Deep Reinforcement Learning with Two-Stage Training Strategy for Practical Electric Vehicle Routing Problem with Time Windows

Jinbiao Chen, Huanhuan Huang, Zizhen Zhang, and Jiahai Wang(✉)

School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China
{chenjb69,huanghh29}@mail2.sysu.edu.cn,
{zhangzzh7,wangjiah}@mail.sysu.edu.cn

Abstract. Recently, it is promising to apply deep reinforcement learning (DRL) to the vehicle routing problem (VRP), which is widely employed in modern logistics systems. A practical extension of VRP is the electric vehicle routing problem with time windows (EVRPTW). In this problem, the realistic traveling distance and time are non-Euclidean and asymmetric, and the constraints are more complex. These characteristics result in a challenge when using the DRL approach to solve it. This paper proposes a novel end-to-end DRL method with a two-stage training strategy. First, a graph attention network with edge features is designed to tackle the graph with the asymmetric traveling distance and time matrix. The node and edge features of the graph are effectively correlated and captured. Then, a two-stage training strategy is proposed to handle the complicated constraints. Some constraints are allowed to be violated to enhance exploration in the first stage, while all the constraints are enforced to be satisfied to guarantee a feasible solution in the second stage. Experimental results show that our method outperforms the state-of-the-art methods and can be generalized well to different problem sizes.

Keywords: Deep reinforcement learning · Electric vehicle routing problem with time windows · Graph attention network · Two-stage training

1 Introduction

Vehicle routing problem (VRP) [2], as a classic combinatorial optimization problem, aims at dispatching a fleet of vehicles to serve a set of customers so as to minimize the total traveling cost. Recently, electric vehicles (EVs) have been extensively popularized. Compared with traditional vehicles, EVs can be beneficial to sustainable transportation systems and environmental protection. They

G. Rudolph et al. (Eds.): PPSN 2022, LNCS 13398, pp. 356–370, 2022.
https://doi.org/10.1007/978-3-031-14714-2_25

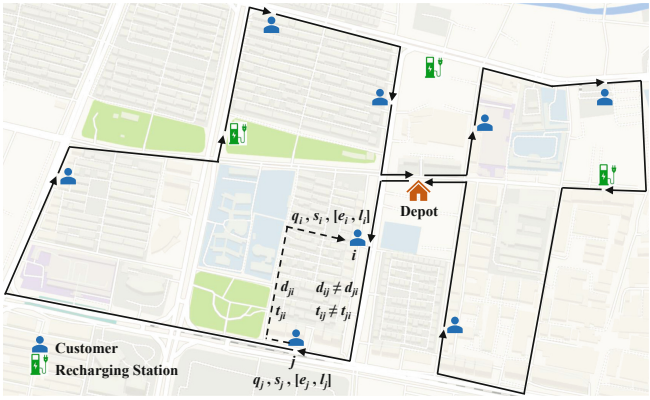


Fig. 1. An illustration of the practical EVRPTW.

will undoubtedly become the mainstream of vehicles in the future. By incorporating EVs into VRP, an interesting problem called the electric VRP (EVRP) attracts researchers' attention [25]. In addition to basic properties of VRP, EVRP further considers limited electricity of EVs. This implies that EVs need to be recharged at recharging stations before running out of battery. A natural and practical variant of EVRP is to impose a specific time window for each customer, known as EVRP with time windows (EVRPTW). Figure 1 illustrates an example of EVRPTW.

To solve VRP and its variants, traditional methods include exact algorithms and heuristic algorithms. Exact algorithms [6] usually employ the branch-and-bound framework to produce an optimal solution, but they can only handle small-scale problems in general. Heuristic algorithms [26] can obtain acceptable solutions in reasonable time, but they require problem-specific experience and knowledge. In addition, heuristic methods, which independently address problem instances and iteratively perform search in solution space, also suffer from long computation time for large-scale problems.

Recently, a novel framework based on deep reinforcement learning (DRL) is developed to rapidly obtain near-optimal solutions of combinatorial optimization problems [22], especially routing problems [21,28]. By using an end-to-end learning paradigm, a deep network model is trained offline with numerous instances. It can automatically learn underlying features from the data and generalize to unseen instances. Then it is leveraged to rapidly construct a solution by a direct forward inference without any iterative search.

The DRL methods have achieved some success for traditional VRPs. However, when these methods are applied to the practical EVRPTW, additional issues need to be addressed.

1) Most of the existing works only focus on routing problems where the symmetric Euclidean traveling distance and time are calculated by the given coordinates of nodes. In the practical EVRPTW, the traveling distance and

time are non-Euclidean and asymmetric, as shown in Fig. 1. The traveling distance is determined by the realistic routes in the transportation network. The traveling time is not only linear with the actual traveling distance but also related to comprehensive traffic factors like terrain, weather, and crowding degree. Therefore, existing methods that exploit coordinates as inputs may fail to extract the edge features between two nodes containing the actual traveling distance and time.
2) Different from the classic VRP, EVRPTW has more complicated constraints including limited electricity and time windows. Although the masking mechanism in most DRL methods can be directly adopted to ensure a feasible solution, such mechanism also restricts the exploration of DRL, which makes it difficult to cross through the infeasible region to learn the relationships among solutions, constraints and objectives.

