draft, Resources, Methodology; **Jie Gao:** Writing – review & editing, Writing – original draft, Supervision, Project administration.

Data availability

Our code and data are available at https://github.com/yyue99/CluVRP-RL.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work is supported by the National Natural Science Foundation of China under grant No. 52472461 from the Second Research Institute of Civil Aviation Administration of China. It is also supported by the National Natural Science Foundation of China [No 72501229], the Sichuan Science and Technology Program [No. 2026NSFSC1320], and the Fundamental Research Funds for the Central Universities [No. 2682025CX185].

Appendix A. Definitions of CluVRP variants

A.1. Generalized VRP

Following the classic CluVRP (Sevaux et al., 2008), as described in Section 3.1 in the main text, the set of customers $V = \{1, \dots, n\}$ are grouped into disjoint clusters denoted by a set $C \in \{C_0, C_1, \dots, C_K\}$. Each customer $i = \{(x_i, y_i), C_i\}$ is assigned to exactly one

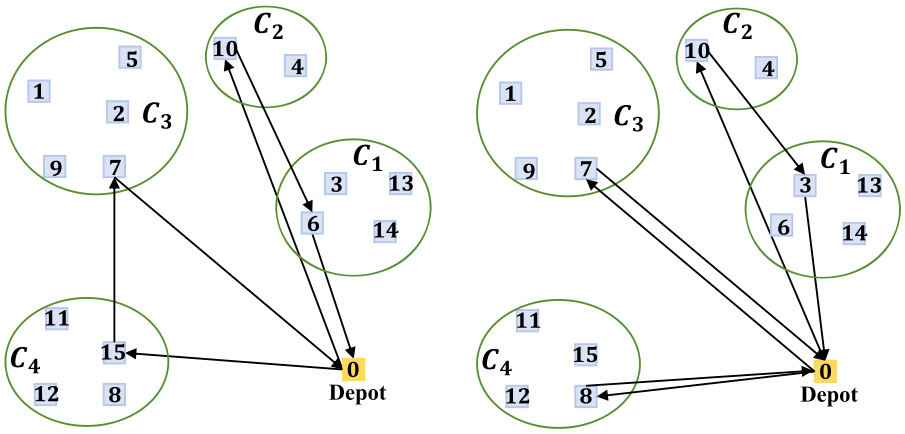

Fig. A.1. The illustration of two solutions for a generalized VRP instance comprising 15 customers in 4 clusters. Differing from the classic CluVRP we introduce in the main text, the generalized VRP is featured by exactly one customer visit in each cluster. In the two solutions, the left one is better due to the less travel cost of the route.

cluster $C_i \in C$, and d_k is a positive value denoting the total demand of customers in the kth cluster. The generalized VRP (Ghiani and Improta, 2000) is different from the classic CluVRP in the following aspect:

- The route should select one customer from each cluster to be visited. This visit fulfills the demand of all customers within the cluster. Therefore, visiting customer i consumes d_{C_i} from the route capacity Q

The Generalized VRP reduces to the classic CVRP when each cluster consists of just one customer. It has wide-ranging applications, such as deliveries to factories, shopping malls, or communities, where goods can only be delivered to one of several designated gates (Baldacci et al., 2010; Pop et al., 2012). We illustrate two solutions for a generalized VRP instance in Fig. A.1.

A.2. Dynamic CluVRP

The dynamic CluVRP is a variant of the classic CluVRP that involves dynamically appearing clusters (with demands) as new information available during the routing process. We do not assume all clusters are known prior to the application of our approach. Instead, clusters are dynamically appearing during the solution construction process. In specific, given half of the clusters known in prior, we apply our approach to construct a solution for these known clusters. The new clusters appear at any construction step following the Poisson distribution, when our approach can naturally involve the new customers and clusters in the policy network and thus the new customers will be selected at the next construction step. Except the dynamic setting, the other constraints and the objective function are not changed.

The metaheuristic and exact solutions are not suitable in the dynamic setting, since they need to re-optimize CluVRPs whenever new customers appear, inducing much more calculation time. Constructive heuristics are advantageous than metaheuristic and exact solutions due to the high efficiency. We compare our approach with traditional and DRL-based constructive heuristics in Section 5.6 in the main text, which has shown that our approach gains the best performance with relatively short runtime.

