Multi-Depot Vehicle Routing Optimization Based on End-to-end Deep Reinforcement Learning (Postprint)

Authors: Lei Kun, Guo Peng, Wang Qixin, Zhao Wenchao, Tang Liansheng

Date: 2022-05-18T16:08:24Z

Abstract

To enhance the solution efficiency for the Multi-Depot Vehicle Routing Problem (MDVRP), we propose an end-to-end deep reinforcement learning framework. First, MDVRP is formulated as a Markov Decision Process (MDP), encompassing definitions of state, action, and reward. Additionally, we propose an improved Graph Attention Network (GAT) as the encoder to perform feature embedding encoding on the graph representation of MDVRP, and design a Transformer-based decoder. An improved REINFORCE algorithm is adopted to train the model. The model is agnostic to graph size; that is, once training is completed, it can be applied to solve problem instances with arbitrary numbers of depots and customers. Finally, the feasibility and effectiveness of the proposed framework are validated through both randomly generated instances and publicly available benchmark instances. Even for MDVRP instances with 100 customer nodes, the trained model requires only 2 milliseconds on average to obtain solutions superior to those of existing methods.

Full Text

Preamble

End-to-end Deep Reinforcement Learning Framework for Multi-Depot Vehicle Routing Problem

Lei Kun[1], Guo Peng[1],[3], Wang Qixin[1], Zhao Wenchao[1], Tang Liansheng[2]†

[1]School of Mechanical Engineering, Southwest Jiaotong University, Chengdu 610031, China;
[2]School of Economics & Management, Ningbo University of Technology, Ningbo, Zhejiang 315211, China;

[3]Technology & Equipment of Rail Transit Operation & Maintenance Key Laboratory of Sichuan Province, Chengdu 610031, China

Abstract: This paper proposes an end-to-end deep reinforcement learning framework to improve the efficiency of solving the Multi-Depot Vehicle Routing Problem (MDVRP). First, the MDVRP is formulated as a Markov Decision Process (MDP), including definitions of its state, action, and reward. An improved Graph Attention Network (GAT) is proposed as the encoder to perform feature embedding on the graph representation of the MDVRP, and a Transformer-based decoder is designed. The model is trained using an improved REINFORCE algorithm. The model is not constrained by graph size; once trained, it can solve problem instances with arbitrary numbers of depots and customers. Finally, the feasibility and effectiveness of the proposed framework are verified through randomly generated instances and publicly available benchmark instances. Even when solving MDVRP with 100 customer nodes, the trained model requires only 2 milliseconds on average to obtain solutions superior to existing methods.

Keywords: multi-depot vehicle routing problem; deep reinforcement learning; graph neural network; REINFORCE algorithm; Transformer model

0 Introduction

With the continuous growth of e-commerce and transportation industries, logistics has developed rapidly, experiencing explosive growth for over a decade in China and worldwide. For instance, in 2021, major logistics companies such as Cainiao, JD.com, and SF Express handled over 108.3 billion parcels nationwide. As the construction of new development patterns accelerates and logistics demand continues to grow, China' s logistics business volume will maintain rapid growth in the future. Meanwhile, the high-speed development of the logistics industry imposes higher requirements on large-scale real-time logistics scheduling systems. However, transportation and warehousing costs generated by logistics distribution remain high. Given the current state of logistics distribution and contemporary demands, seeking efficient logistics distribution models has attracted widespread attention from both academia and industry.

The Multi-Depot Vehicle Routing Problem (MDVRP) has extensive application scenarios, including transportation, logistics distribution, and express delivery. Exploring efficient solution methods for this problem holds important theoretical and practical significance for the development of China' s supply chain. The MDVRP is a variant of the Capacitated Vehicle Routing Problem (CVRP). Since CVRP is already NP-hard, MDVRP has an even larger solution space and thus also belongs to the NP-hard class.

Traditional methods for solving MDVRP mainly include exact algorithms, polynomial-time approximation algorithms, and metaheuristic algorithms.

Exact algorithms can obtain optimal solutions but are difficult to apply to problems with more than 50 customers due to their NP-hard nature [1]. Polynomial-time approximation algorithms can usually obtain solutions with quality guarantees, but their optimality guarantees are weak, and they may not even achieve locally optimal solutions. Metaheuristic algorithms, such as the wolf pack algorithm [2], ant colony optimization [3], bat algorithm [4], and variable neighborhood search [5], are widely used due to their high performance but typically require customization for specific problems and professional domain knowledge [6], and struggle to find good solutions for large-scale problems within polynomial time. The above three methods rarely exploit common features of optimization problems and often repeatedly solve instances of the same problem type, where the coefficient values in the objective function or constraints can be considered as sampled from the same underlying distribution [7]. Despite numerous solution strategies, there remains room for improvement in solution efficiency and a need for more efficient solution frameworks. Therefore, introducing learning-based methods to efficiently find near-optimal solutions is particularly important.

In recent years, increasing research has applied Deep Reinforcement Learning (DRL) techniques to solve combinatorial optimization problems, achieving breakthrough progress. Table 1 summarizes existing reinforcement learning methods for solving routing problems. Reinforcement learning can be further divided into model-based and model-free methods. Model-free reinforcement learning can be categorized into Value-based and Policy-based methods or their combination (Actor-critic). Additionally, routing problems can be classified into the Traveling Salesman Problem (TSP), CVRP, and MDVRP.

