2025 International Conference on Intelligent Computing
July 26-29, Ningbo, China
https://www.ic-icc.cn/2025/index.php

Deep Reinforcement Learning for Solving Electric Vehicle Routing Problems with Battery Swapping Station

Qichao Sun[1], Junqing Li[1,2(✉)], and Xiaolong Chen[1]

[1] Shandong Normal University, Jinan 250014, China
[2] Yunnan Normal University, Kunming 650500, China
lijunqing@lcu-cs.com

Abstract. The increasing adoption of electric vehicles (EVs) in logistics and transportation has led to critical challenges, including long charging times and range limitations. Battery swapping offers an effective solution for reducing energy replenishment time at stations. In this paper, we tackle the Electric Vehicle Routing Problem with Battery Swapping Station (EVRP-BSS) by proposing a Deep Reinforcement Learning (DRL)-based approach. Our approach trains a policy network with an encoder-decoder architecture to sequentially construct vehicle routes. To improve the network ability to capture complex node and edge relationships, we design a graph convolutional network (GCN)-based encoder that separately embeds node features and edge features (e.g., distance, slope), further fusing them through self-attention to generate global representations. This approach enhances the node information, ultimately leading to high-quality solutions. During training, we update the model parameters using the multiple-start sampling trajectory method. Experimental results demonstrate that our method achieves superior performance compared to both traditional heuristics and neural baselines.

Keywords: Deep Reinforcement Learning, Graph Convolutional Network, Electric Vehicle Routing Problem.

1 Introduction

As electric vehicles (EVs) continue to transform the transportation sector, their widespread adoption presents both opportunities and challenges [1]. Although EVs offer a sustainable alternative to traditional vehicles, their limited battery capacity and lengthy charging times remain significant barriers, particularly for long-distance transportation. To overcome these limitations, the efficient planning of EV fleet routes has become critical. This need has given rise to the Electric Vehicle Routing Problem (EVRP) [2], an extension of the classic VRP. Unlike traditional VRP, which focuses solely on optimizing routes, EVRP must also address the unique challenges posed by EVs, such as battery range constraints and the strategic placement of charging stations, ensuring that routes are both feasible and operationally efficient.

Fuel-powered vehicles benefit from rapid refueling, whereas electric vehicles (EVs) often experience extended charging times, posing challenges to operational efficiency

and limiting their broader adoption. To mitigate this issue, the battery swapping station (BSS) model has emerged as a promising alternative [3]. By enabling quick battery replacement, BSS significantly reduce energy replenishment time and improve delivery performance. This study focuses on the Electric Vehicle Routing Problem with Battery Swapping Stations (EVRP-BSS), where EVs start with full batteries from a depot and may require battery swaps during the route. The objective is to minimize the total time, including both travel and swapping time, while meeting all operational constraints.

Solving the EVRP poses substantial computational challenges due to its combinatorial complexity and NP-hard nature, where the solution space expands rapidly with instance size. Traditional approaches, including exact algorithms and handcrafted heuristics, often struggle to scale efficiently in large problem settings [4]. In contrast, recent progress in deep reinforcement learning (DRL) has opened new avenues for tackling such problems. By learning optimization policies through interaction with the environment, DRL methods can produce high-quality solutions with significantly improved computational efficiency, demonstrating notable advantages over classical methods in both speed and performance.

Recent advances in deep learning have significantly impacted routing optimization. The Pointer Network [5] first demonstrated the potential of neural networks in solving the TSP. Building on this foundation, Kool et al. [6] introduced the Attention Model (AM), a Transformer-based architecture capable of solving TSP and CVRP with performance comparable to classical heuristics. Kwon et al. [7] further improved AM through Policy Optimization with Multiple Optima (POMO), which stabilizes training by sampling multiple trajectories. Subsequent works have extended this line of research: Wu et al. [8] proposed an iterative DRL optimization framework, while Ma et al. [9] developed the Dual-Aspect Collaborative Transformer (DACT) to enhance solution quality. To address energy-constrained routing challenges, Tang et al. [10] employed a DRL framework enhanced with a Transformer architecture to reduce energy consumption in electric vehicle applications.