To tackle the aforementioned issues, this paper proposes a novel DRL method with a two-stage training strategy (DRL-TS) to solve the practical EVRPTW. Our contributions can be summarized as follows.

1) A deep neural network based on a graph attention network (GAT) with edge features is proposed to effectively capture both node and edge features of a graph input. The network correlates the node embeddings with the edge embeddings, which can produce a high-quality solution.
2) A two-stage training strategy is proposed to deal with complex constraints. It is a general strategy, as it does not rely on a specific form of constraints. Specifically, in the first stage, some constraints are treated as soft and allowed to be violated to enhance the exploration, but penalties are appended to the objective if constraints are violated. In the second stage, all the constraints must be satisfied by the masking mechanism to guarantee a feasible solution.
3) Computational experiments conducted on real-world asymmetric datasets show that the proposed method outperforms conventional methods and state-of-the-art DRL methods.

2 Literature Review

This section first briefly reviews traditional methods for EVRPTW, including exact and heuristic algorithms. EVRPTW was first introduced by Schneider et al. [26] and solved by a hybrid heuristic algorithm combining a variable neighborhood search heuristic with a tabu search heuristic. Exact branch-and-price-and-cut algorithms relying on customized labeling algorithms [6] were proposed for four variants of EVRPTW. A few conventional methods for EVRPTW have been proposed. Other heuristic algorithms are further developed for some extensions of EVRPTW considering various recharging policies [4,11,12].

Next, we review recent DRL methods for routing problems. A sequence-to-sequence model based on a pointer network [1] was first proposed to solve routing problems. The recurrent neural network (RNN) and attention mechanism [23] were used to improve the policy network for VRP. The framework of [23] was

adapted to solve EVRPTW [19], which is named as DRL-R. Inspired by the Transformer architecture [27], the multi-head self-attention mechanism [14] was adopted to improve the policy network for routing problems. A dynamic attention model with dynamic encoder-decoder architecture [24] was developed to improve the performance for VRP. Policy optimization with multiple optima [15] was introduced to further improve the performance by utilizing the symmetries in the representation of a solution. A structural graph embedded pointer network [10] was presented to iteratively produce tours for the online VRP. A deep model based on the graph convolutional network with node and edge features [7] was proposed for the practical VRP. These methods were further developed for variants of routing problems [16–18,29,33] and their multi-objective versions [30,32,34]. Different from the above end-to-end methods that learn to directly construct solutions, a learn-to-improve framework [5,20,31] that learns local search operators was developed to iteratively improve an initial solution.

In summary, most of the DRL methods were focused on VRP and its simple variants. Only one work, DRL-R [19], applied the DRL method to EVRPTW. This method treating coordinates as inputs cannot be directly applied to the practical EVRPTW with asymmetric traveling distance and time. In addition, it cannot well handle the complex constraints. These facts motivate us to develop a more effective DRL method for EVRPTW.

3 The Proposed DRL-TS for Practical EVRPTW

3.1 Problem Statements and Reinforcement Learning Formulation

EVRPTW is defined on a complete directed graph $G = (N, E)$, where N is the node set and E is the edge set. The node set $N = V \cup F \cup \{0\}$, where V is the set of customers, F is the set of recharging stations, and 0 is the depot. Each node $i \in V$ has a positive demand q_i and a service time s_i. Each node $i \in F$ also has a demand $q_i = 0$ and a service time s_i representing its full recharging time. Also, each node $i \in N$ has a time window $[e_i, l_i]$. Each edge is associated with a traveling distance d_{ij}, a traveling time t_{ij}, and a battery consumption λd_{ij}, where λ is a battery consumption rate. In the practical EVRPTW, the traveling distance and time are both possibly asymmetric, i.e., $d_{ij} \neq d_{ji}$ and $t_{ij} \neq t_{ji}$.

A fleet of homogeneous EVs with identical loading capacity C and battery capacity Q is initially placed at the depot with full battery power. EVs must serve all the customers and finally return to the depot. The objective is to minimize the total traveling distance. The following constraints must be satisfied.

1) *Capacity constraints*: The remaining capacity of EVs for serving node $i \in V$ must be no less than the demand q_i.
2) *Time window constraints*: EVs need to wait if they arrive at node $i \in N$ before e_i. The service after l_i is not allowed.
3) *Electricity constraints*: The remaining electricity of EVs arriving at each node $i \in N$ must be no less than 0.

