Hierarchical Learning-based Graph Partition for Large-scale Vehicle Routing Problems

Yuxin Pan
The Hong Kong University of Science and Technology
Hong Kong, China
yuxin.pan@connect.ust.hk

Ruohong Liu
University of Oxford
Oxford, United Kingdom
ruohong.liu@eng.ox.ac.uk

Yize Chen
University of Alberta
Edmonton, Canada
yize.chen@ualberta.ca

Zhiguang Cao
Singapore Management University
Singapore, Singapore
zhiguangcao@outlook.com

Fangzhen Lin
The Hong Kong University of Science and Technology
Hong Kong, China
flin@cs.ust.hk

ABSTRACT

Neural solvers based on the divide-and-conquer approach for Vehicle Routing Problems (VRPs) in general, and capacitated VRP (CVRP) in particular, integrates the global partition of an instance with local constructions for each subproblem to enhance generalization. However, during the global partition phase, misclusterings within subgraphs have a tendency to progressively compound throughout the multi-step decoding process of the learning-based partition policy. This suboptimal behavior in the global partition phase, in turn, may lead to a dramatic deterioration in the performance of the overall decomposition-based system, despite using optimal local constructions. To address these challenges, we propose a versatile Hierarchical Learning-based Graph Partition (HLGP) framework, which is tailored to benefit the partition of CVRP instances by synergistically integrating global and local partition policies. Specifically, the global partition policy is tasked with creating the coarse multi-way partition to generate the sequence of simpler two-way partition subtasks. These subtasks mark the initiation of the subsequent K local partition levels. At each local partition level, subtasks exclusive for this level are assigned to the local partition policy which benefits from the insensitive local topological features to incrementally alleviate the compounded errors. This framework is versatile in the sense that it optimizes the involved partition policies towards a unified objective harmoniously compatible with both reinforcement learning (RL) and supervised learning (SL). Additionally, we decompose the synchronized training into individual training of each component to circumvent the instability issue. Furthermore, we point out the importance of viewing the subproblems encountered during the partition process as individual training instances. Extensive experiments conducted on various CVRP benchmarks demonstrate the effectiveness and generalization of the HLGP framework. The source code is available at https://github.com/panyxy/hlgp_cvrp.

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

KEYWORDS

Vehicle Routing Problem; Combinatorial Optimization Problem; Hierarchical Learning

ACM Reference Format:
Yuxin Pan, Ruohong Liu, Yize Chen, Zhiguang Cao, and Fangzhen Lin. 2025. Hierarchical Learning-based Graph Partition for Large-scale Vehicle Routing Problems. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

1 INTRODUCTION

The Vehicle Routing Problem (VRP) is a widely-studied NP-hard problem which has many real-world applications including transportation [10], logistic [2], and digital e-commerce [24]. Exact methods for solving VRP often use mixed integer linear programming (MILP) techniques and employ MILP solvers to generate optimal solutions with theoretical guarantees [21]. However, these methods so far are not computationally efficient enough to handle large-scale instances, particularly for the applications with time-sensitive and dynamically changing VRP scenarios. In contrast, heuristic methods such as LKH3 [12] and HGS [37] aim to generate high-quality solutions quickly. They commonly improve the quality of the existing solution incrementally by local search techniques. However, in addition to the heavy reliance on the quality of handcrafted local operators, these methods are not robust and often need to start from scratch for the problem instances with slight variations.

More recently, there has been much work on neural network-based solvers for VRP. Experimentally they have been shown capable of inferring near-optimal efficiently solutions for instances which fall within the training data distribution. These learning-based solvers typically use one of the following methods: constructive, iterative, and divide-and-conquer. The constructive method, as a pioneering paradigm, incrementally deduces a complete solution starting from an empty state [18–20, 25, 32, 38, 40]. However, challenges arise when dealing with out-of-distribution instances due to the limited expressivity of neural network and the intricate search landscape. To mitigate this performance degradation, the iterative method merges a neural network-based policy with heuristic local operators to progressively refine the current solution [3, 13, 26, 29, 30, 41]. Yet, this approach relies on numerous

improvement steps with well-crafted local operators for satisfactory solutions. By comparison, the divide-and-conquer approach embraces either a heuristic-based partition policy [4, 7, 8, 17, 23, 46] or a neural partition policy [14, 33, 42, 44] to globally divide the entire graph into subgraphs and employ a local construction policy to solve subproblems. However, a failure in either component policy may lead to a significant performance drop. Moreover, heuristic-based partition rules often result in local optima, and neural partition policies may be vulnerable to distribution shifts. Hence, there is a pressing need for a more generalizable and meticulous partition policy, which is the focus of this paper.

In the divide-and-conquer paradigm, the local construction policy agent benefits from the local topological features within subproblems insensitive against distribution and scale shifts, contributing to the (near-)optimality of solutions of subproblems [6, 9, 16]. However, during the multi-step decoding process of the learning-based partition policy for Capacitated VRP (CVRP) instances [14, 42], the decoding of clustered nodes in each step relies on the partial partition solution from the preceding step. This implies that errors in the clustering from earlier steps have a tendency to propagate and result in a chain of misclusterings in subsequent steps, called as compounded errors. Consequently, even with an optimal local construction policy, deficiencies in the partition task lead to substantial deviations from the ideal policy in the overall system. Therefore, we argue that the partition task holds a critical position in the overall decomposition-based system for solving CVRP. Furthermore, the success of the local construction policy inspires us to introduce a local partition policy which aims to progressively alleviate compounded errors by harnessing local topological features in the partition task. We thus consider to implement a hierarchical learning (HL) framework specifically designed for the partition task in CVRP, which is capable of seamlessly integrating both global and local partition policies. In prevailing HL frameworks, a high-level policy is adopted to derive a series of simpler sub-tasks which are then delegated to the low-level policy, with the aim of facilitating exploration [34]. These frameworks predominantly focus on reinforcement learning and undergo joint training of the associated policies [22]. Yet, these HL frameworks have not been extensively explored in addressing compounded errors within the graph partition task of large-scale CVRP. In contrast, our study extends the HL framework to the partition task of CVRP and demonstrates its efficacy in mitigating compounded errors.

In this paper, we present a versatile Hierarchical Learning-based Graph Partition (HLGP) framework specifically tailored for the partition task in CVRP, which synergistically integrates both global and local partitioning policies. To be specific, our method formulates the partition problem of CVRP using a multi-level HL framework. At the global partition level, the global partition policy is responsible for initiating a coarse multi-way partition to create a series of simpler 2-way partition subtasks. These subtasks stand as the starting point for the subsequent K local partition levels. At each local partition level, a tailored sequence of subtasks is derived from the partition solution of the preceding level. These subtasks are then fed into the local partition policy. Such a setup allows the local partition policy to mitigate the potential misclustering arising from the previous level by leveraging the insensitive local topological features inherent in the subtask. By enabling the local partition policy to traverse through the subtasks at each local partition level, the compounded misclusterings can be mitigated progressively across levels as a consequence.

Our proposed HLGP framework is versatile, featuring a unified objective that effortlessly accommodates both reinforcement learning (RL) and supervised learning (SL) for training the partition policies. It is worth noting that unlike prior approaches that utilized SL to directly train the neural solver, our method explores the application of SL for training the partition policy, usually omitting the need for permutation information. Additionally, the joint training of the involved policies is disentangled to mitigate training instability. Moreover, by conducting in-depth analyses in both the RL and SL settings, we shed light on the importance of viewing the subproblems encountered during the partition process as individual training instances. Empirically, the proposed HLGP framework convincingly demonstrates its superiority through extensive experiments on various CVRP benchmarks over previous SOTA methods. In particular, our method can scale up to CVRP10K instances with around 10% performance improvement over current literature.