This paper presents a DRL-based neural heuristic for solving the EVRP-BSS. We formulate EVRP-BSS as a graph optimization task and construct solutions by sequentially selecting nodes through an encoder-decoder policy network. To effectively capture spatial and relational information, a graph convolutional network (GCN)-based encoder is employed: node features are embedded via linear projection, while edge features (e.g., distance and slope) are extracted using GCN layers. These are integrated using a self-attention mechanism to produce global representations. A tailored masking strategy is introduced to ensure solution feasibility. The entire network is trained using the REINFORCE algorithm. Extensive experiments show that our approach consistently outperforms traditional heuristics and neural baselines.

The rest of the paper is organized as follows: Section 2 defines the problem of EVRP-BSS; Section 3 describes our DRL-based solution method; Section 4 presents the experimental results; and Section 5 concludes the paper and outlines potential directions for future research.

2 Problem Statement

2.1 Problem Description

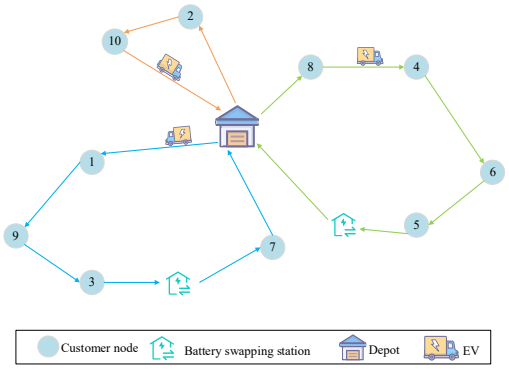


Fig. 1. Examples of the EVEP-BSS

EVRP-BSS can be defined as a directed graph $G=(V,E)$, as shown in Fig. 1, where $V=\{D,F,C\}$ is the set of all nodes and $E=\{(i,j)|i,j\in V,i\neq j\}$ is the set of arcs between nodes. Since the routing problem in this paper involves only one depot, the depot set D is both the starting and the ending point. The total number of nodes in the graph is $N=1+N_F+N_C$, where $F=\{1,2,\cdots,N_F\}$ represents the set of battery swap stations and $C=\{N_F+1,N_F+2,\cdots,N_F+N_C\}$ represents the set of customer nodes.

In the EVRP-BSS, a fleet of K homogeneous electric vehicles distributes goods to N_C customers with demand q_i in a directed graph G. Each vehicle departs from the depot with a maximum load Q and battery level U. During distribution, vehicles may visit any of the N_F battery swapping stations to replace depleted batteries before continuing service. The objective is to minimize total time while ensuring that all customer demands are fulfilled before returning to the depot. Meanwhile, this paper adopts the energy consumption model proposed by [10], which considers multiple factors to better reflect real-world driving conditions. During the travelling, the following conditions must be met:

- Each EV can only serve customers with demand less than or equal to its current load capacity;
- A customer receives service from only one EV;
- EVs can visit battery swapping stations multiple times as needed.

2.2 Mathematical Formulation

The notation used in the EVRP-BSS model is shown in Table 1.

Table 1. The notation definition of the EVRP-BSS model.

Notation	Definition
K	Set of vehicles.
V	Set of all nodes.
D	Set of depot node.
F	Set of battery swapping station nodes.
C	Set of customers.
t_{ij}^k	The travel time of vehicle k from node i to node j.
E_{ij}^k	The energy consumption of vehicle k from node i to node j.
w_{ij}^k	Load of vehicle k when traveling from node i to node j.
e_j^k	The remaining battery level of vehicle k at node j.
q_j	Demand of node j.
t_s	Battery swapping time.
Q	Vehicle load capacity.
U	Vehicle battery capacity.
x_{ij}^k	Binary indicator: 1 if vehicle k travels from node i to node j; 0 otherwise.