Table 1 Survey of deep/reinforcement learning methods in solving routing problems

Routing Problem	Literature	Network Structure	RL Method
TSP	[8]	Transformer	Model-free RL, Actor-Critic
TSP	[9]	NN	Model-free RL, Policy-Based
TSP	[10]	LSTM+Attention	Model-free hierarchical RL, Policy-Based
TSP	[11]	GAT+Attention	Model-free RL, Actor-Critic

Routing Problem	Literature	Network Structure	RL Method
TSP	[12]	GAT+GRU	Model-free RL, Actor-Critic
TSP	[13]	GAT+Attention	Model-free RL, Actor-Critic
TSP	[14]	Transformer	Model-free RL, Policy-Based
TSP	[15]	LSTM+Attention	Model-free RL, Policy-Based
TSP	[16]	LSTM+Attention	Model-free RL, Actor-Critic
CVRP	[8]	Transformer	Model-free RL, Actor-Critic
CVRP	[16]	LSTM+Attention	Model-free RL, Actor-Critic
CVRP	[17]	LSTM+Attention	Model-free RL, Actor-Critic
CVRP	[18]	LSTM+Attention	Model-free RL, Actor-Critic
CVRP	[19]	GAT+GRU	Model-free RL, Actor-Critic
CVRP	[20]	GAT+GRU	Model-free RL, Actor-Critic
CVRP	[13]	GAT+Attention	Model-free RL, Actor-Critic
CVRP	[21]	GAT+GRU	Model-free RL, Actor-Critic
MDVRP	[22]	Transformer	Model-free RL, Actor-Critic

Routing Problem	Literature	Network Structure	RL Method
MDVRP	[23]	Transformer	Model-free RL, Actor-Critic
MDVRP	This paper	RE-GAT+Transformer	Model-free RL, Actor-Critic

Note: In the Network Structure column, LSTM: long short-term memory; NN: neural networks; GRU: gate recurrent unit; GPN: graph pointer network.

Most DRL applications focus on routing problems such as TSP and VRP. Vinyals et al. [24] introduced a supervised learning framework using the sequence-to-sequence Pointer Network (PtrNet) model to solve combinatorial optimization problems like TSP. This model uses a recurrent architecture with Softmax attention mechanism (pointer) to select elements from the input sequence as output. Bello et al. [25] introduced an Actor-Critic style deep reinforcement learning algorithm to train PtrNet in an unsupervised manner for solving TSP, and its performance outperformed most previous approximation algorithms on TSP instances with up to 100 nodes. Nazari et al. [16] extended Bello' s framework to solve VRP.

Most combinatorial optimization problems, including VRP, have graph structures [6] that can be easily modeled using existing graph embedding or graph network embedding techniques to embed graph information into continuous node representations. Recent developments in Graph Neural Networks (GNN) can be used for network design due to their strong capabilities in information embedding and belief propagation over graph topologies [6]. However, the sequence-to-sequence neural network structures used in the above works cannot fully utilize and extract the graph structure information of the problem, such as node information containing customer location and demand, and edge information containing weight. As a powerful tool for processing non-Euclidean data and capturing graph structure information, GNN has been extensively studied in recent years.

In recent years, GNN-based approximate solvers have demonstrated significantly better algorithmic time complexity than traditional operations research algorithms after training. Li et al. [26] applied Graph Convolutional Network (GCN) models [27] with guided tree search algorithms to solve graph-based combinatorial optimization problems such as maximum independent set and minimum vertex cover. Dai et al. [7] encoded problem instances using GNN, which, compared to sequence-to-sequence models, has node order invariance and better reflects the combinatorial structure of TSP. They used DQN [28] to train the structure2vec graph embedding model [29]. Motivated by the Transformer architecture [30], Kool et al. [8] proposed an attention model to solve various combinatorial optimization problems and significantly improved results for small-scale routing problems using Rollout baselines in policy gradient algorithms. Nowak et al. [31] used deep GCN in a supervised learning manner to construct effective TSP graph representations and output tours through highly parallel beam search in a non-autoregressive way. Drori et al. [13] developed a new framework for solving combinatorial optimization problems on graphs using Graph Attention Networks (GAT) for numerous graph combinatorial optimization problems, claiming their framework has generalization capabilities from training on small graphs to testing on large graphs and from training on random graphs to testing on real-world graphs.

However, most machine learning methods focus on solving single-depot vehicle routing problems, with limited research on multi-depot vehicle routing problems. Wang et al. [23] designed a multi-agent reinforcement learning framework based on multi-head attention mechanism to solve MDVRP and trained it using policy gradient algorithms. Their experimental results demonstrated that the proposed multi-agent deep reinforcement learning model, combined with search strategies, could quickly obtain high-quality solutions. However, [23] did not verify the performance of their trained model in generalizing to instances of different scales or to real-world instances (standard benchmarks). In contrast, the encoder-decoder model proposed in this paper is not constrained by problem scale (i.e., number of depots and customers), meaning the trained model can be applied to instances with arbitrary numbers of depots and customers and can provide solutions within milliseconds. This framework possesses strong generalization performance. Its effectiveness is verified through testing on randomly generated datasets. Additionally, its generalization capability from training on random instances to testing on real-world instances is validated through VRPLIB standard benchmarks.