2 RELATED WORKS

Learning-based methods for solving combinatorial optimization problems (COPs) typically fall into three main categories: **constructive methods**, **iterative methods**, and **divide-and-conquer methods**. Constructive methods aim to progressively infer complete solutions using the autoregressive mechanism [1, 9, 11, 15, 18–20, 31, 32, 35, 36, 38, 45]. Impressively, SL-driven constructive policies, such as BQ [5], LEHD [27] and SIL [28] can mitigate the high GPU memory demands associated with gradient backpropagation by eliminating the need for delayed rewards in the training of RL algorithms. Iterative methods offer the benefit of consistently improving a given solution until convergence [3, 13, 26, 29, 30, 41] by integrating local improvement operators into RL policies. The divide-and-conquer paradigm can exploit local topological features that remain insensitive to distribution or scale shifts, thus alleviating performance degradation. Some methods harness heuristic rules for the partitioning process [4, 8, 17, 23, 46]. In contrast, H-TSP [33], TAM [14], GLOP [42], and UDC [44] opt to use a learning-based policy to globally divide the entire instance into subproblems, which are then addressed by a pretrained local construction policy. *For a more detailed review of related algorithms used to solve VRPs, please refer to the Appendix-D.*

3 PRELIMINARIES

CVRP Formulation. A CVRP instance I is defined as a tuple, represented by $I = (G, D, N_{\max})$. The graph G consists of a depot node v_0 and N_v customer nodes v_i $(1 \leq i \leq N_v)$. D and $N_{\max}$ denote the vehicle capacity and maximum allowable number of times vehicles returning to the depot, respectively. Each node is associated with its coordinates (a_i, b_i) and each customer node is further associated with a demand d_i. $N_{\max}$ is accordingly defined as $\lceil \sum_{i=1}^{N_v} d_i/D \rceil + 1$. The distance between any pair of nodes can be measured by the Euclidean distance. In the CVRP, the vehicle needs to visit each customer exactly once, fulfills their demands without exceeding D, and returns to the depot to reload goods if

necessary. A feasible solution $\mathcal{T} \in \mathbb{S}_{\mathcal{T}}$ can be described as a node permutation where the depot node can occur multiple times, while each customer node appears only once. Furthermore, the feasible solution $\mathcal{T}$ also can be decomposed into N_τ feasible subtours. In each subtour τ_i $(1 \le i \le N_\tau)$, the starting and ending nodes are the depot, and the intermediate nodes are customers. The travel cost $e(\tau_i)$ associated with τ_i is the sum of Euclidean distances along this subtour. Thus, the objective is to minimize $e(\mathcal{T}) = \sum_{i=1}^{N_\tau} e(\tau_i)$. $\mathbb{S}_{\mathcal{T}}$ denotes the space of feasible solutions, as does the notation used for $\mathbb{S}$ in the following sections.

Global Partition and Local Construction (GPLC). In the context of CVRP, the partition task involves clustering nodes into distinct groups, ensuring that each cluster includes the depot node, each customer node belongs to a single cluster, and the total demand within each cluster does not surpass D. Each cluster of nodes forms a subgraph. A feasible partition solution $\mathcal{C} \in \mathbb{S}_{\mathcal{C}}$ comprises N_c subgraphs. Each subgraph c_i $(1 \le i \le N_c)$ consists of the depot node along with various customer nodes distinct from those in other subgraphs (i.e., $\cup_{i=1}^{N_c} c_i = G$ and $\cap_{i=1}^{N_c} c_i = \{v_0\}$). The feasible partition solution $\mathcal{C}$ can also be represented as a node list where the order of customer nodes is ignored between the two consecutively visited depot nodes. Therefore, a feasible solution $\mathcal{T}$ can be equivalently seen as a feasible partition solution $\mathcal{C}$ when disregarding the node permutation information in $\mathcal{T}$.

Obviously, both the original CVRP and the partition problem within CVRP revolve around feasible solutions $\mathcal{T}$ and $\mathcal{C}$ composed of discrete value variables. This fact prompts prevalent learning-based methods to employ stochastic policies as the neural solver to handle whatever types of problems (permutation or partition) within the context of CVRP. Let $\Delta(\cdot)$ denote the space of the probability measure. In the GPLC paradigm [14, 42], a stochastic partition policy $\pi_{\text{part}}(\mathcal{C}|I) \in \Delta(\mathbb{S}_{\mathcal{C}})$ is used to derive a feasible partition solution $\mathcal{C}$ by dividing the graph G. Then, a (near-)optimal local permutation policy $\pi^*_{\text{perm}}(\mathcal{T}|\mathcal{C}) \in \Delta(\mathbb{S}_{\mathcal{T}})$ can generate the feasible subtour τ_i for each subproblem $(c_i, D, 1)$. The objective is to identify an optimal partition policy π^*_{part} that minimizes $e(\mathcal{T})$. *However, prior GPLC methods lack theoretical foundations of the partition problem. Therefore, we introduce Theorem 1 to establish the rationality of the partition problem for CVRP.* Please see Appendix-C.1 for proofs.

Theorem 1. *The objective in solving an original CVRP instance I is to identify a (permutation) policy $\pi(\mathcal{T}|I) \in \Delta(\mathbb{S}_{\mathcal{T}})$ so as to minimize the expected cost $\mathbb{E}_{\mathcal{T}\sim\pi}[e(\mathcal{T})]$. If $\pi^*_{\text{perm}} \in \Delta(\mathbb{S}_{\mathcal{T}})$ is optimal for each subproblem $(c_i, D, 1)$, then the original objective can be reframed as identifying an optimal partition policy $\pi^*_{\text{part}} \in \Delta(\mathbb{S}_{\mathcal{C}})$ to minimize the expected cost, expressed as:*

$$\min_{\pi_{\text{part}}} \mathbb{E}_{\mathcal{C}\sim\pi_{\text{part}}}[\sum_{i=1}^{N_c} \mathbb{E}_{\tau_i\sim\pi^*_{\text{perm}}}(e(\tau_i))], \tag{1}$$

where $\pi^*_{\text{perm}}(\mathcal{T}|\mathcal{C}) = \prod_{i=1}^{N_c} \pi^*_{\text{perm}}(\tau_i|c_i)$ implies that τ_i is independently sampled given the corresponding c_i. As aforementioned, viewing a feasible partition solution $\mathcal{C}$ as a node list implies that the partition policy can incrementally construct the partition solution. This process involves conditioning the current selected node $\mathcal{C}[n]$ on the partial partition solution $\mathcal{C}[0:n-1]$ ($\mathcal{C}[0] = \emptyset$) and the given problem instance I, written as:

$$\pi_{\text{part}}(\mathcal{C}|I) = \prod_{n=1}^{N_{\text{sol}}} \pi_{\text{part}}(\mathcal{C}[n]|\mathcal{C}[0:n-1], I), \tag{2}$$

where N_{sol} denotes the length of partition solution. Please note that we abuse the notation of N_{sol} to denote the length of different partition solutions for brevity in the following sections. Since the objective defined in Equation 1 essentially aligns with that associated with RL, the common approach involves training neural policies using RL.

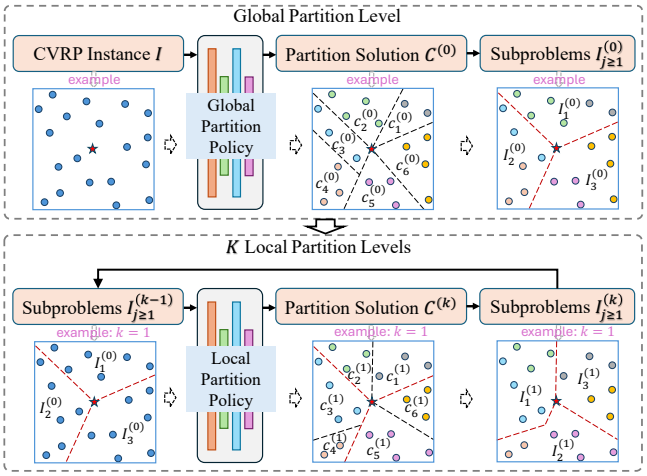


Figure 1: The proposed HLGP framework. $I^k_{j\geq 1}$ represents a sequence of subproblems. Following the HLGP framework, the sequence of subproblems $I^K_{j\geq 1}$ are fed to a permutation policy to derive the respective subtours.

4 HIERARCHICAL LEARNING-BASED GRAPH PARTITION

Our proposed HLGP framework is built upon the GPLC paradigm. Likewise, we assume the optimal local permutation policy π^*_{perm} is obtainable by leveraging LKH3 [12] or the neural solver used in GLOP [42]. It is evident from the partition policy expression in Equation 2 that decoding the nodes at each step hinges on the partial partition solution obtained in the preceding steps. Consequently, inaccuracies in clustering from earlier steps tend to propagate, resulting in a chain of misclustering in subsequent steps. These misclusterings in the partition policy exacerbate notably when confronted with substantial distribution or scale shifts. This empirical challenge in CVRP thus motivates us to develop a HL framework for solving the partition problem in CVRP. We anticipate that this HL framework can progressively mitigate compounded errors by incorporating both global and local partition policies.