A.3. Soft-clustered VRP

Similar to the classic CluVRP, the soft-clustered VRP (SoftCluVRP) (Hintsch and Irnich, 2020) also partition the set of customers $V = \{1, \dots, n\}$ into clusters $C \in \{C_0, C_1, \dots, C_K\}$. However, unlike CluVRP, where all customers within a cluster must be served consecutively by the same vehicle, SoftCluVRP relaxes this constraint. In this problem, a vehicle is allowed to leave and re-enter a cluster while serving customers from the same cluster. This relaxation introduces more flexibility into route construction, enabling the algorithm to find potentially shorter or more balanced routes, especially when customer locations within clusters are spatially dispersed. The SoftCluVRP can model practical applications such as parcel delivery and freight transportation, and has therefore been widely studied in the literature (Sevaux et al., 2008; Zhou et al., 2023).

Appendix B. Mathematical model of CluVRP

Given a complete undirected graph $G = (V, E)$, with node set $V = \{0, 1, \dots, n\}$, we define depot $0 \in V$ and customers $N = V \setminus \{0\}$. Let $K = \{1, \dots, |K|\}$ be the set of identical vehicles, each with capacity $Q > 0$. For each edge $(i, j) \in E$, a nonnegative travel cost c_{ij} is given, and each customer $i \in N$ has demand $d_i > 0$. Customers are partitioned into clusters $\mathcal{R} = \{C_1, \dots, C_m\}$ that form a partition of N. Specially, the depot is the singleton cluster $C_0 = \{0\}$. For any subset $Z \subseteq V$ with $Z \neq \emptyset$ and $Z \neq V$, we define the cut-sets $\delta^+(Z) := \{(i, j) \in E : i \in Z, j \in V \setminus Z\}$ and $\delta^-(Z) := \{(i, j) \in E : i \in V \setminus Z, j \in Z\}$. Following (Expósito-Izquierdo et al., 2016),

the mathematical model of CluVRP can be defined as follows:

$$\min \sum_{(i,j)\in E} \sum_{k\in K} c_{ij}\, x_{ij}^k \tag{B.1}$$

$$\text{s.t.} \sum_{k\in K} y_i^k = 1, \qquad \forall i \in N, \tag{B.2}$$

$$\sum_{k\in K} y_0^k = |K|, \tag{B.3}$$

$$\sum_{j\in V\setminus\{i\}} x_{ij}^k = \sum_{j\in V\setminus\{i\}} x_{ji}^k = y_i^k, \qquad \forall k \in K,\ \forall i \in V, \tag{B.4}$$

$$\sum_{i\in V} d_i\, y_i^k \le Q, \qquad \forall k \in K, \tag{B.5}$$

$$\sum_{(i,j)\in \delta^+(Z)} x_{ij}^k \le y_h^k, \qquad \forall k \in K,\ \forall Z \subseteq N,\ \forall h \in Z, \tag{B.6}$$

$$\sum_{(i,j)\in \delta^+(C_r)} \sum_{k\in K} x_{ij}^k = \sum_{(i,j)\in \delta^-(C_r)} \sum_{k\in K} x_{ij}^k = 1, \qquad \forall C_r \in \mathcal{R}, \tag{B.7}$$

$$x_{ij}^k \in \{0,1\}, \qquad \forall (i,j) \in E,\ \forall k \in K, \tag{B.8}$$

$$y_i^k \in \{0,1\}, \qquad \forall i \in V,\ \forall k \in K, \tag{B.9}$$

where the decision variables represent:

$$x_{ij}^k = \begin{cases} 1, & \text{vehicle } k \text{ travels directly from } i \text{ to } j, \\ 0, & \text{otherwise,} \end{cases} \qquad (i,j) \in E,\ k \in K,$$

$$y_i^k = \begin{cases} 1, & \text{customer } i \text{ is served by vehicle } k, \\ 0, & \text{otherwise,} \end{cases} \qquad i \in V,\ k \in K.$$

The objective (B.1) is used to minimize the total travel cost. Constraint (B.2) ensures each customer is visited exactly once by a single vehicle. Constraint (B.3) makes all vehicles start and end at the depot. By constraint (B.4), a vehicle that visits a customer node enters it once and leaves it once; if it doesn't visit the node, it neither enters nor leaves. Capacity feasibility is guaranteed by (B.5). Cut-set inequalities (B.6) eliminate subtours not connected to the depot. The hard clustering requirement is enforced by (B.7), i.e., each cluster must be entered and exited exactly once by a single vehicle and all customers in a cluster are visited contiguously.