The above GNN-based learning methods motivate this paper to explore their potential in solving MDVRP. We propose an end-to-end deep reinforcement learning framework for efficiently solving MDVRP. In this framework, MDVRP is first modeled as a Markov Decision Process (MDP). We propose a Residual Edge Graph Attention Network (RE-GAT) model as the encoder to extract and embed state features from the graph representation of MDVRP, which is an improvement over Graph Attention Network (GAT). GAT only considers node information while neglecting edge information during graph structure information extraction, whereas edge features can provide more direct information related to optimization objectives (such as weighted distances). Additionally, simultaneously inputting node and edge information facilitates mining spatial adjacency relationships between different nodes. The proposed RE-GAT model fuses and updates information from both nodes and edges (such as weights) in the MDVRP graph representation and adds residual connections between layers to effectively prevent gradient vanishing and model degradation in deep models. Furthermore, a decoder based on the Transformer model is designed to efficiently predict nodes during the solution process. The proposed encoder-decoder model, once trained, can be applied to instances with arbitrary numbers of depots and customers and can provide route optimization solutions within milliseconds. In other words, this framework can serve as a real-time optimization framework with offline training and online testing.

To verify the feasibility and effectiveness of the framework, we design randomly generated instances for training and testing. Additionally, the framework' s generalization performance is tested using VRPLIB standard benchmarks and compared with state-of-the-art machine learning methods and metaheuristic algorithms to verify its superiority. Finally, experiments validate and analyze the time complexity of the framework during training and testing phases.

1 Problem Description

Generally, MDVRP can be described as a capacitated vehicle delivering goods to multiple customers with limited demand. The vehicle returns to the depot when its load is exhausted or cannot meet customer demand, with the objective of minimizing total route length while satisfying all customer demands. Figure 1 illustrates an MDVRP with two depots. This paper defines MDVRP through an undirected graph $G = (V, E, W)$, where nodes include customers and multiple depots $V = \{v_0, v_1, \dots, v_m\}$, and v_0 represents the depot node. Customer nodes $\{v_1, \dots, v_m\}$ have demand δ_i, where δ_0 represents the depot' s demand. Edge $e_{ij} \in E$ represents the edge from node i to node j, and W denotes the distance information. Both depot and customer node coordinates are randomly generated from the unit square $[0, 1] \times [0, 1]$. For problems with customer node counts of 20, 30, and 50, this paper randomly generates three corresponding vehicle capacities of 30, 40, and 50, respectively. Each customer node demand is randomly generated in $[0, 1]$, and customer node demands are normalized to $[0, 1]$, with vehicle capacity D correspondingly transformed to 3, 4, and 5.

This paper introduces a permutation $\hat{\pi}$ of all nodes. The objective is to find a solution $\hat{\pi}$ for a given problem instance such that each customer node is visited exactly once (depot nodes can be visited multiple times, i.e., $\hat{\pi}_t = \hat{\pi}_{t'}$ for some $t \neq t'$), and the total route length is minimized. The length of permutation $\hat{\pi}$ is defined as $L(\hat{\pi})$, representing the solution to the problem. The encoder in this paper contains L layers of RE-GAT. Figure 3 describes how a single-layer RE-GAT integrates edge information into node information and updates each node' s information. The attention coefficient α_{ij} represents the weight coefficient (attention coefficient) of node i relative to node j in layer l, where $\{l = 1, \dots, L\}$. The graph-attention model defines a stochastic policy $p(\hat{\pi}|s)$ for MDVRP instance s. Based on the chain rule of probability, the selection probability of sequence $\hat{\pi}$ can be calculated based on the parameter set θ of the graph-attention model:

$$p(\hat{\pi}|s) = \prod_{t=1}^{m} p(\hat{\pi}_t|\hat{\pi}_{1:t-1}, s; \theta)$$

where $\hat{\pi}_t$ denotes the node selected at time step t.

2 Markov Decision Process Definition for MDVRP

This section models the Markov Decision Process (MDP) for MDVRP, including definitions of state, action, and reward:

a) State: At time step t, the state consists of the visited nodes that constitute the partial solution $\hat{\pi}_{1:t-1}$.

b) Action: At time step t, the action is an unvisited customer node or depot node $\hat{\pi}_t \in \{1, \dots, m\}$.

c) Reward: First, calculate the distance between the two nodes visited from time step $t-1$ to time step t, $(\hat{\pi}_{t-1}, \hat{\pi}_t)$. Then define the agent' s immediate reward as:

$$r_t = -\|\hat{\pi}_{t-1} - \hat{\pi}_t\|_2$$

where $\|\cdot\|_2$ denotes the 2-norm. The reinforcement learning objective is to maximize cumulative reward, hence the negative value.

3.1 Encoder

The encoder takes graph $G = (V, E, W)$ as input, with its structure shown in Figure 2. Input node features are x_i, and input edge features are Euclidean distances e_{ij}. These two features are embedded into d-dimensional and d_e-dimensional features through fully connected layers (FC layers in Figure 2), respectively, and then fed into RE-GAT for encoding. Equations (3) and (4) describe the embedding processes for nodes and edges:

$$x_i^{(0)} = \mathrm{BN}(W_A x_i + b_A), \quad i = 0, 1, \dots, m$$

$$e_{ij}^{(1)} = \mathrm{BN}(W_B e_{ij} + b_B), \quad i, j = 1, \dots, m$$

where W_A and W_B are learnable weight matrices, b_A and b_B are learnable weight vectors, and $\mathrm{BN}(\cdot)$ denotes batch normalization [32].