4.1 HL Formulation of Partition Problem

In this section, we begin by introducing the *feasible cost function* $f(\mathcal{C})$ for a feasible partition solution $\mathcal{C}$ as defined in Definition 1. Following this, various forms of $f(\mathcal{C})$ will be presented in the subsequent sections to align with both RL and SL objectives for training the partition policies.

Definition 1. *Let π^*_{part} denote the optimal partition policy obtained by optimizing the objective in Equation 1. Given a cost function $f(\mathcal{C}) : \mathbb{S}_{\mathcal{C}} \to \mathbb{R}$, if π^*_{part} can be derived by optimizing the objective $\min_{\pi_{\text{part}}} \mathbb{E}_{\mathcal{C}\sim\pi_{\text{part}}}[f(\mathcal{C})]$, then $f(\mathcal{C})$ is a feasible cost function.*

By leveraging this well-defined feasible cost function $f(\mathcal{C})$, the goal of the partition problem is to minimize $\mathbb{E}_{\mathcal{C}}[f(\mathcal{C})]$. Then, we reformulate the partition problem using a multi-level HL framework. In this framework, the global partition policy π_{Gpart} and the local partition policy π_{Lpart} work together in synergy to execute the partition task, as depicted in Figure 1. At the global partition level, π_{Gpart} creates an initial coarse feasible partition $\mathcal{C}^{(0)} = \{c_1^{(0)}, \ldots, c_{N_c}^{(0)}\}$, where $c_i^{(0)}$ denotes the subgraph at this level. In this partition solution $\mathcal{C}^{(0)}$, each pair of subgraphs $(c_i^{(0)}, c_{i\%N_c+1}^{(0)})$ (where $1 \le i \le N_c$) is stipulated as neighboring subgraphs as defined by a specific heuristic rule. For instance, a simple heuristic involves rearranging subgraphs in $\mathcal{C}^{(0)}$ based on the polar angles of their centroids in a Polar coordinate system centered at the depot node. This coarse multi-way partition $\mathcal{C}^{(0)}$ serves as the entry point of the subsequent K local partition levels. At each local partition level $k \in \{1, ..., K\}$, the subproblems are sequentially formed by reuniting pairs of neighboring subgraphs from $\mathcal{C}^{(k-1)}$. Each subproblem $I_j^{(k-1)} (1 \le j \le \lfloor \frac{N_c}{2} \rfloor)$ is defined as:

$$\begin{aligned} I_j^{(k-1)} &= (G_j^{(k-1)}, D, 2); \\ G_j^{(k-1)} &= c^{(k-1)}_{(m+k-1)\%N_c+1} \cup c^{(k-1)}_{(m+k)\%N_c+1}, \end{aligned} \tag{3}$$

where $m = 2(j-1)$. There are $\lfloor \frac{N_c}{2} \rfloor$ subproblems in each local partition level. For each subproblem, the vehicle is only allowed to return twice to the depot by subproblem definition. Please note that each pair of consecutive subproblems $I_j^{(k-1)}$ and $I_{j+1}^{(k-1)}$ do not overlap in terms of the subgraphs they contain. Additionally, this technique for creating subproblems can be described as initially left-shifting the subgraphs in $\mathcal{C}^{(k-1)}$ by $k-1$ places and then merging the neighboring subgraphs without overlaps. At each local partition level $k \ge 1$, the subproblem sequence is directed to the local partition policy π_{Lpart}. This allows the local partition policy π_{Lpart} to address potential misclusterings from the preceding level by utilizing the robust local topological features. As a result, the local partition policy can traverse through subproblems at each local partition level, gradually reducing accumulated misclusterings across levels. Moreover, upon completion of solving the subproblem $I_j^{(k-1)}$, the pair of subgraphs $(c^{(k-1)}_{(m+k-1)\%N_c+1}, c^{(k-1)}_{(m+k)\%N_c+1})$ is transitioned to the corresponding subgraph pair $(c^{(k)}_{(m+k-1)\%N_c+1}, c^{(k)}_{(m+k)\%N_c+1})$. Consequently, the resolution of the subproblem sequence results in an update from $\mathcal{C}^{(k-1)}$ to $\mathcal{C}^{(k)}$.

Within the overall HLGP framework, the global partition policy π_{Gpart} is formulated identical to the partition policy in the GPLC method, written as:

$$\pi_{\text{Gpart}}(\mathcal{C}^{(0)}|I) = \prod_{n=1}^{N_{\text{sol}}} \pi_{\text{Gpart}}(\mathcal{C}^{(0)}[n]|\mathcal{C}^{(0)}[0:n-1], I), \tag{4}$$

where $\mathcal{C}^{(0)}[n]$ and $\mathcal{C}^{(0)}[0:n-1]$ denote the n-th selected node and the partial solution in $\mathcal{C}^0$, respectively. In contrast, the local partition policy addresses the series of subproblems produced from the previous partition solution $\mathcal{C}^{(k-1)}$ to construct the partition solution $\mathcal{C}^{(k)}$. Let $\mathcal{C}_j^{(k-1)}$ denote the partition solution for the subproblem $I_j^{(k-1)}$. Again, the partition solution $\mathcal{C}_j^{(k-1)}$ can be either represented as a node list where $\mathcal{C}_j^{(k-1)}[n]$ and $\mathcal{C}_j^{(k-1)}[0:n-1]$ indicate the n-th node and partial solution within it respectively, or decomposed into two subgraphs $c^{(k)}_{(m+k-1)\%N_c+1}$, $c^{(k)}_{(m+k)\%N_c+1}$ both of which also belong to $\mathcal{C}^{(k)}$. Thus, it can be expressed as:

$$\begin{aligned} \pi_{\text{Lpart}}(\mathcal{C}^{(k)}|\mathcal{C}^{(k-1)}, k) &= \prod_{j=1}^{\lfloor \frac{N_c}{2} \rfloor} \pi_{\text{Lpart}}(\mathcal{C}_j^{(k-1)}|I_j^{(k-1)}) \\ &= \prod_{j=1}^{\lfloor \frac{N_c}{2} \rfloor} \prod_{n=1}^{N_{\text{sol}}} \pi_{\text{Lpart}}(\mathcal{C}_j^{(k-1)}[n]|\mathcal{C}_j^{(k-1)}[0:n-1], I_j^{(k-1)}). \end{aligned} \tag{5}$$

Please note that in the LHS of Equation 5, the parameter k representing the level is included as an input to the local partition policy. This inclusion is necessary as the parameter k governs the construction of different subproblem sequences for each level. As a result, the objective of HLGP framework is to minimize the expected cost by optimizing both π_{Gpart} and π_{Lpart}, written as:

$$\min_{\pi_{\text{Gpart}}, \pi_{\text{Lpart}}} \mathbb{E}_{\mathcal{C}^{(0)}} \mathbb{E}_{\mathcal{C}^{(1)}} \cdots \mathbb{E}_{\mathcal{C}^{(K)}} [f(\mathcal{C}^{(K)})], \tag{6}$$

where $\mathcal{C}^{(0)}$ and $\mathcal{C}^{(k)}$ $(k \ge 1)$ are sampled from $\pi_{\text{Gpart}}(\mathcal{C}^{(0)}|I)$ and $\pi_{\text{Lpart}}(\mathcal{C}^{(k)}|\mathcal{C}^{(k-1)}, k)$, respectively.

4.2 RL-driven HLGP

In the HLGP framework, the imperative task at hand involves the joint optimization for the global and local partition policies, as illustrated in Equation 6. To address this intricate optimization challenge through RL algorithms, a rigorous formulation utilizing a multi-level Markov Decision Process (MDP) is required. However, Equation 6 essentially revolves around evaluating $\mathcal{C}^{(K)}$ at the K-th level. The absence of direct evaluations for $\mathcal{C}^{(k)}, k < K$, primarily contributes to the instability concern during the joint training via RL. We thus equivalently convert it to one involving direct evaluations at each level, as outlined in Theorem 2.