Unlike the traditional CVRP, the CluVRP introduces an explicit partition of customers into disjoint clusters and imposes a hard single-entry/single-exit constraint per cluster, so that each cluster is visited contiguously by a single vehicle. With such cluster-related constraints, solving CluVRP needs to jointly optimize the sequencing of clusters, the identification of inbound and outbound customers (i.e., the first and last customers served within a cluster), and the routing of customers within each cluster. As a result, conventional CVRP methods (designed for homogeneous routing decisions) cannot be directly applied.

References

Baldacci, R., Bartolini, E., Laporte, G., 2010. Some applications of the generalized vehicle routing problem. J. Oper. Res. Soc. 61 (7), 1072–1077.

Battarra, M., Erdoğan, G., Vigo, D., 2014. Exact algorithms for the clustered vehicle routing problem. Oper. Res. 62 (1), 58–71.

Bektaş, T., Erdoğan, G., Røpke, S., 2011. Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. Transp. Sci. 45 (3), 299–316.

Bello, I., Pham, H., Le, Q.V., Norouzi, M., Bengio, S., 2017. Neural combinatorial optimization with reinforcement learning. In: The Fifth International Conference on Learning Representations.

Bi, J., Ma, Y., Zhou, J., Song, W., Cao, Z., Wu, Y., Zhang, J., 2024. Learning to handle complex constraints for vehicle routing problems. Adv. Neural Inf. Process. Syst. 37, 93479–93509.

Bogyrbayeva, A., Yoon, T., Ko, H., Lim, S., Yun, H., Kwon, C., 2023. A deep reinforcement learning approach for solving the traveling salesman problem with drone. Transp. Res. Part C Emerging Technol. 148, 103981.

Choo, J., Kwon, Y.-D., Kim, J., Jae, J., Hottung, A., Tierney, K., Gwon, Y., 2022. Simulation-guided beam search for neural combinatorial optimization. In: Proceedings of the 36th Advances in Neural Information Processing Systems, pp. 8760–8772.

Cosma, O., Pop, P.C., Sitar, C.P., 2022. A two-level based genetic algorithm for solving the soft-clustered vehicle routing problem. Carpathian J. Math. 38 (1), 117–128.

Cplex, I.I., 2009. V12. 1: User's manual for CPLEX. Int. Bus. Mach. Corporation 46 (53), 157.

Dai, H., Khalil, E.B., Zhang, Y., Dilkina, B., Song, L., 2017. Learning combinatorial optimization algorithms over graphs. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 6351–6361.

Defryn, C., Sörensen, K., 2017. A fast two-level variable neighborhood search for the clustered vehicle routing problem. Comput. Oper. Res. 83, 78–94.

Expósito-Izquierdo, C., Rossi, A., Sevaux, M., 2016. A two-level solution approach to solve the clustered capacitated vehicle routing problem. Comput. Indus. Eng. 91, 274–289.

Fang, H., Song, Z., Weng, P., Ban, Y., 2024. InViT: a generalizable routing problem solver with invariant nested view transformer. In: Proceedings of the 41st International Conference on Machine Learning, pp. 12973–12992.

Freitas, M., Silva, J. M.P., Uchoa, E., 2023. A unified exact approach for clustered and generalized vehicle routing problems. Comput. Oper. Res. 149, 106040.

Fu, Z.-H., Qiu, K.-B., Zha, H., 2021. Generalize a small pre-trained model to arbitrarily large tSP instances. In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 35, pp. 7474–7482.

Gao, C., Shang, H., Xue, K., Li, D., Qian, C., 2023. Towards generalizable neural solvers for vehicle routing problems via ensemble with transferrable local policy. In: International Joint Conference on Artificial Intelligence, pp. 6914–6922.