RE-GAT' s each layer updates each node' s feature vector through the attention mechanism described by Equations (5) and (7). The model uses residual connections between every two layers (represented by Equation (6)). That is, the output of layer l is calculated as:

$$x_i^{(l)} = x_i^{(l-1)} + \mathrm{BN}\left(\sum_{j=1}^{m} \alpha_{ij}^{(l)} W_1^{(l)} x_j^{(l-1)}\right)$$

where the attention coefficient $\alpha_{ij}^{(l)}$ is computed as:

$$\alpha_{ij}^{(l)} = \frac{\exp\left(\sigma\left(g(W_2^{(l)}[x_i^{(l-1)} \| x_j^{(l-1)} \| e_{ij}^{(l)}])\right)\right)}{\sum_{k=1}^{m} \exp\left(\sigma\left(g(W_2^{(l)}[x_i^{(l-1)} \| x_k^{(l-1)} \| e_{ik}^{(l)}])\right)\right)}$$

where $W_1^{(l)}$ and $W_2^{(l)}$ are learnable weight matrices, $g(\cdot)$ is the LeakyReLU activation function, and $\|$ denotes the concatenation operator.

The L-th layer RE-GAT outputs the final embedded feature vector $x_i^{(L)}$ for each node. These are used to compute the final graph embedding vector h and the final node embeddings. For each node i, the final embedding is represented by Equation (8):

$$h_i = \mathrm{BN}(W_3 x_i^{(L)} + b_3)$$

where W_3 and b_3 are learnable parameters. The encoder embeds the raw features of all input nodes into high-dimensional features.

3.2 Decoder

The decoder selects a node at each time step t based on the encoder' s output, thereby generating a permutation $\hat{\pi}$ of input nodes. The decoding process proceeds sequentially. At time step t, the context vector c_t is first calculated using the graph embedding vector h, the embedding vector of the node selected at time $t-1$, and the vehicle' s remaining capacity D_t:

$$c_t = W_c[h \| x_{\hat{\pi}_{t-1}}^{(L)} \| D_t]$$

where W_c is a learnable weight matrix, and D_t represents the vehicle's remaining capacity at two consecutive decoding steps. The update formula for D_t is:

$$D_t = \begin{cases} D_{t-1} - \delta_{\hat{\pi}_{t-1}}, & \text{if } \hat{\pi}_{t-1} \text{ is a customer node} \\ D, & \text{if } \hat{\pi}_{t-1} \text{ is a depot node} \end{cases}$$

The first layer of the decoder takes the context vector $c_t^{(0)}$ as input and produces a new context vector $c_t^{(1)}$. Specifically, this context vector is obtained through a multi-head attention mechanism with H heads. Equation (11) describes the multi-head attention mechanism, which computes query vectors q_i, key vectors k_i, and value vectors v_i using the node embedding vectors output by the encoder and the context vector:

$$q_i = W_q c_t^{(0)}, \quad k_i = W_k x_i^{(L)}, \quad v_i = W_v x_i^{(L)}, \quad i = 1, 2, \dots, m$$

where W_q, W_k, and W_v are learnable weight matrices.

The first decoding layer' s attention coefficients $u_{i,t}^{(1)}$ are computed using the query vectors q_i and key vectors k_i from the encoder' s node embeddings. For each head $h \in \{1, \dots, H\}$, the attention coefficient is calculated as:

$$u_{i,t}^{(1),h} = \frac{(q_i^h)^T k_i^h}{\sqrt{d_k}}$$

These are then concatenated and passed through a fully connected layer to obtain the final context vector $c_t^{(1)}$:

$$c_t^{(1)} = W_f \left\|_{h=1}^{H} \left(\sum_{i=1}^{m} \text{softmax}(u_{i,t}^{(1),h}) v_i^h \right) \right\|$$

where W_f is a learnable weight matrix. This multi-head attention mechanism helps improve the stability of the attention learning process [33].

The second layer of the decoder is based on single-head attention, taking the context vector $c_t^{(1)}$ as input. The attention coefficient at time t for the second decoding layer is calculated using Equation (15):

$$u_{i,t}^{(2)} = \begin{cases} \tanh\left(\frac{(c_t^{(1)})^T k_i}{\sqrt{d_k}}\right), & \text{if node } i \text{ is feasible} \\ -\infty, & \text{otherwise} \end{cases}$$

Based on the work of Bello et al. [25], the tanh activation function is used to clip the coefficient within $[-C, C]$ (this paper selects $C = 10$). Then, the selection probability $p_{i,t}$ for each node is obtained using the Softmax activation function via Equation (16):

$$p_{i,t} = \frac{\exp(u_{i,t}^{(2)})}{\sum_{j=1}^{m} \exp(u_{j,t}^{(2)})}$$

Finally, based on the policy probability distribution $p_{i,t}$, the next node to visit (depot or customer) is predicted using sampling or greedy decoding (described below).

Masking is used to avoid repeatedly selecting customer nodes, selecting customer nodes that exceed vehicle remaining capacity, and selecting depot nodes consecutively (i.e., selecting a depot node at time t when $\hat{\pi}_{t-1}$ is also a depot). Specifically, in Equation (12), the attention coefficients for these cases are set to $-\infty$ to mask them. Then, the attention coefficients $u_{i,t}^{(1)}$ are normalized using the Softmax activation function via Equation (13).