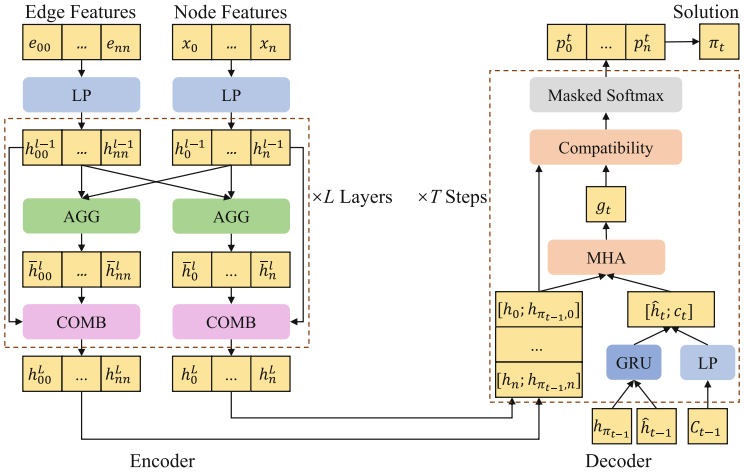


Fig. 2. The GAT-based policy network. LP denotes a linear projection, AGG denotes the aggregation sub-layer, COMB denotes the combination sub-layer, GRU denotes a gated recurrent unit, MHA denotes a multi-head attention layer, and $N = \{0, 1, ..., n\}$.

To formulate EVRPTW as a reinforcement learning (RL) form, it can be naturally deemed as a sequential decision problem by constructing the routes step-by-step. Especially, a solution is represented by a sequence $\boldsymbol{\pi} = \{\pi_0, ..., \pi_T\}$, where $\pi_t \in N$, $\pi_0 = \pi_T = 0$. Note that the sequence length T is unfixed, because there are multiple routes and the recharging stations can be visited any times.

In RL, the state is defined as a partial solution $\boldsymbol{\pi}_{0:t-1}$. The action is defined as visiting a node $\pi_t \in V \backslash \{\boldsymbol{\pi}_{0:t-1}\} \cup F \cup \{0\}$ satisfying all the constraints. The state transition is defined as $\boldsymbol{\pi}_{0:t} = \{\boldsymbol{\pi}_{0:t-1}, \pi_t\}$. The reward is defined as $R = \sum_{t=1}^{T} -d_{\pi_{t-1},\pi_t}$, where d_{π_{t-1},π_t} represents the traveling distance at step t. A stochastic policy $p(\boldsymbol{\pi}|G)$ generating a solution $\boldsymbol{\pi}$ from the graph G of an instance is calculated as $p(\boldsymbol{\pi}|G) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\pi_t|\boldsymbol{\pi}_{0:t-1}, G)$, where $p_{\boldsymbol{\theta}}(\pi_t|\boldsymbol{\pi}_{0:t-1}, G)$ is the probability of the node selection parameterized by $\boldsymbol{\theta}$.

3.2 GAT-based Policy Network with Edge Features

To learn the policy $p_{\boldsymbol{\theta}}$, a GAT-based policy network with edge features is designed, which follows the encoder-decoder architecture (see Fig. 2). It can extract both the node and edge features including the traveling distance and time matrix of the practical EVRPTW. The encoder produces the correlated node and edge embeddings of the graph. At each time step, the decoder aggregates the embeddings with the context to generate a probability vector and selects a node accordingly. The process is iteratively repeated until all the customers are served.

Encoder. The encoder first computes initial node and edge embeddings of the graph input with node features $x_i = (q_i, e_i, l_i, z_i)$ and edge features $e_{ij} = (d_{ij}, t_{ij}, a_{ij})$, $i, j \in N$, where z_i denotes the node type (depot, customer or recharging station) and a_{ij} denotes whether i and j are adjacent as follows.

$$z_i = \begin{cases} 0, & i \in \{0\} \\ 1, & i \in V \\ -1, & i \in F \end{cases}, \tag{1}$$

$$a_{ij} = \begin{cases} 1, & i \text{ and } j \text{ are } r\text{-nearest neighbors} \\ -1, & i = j \\ 0, & \text{otherwise} \end{cases}, \tag{2}$$

where r is empirically set to 10 as in [7]. The inputs are linearly projected to initial node embeddings h_i^0 and edge embeddings h_{ij}^0 with the same dimension d_h.

$$\begin{aligned} h_i^0 &= W_N^0 x_i + b_N, \\ h_{ij}^0 &= W_E^0 e_{ij} + b_E, \end{aligned} \tag{3}$$

where W_N, W_E, b_N, and b_E are all trainable parameters. Let h_i^l and h_{ij}^l respectively denote the embeddings of node i and edge (i, j) produced by layer $l \in \{1, ..., L\}$. The final embeddings h_i^L and h_{ij}^L are obtained by the encoder. Each layer l contains an aggregation and a combination sub-layer, which aggregate each embedding with its neighbors and combine itself with the aggregated embeddings [7], respectively.

In the aggregation sub-layer, the node and edge embeddings are simultaneously computed as follows.

$$\begin{aligned} \tilde{h}_i^l &= \mathrm{MHA}^l(h_i^{l-1}, \{[h_j^{l-1}; h_{ij}^{l-1}] \mid j \in N\}), \\ \bar{h}_i^l &= \sigma(\mathrm{BN}^l(W_N^l \tilde{h}_i^l)), \\ \tilde{h}_{ij}^l &= W_{E1}^l h_{ij}^{l-1} + W_{E2}^l h_i^{l-1} + W_{E3}^l h_j^{l-1}, \\ \bar{h}_{ij}^l &= \sigma(\mathrm{BN}^l(W_E^l \tilde{h}_{ij}^l)), \end{aligned} \tag{4}$$

where $[;]$ is the concatenation of two vectors, BN is a batch normalization layer [9], σ is the ReLU activation, and W_N^l, W_E^l, W_{E1}^l, W_{E2}^l, W_{E3}^l are all trainable parameters. MHA is a multi-head attention layer with M heads [27], in which $q_i^{m,l-1} = W_Q^m h_i^{l-1}$, $k_{ij}^{m,l-1} = W_K^m [h_j^{l-1}; h_{ij}^{l-1}]$, $v_{ij}^{m,l-1} = W_V^m [h_j^{l-1}; h_{ij}^{l-1}]$, and $m \in \{1, ..., M\}$.

