Hierarchical Deep Reinforcement Learning for Vehicle Routing Problem

1st Qianyu Liu
Xi'an Jiaotong University
Xi'an, China

***Abstract*—The Vehicle Routing Problem (VRP) is a fundamental combinatorial optimization problem with broad applications in logistics, supply chain management, and transportation. Traditional approaches, including exact algorithms and metaheuristics, often struggle with scalability and computational efficiency when applied to large-scale VRP instances. In this work, we propose a novel Hierarchical Deep Reinforcement Learning (HDRL) framework designed to tackle large-scale VRP efficiently. Our method employs a two-level hierarchical model, where a high-level meta-controller partitions the problem space into manageable subproblems, and a lower-level controller generates high-quality routes for these subproblems using deep reinforcement learning. This hierarchical decomposition significantly reduces the complexity of solving large-scale instances while maintaining competitive solution quality. We evaluate our approach on benchmark VRP datasets with up to 10,000 customers and compare its performance against state-of-the-art solvers, including LKH-3 and OR-Tools. Experimental results demonstrate that HDRL achieves near-optimal solutions with significantly reduced computational time, making it a promising approach for solving real-world VRP instances at scale.**

***Index Terms*—Vehicle Routing Problem, Reinforcement Learning, Divide and Conquer, Markov Decision Process, Hierarchical Reinforcement Learning.**

I. Introduction

Vehicle routing problem(VRP) is a well-known question in combinatorial optimization with significant practical application, such as supply chain management, transportation and robotic routing. The complexity of VRP grows rapidly with the scale of the network and the intricacy of the constraints, classifying them as NP-hard problems that can not be solved efficiently by traditional algorithms.

Historically, a wide range of methods from exact algorithms to metaheuristic approaches have been employed to solve VRP. Exact algorithms like search algorithm [1] can provide optimal solution with an exponential time cost. Metaheuristic algorithms like evolutionary algorithm [2] and ant colony system [3] can costs much less time to find an approximate optimal solution. However, these methods are no longer commonly used due to the limitation of their scalability.

To overcome those limitations, recent advancements in deep reinforcement learning (DRL) have provided new choices for addressing VRP. Deep learning was first introduced in VRP [4] by supervised machine learning method with a seq2seq model. After that, reinforcement learning also shows a potential to drastically reduce inference time for VRP.

DRL methods enable the development of policies that learn directly from interaction with the environment, offering a powerful alternative to classical optimization techniques. However, directly applying DRL to VRP often struggle with the vast and intricate action spaces involved in vehicle routing, leading to more cost to find the optimal result. The adoption of hierarchical deep reinforcement learning(HDRL) is motivated by the inherent complexity and nature of multiple vehicle routing times.By decomposing the decision-making process into hierarchical layers, HDRL enables the high-level policy to focus on long-term strategic planning and gives context-specific tasks to lower-level controllers. This separation not only enhances learning efficiency and solution scalability but also improves the interpretability of the learned policies. These effects make HDRL a particularly powerful approach for solving large-scale VRP.

In this paper, we propose a novel HDRL framework tailored for VRP. Our algorithm integrates domain-specific insights with advanced reinforcement learning techniques to address the scalability and adaptability challenges of traditional methods. The experiment shows that with HDRL we can significantly scale the solvable size to 10k nodes and still remains a very close result with former SOTA approach LKH3 [11], [12] based on metaheuristic method. The key contributions of our work are as follows:

- We introduce a Hierarchical Deep Reinforcement Learning (HDRL) framework specifically tailored for large-scale VRP. By decomposing the decision-making process into hierarchical layers, our method effectively tackles the complexity of VRP, enabling both strategic planning and fine-grained optimization.
- Our approach employs an adaptive and dynamic decomposition mechanism in the upper-level model. This strategy interleaves decomposition with merging, allowing the meta-controller to continuously update its partitioning decisions based on the evolving partial solution and the distribution of remaining nodes, thus ensuring that subproblems integrate seamlessly into a high-quality overall solution.
- We design a robust lower-level controller that leverages state-of-the-art deep reinforcement learning techniques, including a Transformer-based encoder-decoder architecture. This model efficiently solves open-loop VRP subproblems with fixed endpoints, maintaining high solution quality while operating under minimal computational overhead.