Based on the above notation, EVRP-BSS can be represented as the following nonlinear mixed integer programming model:

$$\min t_O = \sum_{k\in K}\sum_{i,j\in V, i\neq j} t_{ij}^k x_{ij}^k + \sum_{k\in K}\sum_{j\in F} t_s \tag{1}$$

$$\sum_{j\in V} x_{ij}^k = 1, \forall i \in C, k \in K \tag{2}$$

$$\sum_{j\in V} x_{ij}^k \leq 1, \forall i \in F, k \in K \tag{3}$$

$$\sum_{j\in V} x_{ij}^k - \sum_{j\in V} x_{ji}^k = 0, i \in C \bigcup F, k \in K \tag{4}$$

$$w_i^k - q_i x_{ij}^k + Q\left(1 - x_{ij}^k\right) \geq w_j^k, \forall i, j \in V, i \neq j, k \in K \tag{5}$$

$$w_0 = Q \tag{6}$$

$$0 \leq w_i^k \leq Q, \forall i, j \in V, i \neq j, k \in K \tag{7}$$

$$e_i^k - E_{ij}^k - U\left(1 - x_{ij}^k\right) \geq e_{ij}^k, \forall i, j \in V, i \neq j, k \in K \tag{8}$$

$$0 \leq e_i^k \leq U, \forall i \in V, k \in K \tag{9}$$

$$e_i^k = U, \forall i \in F \bigcup \{0\}, k \in K \tag{10}$$

$$x_{ij}^k \in \{0,1\}, \forall i, j \in V, i \neq j, k \in K \tag{11}$$

Specifically, the objective function in Eq. (1) minimizes the total travel time. Constraint (2) guarantees that every customer is visited by exactly one vehicle. Constraint (3) allows that visiting battery swapping stations is optional. Constraint (4) ensures flow conservation between nodes. Constraint (5) calculates the remaining vehicle load. Constraints (6) and (7) specify the vehicle initial load upon departure from the depot and ensure it stays within the weight limit during travel. Constraint (8) tracks the remaining battery charge. Constraint (9) keeps the battery charge within the allowable range. Constraint (10) requires vehicles to leave the depot or a swapping station with a full battery, and Constraint (11) specifies binary decision variables.

3 Methodology

In this section, we present the DRL framework for solving the EVRP-BSS, as shown in Fig. 2. The EVRP-BSS is first modeled as an MDP, and the policy is learned via an encoder-decoder neural network.

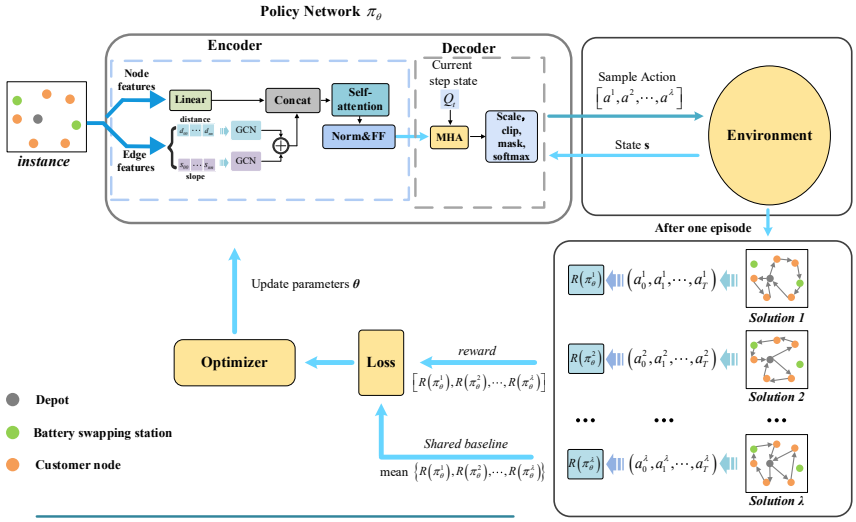


Fig. 2. Framework of the DRL Method.

3.1 Markov Decision Process

To model EVRP-BSS as MDP, we treat it as a sequential decision-making problem, where routes are progressively constructed at each step. Specifically, the solution can be represented as a sequence $\pi = \{a_0, a_1, \cdots, a_T\}$, where $a_t \in N$ and both $a_0 = a_t = 0$.