In the combination sub-layer, the node and edge embeddings are both combined by a skip-connection [8], as follows.

$$\begin{aligned} h_i^l &= \sigma(h_i^{l-1} + \mathrm{BN}^l(\mathrm{FF}^l(\bar{h}_i^l))), \\ h_{ij}^l &= \sigma(h_{ij}^{l-1} + \mathrm{BN}^l(\mathrm{FF}^l(\bar{h}_{ij}^l))), \end{aligned} \tag{5}$$

where FF is a fully connected feed-forward layer.

Decoder. The decoder sequentially selects a node according to a probability distribution obtained by the node embeddings h_i^L and edge embeddings h_{ij}^L from the encoder (the superscript L is omitted for readability in the following).

Specifically, at the decoding step t, the *context* c_t, previous partial tour, node embeddings, and edge embeddings are firstly aggregated as the *glimpse* g^t [1].

$$\begin{aligned} c_t &= W_C C_{t-1} + b_C, \\ \hat{h}_t &= \mathrm{GRU}(h_{\pi_{t-1}}, \hat{h}_{t-1}), \\ g^t &= \mathrm{MHA}([\hat{h}_t; c_t], \{[h_j; h_{\pi_{t-1},j}] \mid j \in N\}), \end{aligned} \tag{6}$$

where W_C and b_C are trainable parameters, GRU is a gated recurrent unit, which can better handle the partial solution generated by sequential steps. $C_t = (\mathcal{T}_t, D_t, B_t)$ is composed of the traveling time $\mathcal{T}_t$, remaining capacity D_t, and remaining electricity B_t when leaving node π_t, which are updated as follows.

$$\begin{aligned} \mathcal{T}_t &= \begin{cases} \mathcal{T}_{t-1} + t_{\pi_{t-1},\pi_t} + s_{\pi_t}, & \pi_t \in V \cup F \\ 0, & \pi_t \in \{0\} \end{cases}, \\ D_t &= \begin{cases} D_{t-1} - q_{\pi_t}, & \pi_t \in V \cup F \\ C, & \pi_t \in \{0\} \end{cases}, \\ B_t &= \begin{cases} B_{t-1} - \lambda d_{\pi_{t-1},\pi_t}, & \pi_t \in V \\ Q, & \pi_t \in F \cup \{0\} \end{cases}. \end{aligned} \tag{7}$$

Then, the *compatibility* u^t is calculated by the *query* $q^t = W_Q g^t$ and *key* $k_i^t = W_K[h_i; h_{\pi_{t-1},i}]$ with trainable parameters W_Q and W_K.

$$u_i^t = \begin{cases} \zeta \cdot \tanh\left(\frac{(q^t)^T k_i^t}{\sqrt{d_h}}\right), & \mathrm{mask}_i^t = 1 \\ -\infty, & \text{otherwise} \end{cases}, \tag{8}$$

where ζ is used to clip the result. $\mathrm{mask}_i^t = 1$ represents that the feasible node i is unmasked at step t. A node is called feasible or unmasked, if upon its arrival, the capacity, time window and electricity constraints are not violated. Finally, the probability distribution to select a node at step t is computed using the softmax function as follows.

$$p_{\boldsymbol{\theta}}(\pi_t | \boldsymbol{\pi}_{0:t-1}, G) = \mathrm{softmax}(u^t). \tag{9}$$

Two common decoding strategies, which are sampling decoding and greedy decoding, are adopted to choose a node at each step. A node is chosen according to the probability distribution in the sampling decoding, or according to the maximum probability in the greedy decoding.

3.3 Two-Stage Training Strategy

The masking mechanism in DRL methods can be directly applied to EVRPTW by including the capacity, time window, and electricity constraints [19]. However, such mechanism would limit the exploration of those infeasible regions,

thereby impairing the seeking of global-best solutions. In order to enrich the search space, we propose a two-stage training strategy, which tries to well balance the exploration and feasibility of the search.

Specifically, in the first stage, the capacity, time window and electricity constraints are treated as the soft constraints. Only the constraints ensuring a tour are retained hard. In this case, node i is masked, i.e., $\text{mask}_i^t = 0$, if one of the following conditions is satisfied.

1) The customer has already been visited before, i.e., $i \in V$ and $i \in \{\boldsymbol{\pi}_{0:t-1}\}$.
2) The depot has been visited in the last step, i.e., $i \in \{0\}$ and $\pi_{t-1} = 0$.
3) The recharging station is visited by the EV with full electricity, i.e., $i \in F, \pi_{t-1} \in F \cup \{0\}$.