II. Related Work

In this section, we will briefly list existing VRP researches with some related TSP work, since they share a lot of similarities to a certain extent.

A. Reinforcement learning in VRP

Recent advances in deep reinforcement learning (DRL) have paved the way for novel approaches to solving the Vehicle Routing Problem. Pioneering works such as Bello et al. [5] choosed pointer networks and policy gradient methods to construct solutions for routing problems. Subsequent research [6]–[8] extended these ideas, incorporating attention mechanisms to better capture the spatial and sequential relationships inherent in VRP. These methods learn to generate high-quality routes directly from data, often outperforming traditional methods on benchmark instances.

B. Divide and conquer in VRP

Divide and conquer method has been applied to TSP and VRP in both traditional algorithms [9], [10] and reinforcement learning methods [13]–[17]. Traditional algorithms often decompose the original large-scale problem into smaller, more manageable subproblems by clustering customer nodes based on geographical or demand-related criteria. These subproblems are then solved independently, and their solutions are merged to form a complete route for the entire network. Such methods not only reduce computational complexity but also enable the application of tailored heuristics and optimization techniques to localized regions.

C. Hierarchical Reinforcement learning

Hierarchical reinforcement learning (HRL) [19] is an advanced approach in reinforcement learning that decomposes complex tasks into a hierarchy of subtasks. In HRL, the overall decision-making process is to split the problem into multiple levels of abstraction. At the higher level, a high-level policy makes strategic decisions and manage sub-tasks. At the lower level, lower-level policies focus on executing the specific actions needed to achieve those goals. In VRP, the nodes are divided into separate groups by high-level manager to reduce problem size for low-level workers. Both levels are managed by deep reinforcement learning, which differs from former divide and conquer methods based on other algorithms at higher-level. Some TSP solver [15] used HRL, but no current work has introduced it into VRP so far.

III. Problem Definition

Given:

- A set of customers $C = \{c_1, c_2, \ldots, c_n\}$, where each customer c_i has a specific demand d_i.
- A depot c_0 where all vehicles begin and end their routes.
- A fleet of m vehicles $V = \{v_1, v_2, \ldots, v_m\}$, each with a maximum capacity Q.
- A cost matrix $D = [d_{ij}]$ representing the distance or cost between customer locations c_i and c_j.

The goal is to determine a set of routes $R = \{r_1, r_2, \ldots, r_m\}$, where each route r_k is a sequence of customer visits for vehicle v_k, such that:

1) Each customer is visited exactly once by one vehicle.
2) The total demand on each route does not exceed the vehicle's capacity:

$$\sum_{c_i \in r_k} d_i \leq Q \quad \forall k.$$

3) Each route starts and ends at the depot c_0.
4) The total cost across all routes is minimized:

$$\text{Minimize} \sum_{k=1}^{m} \sum_{i,j \in r_k} d_{ij}.$$

IV. Model Architecture

In this section, we proposed a two-level architecture based on HDRL method to solve large scale VRP. The model pipeline consists of both controller and meta controller, which are responsible for partitioning nodes and solving sub-problems respectively. Figure 1 illustrates the overall architecture of our model.