Gendreau, M., Potvin, J.-Y., Bräumlaysy, O., Hasle, G., Løkketangen, A., 2008. Metaheuristics for the vehicle routing problem and its extensions: a categorized bibliography. In: The Vehicle Routing Problem: Latest Advances and New Challenges, pp. 143–169.
Ghiani, G., Improta, G., 2000. An efficient transformation of the generalized vehicle routing problem. Eur. J. Oper. Res. 122 (1), 11–17.
Helsgaun, K., 2017. An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. Roskilde Roskilde Univ. 12, 966–980.
Heßler, K., Irnich, S., 2021. A branch-and-cut algorithm for the soft-clustered vehicle-routing problem. Discrete Appl. Math. 288, 218–234.
Hintsch, T., 2021. Large multiple neighborhood search for the soft-clustered vehicle-routing problem. Comput. Oper. Res. 129, 105132.
Hintsch, T., Irnich, S., 2018. Large multiple neighborhood search for the clustered vehicle-routing problem. Eur. J. Oper. Res. 270 (1), 118–131.
Hintsch, T., Irnich, S., 2020. Exact solution of the soft-clustered vehicle-routing problem. Eur. J. Oper. Res. 280 (1), 164–178.
Ho, G., Tang, Y.M., Leung, E. K.H., Tong, P.H., 2025. Integrated reinforcement learning of automated guided vehicles dynamic path planning for smart logistics and operations. Transp. Res. Part E Logist. Transp. Rev. 196, 104008.
Hoogeboom, M., Battarra, M., Erdoğan, G., Vigo, D., 2016. Erratum-exact algorithms for the clustered vehicle routing problem. Oper. Res. 64 (2), 456–457.
Hottung, A., Tierney, K., 2022. Neural large neighborhood search for routing problems. Artif. Intell. 313, 103786.
Hou, Q., Yang, J., Su, Y., Wang, X., Deng, Y., 2023. Generalize learned heuristics to solve large-scale vehicle routing problems in real-time. In: The Eleventh International Conference on Learning Representations.
Hu, Y., Yao, Y., Chen, J., Wang, Z., Jia, Q., Pan, Y., 2025. Solving scalable multiagent routing problems with reinforcement learning. IEEE Trans. Neural Netw. Learn. Syst. .
Hudson, B., Li, Q., Malencia, M., Prorok, A., 2021. Graph neural network guided local search for the traveling salesperson problem. In: The Ninth International Conference on Learning Representations.
Islam, M.A., Gajpal, Y., ElMekkawy, T.Y., 2021. Hybrid particle swarm optimization algorithm for solving the clustered vehicle routing problem. Appl. Soft. Comput. 110, 107655.
Jin, Y., Ding, Y., Pan, X., He, K., Zhao, L., Qin, T., Song, L., Bian, J., 2023. PointerFormer: deep reinforced multi-pointer transformer for the traveling salesman problem. In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 37, pp. 8132–8140.
Kim, J., Manna, A., Roy, A., Moon, I., 2025. Clustered vehicle routing problem for waste collection with smart operational management approaches. Int. Trans. Oper. Res. 32 (2), 863–887.
Kim, M., Choi, S., Kim, H., Son, J., Park, J., Bengio, Y., 2024. Ant colony sampling with Gflownets for combinatorial optimization. In: The 28th International Conference on Artificial Intelligence and Statistics.
Kim, M., Park, J., Park, J., 2022. Sym-NCO: leveraging symmetricity for neural combinatorial optimization. Adv. Neural Inf. Process. Syst. 35, 1936–1949.
Kong, D., Ma, Y., Cao, Z., Yu, T., Xiao, J., 2024. Efficient neural collaborative search for pickup and delivery problems. IEEE Trans. Pattern Anal. Mach. Intell. .
Kool, W., van Hoof, H., Welling, M., 2018. Attention, learn to solve routing problems! In: The Sixth International Conference on Learning Representations.
Kwon, Y.-D., Choo, J., Kim, B., Yoon, I., Gwon, Y., Min, S., 2020. POMO: policy optimization with multiple optima for reinforcement learning. Adv. Neural Inf. Process. Syst. 33, 21188–21198.
Kwon, Y.-D., Choo, J., Yoon, I., Park, M., Park, D., Gwon, Y., 2021. Matrix encoding networks for neural combinatorial optimization. Adv. Neural Inf. Process. Syst. 34, 5138–5149.
Latorre, V., 2025. An application of a two-level genetic search for the soft-clustered vehicle routing problem. Evol. Intell. 18 (4), 1–19.