In the MDP framework, the state is represented by the partial solution $a_{0:t-1}$, and an action involves visiting the node $a_t \in V \setminus \{a_{0:t-1}\} \cup F \cup \{0\}$. The reward is represented by $R = \sum_{t=1}^{T} t_O^{a_{t-1}}$, where $t_O^{a_{t-1}}$ indicates the travel time at step t, and the state transition is described by $a_{0:t} = \{a_{0:t-1}, a_t\}$. The policy $p(\pi|G)$ maps the graph G to a solution π. It is expressed as $p(\pi|G) = \prod_{t=1}^{T} \pi_\theta(a_t|a_{0:t-1}, G)$, where $\pi_\theta(a_t|a_{0:t-1}, G)$ denotes the node selection probabilities.

3.2 Policy Network with GCN and Edge Feature Integration

A GCN-based policy network is used to process edge features and extract information from both nodes and edges, enabling the learning of the strategy π_θ, The encoder generates partial embeddings for nodes and edges, which are then combined using a self-attention mechanism. During decoding, the model incorporates node embeddings and contextual information at each step to generate a probability distribution, based on which the next node is selected. This process repeats until all customers have been visited.

Encoder. The encoder extracts the node features $x_i = (v_i, q_i)$ using linear projection into a d_h-dimensional space ($d_h = 128$), while the edge features $e_{ij} = (d_{ij}, s_{ij})$ using a GCN to generate the corresponding embeddings. Here, v_i represents the coordinates of the node, q_i denotes the customer demand, d_{ij} and s_{ij} represent the distance and slope between nodes, respectively. The process is as follows:

$$\begin{aligned} h_{e_i}^0 &= \mathrm{GCN}(d_{ij}) + \mathrm{GCN}(s_{ij}) \\ h_i^0 &= \left[W_N^0 x_i + b_N, h_{e_i}^0 \right] W^0 + b_0 \end{aligned} \tag{12}$$

Where W_N^0, b_N, W^0, and b_0 are trainable parameters. The encoder is processed through L (L=3) sublayers. Specifically, for the l-th sublayer, its input h^{l-1} first computes the attention value through a multi-head attention (MHA)layer with M (M=8) heads. Each attention head h^{l-1} computes the query Q, key K, and value V. The outputs of all heads are concatenated to form the final MHA output, preserving the input shape. This process can be expressed as:

$$\begin{aligned} Q_{lm}, K_{lm}, V_{lm} &= W_{lm}^Q h^{l-1}, W_{lm}^K h^{l-1}, W_{lm}^V h^{l-1} \\ H_{lm} &= soft\max\left(\frac{Q_{lm} \times (K_{lm})^T}{\sqrt{d_k}} \right) V_{lm} \\ \mathrm{MHA}(h^{l-1}) &= \left[H_{l1}, H_{l2}, \cdots H_{lM} \right] W_l^O \end{aligned} \tag{13}$$

where $\forall m \in \{1,\cdots,M\}$, $\forall l \in \{1,\cdots,L\}$, and $d_q = d_k = d_v = d_h / M$. The trainable parameters are $W_{lm}^{Q} h^{l-1}$, $W_{lm}^{K} h^{l-1}$, $W_{lm}^{V} h^{l-1}$, and W_l^{O}. The output of the $\mathrm{MHA}\left(h^{l-1}\right)$ is then further processed through skip connections[11], followed by instance normalization (IN) [12]and the feed-forward (FF) layer, resulting in the output of the current layer node embedding. The specific process is as follows.

$$\begin{aligned} \tilde{h}^{l} &= \mathrm{IN}^{l}\left(h^{l-1} + \mathrm{MHA}\left(h^{l-1}\right)\right) \\ h^{l} &= \mathrm{IN}^{l}\left(\tilde{h}^{l} + \mathrm{FF}\left(\tilde{h}^{l}\right)\right) \end{aligned} \tag{14}$$

The graph embedding $\bar{h}$ is the average of the node embeddings h^{L}.