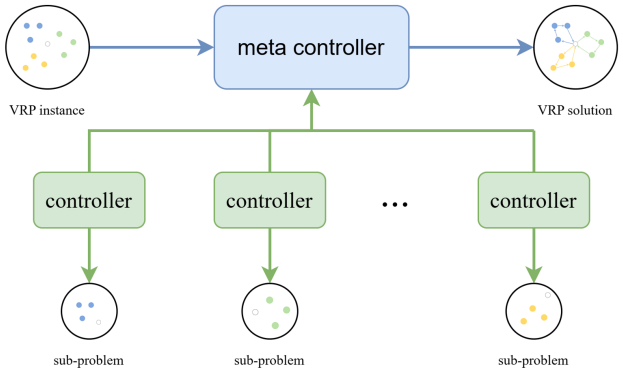

Fig. 1. Overall Architecture

When a new VRP instance is presented, the meta controller first processes the overall network information and strategically partitions the nodes into several clusters. Once partitioned, each cluster is forwarded to the controller, which uses a specialized reinforcement learning policy to solve the corresponding sub-problem. The sub-problem solution will be then passed back to the meta controller to merge into the final routes. Algorithm 1 illustrates the whole model procedure.

A. High-level model

Our upper-level model is designed to break down a large-scale VRP into manageable sub-problems that can be solved efficiently without compromising the final solution quality. To achieve this, we employ an adaptive and dynamic decomposition strategy. By interleaving decomposition and merging processes, our model continuously updates its strategy based on the evolving partial solution and the current distribution of remaining nodes. This approach enables the upper-level

Algorithm 1 Hierarchical VRP algorithm

Require: VRP instance $V = \{v_1, v_2, \ldots, v_N\}$, initial solution $\tau_{\text{init}} = \{v_d\}$
Ensure: Solution routes $\tau = \{\tau_1, \tau_2, \ldots, \tau_N\}$
$\tau \leftarrow \tau_{\text{init}}$, the nearest node v of v_d
while len$(\tau) < N$ **do**
  SubProb $\leftarrow$ GenerateSubProb(V, τ)
  SubSol $\leftarrow$ SolveSubProb(SubProb)
  $\tau \leftarrow$ MergeSubSol(SubSol, τ)
end while
return τ

Algorithm 2 Sub-problem Partition

Require: k-NN graph $G_{\text{kNN}} = (V, E)$, the partial solution at step t $\tau_t = \{v_1^t, v_2^t, \ldots\}$, length of the sub-problem subLength, maximum number of unvisited nodes maxNum, upper layer model UpperModel
Ensure: Sub-problem $P = \{v_1^s, v_2^s, \ldots, v_{\text{subLength}}^s\}$, two endpoints $v_s, v_t \in P$
$P \leftarrow \emptyset$, $S_v \leftarrow \tau_t$, $Q_{\text{new}} \leftarrow$ Deque()
UpperModel inputs G_{kNN} and τ_t, outputs Coordpred
v_c is the unvisited node closest to Coordpred
v_b is the visited node closest to v_c
Push v_b to the end of Q_{new}
while len$(P) \leq$ maxNum and Q_{new} is not empty **do**
  $v_i \leftarrow$ PopFront(Q_{new})
  for each $v_j \in \text{NG}_{kNN}(v_i)$ and $v_j \notin S_v$ **do**
    Push v_j to the end of Q_{new}
    Add v_j into S_v and P
  end for
end while
oldLength $\leftarrow$ subLength $-$ len(P)
$P_t \leftarrow$ SelectFragment$(\tau_t, v_b,$ oldLength$)$
$P \leftarrow P \cup P_t$
$v_s, v_t \leftarrow$ SetEndpoints(P_t)
return P, v_s, v_t

policy to make informed, context-aware decisions at every step, ensuring that the sub-solutions integrate seamlessly into a coherent and high-quality final route.

Sub-problem generation and merging are core components of our hierarchical framework. The action space of the upper-level policy is continuous with two dimensions, where each action corresponds to a coordinate in a unit grid. Given an action Coordpred, the process begins by selecting vc, the unvisited node closest to Coordpred, and then identifying vb, the nearest already visited node to vc. The sub-problem is expanded by exploring unvisited nodes from the neighborhood of vb using a k-Nearest Neighbor (kNN) graph; this expansion follows a breadth-first search strategy until the sub-problem reaches a predefined maximum size (maxNum) or all nodes have been either visited or selected. To ensure the sub-problem remains well-integrated with the existing route, a fragment of visited nodes centered around vb (SelectFragment) is appended, resulting in a segment with two endpoints. Solving this segment as an open-loop VRP yields a refined path that can be seamlessly merged with the overall solution.