Theorem 2. *Let $g(c_i)$ denote $\mathbb{E}_{\tau_i \sim \pi^*_{\text{perm}}(\cdot|c_i)}(e(\tau_i))$. It is clear that $f(\mathcal{C}) = \sum_{i=1}^{N_c} g(c_i)$ acts as a feasible cost function. Then, the optimization problem defined in Equation 6 can be transformed equivalently as follows:*

$$\begin{aligned} \min_{\pi_{\text{Gpart}}, \pi_{\text{Lpart}}} \mathbb{E}_{\mathcal{C}^{(0)}}[f(\mathcal{C}^{(0)})] + \mathbb{E}_{\mathcal{C}^{(0)}} \mathbb{E}_{\mathcal{C}^{(1)}}[f(\mathcal{C}^{(1)}) - f(\mathcal{C}^{(0)})] + \\ \cdots + \mathbb{E}_{\mathcal{C}^{(0)}} \mathbb{E}_{\mathcal{C}^{(1)}} \cdots \mathbb{E}_{\mathcal{C}^{(K)}}[f(\mathcal{C}^{(K)}) - f(\mathcal{C}^{(K-1)})]. \end{aligned} \tag{7}$$

The evaluation for $\mathcal{C}^{(k)}, k \ge 1$, can further be derived as:

$$\begin{aligned} f(\mathcal{C}^{(k)}) - f(\mathcal{C}^{(k-1)}) &= \sum_{j=1}^{\lfloor \frac{N_c}{2} \rfloor} [h(\mathcal{C}^{(k)}, k, m) - h(\mathcal{C}^{(k-1)}, k, m)]; \\ h(\mathcal{C}^{(k)}, k, m) &= g(c^{(k)}_{(m+k-1)\%N_c+1}) + g(c^{(k)}_{(m+k)\%N_c+1}), \end{aligned} \tag{8}$$

where $m = 2(j-1)$.

Please see Appendix-C.2 for proofs. Theorem 2 breaks down the objective described in Equation 6 into $K+1$ components, with each component associated with the direct evaluation of the respective partition solution. Notably, except for the evaluation of $\mathcal{C}^{(0)}$ which solely considers its own cost $f(\mathcal{C}^{(0)})$, the evaluation of $\mathcal{C}^{(k)}$, $k \geq 1$ hinges on the difference between its own cost $f(\mathcal{C}^{(k)})$ and the cost $f(\mathcal{C}^{(k-1)})$ from the preceding level. At each local partition level $k \geq 1$, the local partition policy is responsible for resolving each subproblem I_j^{k-1}, leading to the modification of each pair of subgraphs $(c^{(k-1)}_{(m+k-1)\%N_c+1}, c^{(k-1)}_{(m+k)\%N_c+1})$ to the respective subgraph pair $(c^{(k)}_{(m+k-1)\%N_c+1}, c^{(k)}_{(m+k)\%N_c+1})$. We are thus allowed to proceed with the derivation as indicated in Equation 8. Given the optimization problem stated above, we present the formulation utilizing a multi-level MDP in Proposition 1.

Proposition 1. *In the multi-level MDP framework, at the global partition level, for* $t \geq 1$, *the state* $x_t^{(0)} \in \mathbb{X}^{(0)}$ *comprises problem instance* I *and the partial partition solution* $\mathcal{C}^{(0)}[0:t-1]$ *(*$\mathcal{C}^{(0)}[0] = \emptyset$*). The initial distribution* $\mu^{(0)}$ *aligns with the problem instance distribution* p_I. *The action* $u_t^{(0)} \in \mathbb{U}^{(0)}$ *involves selecting a node denoted as* $\mathcal{C}^{(0)}[t]$, *from unvisited customer nodes and the depot node. Let* i_t *index subgraphs such that at timestep* t, *the agent is constructing* i_t*-th subgraph* $c_{i_t}^{(0)}$. *If the subgraph* $c_{i_t}^{(0)}$ *is created, then the reward* $r_t^{(0)}$ *is set as* $-g(c_{i_t}^{(0)})$; *otherwise, it remains at* 0. *The global partition policy, parameterized by* θ_G, *is thus specified as* $\pi_{\theta_G}(u_t^{(0)}|x_t^{(0)})$.

At each local partition level $k \geq 1$, *the local partition policy is tasked with solving the sequence of subproblems obtained from* $\mathcal{C}^{(k-1)}$. *In this context, we use* j_t *as an index for subproblems, indicating that the* j_t*-th subproblem denoted as* $I_{j_t}^{(k-1)}$, *is currently being addressed but remains incomplete at timestep* t. *The state* $x_t^{(k)} \in \mathbb{X}^{(k)}$ *consists of the subproblem sequence and the partial solution of* $I_{j_t}^{(k-1)}$. *The initial state distribution* $\mu^{(k)}$ *corresponds to the distribution of the subproblem sequence. The action* $u_t^{(k)} \in \mathbb{U}^{(k)}$ *involves selecting a node for solving* $I_{j_t}^{(k-1)}$. *When* $I_{j_t}^{(k-1)}$ *is successfully solved, the index* j_t *will proceed to the next subproblem, and the reward* $r_t^{(k)}$ *is set as* $-(h(\mathcal{C}^{(k)}, k, m) - h(\mathcal{C}^{(k-1)}, k, m))$ *(where* $m = 2(j_t - 1)$*). Otherwise, the reward remains at* 0. *Thus, the local partition policy parameterized by* θ_L, *is defined as* $\pi_{\theta_L}(u_t^{(k)}|x_t^{(k)})$. *The objective is to maximize the sum of expected returns across levels, as defined below:*

$$J(\theta_G, \theta_L) = \mathbb{E}_{\omega^{(0)}}[\sum_{t=1}^{T^{(0)}} r_t^{(0)}] + \cdots + \mathbb{E}_{\omega^{(0)}} \cdots \mathbb{E}_{\omega^{(K)}}[\sum_{t=1}^{T^{(K)}} r_t^{(K)}], \quad (9)$$

where $T^{(k)}$ *and* $\omega^{(k)}$ *denote the horizon and the trajectory at level* k.

Notably, although Equation 9 isolates the evaluation exclusively for $\omega^{(k)}$, the evaluation impacted by the trajectories $\omega^{(k+1)}, ..., \omega^{(K)}$ still remains. This implies that the underlying MDP at level k remains nonstationary. We thus take the following optimization problem as an approximation:

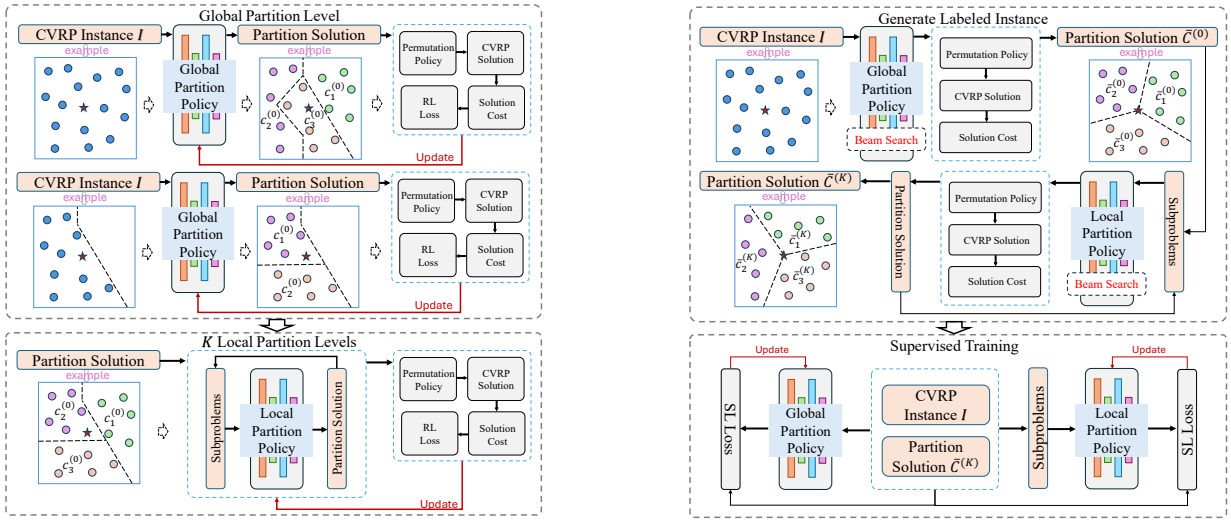
$$\begin{aligned} \hat{J}(\theta_G, \theta_L) &= L(\theta_G, \lambda_G, 0) + \sum_{k=1}^{K} L(\theta_L, \lambda_L, k); \\ L(\theta, \lambda, k) &= \mathbb{E}_{\omega^{(k)} \sim \hat{\mu}^{(k)}, \pi_\theta}[\sum_{t=1}^{T^{(k)}} r_t^{(k)}] + \lambda \mathcal{H}(\pi_\theta), \end{aligned} \quad (10)$$

where $\hat{\mu}^{(k)}$ is a surrogate initial state distribution at level k, $\mathcal{H}(\pi_\theta)$ is the entropy term, and λ denotes the hyperparameter. The entropy term is typically defined to minimize the KL divergence between the policy and a uniform distribution. In Equation 9, $\omega^{(k)}, k \geq 1$ is drawn from the initial distribution $\mu^{(k)}$ and the local partition policy π_{θ_L}. However, $\mu^{(k)}$ heavily relies on preceding partition solutions derived from both the global and local partition policies. Therefore, in Equation 10, the surrogate initial distribution $\hat{\mu}^{(k)}$ is introduced to eliminate this dependency. Please note that $\hat{\mu}^{(0)} = \mu^{(0)}$. As a result, the optimization for π_{θ_G} and π_{θ_L} is decoupled.