Traditional heuristic algorithms typically adopt a "cluster-first-route-second" approach [23] to solve MDVRP, which has the following disadvantages [23]: (a) different clusters are routed separately, leading to a lack of overall correlation between clusters; (b) the quality of grouping in heuristic methods often determines the quality of overall planning, and the formulation of grouping rules requires expert domain knowledge, making it difficult for manually selected grouping rules to achieve optimal results. In contrast, deep reinforcement learning agents can interact with the scheduling environment in a data-driven manner, continuously evolving their strategies to maximize reward (the negative of total route length for routing problems). That is, during the decoding process, the reinforcement learning policy can select scheduling centers without relying on manual heuristic intervention.

4 Deep Reinforcement Learning Algorithm Based on Policy Gradient

This paper introduces an improved REINFORCE algorithm to train the proposed model. The loss function is defined as:

$$\mathrm{Loss}(\theta) = \mathbb{E}_{\hat{\pi}\sim p(\cdot|s;\theta)}[L(\hat{\pi}) - b]$$

where b is the baseline. The improved REINFORCE algorithm has an actor-critic style but differs from traditional critics that use state-value estimation functions. Compared to the original version, this implementation adds the Rollout baseline method [8] to accelerate convergence and enhance optimization performance. The gradient calculation process for the loss function is:

$$\nabla_\theta \mathrm{Loss}(\theta) = \mathbb{E}_{\hat{\pi}\sim p(\cdot|s;\theta)}[(L(\hat{\pi}) - b)\nabla_\theta \log p(\hat{\pi}|s;\theta)]$$

In the proposed REINFORCE algorithm with Rollout baseline, the critic network is replaced by a baseline actor. The algorithm' s flow diagram is shown in Figure 4. The algorithm can be described as having a structure with two actors. The baseline actor' s policy network $\pi_{\theta_{BL}}$ (where θ_{BL} is the parameter set) is fixed within each epoch (i.e., its parameters are not updated), similar to the fixed target Q-network in DDQN [34]. At the end of each epoch, greedy decoding is used to compare the results of the current training actor and the baseline actor. The baseline actor's policy network parameters are only updated when there is significant improvement on test instances (tested with a t-test at significance level $\alpha = 5\%$). During training, "code-level optimization" strategies are also employed to improve algorithm performance, including learning rate decay for the Adam optimizer and reward normalization.

Effective combinatorial optimization search algorithms mainly include beam search, neighborhood search, and tree search. Bello et al. [25] proposed search strategies such as sampling, greedy search, and active search. This paper uses the following two decoding strategies:

a) Greedy Decoding: Generally, greedy algorithms construct locally optimal solutions and provide fast approximations of global optimal solutions. At each decoding step, the node with the highest probability is selected greedily. The search terminates when all customer demands are satisfied, thereby constructing a valid solution.

b) Sampling Decoding: Nodes are selected randomly according to the probability distribution to construct valid solutions. During testing, Bello et al. [25] utilized a temperature hyperparameter λ to modify Equation (16) to ensure sampling diversity. The modified formula is:

$$p_{i,t} = \frac{\exp(u_{i,t}^{(2)}/\lambda)}{\sum_{j=1}^{m} \exp(u_{j,t}^{(2)}/\lambda)}$$

Through grid search of the temperature hyperparameter, values of 2.5, 1.8, and 1.2 were found to work best for MDVRP20 (20 customer nodes), MDVRP50, and MDVRP100, respectively.

During training, the model typically needs to explore the environment to achieve better performance, so a random sampling decoding strategy is adopted. During testing, this paper uses greedy decoding. Additionally, following existing research testing methods [8][25], 1280 solutions are obtained through random sampling and the best one is reported.

5 Computational Experiments

This section verifies the feasibility and effectiveness of the proposed framework through experiments. Experiments include training and testing phases. Since training requires large amounts of data, training data is generated from uniform random distributions. Test datasets include randomly generated instances and publicly available standard benchmarks (proposed by Cordeau et al. [35, 36]), which are used to test the effectiveness and generalization performance of the framework, respectively. The proposed framework is also compared with other learning-based methods, Google OR-tools, and metaheuristic algorithms.

5.1 Dataset and Hyperparameter Selection

For readability, MDVRP scale is represented in the format "number of customers-number of depots." The datasets used include randomly generated training data, validation data, and random test data. MDVRP instances are randomly generated from the unit square $[0, 1] \times [0, 1]$ for scales 20-2, 50-2, and 100-2. The training set contains 819,200 instances for scale 20-2 and 768,000 instances each for scales 50-2 and 100-2, with each model trained for 100 epochs. For validation and random test data, the same distribution as training data is used, generating 10,000 instances for each scale. Additionally, publicly available standard benchmarks (proposed by Cordeau et al. [35, 36])

are used to evaluate the model' s generalization performance from training on random instances to testing on real-world instances and across different scales (varying numbers of depots and customers). All experiments are conducted on a computer with an Intel Turbo HT (100W) DDR4-2400 CPU and an Nvidia GeForce RTX 3090 GPU. Table 2 lists other relevant hyperparameter values for the training process. The model is constructed using PyTorch and implemented with Python 3.7.

Table 2 Value of hyper-parameters

Hyperparameter	Value
Encoder layers	3
Learning rate decay factor	0.96
Multi-head attention heads	8
Node embedding dimension	128
Edge embedding dimension	64

5.2.1 Random Instance Analysis

Table 3 lists the test results of the proposed framework (denoted as "Greedy," "Sampling128," and "Sampling1280" based on different decoding strategies), Google OR-tools, and another DRL method [23] on randomly generated MDVRP instances of different scales. The distances (lower is better) and relative optimality gap values are averages over 10,000 instances. Additionally, the average computation time for all test instances is provided.