We formalize this process within a Markov Decision Process (MDP) framework, denoted as $MG =< S, A, P, R, \gamma >$ for a given VRP instance $G = (V, E)$. In this formulation, S represents the set of all states, each corresponding to a possible path fragment τ, while the action space A is defined as $[0, 1] \times [0, 1]$, encompassing all points in a unit grid. The deterministic transition function $P : S \times A \to S$ captures the evolution of the state based on both upper-level and lower-level policies. The reward function $R : S \times A \times S \to$ is defined as $R(\tau, a, \tau^{)=L(\tau)-L(\tau^{)}}$, where L denotes the route length, thus quantifying the improvement from taking action a. With a discount factor γ set to 1, the MDP framework emphasizes the cumulative benefits of each decision throughout the routing process, ensuring that the upper-level policy is trained to generate and merge sub-problems in a manner that progressively refines the overall solution.

B. Low-level model

The lower-level model is designed to efficiently solve VRP with fixed nodes provided by the upper-level model. To ensure robust and rapid solutions for these smaller subproblems, we build on recent end-to-end approaches that have proven effective in solving VRP. By adapting their core ideas, our lower-level policy gives high-quality routes with minimal computational overhead, even when executed frequently. This integration not only enhances the efficiency of solving individual subproblems but also significantly contributes to the overall scalability and performance of our approach in tackling large-scale VRP.

The neural network behind our lower-level model is a Transformer, chosen for its proven effectiveness in capturing long-range dependencies in both natural language processing and computer vision. Our network comprises a Multi-Head Attention module and a Multi-Layer Perceptron (MLP) layer, augmented by a masking mechanism that eliminates invalid actions during node selection. The model follows an encoder-decoder architecture where the encoder leverages self-attention layers to encode the input node sequence, and the decoder generates the solution in an auto-regressive fashion.

The lower-level model's decision-making process is formalized as a Markov Decision Process (MDP) similar to that in prior works. Here, the state space S comprises all possible contexts defined by the extended encoder input. The action space A includes all nodes in the VRP, with a dynamic masking mechanism ensuring that visited nodes are excluded from future selections. The transition function P deterministically updates the state based on the chosen actions, while the reward function R assigns a reward equal to the negative cost of the route when a state corresponding to a complete and feasible solution is reached; otherwise, the reward remains 0. This MDP formulation ensures that the lower-level policy is continuously optimized to generate efficient and cost-effective routes under the constraints imposed by the VRP framework.

V. TRAINING

In our HDRL methods, the meta controller and controller use different reinforcement learning algorithms to tackle with their task which are Markov Decision Processes.

A. Upper-level model

The upper-level model leverages Proximal Policy Optimization (PPO) [19], a popular policy gradient method known for its balance between robust performance and stable policy updates. The optimize objective function defined as:

$$\hat{L}_{CLIP}(\theta) = E_t[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$$

Where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the probability ratio comparing the new policy π_θ to the old policy $\pi_{\theta_{old}}$ for action a_t in state s_t, $\hat{A}_t$ is an estimator of the advantage at time t, and ϵ is a small hyperparameter that controls the clipping range. This clipped objective ensures that updates do not deviate too far from the previous policy, thereby maintaining training stability while still allowing for effective policy improvement. Additionally, the overall loss function typically includes a value function loss and an entropy bonus to encourage exploration, further enhancing the robustness of the learning process in our hierarchical framework.