In the second stage, all the original constraints must be respected. In this case, $\text{mask}_i^t = 0$ if one of the following conditions in addition to above three conditions in the first stage is satisfied.

1) The capacity constraint: $D_{t-1} < q_i$.
2) The time window constraint: $\mathcal{T}_{t-1} + t_{\pi_{t-1},i} > l_i$.
3) The electricity constraint: $B_{t-1} < \lambda d_{\pi_{t-1},i} + \min_{j \in F \cup \{0\}} \lambda d_{ij}$.

The loss is defined as $\mathcal{L}(\boldsymbol{\theta}|G) = \mathbf{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|G)}[y(\boldsymbol{\pi})]$, where $y(\boldsymbol{\pi})$ is defined as an uniform form for two training stages as follows.

$$\begin{aligned} y(\boldsymbol{\pi}) = &\sum_{t=1}^{T} d_{\pi_{t-1},\pi_t} + \alpha \sum_{t=1}^{T} \max(\mathcal{T}_{t-1} + t_{\pi_{t-1},\pi_t} - l_{\pi_t}, 0) \\ &+ \beta \sum_{t=1}^{T} \max(q_{\pi_t} - D_{t-1}, 0) + \gamma \sum_{t=1}^{T} \max(\lambda d_{\pi_{t-1},\pi_t} - B_{t-1}, 0) \end{aligned} . \tag{10}$$

For the first stage, α, β, and γ are respectively three penalties for the violation of the capacity, time window, and electricity constraints. For the second stage, $y(\boldsymbol{\pi})$ is degenerated to the total traveling distance $\sum_{t=1}^{T} d_{\pi_{t-1},\pi_t}$, since the hard constraints are considered. $\mathcal{L}(\boldsymbol{\theta}|G)$ is optimized by gradient descent using the well-known REINFORCE algorithm with a rollout baseline $b(G)$, as follows.

$$\nabla \mathcal{L}(\boldsymbol{\theta}|G) = \mathbf{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|G)}[(y(\boldsymbol{\pi}) - b(G)) \nabla \log p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|G)], \tag{11}$$

$$\boldsymbol{\theta} \leftarrow \text{Adam}(\boldsymbol{\theta}, \nabla \mathcal{L}), \tag{12}$$

where Adam is the Adam optimizer [13]. The training algorithm is similar to that in [14], and $b(G)$ is a greedy rollout produced by the current model.

The proportions of the epochs of the first and second stage are respectively controlled by η and $1 - \eta$, where η is a user-defined parameter.

3.4 Characteristics of DRL-TS

The characteristics of DRL-TS are summarized as follows.

1) For the policy network, DRL-TS uses GAT with edge features to effectively tackle the graph input with the asymmetric traveling distance and time matrix, while DRL-R [19] exploiting coordinates as inputs can only deal with the symmetric Euclidean traveling distance. In addition, unlike DRL-R adopting a dynamic T-step encoder based on RNN, which consumes enormous memory for large-scale cases, DRL-TS uses an efficient one-step encoder.
2) For the training, like most of DRL-based methods, DRL-R [19] directly adopts the masking mechanism to EVRPTW to ensure a feasible solution. However, DRL-TS uses a two-stage training strategy. The capacity, time window, and electricity constraints are all allowed to be violated to enhance exploration in the first stage, while the original constraints must be satisfied by the masking mechanism to guarantee feasibility in the second stage.

4 Experimental Results

In this section, we conduct computational experiments to evaluate the proposed method on practical EVRPTW instances. All the experiments are implemented on a computer with an Intel Xeon 4216 CPU and an RTX 3090 GPU. The source code of the proposed algorithm is available on request.

4.1 Experimental Settings

Benchmarks. The practical EVRPTW instances are generated from the real-world data of the Global Optimization Challenge competition[1] held by JD Logistics, where the traveling distance and time are both asymmetric. For each instance, each customer i is randomly selected from the entire customer set of the data. The time window $[e_i, l_i]$, service time (or recharging time) s_i, traveling distance d_{ij}, and traveling time t_{ij} are all directly obtained from the original data. The time window of the depot $[e_0, l_0]$ is [8:00,20:00], as the working time is 720 min. Each demand q_i is randomly chosen from {1,...,9}. Following the previous work [14,19], we conduct experiments on the instances with different customer sizes $|V| = 10/20/50/100$, denoted as C10/C20/C50/C100. The capacities of EVs Q are correspondingly set to 20/30/40/50. The number of recharging stations $|F|$ is set to $|V|/10$. The battery capacity Q is set to 1.6×10^5. The battery consumption rate λ is fixed to 1.

Baselines. The following representative methods, including the state-of-the-art DRL method, exact method, and heuristic method, are considered as baselines for the comparisons.

[1] https://jdata.jd.com/html/detail.html?id=5.