In the context of RL-driven HLGP, we incorporate the surrogate objective defined in Equation 10 into the REINFORCE algorithm [39] to update π_{θ_G} and π_{θ_L}. In each iteration $n \geq 0$ of REINFORCE, the existing global partition policy denoted as $\pi_{\theta_G^n}$ is employed to sample $\omega^{(0)}$ for the update of $\pi_{\theta_G^n}$. At each local partition level $k \geq 1$, the current local partition policy denoted as $\pi_{\theta_L^n}$ is additionally leveraged to sample the partition solution $\mathcal{C}^{k-1}$, crucial for $\hat{\mu}^{(k)}$. Following this, $\omega^{(k)}$ is sampled to update $\pi_{\theta_L^n}$. Please refer to Appendix-B for the pseudocode.

Furthermore, in the standard theoretical analysis of REINFORCE algorithm conducted in [43], the upper bound of regret includes the term represented by $||\frac{d}{\mu}||_\infty$, where d and μ stand for the stationary state distribution and the initial state distribution. However, the existing method using REINFORCE algorithm for whatever types of problems (permutation or partition) in the context of CVRP ignores the potential risks highlighted in the regret bound. We exemplify the partition problem as a case study to elucidate this issue. The support set of μ consists solely of the problem instances I. Let $N_v(t)$ denote the number of customer nodes selected before timestep t. In contrast, during the partition process, at each step $t > 1$, the partition policy is indeed responsible to solve the subproblem denoted as $I_{N_v(t)}$ in which the graph comprises depot and unvisited customers. Let c_{i_t} denote the subgraph under construction. The capacity in $I_{N_v(t)}$ is accordingly subtracted from the total demand of the visited node in c_{i_t}, reverting back to D once c_{i_t} is fully formed. The support set of d thus includes the subproblems $I_{N_v(t)}$. This significant discrepancy in support sets inevitably results in an infinite $||\frac{d}{\mu}||_\infty$ in the regret bound. This observation inspires us to incorporate the subproblems $I_{N_v(t)}$ encountered during the partition process into the training of involved partition policies to reduce the mismatch between support sets.

In the practical implementation, a problem instance I is initially generated from the instance distribution p_I, which is used to train π_{θ_G} via RL. If a new subgraph c_{i_t} is formed at timestep t, then the subproblem $I_{N_v(t)}$ is treated as an individual problem instance, denoted as $I \leftarrow I_{N_v(t)}$, for the training of π_{θ_G}. This procedure



(a) RL-driven HLGP (b) SL-driven HLGP

Figure 2: RL-driven HLGP replaces the initially generated partial partition solution with the complete partition solution of subproblems within $\mathcal{C}^{(0)}$ at level 0. SL-driven HLGP requires labeled instances for training π_{θ_G} and π_{θ_L}.

continues until $G = \emptyset$ in I, and reverts back to p_I for a new instance I. For efficiency reasons, we do not include all subproblems. In inference, the partition solution $\mathcal{C}^{(0)}$ is formed by sequentially replacing the partial partition solution with the corresponding complete partition solution of the subproblem. An example is shown in Figure 2(a). The training and inference procedure utilizing the encountered subproblem can similarly be applied to π_{θ_L}. Additionally, we utilize the isomorphic Graph Neural Netwok (GNN) as presented in GLOP [42] to serve as the backbones of π_{θ_G} and π_{θ_L} correspondingly.

4.3 SL-driven HLGP

In this section, we pivot towards an SL training strategy to optimize the objective of the partition problem as defined in Equation 6. Here, the optimal partition solver π^*_{part} is presumed to be available in advance. The optimal partition solution $\bar{\mathcal{C}} = \{\bar{c}_1, \ldots, \bar{c}_{N_c}\}$ for each instance I is accordingly obtainable from π^*_{part}. We thus adopt $f(\mathcal{C}) = -\mathbb{1}(\mathcal{C} = \bar{\mathcal{C}}) = -\mathbb{1}(c_1 = \bar{c}_1, \ldots, c_{N_c} = \bar{c}_{N_c})$, where $\mathbb{1}(\cdot)$ denotes the indicator function. Recall that the optimal solution $\bar{\mathcal{T}}$ of the CVRP instance can be equivalently seen as the optimal partition solution $\bar{\mathcal{C}}$ when disregarding the node permutation information in $\bar{\mathcal{T}}$. Therefore, by setting $\bar{\mathcal{C}} = \bar{\mathcal{T}}$, the feasible cost function can be defined as $f(\mathcal{C}) = -\mathbb{1}(\mathcal{C}[0] = \bar{\mathcal{C}}[0], \ldots, \mathcal{C}[N_{\text{sol}}] = \bar{\mathcal{C}}[N_{\text{sol}}])$. Upon utilizing this design of $f(\mathcal{C})$, Theorem 3 simplifies the optimization objective of HLGP framework.

Theorem 3. *Given $f(\mathcal{C}) = -\mathbb{1}(\mathcal{C} = \bar{\mathcal{C}})$, the optimization objective in the HLGP framework for a problem instance I is to identify π_{θ_G} and π_{θ_L} so as to minimize:*

$$L(\theta_G, \theta_L, \bar{\mathcal{C}}) = -\log \pi_{\theta_G}(\bar{\mathcal{C}}|I) - \sum_{i=1}^{N_c} \log \pi_{\theta_L}(\bar{\mathcal{C}}_i|I_i), \tag{11}$$

where $I_i = (G_i, D, 2)$ denotes the subproblem, with $G_i = \bar{c}_i \cup \bar{c}_{i\%N_c+1}$, and $\bar{\mathcal{C}}_i = \{\bar{c}_i, \bar{c}_{i\%N_c+1}\}$ represents the corresponding label.

Please see Appendix-C.3 for proofs. We can observe that the optimization objective for π_{θ_G} and π_{θ_L} is totally disentangled in Theorem 3. Accordingly, the optimization objective for instances sampled from p_I is to minimize:

$$J(\theta_G, \theta_L) = \mathbb{E}_{(I,\bar{\mathcal{C}})\sim p_I, \pi^*_{\text{part}}}[L(\theta_G, \theta_L, \bar{\mathcal{C}})]. \tag{12}$$

This objective involves evaluating the performance of π_{θ_G} and π_{θ_L} on the sampled trajectories induced by π^*_{part}. However, π^*_{part} is practically unavailable for the NP-hard partition problem, since it is impossible to directly get supervised labels. Inspired by recent techniques known as self-imitation learning [28, 36], our goal is to acquire high-quality labeled instances from a behavioral policy $\hat{\pi}_{\text{part}}$. The working pipeline of $\hat{\pi}_{\text{part}}$ can be described as follows (see Figure 2(b)): At the global partition level, $\hat{\mathcal{C}}^{(0)}$ is first generated using beam search with π_{θ_G}. Then, at each local partition level $k \geq 1$, the partition solution is further refined locally using beam search with π_{θ_L} to obtain the ultimate partition solution $\hat{\mathcal{C}}^{(K)}$. Thus, during training, $\bar{\mathcal{C}} = \hat{\mathcal{C}}^{(K)}$ serves as the label. The practical loss function is thus defined as:

$$\hat{J}(\theta_G, \theta_L) = \mathbb{E}_{(I,\bar{\mathcal{C}})\sim p_I, \hat{\pi}_{\text{part}}}[L(\theta_G, \theta_L, \bar{\mathcal{C}})] + \text{reg}(\theta_G, \theta_L), \tag{13}$$

where $\text{reg}(\theta_G, \theta_L) = \lambda_G \frac{||\theta_G||^2}{2} + \lambda_L \frac{||\theta_L||^2}{2}$, with hyperparameters λ_G and λ_L. Therefore, this loss function is incorporated into the imitation learning algorithm to iteratively optimize π_{θ_G} and π_{θ_L}. In each iteration $n \geq 0$, the algorithm deploys $\hat{\pi}^n_{\text{part}}$ (which relies on θ^n_G and θ^n_L) and gathers the labeled instance $(I, \bar{\mathcal{C}})$. Online gradient updates are then executed based on estimated gradients to update θ^n_G and θ^n_L. Please refer to Appendix-B for the pseudocode.