B. Lower-level model

The lower-level model is designed as an end-to-end approach to solve VRP that involve a relatively small number of nodes. It is trained using the classic REINFORCE algorithm [20] with a shared baseline, following the methodologies presented in prior works [21], [22] . In this setup, experience is collected via Monte Carlo sampling, and the policy gradient is computed as:

$$\begin{aligned}\nabla_\theta J(\theta) &= E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(\tau|s) A_{\pi_\theta}(\tau)] \\ &\approx \frac{1}{N}\sum_{i=1}^{N}(R(\tau_i) - b(s))\nabla_\theta \log \pi_\theta(\tau_i|s)\end{aligned}$$

where τ denotes a trajectory corresponding to a feasible solution of a VRP instance, and the reward $R(\tau_i) = L(\tau_i)$ is defined as the negative cost of the route τ_i. The shared baseline $b(s)$ is used to reduce the variance of the gradient estimates and improve training stability by averaging returns over trajectories generated from the same instance.

Together, these upper- and lower-level models form a hierarchical framework that effectively tackles large-scale VRPs. The upper-level model, optimized via PPO, is responsible for high-level decision-making and decomposing the problem, while the lower-level model efficiently solves the smaller subproblems generated during the decomposition process.

VI. EXPERIMENT

This paper focuses on the large-scale VRP problem. To demonstrate the effectiveness of our approach, we evaluate it on four datasets containing VRP instances with problem sizes of 1000, 2000, 5000, and 10000 customers, denoted as Random1000, Random2000, Random5000, and Random10000, respectively. Each dataset contains 16 VRP instances except Random1000, which contains 128 instances.

TABLE I
COMPARISONS WITH OTHER SOLVERS ON VRP

Algorithm	**Random1000**		
	Length (%)	**Gap (%)**	**Time (s)**
LKH-3	23.16	0.17	22.01
OR-Tools	24.23	4.82	104.34
HDRL	24.65	6.62	0.33

Algorithm	**Random2000**		
	Length (%)	**Gap (%)**	**Time (s)**
LKH-3	32.64	0.49	79.75
OR-Tools	34.04	4.82	532.14
HDRL	34.88	7.39	0.72

Algorithm	**Random5000**		
	Length (%)	**Gap (%)**	**Time (s)**
LKH-3	51.36	0.00	561.74
OR-Tools	53.35	3.86	5368.24
HDRL	55.01	7.10	1.66

Algorithm	**Random10000**		
	Length (%)	**Gap (%)**	**Time (s)**
LKH-3	72.45	0.00	4746.59
OR-Tools	74.95	3.44	21358.66
HDRL	77.75	7.32	3.32

Table 1 presents the performance comparison between HDRL, LKH-3, and OR-Tools across different VRP instance sizes. HDRL achieves competitive results, producing solutions close to LKH-3 in quality while being significantly faster. On the Random1000 dataset, HDRL achieves an optimality gap of 6.62% in only 0.33 seconds, while LKH-3 produces better-quality solutions with a gap of 0.17% but requires 22.01 seconds. Similarly, on Random10000, HDRL maintains an optimality gap of 7.32% while running in 3.32 seconds, whereas LKH-3 achieves better solution quality but takes 4746.59 seconds.

OR-Tools, while offering an efficient solution, tends to have a higher optimality gap than both HDRL and LKH-3. On Random5000, OR-Tools achieves a 3.86% gap with a runtime of 5368.24 seconds, whereas HDRL completes the task in only 1.66 seconds with a slightly higher gap of 7.10%. The efficiency of HDRL makes it particularly well-suited for large-scale and real-time VRP applications.

Overall, our experimental results confirm that HDRL balances solution quality and computational efficiency effectively, making it a strong candidate for solving large-scale VRP instances in real-world applications.

VII. Conclusion

In this paper, we introduced a Hierarchical Deep Reinforcement Learning (HDRL) framework for solving large-scale Vehicle Routing Problems. By leveraging a hierarchical structure, our method effectively decomposes complex VRP instances into smaller, more manageable subproblems while preserving solution quality. The upper-level meta-controller strategically partitions customer nodes, while the lower-level controller efficiently optimizes routes within each subproblem using a deep reinforcement learning-based approach. Our experiments demonstrate that HDRL achieves near-optimal solutions with substantial improvements in computational efficiency compared to traditional solvers such as LKH-3 and OR-Tools. The scalability of our approach enables the effective handling of VRP instances with up to 10,000 customers, making it highly suitable for real-world logistics applications. Future work will explore further refinements in subproblem partitioning strategies and adaptive learning mechanisms to enhance generalization and performance across diverse VRP variants.