1) SCIP[2]: an open-source exact solver for combinatorial optimization problems, which solves EVRPTW using the mathematical model described in [26]. SCIP(180s) indicates running the SCIP solver for 180 s.
2) DRL-R [19]: the state-of-the-art DRL method for EVRPTW, which adopts an RNN-based policy network and uses an ordinary masking mechanism to train the network.
3) SA-VNS: the combination of simulated annealing (SA) and variable neighborhood search (VNS), which is an especially designed meta-heuristic method based on [26] for EVRPTW. To efficiently explore the solution space, different neighborhood operators are employed, including 2-opt, 2-opt*, or-opt, cross-exchange, merge, and stationInRe. The last operator is specially designed for EVRPTW [26], while the others are widely used in routing problems [3]. The SA framework is adopted to avoid local optima. The number of outer iterations is 1000. The linear annealing rate is 0.99. The initial temperature is 10. The number of inner iterations is 10. The six operators are successively executed in a random order in each inner iteration.

Hyper-parameters. In the encoder, $L = 2$ GAT-layers are used as in [7]. d_h, M, and ζ are respectively set to 128, 8, and 10 as in [14]. The Adam optimizer with a constant learning rate 10^{-4} is adopted to train the model. The penalties α, β, and γ are all set to 1. Regarding the training, 200 epochs are run. In each epoch, 250 batches are generated. The batch size is set to 128 for C10/C20/C50 and 64 for C100 due to memory constraints. η is set to 0.5, i.e., 100 epochs are run both for the first and second stage.

4.2 Comparison Analysis

The experimental results of SCIP, SA-VNS, DRL-R [19] and DRL-TS (ours) are recorded in Table 1, where the objective value, optimality gap, and average computing time of an instance are shown. The objective values are normalized by a scale of 10^5. The gap of a method is calculated by $(O_m - O_b)/O_b \times 100\%$, where O_m is the objective value of the compared method and O_b is the best objective value among all methods. The performance of our method and that of other methods are all statistically different by the Wilcoxon rank-sum test with a significance level 1% for each experiment.

From Table 1, it can be seen that our method outperforms DRL-R with both greedy and sampling decoding. DRL-R is first trained on C20 and then tested on instances of various sizes like that in [19], since it is a dynamic encoding model with T steps consuming enormous memory. DRL-TS20 uses the same training and testing form as DRL-R, but it still outperforms DRL-R. Regarding sampling decoding, it achieves smaller objective values and gaps than greedy

[2] https://scip.zib.de/.

Table 1. The average performance of our method and baselines on 1000 random instances. DRL-TS20/50 means that our proposed model trained on the C20/C50 is tested on instances of various sizes. G denotes greedy decoding and S denotes sampling decoding that chooses the best solution from 1280 sampled solutions.

Method	C10			C20		
	Obj.	Gap	Time/s	Obj.	Gap	Time/s
SCIP(180s)	**3.665**	**0.00%**	4.01	**5.626**	**0.00%**	146.20
SA-VNS	3.687	0.58%	20.23	5.771	2.58%	27.79
DRL-R(G)	4.172	13.81%	0.01	6.554	16.49%	0.01
DRL-TS20(G)	4.038	10.16%	<0.01	6.247	11.04%	<0.01
DRL-TS50(G)	4.184	14.15%	<0.01	6.425	14.19%	<0.01
DRL-TS(G)	3.950	7.76%	<0.01	6.247	11.04%	<0.01
DRL-R(S)	3.930	7.23%	0.71	6.112	8.63%	1.99
DRL-TS20(S)	3.819	4.18%	0.95	5.903	4.91%	1.67
DRL-TS50(S)	3.818	4.17%	1.24	5.975	6.19%	1.94
DRL-TS(S)	3.796	3.57%	0.91	5.903	4.91%	1.67
Method	C50			C100		
	Obj.	Gap	Time/s	Obj.	Gap	Time/s
SCIP(180s)	14.499	26.22%	180.00	62.910	217.62%	181.06
SA-VNS	11.693	1.80%	49.87	20.470	3.35%	105.22
DRL-R(G)	13.147	14.46%	0.01	23.197	17.12%	0.03
DRL-TS20(G)	12.642	10.06%	<0.01	22.008	11.11%	0.01
DRL-TS50(G)	12.187	6.10%	<0.01	21.084	6.45%	0.01
DRL-TS(G)	12.187	6.10%	<0.01	20.847	5.25%	0.01
DRL-R(S)	12.155	5.82%	5.26	21.515	8.63%	30.91
DRL-TS20(S)	11.604	1.03%	3.72	20.879	5.41%	12.85
DRL-TS50(S)	11.486	0.00%	3.77	19.844	0.19%	14.97
DRL-TS(S)	**11.486**	**0.00%**	3.77	**19.806**	**0.00%**	13.10

decoding despite using slightly more computing time, which shows that the sampling strategy can effectively improve the solution quality. SCIP can engender the optimal solutions for all of C10 instances and 400 instances of C20, but only feasible solutions for all of C50 and C100 instances, as it suffers from exponentially growing computing time. SA-VNS can produce near-optimal solutions within reasonable computing time. Our method can rapidly achieve better performance in terms of objective values and gaps for large problem sizes, i.e., C50 and C100, and achieve acceptable performance for small problem sizes. Moreover, compared with results of DRL-TS that performs best correspondingly trained on each problem size, the results of DRL-TS20 and DRL-TS50 demonstrate that the model has a desirable ability of generalization for different problem sizes.