Here, let us delve deeper into illustrating the training process for the global partition policy π_{θ_G} as a case study to elucidate

Table 1: Comparative results on uniformly distributed CVRP instances. OOM stands for out-of-memory. The symbol $*$ indicates that the results are obtained from the original paper. The notation $\downarrow$ indicates that a lower value is better.

Methods	CVRP1K			CVRP2K			CVRP5K			CVRP7K			CVRP10K		
	$Avg.\downarrow$	$Std.\downarrow$	$Time\downarrow$	$Avg.\downarrow$	$Std.\downarrow$	$Time\downarrow$	$Avg.\downarrow$	$Std.\downarrow$	$Time\downarrow$	$Avg.\downarrow$	$Std.\downarrow$	$Time\downarrow$	$Avg.\downarrow$	$Std.\downarrow$	$Time\downarrow$
LKH	42.17	0.80	14.18s	58.06	1.06	31.92s	126.59	2.81	6.80m	172.80	4.04	18.21m	240.23	5.42	41.29m
HGS	41.12	0.77	4.57m	56.24	1.07	12.68m	122.84	2.87	18.80m	165.37	3.95	20.15m	226.59	5.29	24.64m
AM (ICLR'19)	59.18	2.81	8.84s	OOM			OOM			OOM			OOM		
POMO (NeurIPS'20)	100.99	7.43	4.63s	255.13	30.02	39.35s	OOM			OOM			OOM		
Sym-POMO (NeurIPS'22)	98.04	2.86	5.71s	157.89	2.96	45.23s	OOM			OOM			OOM		
AMDKD (NeurIPS'22)	84.16	0.98	4.27s	188.75	4.39	34.39s	OOM			OOM			OOM		
Omni-POMO (ICML'23)	47.80	0.77	4.45s	74.01	1.05	35.86s	OOM			OOM			OOM		
ELG-POMO (IJCAI'24)	46.41	0.40	9.53s	66.07	0.55	67.19s	OOM			OOM			OOM		
INViT (ICML'24)	46.56	0.81	17.08s	64.94	1.09	36.67s	139.45	2.86	141.07s	186.57	3.93	4.81m	254.17	5.39	6.96m
LEHD (NeurIPS'23)	42.80	0.82	40.25s	60.48	1.12	72.48s	136.80	2.86	3.22m	188.11	4.00	6.52m	266.06	5.56	11.92m
BQ (NeurIPS'23)	43.12	0.80	4.75s	60.95	1.10	15.66s	137.14	3.00	79.89s	188.78	4.23	1.88m	265.81	5.97	3.30m
L2I (ICLR'20)	49.79	1.10	18.60s	95.58	5.44	44.88s	262.84	9.99	2.64m	506.73	25.25	3.61m	1263.23	4.00	4.07m
NLNS (ECAI'20)	53.51	0.83	12.08s	81.54	1.12	12.15s	180.84	2.99	12.62s	243.50	4.17	13.16s	331.27	5.53	13.97s
DACT (NeurIPS'21)	50.43	0.35	75.47s	71.17	0.51	5.40m	OOM			OOM			OOM		
L2D (NeurIPS'21)	46.45	0.87	4.67s	64.04	1.21	7.54s	135.09	3.02	16.11s	182.21	4.13	24.37s	246.38	5.55	27.59s
RBG* (KDD'22)	74.00	-	13.00s	137.60	-	42.00s		-			-			-	
TAM-AM* (ICLR'23)	50.06	0.98	0.76s	74.31	1.42	2.20s	172.22	-	11.78s	233.44	-	26.47s		-	
TAM-LKH* (ICLR'23)	46.34	0.84	1.82s	64.78	1.18	5.63s	144.64	-	17.19s	196.91	-	33.21s		-	
GLOP-G (AAAI'24)	47.21	0.90	0.73s	63.60	1.41	1.74s	141.67	3.67	2.37s	191.75	4.99	3.50s	266.96	6.46	13.74s
GLOP-LKH (AAAI'24)	46.03	0.99	0.78s	63.10	1.43	1.83s	140.51	3.72	4.31s	191.45	5.06	7.34s	267.50	6.50	16.47s
RL-driven HLGP	43.78	0.85	3.72s	59.58	1.17	10.03s	128.12	3.06	82.59s	173.71	4.39	1.96m	238.62	6.03	5.13m
SL-driven HLGP	**41.95**	0.78	8.31s	**57.67**	1.08	32.40s	**124.13**	2.79	97.27s	**166.73**	3.91	2.15m	**227.07**	5.25	3.39m

the intricacies of the SL algorithm for the partition problem. A similar analysis can be conducted for the local partition policy π_{θ_L}. The global partition policy π_{θ_G} requires to imitate the whole trajectory induced by the behavioral policy $\hat{\pi}_{\text{part}}$. Following the formulation in Equation 4, the global partition policy can directly output the node sequence. Subsequently, the log-probability of this node sequence in $\hat{\mathcal{C}}$ is maximized to update θ_G. This log-probability of the node sequence contains the product of a series of conditional probabilities, represented as $\log \prod_{t=1}^{N_{\text{sol}}} \pi_{\theta_G}(\bar{\mathcal{C}}[t]|\bar{\mathcal{C}}[0:t-1], I) = \sum_{t=1}^{N_{\text{sol}}} \log \pi_{\theta_G}(\bar{\mathcal{C}}[t]|\bar{\mathcal{C}}[0:t-1], I)$. This sum of log-probabilities prompts us to consider $(\bar{\mathcal{C}}[0:t-1], I)$ as an individual training sample, with its corresponding label being a singular $\bar{\mathcal{C}}[t]$. In this context, at timestep $t \geq 1$, the global partition policy is addressing a subproblem $I_{N_v(t)}$ determined by $(\bar{\mathcal{C}}[0:t-1], I)$, and it aligns with the same definition as in the RL setting. Hence, rather than generating the entire node sequence for behavioral imitation, the labeled instance $(I_{N_v(t)}, \bar{\mathcal{C}}[t])$ is fed to the global partition policy to imitate one-step behavior at each time step $t \geq 1$. In practice, we employ a variant Transformer model of BQ [5] as the backbones of π_{θ_G} and π_{θ_L}, which aligns with the analysis above. Therefore, we underscore the importance of viewing the subproblems encountered during training as individual training instances within both the contexts of RL and SL training paradigms.

5 EXPERIMENTS

5.1 Training and Evaluation Settings

In the training for both RL-driven and SL-driven HLGP, we adhere to the problem settings used in GLOP [42]. During the training phase, each CVRP instance consists of 1000 customer nodes distributed uniformly, with a vehicle capacity of 200. During the evaluation phase, our focus is on assessing the generalization capabilities of various models. Therefore, we utilize diverse datasets with varying scales and distributions. Each evaluation dataset can specify the number of customer nodes as 1000, 2000, 5000, 7000, or 10000. The customer nodes in each dataset are distributed according to a Uniform distribution, a Gaussian distribution, an explosion pattern, or a rotation pattern. Except for the dataset with 1000 customer nodes setting capacity as 200, the capacity for the other datasets is set to 300. Each dataset comprises 128 instances. The process of generating problem instances aligns with the methodologies outlined in [19, 45]. Please refer to Appendix-A.1 for more training details. Note that the code of our implementation, along with the Appendix, has been uploaded as the supplementary material.