References

[1] G. Araque, J. R. Kudva, T. L. Morin, et al., "A branch-and-cut algorithm for vehicle routing problems," *Annals of Operations Research*, vol. 50, pp. 37–59, 1994.

[2] C. Prins, et al., "A simple and effective evolutionary algorithm for the vehicle routing problem," *Computers & Operations Research*, 2004.

[3] M. Dorigo, et al., "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, 1997.

[4] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'15)*, vol. 2, MIT Press, Cambridge, MA, USA, 2015, pp. 2692–2700.

[5] Z. Zong, X. Tong, M. Zheng, and Y. Li, "Reinforcement learning for solving multiple vehicle routing problems with time window," *ACM Transactions on Intelligent Systems and Technology*, vol. 15, no. 2, Article 32, Apr. 2024, pp. 1–19.

[6] W. Kool, H. Van Hoof, and M. Welling, "Attention, learn to solve routing problems!" in *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*, 2019.'

[7] B. Peng, J. Wang, and Z. Zhang, "A deep reinforcement learning algorithm using dynamic attention model for vehicle routing problems," *arXiv preprint arXiv:2002.03282*, 2020.

[8] S. Foa, et al., "Solving the vehicle routing problem with deep reinforcement learning," *ArXiv*, 2022.

[9] F. Alesiani, G. Ermis, and K. Gkiotsalitis, "Constrained Clustering for the Capacitated Vehicle Routing Problem (CC-CVRP)," *Applied Artificial Intelligence*, vol. 36, no. 1, p. 1995658, 2022.

[10] E. D. Taillard and K. Helsgaun, "POPMUSIC for the travelling salesman problem," *European Journal of Operational Research*, vol. 272, no. 2, pp. 420–429, 2019.

[11] K. Helsgaun, "An extension of the Linkernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems," Technical report, 2017.

[12] K. Helsgaun, "An effective implementation of the Lin–Kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, 2000.

[13] Z.-H. Fu, K.-B. Qiu, and H. Zha, "Generalize a small pre-trained model to arbitrarily large TSP instances," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 8, 2021, pp. 7474–7482.

[14] H. Ye, J. Wang, H. Liang, Z. Cao, Y. Li, and F. Li, "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, Article 2261, 2024, pp. 20284–20292.

[15] X. Pan, Y. Jin, Y. Ding, M. Feng, L. Zhao, L. Song, and J. Bian, "H-TSP: hierarchically solving the large-scale traveling salesman problem," in *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI'23)*, vol. 37, AAAI Press, Article 1051, 2023, pp. 9345–9353.

[16] C. Gao, H. Shang, K. Xue, D. Li, and C. Qian, "Towards generalizable neural solvers for vehicle routing problems via ensemble with transferrable local policy," in *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI'24)*, Article 764, 2024, pp. 6914–6922.

[17] Q. Hou, J. Yang, Y. Su, X. Wang, and Y. Deng, "Generalize Learned Heuristics to Solve Large-scale Vehicle Routing Problems in Real-time," in *The Eleventh International Conference on Learning Representations*, 2023.

[18] T. D. Kulkarni, K. R. Narasimhan, A. Saeedi, and J. B. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS'16)*, Curran Associates Inc., Red Hook, NY, USA, 2016, pp. 3682–3690.

[19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[20] R. J. Williams, "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, 1992.

[21] W. Kool, H. Van Hoof, and M. Welling, "Attention, learn to solve routing problems!" in *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[22] Y. D. Kwon, J. Choo, B. Kim, I. Yoon, Y. Gwon, and S. Min, "POMO: Policy optimization with multiple optima for reinforcement learning," in *Advances in Neural Information Processing Systems*, Dec. 2020.