Table 3 Results of the proposed framework, a reinforcement learning method and Google OR-tools on random generated MDVRP instances

Method	MDVRP20-2	MDVRP50-2	MDVRP100-2
	Distance	Gap	Time
Greedy[23]	6.23	13.68%	43.9 ms
Greedy (Ours)	5.52	1.71%	0.18 ms
OR-tools	5.54	1.71%	19 ms
Sampling128[23]	5.89	8.23%	1.8 ms
Sampling128 (Ours)	5.44	0.00%	0.21 s
Sampling1280 (Ours)	5.44	0.00%	91 ms

The table shows that sampling decoding strategies obtain the best solutions among all methods. This strategy performs 128 or 1280 samplings per instance, constructing 128 or 1280 solutions and reporting the best one. In contrast, greedy decoding constructs a single solution by greedily selecting the node with the highest probability at each decoding step using the trained model. Additionally, neural network-based parallel computation enables batch processing of multiple instances, making the trained model extremely fast with greedy decoding. For example, for 10,000 MDVRP instances of scale 100-2, the proposed method requires only 1.8 ms per instance with greedy decoding, while the sampling decoding approach requires 1.58 s.

Google OR-tools is an efficient solver based on local search, and [23] is a deep reinforcement learning method. However, on all scales of MDVRP, the proposed framework outperforms both OR-tools and the method in [23] regardless of whether greedy or sampling decoding is used, and greedy decoding is far superior in running time. Furthermore, Figure 5 shows the convergence curves for MDVRP of various scales during training, demonstrating that all scales converge well after 80 epochs.

5.2.2 Public Benchmark Analysis

To evaluate the proposed framework' s generalization from randomly generated instances to real-world instances and across different scales (i.e., different numbers of customers and depots), the trained model (trained on randomly generated MDVRP instances of scale 100-2) is used to solve publicly available standard benchmarks for MDVRP proposed by Cordeau et al. [35, 36] (with 50-160 customers). All results are listed in Table 4. To further evaluate performance, comparisons are made with a state-of-the-art metaheuristic algorithm (improved ACO [38]), with results also shown in Table 4. Gap values for both methods are calculated based on the best known solutions (BKS).

Table 4 Results of the proposed framework and improved ACO on benchmarks

Problem	Customers	Greedy Gap	Greedy Time	Improved ACO[38] Gap	Improved ACO[38] Time
p01	50	6.23%	0.06 s	5.34%	1.7 s
p02	50	4.18%	0.06 s	4.61%	1.8 s
p03	50	6.63%	0.09 s	4.62%	4.3 s
p04	100	10.28%	0.10 s	1.97%	28.4 s
p05	100	7.96%	0.10 s	0.09%	25.6 s
p06	100	9.72%	0.09 s	3.01%	31.9 s
p07	100	10.64%	0.09 s	2.46%	30.9 s
p08	160	0.82%	0.04 s	0.00%	15.4 s
p09	160	0.82%	0.04 s	0.00%	16.0 s
p10	160	0.00%	0.04 s	0.41%	16.9 s
p11	160	3.24%	0.15 s	1.94%	167.1 s
p12	160	0.56%	0.15 s	1.32%	188.1 s
p13	160	0.00%	0.15 s	0.00%	147.3 s
Average		**4.97%**	**0.09 s**	**1.98%**	**51.59 s**

Although the improved ACO algorithm outperforms the proposed framework on most instances and has a better average gap (1.98% vs. 4.97%), the improved

ACO method runs 100 times per instance and reports the best result, with times shown as averages. In contrast, the proposed framework uses greedy decoding to solve each instance once. The proposed framework is far superior in running time (0.09 s vs. 51.59 s). Therefore, the proposed framework achieves a good balance between solution quality and running time. Moreover, the framework can serve as a real-time solver with offline training and online testing.

5.2.3 Extended Computational Analysis (CVRP)

To further evaluate the proposed framework' s performance and generalization capability for vehicle routing problems in different scenarios, the framework is applied to solve the single-depot CVRP. This section uses random instances for experimental verification, generating 10,000 instances from uniform distributions (consistent with [8][16]). The proposed framework is compared with existing learning-based methods (both "PtrNet" and "AM" in Table 5 are deep reinforcement learning methods), Google OR-tools, the Gurobi exact solver, and the LKH3 solver to verify its effectiveness on single-depot CVRP. Test results for different scales of CVRP instances are listed in Table 5.

Table 5 Results of different methods on random generated CVRP instances

Method	Type	VRP20	VRP50	VRP100
		Gap	Time	Gap
Gurobi	Solver	0.00%	7.2 ms	0.00%
Concorde	Solver	0.58%	0.1 ms	9.78%
LKH3	Solver	8.03%	0.2 ms	5.86%
PtrNet [16]	SL, BS	4.97%	36 ms	4.81%
AM [8]	RL, G	5.41%	84 ms	9.01%
Greedy (Ours)	RL, G	4.92%	0.2 ms	7.46%
OR Tools	H, G	2.49%	0.1 ms	2.40%
PtrNet [16] Sampling	RL, S	1.47%	0.1 ms	1.54%

Note: In the Type column, RL: reinforcement learning, H: heuristic, SL: supervised learning, S: sampling/search, G: greedy search, BS: beam search, "-" : cannot solve in reasonable time.