In summary, our method has fast solving speed and high generalization ability, and it is superior to the state-of-the-art DRL method. For small-scale cases,

compared with the exact and heuristic methods carefully calibrated for problems, our method can still produce promising solutions with acceptable gaps.

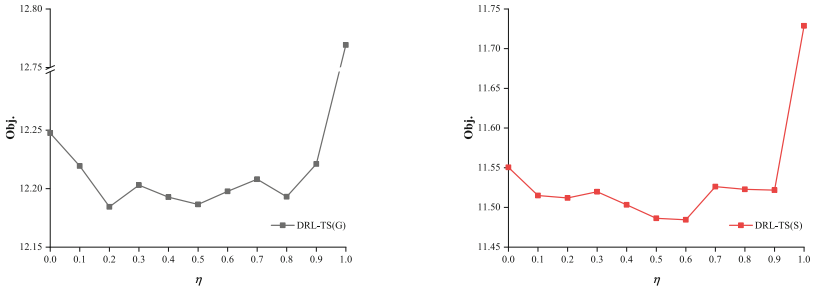
4.3 Ablation Study

To verify the significance of the two-stage training strategy and the GAT-based network considering the feature edges, the results of ablation experiments are shown in Table 2. DRL-H/DRL-S is our proposed GAT-based network trained only with purely hard/soft constraints, i.e., the same as the second/first stage, which is used to evaluate the effects of the proposed two-stage training strategy. DRL-TS w/o E is our proposed policy network without edge features, which is used to evaluate the effects of the edge features. The results indicate that our model performs better for both greedy and sampling decoding than that trained only with the hard or soft constraints and without edge features.

Furthermore, for the two-stage training strategy, the sensitivity of the proportion of the epochs of the first stage η is studied. $\eta = 0$ and $\eta = 1$ represent DRL-H and DRL-S, respectively. The results on C50 instances are shown in Fig. 3. Except $\eta = 0$ and $\eta = 1$, η has a little influence on the objective value and $\eta = 0.5$ is always a robust strategy. This again verifies the effectiveness of the two-stage training strategy.

Table 2. Different DRL methods on 1000 random instances.

Method	C10		C20		C50		C100	
	Obj.	Gap	Obj.	Gap	Obj.	Gap	Obj.	Gap
DRL-TS(S)	**3.796**	**0.00%**	**5.903**	**0.00%**	**11.486**	**0.00%**	**19.806**	**0.00%**
DRL-H(S)	3.833	0.97%	5.931	0.48%	11.551	0.56%	19.965	0.80%
DRL-S(S)	3.819	0.60%	5.990	1.48%	11.729	2.11%	20.298	2.48%
DRL-TS w/o E(S)	4.210	10.91%	6.875	16.47%	15.817	37.70%	30.437	53.67%
DRL-TS(G)	**3.950**	**4.05%**	**6.247**	**5.83%**	**12.187**	**6.10%**	**20.847**	**5.25%**
DRL-H(G)	3.971	4.60%	6.257	6.01%	12.248	6.63%	21.121	6.64%
DRL-S(G)	4.024	6.00%	6.483	9.83%	12.769	11.17%	21.515	8.62%
DRL-TS w/o E(G)	4.781	25.95%	8.545	44.76%	18.841	64.03%	34.417	73.77%



(a) (b)

Fig. 3. Effect of the parameter η. (a) Greedy. (b) Sampling.

5 Conclusion

This paper proposes a novel DRL method for the practical EVRPTW. A GAT-based policy network with edge features is designed to cope with the non-Euclidean and asymmetric traveling distance and time. A two-stage training strategy is presented to handle the complicated constraints. The experimental results based on the real-world asymmetric data verify the effectiveness of the proposed approach. In the future, the proposed method can be extended to other VRP variants with practical characteristics and their multi-objective versions.

Acknowledgements. This work is supported by the National Key R&D Program of China (2018AAA0101203), the National Natural Science Foundation of China (62072483), and the Guangdong Basic and Applied Basic Research Foundation (2022A1515011690, 2021A1515012298).

References

1. Bello, I., Pham, H., Le, Q.V., Norouzi, M., Bengio, S.: Neural combinatorial optimization with reinforcement learning. In: International Conference on Learning Representations (2017)
2. Braekers, K., Ramaekers, K., Van Nieuwenhuyse, I.: The vehicle routing problem: state of the art classification and review. Comput. Ind. Eng. **99**, 300–313 (2016)
3. Bräysy, O., Gendreau, M.: Vehicle routing problem with time windows, Part I: route construction and local search algorithms. Transp. Sci. **39**(1), 104–118 (2005)
4. Bruglieri, M., Pezzella, F., Pisacane, O., Suraci, S.: A variable neighborhood search branching for the electric vehicle routing problem with time windows. Electron. Notes Discret. Math. **47**, 221–228 (2015)
5. Chen, X., Tian, Y.: Learning to perform local rewriting for combinatorial optimization. In: Advances in Neural Information Processing Systems (2019)
6. Desaulniers, G., Errico, F., Irnich, S., Schneider, M.: Exact algorithms for electric vehicle-routing problems with time windows. Oper. Res. **64**(6), 1388–1405 (2016)