Li, J., Xin, L., Cao, Z., Lim, A., Song, W., Zhang, J., 2021a. Heterogeneous attentions for solving pickup and delivery problem via deep reinforcement learning. IEEE Trans. Intell. Transp. Syst. 23 (3), 2306–2315.
Li, M., Cai, K., Zhao, P., 2025. Optimizing same-day delivery with vehicles and drones: a hierarchical deep reinforcement learning approach. Transp. Res. Part E Logist. Transp. Rev. 193, 103878.
Li, S., Yan, Z., Wu, C., 2021b. Learning to delegate for large-scale vehicle routing. Adv. Neural Inf. Process. Syst. 34, 26198–26211.
Liu, F., Lin, X., Wang, Z., Zhang, Q., Xialiang, T., Yuan, M., 2024. Multi-task learning for routing problem with cross-problem zero-shot generalization. In: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 1898–1908.
Luo, F., Lin, X., Liu, F., Zhang, Q., Wang, Z., 2023. Neural combinatorial optimization with heavy decoder: toward large scale generalization. Adv. Neural Inf. Process. Syst. 36, 8845–8864.
Mak, S., Xu, L., Pearce, T., Ostroumov, M., Brintrup, A., 2023. Fair collaborative vehicle routing: a deep multi-agent reinforcement learning approach. Transp. Res. Part C Emerging Technol. 157, 104376.
Min, Y., Bai, Y., Gomes, C.P., 2023. Unsupervised learning for solving the travelling salesman problem. Adv. Neural Inf. Process. Syst. 36, 47264–47278.
Pan, W., Xiong, H., Ma, J., Zhao, W., Li, Y., Yan, J., 2025. UniCO: on unified combinatorial optimization via problem reduction to matrix-encoded general TSP. In: The Thirteenth International Conference on Learning Representations.
Peng, W., Wang, D., Yin, Y., Cheng, T., 2025. Multi-agent deep reinforcement learning-based truck-drone collaborative routing with dynamic emergency response. Transp. Res. Part E Logist. Transp. Rev. 195, 103974.
Pop, P.C., Fuksz, L., Marc, A.H., Sabo, C., 2018. A novel two-level optimization approach for clustered vehicle routing problem. Comput. Indus. Eng. 115, 304–318.
Pop, P.C., Kara, I., Marc, A.H., 2012. New mathematical models of the generalized vehicle routing problem and extensions. Appl. Math. Model. 36 (1), 97–107.
Qiu, R., Sun, Z., Yang, Y., 2022. DIMES: a differentiable meta solver for combinatorial optimization problems. Adv. Neural Inf. Process. Syst. 35, 25531–25546.
Sevaux, M., Sörensen, K., et al., 2008. Hamiltonian paths in large clustered routing problems. In: Proceedings of the EU/MEeting 2008 Workshop on Metaheuristics for Logistics and Vehicle Routing. Vol. 8, pp. 411–417.
Sun, Z., Yang, Y., 2023. DIFUSCO: graph-based diffusion solvers for combinatorial optimization. Adv. Neural Inf. Process. Syst. 36, 3706–3731.
Sutton, R.S., 2018. Reinforcement learning: an introduction. Bradford Book .
Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T., Subramanian, A., 2017. New benchmark instances for the capacitated vehicle routing problem. Eur. J. Oper. Res. 257 (3), 845–858.
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need. Adv. Neural Inf. Process. Syst. 30, 5998–6008.
Vidal, T., Battarra, M., Subramanian, A., Erdoğan, G., 2015. Hybrid metaheuristics for the clustered vehicle routing problem. Comput. Oper. Res. 58, 87–99.
Williams, R.J., 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach. Learn. 8, 229–256.
Ye, H., Wang, J., Cao, Z., Liang, H., Li, Y., 2024. DeepACO: neural-enhanced ant systems for combinatorial optimization. Adv. Neural Inf. Process. Syst. 36, 43706–43728.
Zhou, F., Lischka, A., Kulcsár, B., Wu, J., Chehreghani, M.H., Laporte, G., 2025a. Learning for routing: a guided review of recent developments and future directions. Transp. Res. Part E Logist. Transp. Rev. 202, 104278.
Zhou, Y., Kou, Y., Zhou, M., 2023. Bilevel memetic search approach to the soft-clustered vehicle routing problem. Transp. Sci. 57 (3), 701–716.
Zhou, Y., Liu, L., Benlic, U., Li, Z.-C., Wu, Q., 2025b. Solving soft and hard-clustered vehicle routing problems: a bi-population collaborative memetic search approach. Eur. J. Oper. Res. 324 (3), 825–838.