The results in Table 5 are categorized into three groups: solvers, greedy methods, and sampling/search methods. Except for the results of the proposed model, all other results are taken from Kool et al. [8]. In terms of model performance, the proposed framework achieves better results than other listed learning-based methods under both sampling and greedy decoding strategies. These results demonstrate that the proposed framework has good generalization performance in single-depot CVRP scenarios and verifies its potential for transfer to other VRP scenarios.

5.2.4 Framework Computational Time Complexity Analysis

Next, the relationship between the proposed model' s running time and graph scale (i.e., number of nodes) during training and testing phases is evaluated by solving TSP problems (most literature [7][13] analyzes time complexity through TSP; for comparison, this paper also uses TSP). All instances used in this section are randomly generated from uniform distributions (consistent with [7][13]).

For the training phase, training time depends not only on graph scale but also on the amount of training data and batch size. For generality, the running time for a single epoch is tested with 10,000 training instances and the same batch size (128) while increasing the number of nodes from 1 to 100. Figure 6(a) shows that the proposed framework' s running time grows linearly with the number of graph nodes during training.

For the testing phase, the running time of the entire encoder-decoder model is tested as the graph scale (number of nodes) increases from 1 to 500. Figure 6(b) shows that the proposed model' s running time grows linearly with graph scale during testing. Table 6 summarizes the time complexity, running time (average), and average optimality gap for several methods including exact algorithms, heuristic algorithms, and learning-based methods on TSP with 100 nodes. Except for the proposed framework' s results, all others are taken from Table 1 in Drori et al. [13]. The proposed framework' s results are listed in bold.

Table 6 Running time complexity of each method

Method	Time Complexity	Running Time (ms)	Gap
Gurobi	$O(n^3)$	3,220	0.00%
Concorde	$O(n^2)$	254.1	0.58%
Christofides	$O(n^2)$	5,002	24.5%
2-opt	$O(n^2)$	2,879	30.08%
Farthest Insertion	$O(n^2)$	8.35	8.4%
Nearest Insertion	$O(n^2)$	9.35	8.4%
S2V-DQN [7]	$O(n^2)$	61.72	8.4%
GAT [13]	$O(n)$	1.17	1.06%
Greedy (Ours)	$O(n)$	1.06	1.06%

Exact algorithms, approximation algorithms, and heuristic algorithms have at least quadratic time complexity in graph scale. S2V-DQN [7] is a reinforcement learning method with $O(n^2)$ time complexity and a large optimality gap (8.4%). The proposed framework has $O(n)$ time complexity, achieving significant improvements in both running time and optimality gap compared to $O(n^2)$ methods. GAT [13] has the same time complexity as the proposed framework but with a larger optimality gap.

6 Conclusion

This paper proposes an end-to-end deep reinforcement learning framework to improve the efficiency of solving MDVRP. The MDVRP is modeled as a Markov Decision Process, an improved graph attention network is designed as the encoder to encode state information during the solution process, and a decoder model based on the Transformer is designed. An improved REINFORCE algorithm is designed to train the proposed encoder-decoder model. The designed framework is not constrained by problem scale; once trained, it can be applied to instances of different scales (different numbers of customers and depots). To verify feasibility and effectiveness, numerical experiments are conducted on randomly generated instances and publicly available standard benchmarks, with comparisons to existing learning-based methods, Google OR-tools, and metaheuristic algorithms. Computational results demonstrate the feasibility and efficiency of the proposed framework for solving vehicle routing problems of different scales and scenarios.

This paper considers MDVRP in static environments. However, in actual logistics transportation, the environment is constantly changing, facing situations with dynamically arriving orders. Based on the rapid solution capability of the proposed framework, it has potential for real-time vehicle scheduling in dynamic environments. Therefore, future research will focus on building an end-to-end deep reinforcement learning framework for dynamic environments, solving routing problems through offline training and online testing.

References

[1] Sharma N, Monika. A Literature Survey on Multi Depot Vehicle Routing Problem [J]. International Journal for Scientific Research Development, 2015, 3 (4): 1752-1757.

[2] Ye Yong, Zhang Huizhen. Wolf pack algorithm for multi-depot vehicle routing problem [J]. Application Research of Computers, 2017, 34 (9): 2590-2593.

[3] Hu Rong, Chen Wenbo, Qian Bin, et al. Learning ant colony algorithm for green multi-depot vehicle routing problem [J]. Journal of System Simulation, 2021, 33 (9): 2095-2108.

[4] Qi Yuanhang, Cai Yanguang, Cai Hao, et al. Voronoi diagram-based discrete bat algorithm for multi-depot vehicle routing problem [J]. Control Theory & Applications, 2018, 35 (8): 1142-1150.

[5] Li Yang, Hu Rong, Qian Bin, et al. Two-stage algorithm for multi-depot vehicle routing problem [J]. Information and Control, 2020, 49 (6): 752-760.

[6] Bengio Y, Lodi A, Prouvost A. Machine learning for combinatorial optimization: A methodological tour d’ horizon [J]. European Journal of Operational Research, 2021, 290 (2): 405-421.

[7] Khalil E, Dai H, Zhang Y, et al. Learning combinatorial optimization algorithms over graphs [J]. In Advances in Neural Information Processing Systems, 2017, 30: 6348-6358.

[8] Kool W, Van H H, Welling M. Attention, Learn to Solve Routing Problems! [C]// proceedings of the International Conference on Learning Representations, 2018.

[9] Malazgirt G A, Unsal O S, Kestelman A C. Tauriel: Targeting traveling salesman problem with a deep reinforcement learning inspired architecture [J/OL]. 2019, arXiv preprint arXiv: 1905. 05567.