Decoder. After receiving the encoder output, the decoder calculates the node selection probabilities, thereby determining the nodes to visit at each step. It utilizes the graph embedding $\bar{h}$, the embedding of the previously visited node h_{t-1}^{L}, and the current vehicle state v^{t} as contextual information H_t^{c}, and computes an attention glimpse over this context using a multi-head attention mechanism. The specific process is as follows:

$$\begin{aligned} Q_t &= \mathrm{FF}\left(\left[W_1 v^{t} + b_1, h_{t-1}^{L}\right]\right) \\ H_t^{c} &= \left[\bar{h}, Q_t\right] \\ H_t^{g} &= \mathrm{MHA}\left(W_g^{Q} H_t^{c}, W_g^{K} h^{L}, W_g^{V} h^{L}\right) \end{aligned} \tag{15}$$

The parameters W_1, W_g^{Q}, W_g^{K}, W_g^{V} and b_1 are trainable. The vehicle state $v^{t} = \left[w^{t}, e^{t}\right]$, where w^{t} and e^{t} represent the vehicle load capacity and remaining battery level respectively. Then, the compatibility u_t between H_t^{g} and all vertices is computed using single-head attention.

$$u_t^{i} = \begin{cases} C \bullet \tanh\left(\dfrac{Q_t^{c}\left(K_t^{c}\right)^{T}}{\sqrt{d_k}}\right), & mask_t^{i} = 1 \\ -\inf, & otherwise \end{cases} \tag{16}$$

Where $Q_t^{c} = W_c^{Q} H_t^{g}$, $K_t^{c} = W_c^{K} h^{L}$, $C = 10$. When $mask_t^{i} = 1$, it indicates that the feasible node *i* is not masked at step *t*. Finally, the *softmax* function is applied to compute the probability distribution for selecting a node at step t, as follows:

$$P^{t} = softmax\left(u_t\right) \tag{17}$$

The decoder will perform node selection at each step until the demands of all nodes are met and all vehicles return to the depot.

3.3 Policy Optimization

To improve training stability and efficiency, multiple sampling tracks inspired by POMO [7] are used, each starting from a different customer to generate N_C distinct solutions. Policy optimization follows the REINFORCE algorithm [13], using gradient ascent to maximize reward (i.e., minimize total time). The baseline $b(s)$, shared across trajectories, is the average reward over all tracks. This process is illustrated below.

$$\begin{aligned} \nabla_\theta J(\theta) &= \frac{1}{N_C}\sum_{\lambda=1}^{N_C}\left(R\left(\pi_\theta^\lambda\right)-b(s)\right)\nabla_\theta \log P_\theta^\lambda\left(a|s\right) \\ b(s) &= \frac{1}{N_C}\sum_{\gamma=1}^{N_C} R\left(\pi_\theta^\gamma\right) \end{aligned} \tag{18}$$

Where $\nabla_\theta J(\theta)$ represents the gradient of the loss function, and $R\left(\pi_\theta^\gamma\right)$ denotes the reward of the λ-th trajectory. The overall training process is shown in Algorithm 1.

Algorithm 1 Training of our policy network

Input: Instance with a set of nodes V, number of epochs E, batched size B, step limits T;
Initialization: initial parameters for policy network π_θ^λ for sub-track λ;
for $epoch = 1,2,\ldots,E$ do
  for $batch = 1,2,\ldots,B$ do
    Select multiple starting nodes $\left\{a_1^1,\cdots,a_1^\lambda,\cdots,a_1^{N_C}\right\}$;
    for $\lambda = 1,2,\ldots,N_C$ do
      while $t < T$ do
        Calculate the action $a_t^\lambda \sim \pi_\theta\left(a|s_t^\lambda\right)$, $t \leftarrow t+1$;
      end
      Calculate the reward $R\left(\pi_\theta^\lambda\right)$;
    end
    $R(\pi_\theta) = -\max\left(R\left(\pi_\theta^\lambda\right)\right)$;
    Update baseline $b(s)$ and Calculate gradient $\nabla_\theta J(\theta)$;
    Update θ by $\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta)$.
  end
end

4 Experimentally

In this section, we evaluate the proposed DRL method for EVRP-BSS. All experiments are conducted on a device with a 10-core Intel i7 CPU and a single RTX 4090 GPU.