7. Duan, L., et al.: Efficiently solving the practical vehicle routing problem: a novel joint learning approach. In: ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 3054–3063 (2020)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
9. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning, vol. 37, pp. 448–456 (2015)
10. James, J., Yu, W., Gu, J.: Online vehicle routing with neural combinatorial optimization and deep reinforcement learning. IEEE Trans. Intell. Transp. Syst. **20**(10), 3806–3817 (2019)
11. Keskin, M., Çatay, B.: Partial recharge strategies for the electric vehicle routing problem with time windows. Transp. Res. Part C Emerg. Technol. **65**, 111–127 (2016)
12. Keskin, M., Çatay, B.: A matheuristic method for the electric vehicle routing problem with time windows and fast chargers. Comput. Oper. Res. **100**, 172–188 (2018)
13. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: International Conference on Learning Representations (2015)
14. Kool, W., Van Hoof, H., Welling, M.: Attention, learn to solve routing problems! In: International Conference on Learning Representations (2019)
15. Kwon, Y.D., Choo, J., Kim, B., Yoon, I., Gwon, Y., Min, S.: POMO: policy optimization with multiple optima for reinforcement learning. In: Advances in Neural Information Processing Systems (2020)
16. Li, J., et al.: Deep reinforcement learning for solving the heterogeneous capacitated vehicle routing problem. IEEE Trans. Cybern. (2021)
17. Li, J., Xin, L., Cao, Z., Lim, A., Song, W., Zhang, J.: Heterogeneous attentions for solving pickup and delivery problem via deep reinforcement learning. IEEE Trans. Intell. Transp. Syst. **23**(3), 2306–2315 (2022)
18. Li, K., Zhang, T., Wang, R., Wang, Y., Han, Y., Wang, L.: Deep reinforcement learning for combinatorial optimization: covering salesman problems. IEEE Trans. Cybern. (2021)
19. Lin, B., Ghaddar, B., Nathwani, J.: Deep reinforcement learning for the electric vehicle routing problem with time windows. IEEE Trans. Intell. Transp. Syst. (2021)
20. Lu, H., Zhang, X., Yang, S.: A learning-based iterative method for solving vehicle routing problems. In: International Conference on Learning Representations (2020)
21. Mazyavkina, N., Sviridov, S., Ivanov, S., Burnaev, E.: Reinforcement learning for combinatorial optimization: a survey. Comput. Oper. Res. **134**, 105400 (2021)
22. Mirhoseini, A., et al.: A graph placement methodology for fast chip design. Nature **594**, 207–212 (2021)
23. Nazari, M., Oroojlooy, A., Takáč, M., Snyder, L.V.: Reinforcement learning for solving the vehicle routing problem. In: Advances in Neural Information Processing Systems (2018)
24. Peng, B., Wang, J., Zhang, Z.: A deep reinforcement learning algorithm using dynamic attention model for vehicle routing problems. In: Li, K., Li, W., Wang, H., Liu, Y. (eds.) ISICA 2019. CCIS, vol. 1205, pp. 636–650. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-5577-0_51
25. Qin, H., Su, X., Ren, T., Luo, Z.: A review on the electric vehicle routing problems: variants and algorithms. Front. Eng. Manag. **8**(3), 370–389 (2021). https://doi.org/10.1007/s42524-021-0157-1

26. Schneider, M., Stenger, A., Goeke, D.: The electric vehicle-routing problem with time windows and recharging stations. Transp. Sci. **48**(4), 500–520 (2014)
27. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems (2017)
28. Wang, Q., Tang, C.: Deep reinforcement learning for transportation network combinatorial optimization: a survey. Knowl.-Based Syst. **233**, 107526 (2021)
29. Wu, G., Zhang, Z., Liu, H., Wang, J.: Solving time-dependent traveling salesman problem with time windows with deep reinforcement learning. In: IEEE International Conference on Systems, Man, and Cybernetics (2021)
30. Wu, H., Wang, J., Zhang, Z.: MODRL/D-AM: multiobjective deep reinforcement learning algorithm using decomposition and attention model for multiobjective optimization. In: Li, K., Li, W., Wang, H., Liu, Y. (eds.) ISICA 2019. CCIS, vol. 1205, pp. 575–589. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-5577-0_45
31. Wu, Y., Song, W., Cao, Z., Zhang, J., Lim, A.: Learning improvement heuristics for solving routing problems. IEEE Trans. Neural Netw. Learn. Syst. (2021)
32. Zhang, Y., Wang, J., Zhang, Z., Zhou, Y.: MODRL/D-EL: multiobjective deep reinforcement learning with evolutionary learning for multiobjective optimization. In: International Joint Conference on Neural Networks (2021)
33. Zhang, Z., Liu, H., Zhou, M., Wang, J.: Solving dynamic traveling salesman problems with deep reinforcement learning. IEEE Trans. Neural Netw. Learn. Syst. (2021)
34. Zhang, Z., Wu, Z., Zhang, H., Wang, J.: Meta-learning-based deep reinforcement learning for multiobjective optimization problems. IEEE Trans. Neural Netw. Learn. Syst. (2022)