[10] Ma Q, Ge S, He D, et al. Combinatorial Optimization by Graph Pointer Networks and Hierarchical Reinforcement Learning [J/OL]. 2019, arXiv: 1911. 04907.

[11] Cappart Q, Moisan T, Rousseau L M, et al. Combining Reinforcement Learning and Constraint Programming for Combinatorial Optimization [J/OL]. 2020, arXiv: 2006. 01610.

[12] Subramaniam V, Lee G K, Ramesh T, et al. Machine Selection Rules in a Dynamic Job Shop [J]. The International Journal of Advanced Manufacturing Technology, 2000, 16 (12): 902-908.

[13] Drori I, Kharkar A, Sickinger W R, et al. Learning to Solve Combinatorial Optimization Problems on Real-World Graphs in Linear Time [J/OL]. 2020, arXiv: 2006. 03750.

[14] Zhang R, Prokhorchuk A, Dauwels J. Deep Reinforcement Learning for Traveling Salesman Problem with Time Windows and Rejections [C]// International Joint Conference on Neural Networks (IJCNN), 2020: 1-8.

[15] Hu Y, Yao Y, Lee W S. A reinforcement learning approach for optimizing multiple traveling salesman problems over graphs [J]. Knowledge-Based Systems, 2020, 204: 106244.

[16] Nazari M, Oroojlooy A, Snyder L, et al. Reinforcement learning for solving the vehicle routing problem [C]// Advances in Neural Information Processing Systems, 2018: 9839-9849.

[17] Chen X, Tian Y. Learning to perform local rewriting for combinatorial optimization [C]// proceedings of the Advances in Neural Information Processing Systems, 2019: 6281-6292.

[18] Zhao J, Mao M, Zhao X, et al. A Hybrid of Deep Reinforcement Learning and Local Search for the Vehicle Routing Problems [J]. Ieee Transactions on Intelligent Transportation Systems, 2020: 1-11.

[19] Lu H, Zhang X, Yang S. A Learning-based Iterative Method for Solving Vehicle Routing Problems [C]// proceedings of the International Conference on Learning Representations, 2019.

[20] Gao L, Chen M, Chen Q, et al. Learn to Design the Heuristics for Vehicle Routing Problem [J/OL]. 2020, arXiv: 2002. 08539.

[21] Chen M, Gao L, Chen Q, et al. Dynamic Partial Removal: A Neural Network Heuristic for Large Neighborhood Search [J/OL]. 2020, arXiv: 2006. 01610.

[22] Zhang K, He F, Zhang Z, et al. Multi-vehicle routing problems with soft time windows: A multi-agent reinforcement learning approach [J]. Transportation Research Part C: Emerging Technologies, 2020, 121: 103834.

[23] Wang Wanliang, Chen Haoli, Li Guoqing, et al. Deep Reinforcement Learning for Multi-depot Vehicle Routing Problem [J/OL]. Control and Decision: DOI: 10.13195/j.kzyjc.2021.1381.

[24] Vinyals O, Fortunato M, Jaitly N. Pointer networks [C]// proceedings of the Advances in Neural Information Processing Systems, 2015: 2692-2700.

[25] Bello I, Pham H, Le Q V, et al. Neural combinatorial optimization with reinforcement learning [J/OL]. arXiv preprint arXiv: 1611.09940, 2016.

[26] Li Z, Chen Q, Koltun V. Combinatorial optimization with graph convolutional networks and guided tree search [C]// proceedings of the Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018: 537-546.

[27] Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks [J/OL]. arXiv preprint arXiv: 1609.02907, 2016.

[28] Mnih V, Kavukcuoglu K, Silver D, et al. Playing Atari with Deep Reinforcement Learning [J/OL]. 2013, arXiv: 1312.5602.

[29] Dai H, Dai B, Song L. Discriminative embeddings of latent variable models for structured data [J]. International conference on machine learning, 2016: 2702-2711.

[30] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need [C]// In Advances in Neural Information Processing Systems, 2017, 30: 5998-6008.

[31] Nowak A, Villar S, Bandeira A S, et al. A note on learning algorithms for quadratic assignment with graph neural networks [C]// proceedings of the Proceeding of the 34th International Conference on Machine Learning (ICML), 2017: 22.

[32] Ioffe S, Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift [C]// Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37. Lille, France; JMLR. org. 2015: 448–456.

[33] Velickovic P, Cucurull G, Casanova A, et al. Graph Attention Networks [C]// proceedings of the International Conference on Learning Representations, 2018.

[34] Van H H, Guez A, Silver D. Deep Reinforcement Learning with Double Q-Learning [C]// Proceedings of the AAAI Conference on Artificial Intelligence, 2016, 30 (1).

[35] Cordeau J F, Gendreau M, Laporte G. A tabu search heuristic for periodic and multi-depot vehicle routing problems [J]. Networks, 1997, 30 (2): 105-119.

[36] Cordeau J F, Maischberger M. A parallel iterated tabu search heuristic for vehicle routing problems [J]. Computers & Operations Research, 2012, 39 (9): 2033-2050.

[37] Kingma D P, Ba J. Adam: A Method for Stochastic Optimization [C]// International Conference on Learning Representations, 2015.

[38] Stodola P. Using Metaheuristics on the Multi-Depot Vehicle Routing Problem with Modified Optimization Criterion [J]. 2018, 11 (5): 74.