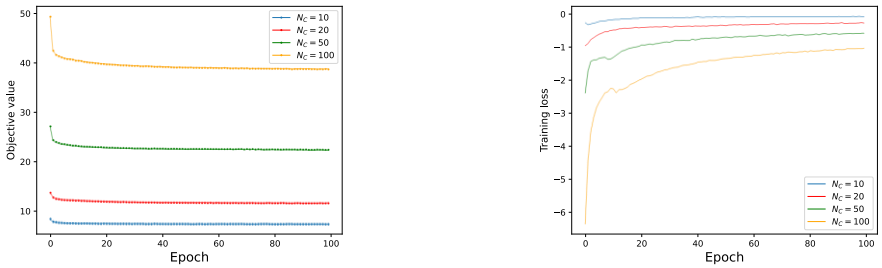
4.1 Experimental Settings

Following prior studies [10], we generate four types of stochastic instances with 10, 20, 50, and 100 customer nodes, denoted as C10, C20, C50, and C100, respectively. Each instance includes 4 (for 10/20 customers) or 8 (for 50/100 customers) battery swapping stations. Customer and swapping station coordinates are uniformly sampled within [0, 100]×[0, 100], and the depot is placed within [25, 75]×[25, 75]. Customer demand values are randomly sampled from the discrete set $\{0.25,\ 0.5,\ 0.75,\ 1\}$. The vehicle physical parameters are set according to the literature [14]. Meanwhile, the vehicle average traveling speed is set to 50 km/h, and the battery swapping time is set to 5 min.

During training, the model is trained for 100 epochs using 10,000 dynamically generated instances per epoch and optimized with Adam (learning rate $\eta = 10^{-4}$). For testing, 200 random instances are generated per problem size, following the same distribution as training.

4.2 Comparison Analysis

The proposed DRL model is compared with six baseline methods, including: exact solver (GUROBI [15]), classical heuristics (ACO [16], ALNS [14]), and DRL-based models (AM [6], POMO [7], EV-RL [10]). We adopt three decoding strategies: greedy (g.); sampling (s.), with s.1280 indicating 1,280 samples per instance; and coordinate augmentation (aug.8), involving seven geometric transformations to diversify solutions. Fig. 3 shows the training process of EVRP-BSS, where the average negative reward (total time) decreases steadily, and the loss gradually stabilizes, indicating convergence to a stable policy.



(a) Training reward (b) Training loss

Fig. 3. training curves for different problem sizes

Table 2 and Table 3 report the experimental results on randomized instances of varying sizes, including average objective values, gaps, and running times. Due to the exponential increase in computation time, GUROBI is only applied to smaller instances (C10 and C20). Overall, the proposed DRL method achieves superior solution quality and computational efficiency. Compared with traditional heuristics and other DRL baselines, it consistently yields better results, and the performance advantage becomes more pronounced as problem size increases. For example, the gap between our method and POMO grows from 0.29% to 0.96% as the number of customers increases from 10 to 100.

Table 2. Comparison of EVRP-BSS on C10-4 and C20-4.

Method	C10			C20		
	Obj.	Gap	Time(s)	Obj.	Gap	Time(s)
GUROBI	**6.92**	**0.00%**	**0.47**	**10.74**	**0.00%**	**21.50**
ALNS	7.65	10.55%	1.47	12.32	14.71%	20.78
ACO	7.42	7.22%	2.39	11.53	7.36%	6.12
AM (aug.8, g.)	8.73	16.97%	0.10	13.18	22.72%	0.18
AM (s.1280)	7.71	11.42%	0.10	11.98	11.55%	0.14
EV-RL (aug.8, g.)	8.06	16.47%	0.12	13.09	21.88%	0.21
EV-RL (s.1280)	7.44	7.52%	0.07	11.90	10.80%	0.13
POMO (aug.8, g.)	7.43	7.37%	0.06	11.58	7.82%	0.12
Ours (aug.8, g.)	7.41	7.08%	0.07	11.50	7.08%	0.12

Table 3. Comparison of EVRP-BSS on C50-8 and C100-8.