During the evaluation phase, we compare our proposed RL-driven and SL-driven HLGP models with various methods. The classical heuristic methods include LKH3 [12] and HGS [37]. In learning-based constructive methods, AM [19], POMO [20], and Sym-POMO [18] serve as baselines trained with RL. AMDKD [1] and Omni-POMO [45] target generalization issues specifically. ELG-POMO [9] and INViT [6] harness local topological features. Within the realm of iterative methods, L2I [26], NLNS [13], and DACT [30] integrate RL-based policies with local operators to iteratively refine a given solution. In the context of the divide-and-conquer paradigm, RBG [46] and L2D [23] employ heuristic rules for the partition policy, while GPLC paradigms TAM [14] and GLOP [42] utilize neural partition policies. We adhere to the official implementations of these methods and the instructions provided by other works [6, 14, 42] that cite these methods to replicate the experimental results. For

Table 2: Comparative results on various distributed CVRP instances. "G" denotes the Gaussian distribution. "E" denotes the Explosion distribution. "R" denotes the Rotation distribution. The notation ↓ indicates that a lower value is better.

Methods	CVRP1K+G			CVRP1K+E			CVRP1K+R			CVRP2K+G			CVRP5K+E			CVRP7K+R		
	$Avg.\downarrow$	$Std.\downarrow$	$Time\downarrow$	$Avg.\downarrow$	$Std.\downarrow$	$Time\downarrow$	$Avg.\downarrow$	$Std.\downarrow$	$Time\downarrow$	$Avg.\downarrow$	$Std.\downarrow$	$Time\downarrow$	$Avg.\downarrow$	$Std.\downarrow$	$Time\downarrow$	$Avg.\downarrow$	$Std.\downarrow$	$Time\downarrow$
LKH3	32.52	1.21	37.35s	38.01	1.48	15.55s	37.50	1.33	15.35s	42.60	1.62	64.06s	103.45	4.39	6.67m	156.04	6.81	25.69m
HGS	31.84	1.19	15.57m	37.13	1.46	6.52m	36.62	1.32	7.78m	41.64	1.61	19.80m	101.16	4.40	16.53m	151.04	6.72	21.04m
AM (ICLR'19)	93.62	20.23	9.32s	58.99	4.79	8.74s	60.80	5.42	8.72s	OOM			OOM			OOM		
POMO (NeurIPS'20)	56.83	2.72	4.78s	74.88	4.84	4.54s	75.26	5.52	4.41s	101.75	7.32	38.31s	OOM			OOM		
Sym-POMO (NeurIPS'22)	97.59	5.35	5.57s	95.08	5.11	5.65s	106.88	6.11	5.53s	134.32	5.20	40.56s	OOM			OOM		
AMDKD (NeurIPS'22)	58.71	1.98	4.14s	71.10	2.04	4.17s	73.32	2.00	4.11s	108.11	3.82	32.96s	OOM			OOM		
Omni-POMO (ICML'23)	35.47	1.20	4.30s	41.80	1.47	4.30s	41.30	1.34	4.28s	51.02	1.76	35.31s	OOM			OOM		
ELG-POMO (IJCAI'24)	36.49	0.63	9.86s	41.64	0.75	9.67s	41.31	0.69	9.48s	49.34	0.86	68.73s	OOM			OOM		
INViT (ICML'24)	35.67	1.28	19.80s	42.11	1.53	19.80s	41.22	1.37	19.74s	47.31	1.75	46.92s	113.26	4.79	3.17m	169.38	7.47	4.66m
LEHD (NeurIPS'23)	33.99	1.23	36.27s	38.96	1.50	36.19s	38.44	1.36	36.16s	47.48	1.65	64.80s	116.70	4.38	2.85m	176.14	6.91	5.93m
BQ (NeurIPS'23)	34.71	1.25	3.88s	39.64	1.50	3.90s	39.17	1.39	3.91s	47.74	1.67	11.34s	120.23	4.75	72.12s	181.04	7.59	76.47s
L2I (ICLR'20)	37.42	1.28	13.56s	44.05	1.58	14.15s	43.56	1.41	14.01s	63.33	3.26	26.14s	204.51	10.31	2.10m	348.70	17.47	4.37m
NLNS (ECAI'20)	41.31	1.27	12.15s	46.52	1.51	12.15s	47.44	1.36	12.16s	60.38	1.89	12.22s	142.87	4.65	12.73s	221.69	6.51	13.02s
DACT (NeurIPS'21)	37.03	0.57	68.63s	43.10	0.66	67.92s	43.50	0.57	67.98s	49.30	0.83	4.52m		OOM			OOM	
L2D (NeurIPS'21)	35.26	1.24	2.60s	41.09	1.50	2.52s	40.40	1.37	2.63s	46.29	1.69	4.24s	108.95	4.63	10.15s	162.90	6.99	19.04s
GLOP-G (AAAI'24)	39.20	1.40	0.44s	43.44	1.63	0.43s	43.46	1.46	0.41s	50.55	1.97	1.88s	117.65	4.80	7.07s	178.37	6.84	7.98s
GLOP-LKH (AAAI'24)	38.71	1.42	1.22s	42.83	1.67	0.93s	42.80	1.49	0.77s	50.42	1.98	3.84s	117.04	4.83	9.85s	178.08	6.90	11.35s
RL-driven HLGP	34.58	1.26	3.47s	39.85	1.54	3.43s	39.36	1.38	3.47s	44.80	1.70	10.37s	106.27	4.52	70.80s	160.73	7.20	3.04m
SL-driven HLGP	**32.55**	1.21	7.21s	**37.96**	1.48	7.55s	**37.40**	1.34	7.41s	**42.85**	1.65	30.61s	**102.27**	4.47	84.12s	**152.47**	6.91	98.49s

RBG and TAM, we directly use the reported results from the papers. For a fair comparison, we only consider baseline methods that either have official code available for reproduction or have been extensively reported in previous papers. Further details are deferred to the Appendix-A.6. In addition, for comparison, the metrics include the average solution costs (*Avg.*), the standard deviation of solution costs (*Std.*), and average inference time (*Time*).

5.2 Performance Comparisons

In Table 1, our proposed RL-driven and SL-driven HLGP are compared with various previous methods on cross-size datasets. When compared to the state-of-the-art method, LEHD, RL-driven HLGP shows only a slight performance drop, notably on the CVRP1K dataset. Across other cross-size datasets, RL-driven HLGP consistently delivers superior solutions and is notably more efficient than LEHD. In comparison to all other learning-based baselines, SL-driven HLGP consistently demonstrates its superiority in terms of average cost while maintaining efficiency. Moreover, compared to classical heuristic solvers, SL-driven HLGP can swiftly produce high-quality solutions. In the most challenging case, CVRP10K, SL-driven HLGP stands out as the only method capable of generating high-quality solutions within 4 minutes for each instance. Additionally, owing to the adopted HL framework to mitigate compounded errors, both RL-driven HLGP and SL-driven HLGP outperform their respective baselines (GLOP and BQ).

Table 2 displays the comparison of our proposed methods with various existing methods on both cross-distribution datasets and cross-size and distribution datasets. When compared to BQ and LEHD, our RL-driven HLGP exhibits a minor performance decline on cross-distribution datasets. However, on the cross-size and distribution datasets, RL-driven HLGP consistently showcases superior generalization by efficiently producing improved solutions. In comparison to all previous learning-based methods, SL-driven HLGP consistently outperforms on both cross-size and cross-size and distribution datasets. Moreover, the performance of SL-driven HLGP closely approaches that of HGS while being dramatically more efficient. This justifies the use of supervised partition policy especially for larger instances. Please refer to the Appendix-A.3 and Appendix-A.4 for the hyperparameter studies, ablation studies, and visualization results.

6 CONCLUSION

In this work, we propose a novel hierarchical learning-based framework for the graph partition in CVRP. The global partition policy and local partition policy synergistically tackle the partition task to progressively alleviate the compounded misclusterings. Our methods adopt a unified objective function harmoniously compatible with both RL and SL training methods. Meanwhile, we thoroughly analyze the significance of treating the subproblems encountered during training as individual instances in both the RL and SL settings. Experimental results unequivocally demonstrate the generalization capability of proposed HLGP framework in finding low-cost CVRP solutions under distribution and scale shifts. In future, we plan to extend our HLGP to more different VRP variants like CVRPTW and Min-max CVRP, as well as other types of COPs.

ACKNOWLEDGMENTS

This research is supported by a generous research grant from Xiaoi Robot Technology Limited, the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG3-RP-2022-031), and the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1 grant.

REFERENCES

[1] Jieyi Bi, Yining Ma, Jiahai Wang, Zhiguang Cao, Jinbiao Chen, Yuan Sun, and Yeow Meng Chee. 2022. Learning generalizable models for vehicle routing problems via knowledge distillation. *Advances in Neural Information Processing Systems* 35 (2022), 31226–31238.

[2] Diego Cattaruzza, Nabil Absi, Dominique Feillet, and Jesús González-Feliu. 2017. Vehicle routing problems for city logistics. *EURO Journal on Transportation and Logistics* 6, 1 (2017), 51–79.

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

[4] Hanni Cheng, Haosi Zheng, Ya Cong, Weihao Jiang, and Shiliang Pu. 2023. Select and optimize: Learning to solve large-scale tsp instances. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 1219–1231.

[5] Darko Drakulic, Sofia Michel, Florian Mai, Arnaud Sors, and Jean-Marc Andreoli. 2024. Bq-nco: Bisimulation quotienting for efficient neural combinatorial optimization. *Advances in Neural Information Processing Systems* 36 (2024).

[6] Han Fang, Zhihao Song, Paul Weng, and Yutong Ban. 2024. INViT: A Generalizable Routing Problem Solver with Invariant Nested View Transformer. In *Forty-first International Conference on Machine Learning*.

[7] Marshall L Fisher and Ramchandran Jaikumar. 1981. A generalized assignment heuristic for vehicle routing. *Networks* 11, 2 (1981), 109–124.

[8] Zhang-Hua Fu, Kai-Bin Qiu, and Hongyuan Zha. 2021. Generalize a small pre-trained model to arbitrarily large tsp instances. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 7474–7482.

[9] Chengrui Gao, Haopu Shang, Ke Xue, Dong Li, and Chao Qian. 2024. Towards generalizable neural solvers for vehicle routing problems via ensemble with transferrable local policy. In *The 33rd International Joint Conference on Artificial Intelligence (IJCAI-24)*.

[10] Thierry Garaix, Christian Artigues, Dominique Feillet, and Didier Josselin. 2010. Vehicle routing problems with alternative paths: An application to on-demand transportation. *European Journal of Operational Research* 204, 1 (2010), 62–75.

[11] Nathan Grinsztajn, Daniel Furelos-Blanco, Shikha Surana, Clément Bonnet, and Tom Barrett. 2023. Winner takes it all: Training performant RL populations for combinatorial optimization. *Advances in Neural Information Processing Systems* 36 (2023), 48485–48509.

[12] Keld Helsgaun. 2017. An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University* 12 (2017), 966–980.

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

[14] Qingchun Hou, Jingwei Yang, Yiqiang Su, Xiaoqing Wang, and Yuming Deng. 2023. Generalize learned heuristics to solve large-scale vehicle routing problems in real-time. In *The Eleventh International Conference on Learning Representations*.

[15] Yuan Jiang, Zhiguang Cao, Yaoxin Wu, Wen Song, and Jie Zhang. 2024. Ensemble-based deep reinforcement learning for vehicle routing problems under distribution shift. *Advances in Neural Information Processing Systems* 36 (2024).

[16] Yuan Jiang, Zhiguang Cao, Yaoxin Wu, and Jie Zhang. 2023. Multi-view graph contrastive learning for solving vehicle routing problems. In *Uncertainty in Artificial Intelligence*. PMLR, 984–994.

[17] Minsu Kim, Jinkyoo Park, et al. 2021. Learning collaborative policies to solve np-hard routing problems. *Advances in Neural Information Processing Systems* 34 (2021), 10418–10430.

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

[19] Wouter Kool, Herke van Hoof, and Max Welling. 2019. Attention, Learn to Solve Routing Problems!. In *International Conference on Learning Representations*.

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

[21] Gilbert Laporte and Yves Nobert. 1983. A branch and bound algorithm for the capacitated vehicle routing problem. *Operations-Research-Spektrum* 5 (1983), 77–85.

[22] Andrew Levy, Robert Platt, and Kate Saenko. 2019. Hierarchical Reinforcement Learning with Hindsight. In *International Conference on Learning Representations*.

[23] Sirui Li, Zhongxia Yan, and Cathy Wu. 2021. Learning to delegate for large-scale vehicle routing. *Advances in Neural Information Processing Systems* 34 (2021), 26198–26211.

[24] Ling Liu, Kunpeng Li, and Zhixue Liu. 2017. A capacitated vehicle routing problem with order available time in e-commerce industry. *Engineering Optimization* 49, 3 (2017), 449–465.

[25] Qidong Liu, Chaoyue Liu, Shaoyao Niu, Cheng Long, Jie Zhang, and Mingliang Xu. 2024. 2D-Ptr: 2D Array Pointer Network for Solving the Heterogeneous Capacitated Vehicle Routing Problem. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*. 1238–1246.

[26] Hao Lu, Xingwen Zhang, and Shuang Yang. 2019. A learning-based iterative method for solving vehicle routing problems. In *International conference on learning representations*.

[27] Fu Luo, Xi Lin, Fei Liu, Qingfu Zhang, and Zhenkun Wang. 2024. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. *Advances in Neural Information Processing Systems* 36 (2024).

[28] Fu Luo, Xi Lin, Zhenkun Wang, Tong Xialiang, Mingxuan Yuan, and Qingfu Zhang. 2024. Self-Improved Learning for Scalable Neural Combinatorial Optimization. *arXiv preprint arXiv:2403.19561* (2024).

[29] Yining Ma, Zhiguang Cao, and Yeow Meng Chee. 2024. Learning to search feasible and infeasible regions of routing problems with flexible neural k-opt. *Advances in Neural Information Processing Systems* 36 (2024).

[30] Yining Ma, Jingwen Li, Zhiguang Cao, Wen Song, Le Zhang, Zhenghua Chen, and Jing Tang. 2021. Learning to iteratively solve routing problems with dual-aspect collaborative transformer. *Advances in Neural Information Processing Systems* 34 (2021), 11096–11107.

[31] Sahil Manchanda, Sofia Michel, Darko Drakulic, and Jean-Marc Andreoli. 2022. On the generalization of neural combinatorial optimization heuristics. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 426–442.

[32] Mohammadreza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takác. 2018. Reinforcement learning for solving the vehicle routing problem. *Advances in neural information processing systems* 31 (2018).

[33] Xuanhao Pan, Yan Jin, Yuandong Ding, Mingxiao Feng, Li Zhao, Lei Song, and Jiang Bian. 2023. H-tsp: Hierarchically solving the large-scale traveling salesman problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 9345–9353.

[34] Shubham Pateria, Budhitama Subagdja, Ah-hwee Tan, and Chai Quek. 2021. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)* 54, 5 (2021), 1–35.

[35] Ruizhong Qiu, Zhiqing Sun, and Yiming Yang. 2022. Dimes: A differentiable meta solver for combinatorial optimization problems. *Advances in Neural Information Processing Systems* 35 (2022), 25531–25546.

[36] Jiwoo Son, Minsu Kim, Hyeonah Kim, and Jinkyoo Park. 2023. Meta-sage: Scale meta-learning scheduled adaptation with guided exploration for mitigating scale shift on combinatorial optimization. In *International Conference on Machine Learning*. PMLR, 32194–32210.

[37] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, Nadia Lahrichi, and Walter Rei. 2012. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research* 60, 3 (2012), 611–624.

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

[39] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8 (1992), 229–256.

[40] Liang Xin, Wen Song, Zhiguang Cao, and Jie Zhang. 2020. Step-wise deep learning models for solving routing problems. *IEEE Transactions on Industrial Informatics* 17, 7 (2020), 4861–4871.

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

[42] Haoran Ye, Jiarui Wang, Helan Liang, Zhiguang Cao, Yong Li, and Fanzhang Li. 2024. Glop: Learning global partition and local construction for solving large-scale routing problems in real-time. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 20284–20292.

[43] Junzi Zhang, Jongho Kim, Brendan O'Donoghue, and Stephen Boyd. 2021. Sample efficient reinforcement learning with REINFORCE. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 10887–10895.

[44] Zhi Zheng, Changliang Zhou, Tong Xialiang, Mingxuan Yuan, and Zhenkun Wang. 2024. UDC: A Unified Neural Divide-and-Conquer Framework for Large-Scale Combinatorial Optimization Problems. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

[45] Jianan Zhou, Yaoxin Wu, Wen Song, Zhiguang Cao, and Jie Zhang. 2023. Towards omni-generalizable neural methods for vehicle routing problems. In *International Conference on Machine Learning*. PMLR, 42769–42789.

[46] Zefang Zong, Hansen Wang, Jingwei Wang, Meng Zheng, and Yong Li. 2022. Rbg: Hierarchically solving large-scale routing problems in logistic systems via reinforcement learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4648–4658.