Method	C50			C100		
	Obj.	Gap	Time(s)	Obj.	Gap	Time(s)
GUROBI	-	-	-	-	-	-
ALNS	31.02	39.98%	236.91	-	-	-
ACO	24.59	10.97%	23.52	46.33	19.84%	75.69
AM (aug.8, g.)	26.61	20.08%	0.59	46.47	20.20%	1.34
AM (s.1280)	25.67	15.84%	0.32	46.28	19.71%	0.71
EV-RL (aug.8, g.)	26.48	19.50%	0.44	45.19	16.89%	1.08
EV-RL (s.1280)	25.49	15.03%	0.33	45.01	16.43%	0.69
POMO (aug.8, g.)	22.37	0.95%	0.30	39.03	0.96%	0.70
Ours (aug.8, g.)	**22.16**	**0.00%**	**0.30**	**38.66**	**0.00%**	**0.71**

4.3 Generalization on Benchmark Datasets

We evaluated the generalization performance of our method on larger-scale benchmark instances. Instances were selected from the Solomon benchmark [17] and the Gehring and Homberger benchmark [18], and modified to construct the test dataset. Specifically, each customer node retains its coordinates and demand-to-capacity ratio, while the coordinates of battery swapping stations are randomly generated within predefined bounds. Each instance is defined by its name (“Ins.”), the number of swapping stations

("Sta."), and customers ("Cus.").

To evaluate generalization, we use the four models trained in Section 4.2 to solve benchmark instances and compare the proposed DRL method with POMO (aug.8, g.) and EV-RL (s.1280). Table 4 and Table 5 report the optimal and average objective values, along with the corresponding gaps. Although the proposed DRL method is slightly outperformed by POMO on C221 and C241 in terms of average objective values, it consistently surpasses other DRL-based baselines across all benchmark instances, demonstrating robust generalization ability.

Table 4. Best results on benchmark instances.

Ins.	Sta.	Cus.	Ours (aug.8, g.)		POMO (aug.8, g.)		EV-RL (s.1280)	
			B.O.	B.G.	B.O.	B.G.	B.O.	B.G.
C101	8	100	19.20	**0.00%**	19.93	3.80%	23.08	20.21%
R101	8	100	19.62	**0.00%**	19.86	1.22%	28.26	44.04%
RC101	8	100	22.48	**0.00%**	23.74	5.61%	27.22	21.09%
C121	12	200	39.98	**0.00%**	40.75	1.93%	106.02	165.18%
C221	12	200	42.90	**0.00%**	44.66	4.10%	94.30	119.81%
C141	16	400	161.28	**0.00%**	166.67	3.34%	360.27	123.38%
C241	16	400	101.90	**0.00%**	102.88	0.96%	368.47	261.60%
Average			58.19	**0.00%**	59.78	2.73%	143.95	147.35%

Table 5. Average results on benchmark instances.

Ins.	Cus.	Best	Ours (aug.8, g)		POMO (aug.8, g)		EV-RL (s.1280)	
			A.O.	A.G.	A.O.	A.G.	A.O.	A.G.
C101	100	19.20	20.15	**4.95%**	20.80	8.33%	31.29	62.97%
R101	100	19.62	20.14	**2.65%**	20.38	3.87%	39.06	99.08%
RC101	100	22.48	23.55	**4.76%**	24.48	8.90%	35.22	56.67%
C121	200	39.98	41.36	**3.45%**	42.31	5.83%	116.61	191.67%
C221	200	42.90	46.19	7.67%	46.07	**7.39%**	102.86	139.77%
C141	400	161.28	176.78	**9.61%**	178.11	10.44%	561.67	248.26%
C241	400	101.90	109.75	7.70%	109.19	**7.15%**	571.46	460.80%
Average		58.19	62.56	**7.50%**	63.05	8.34%	208.31	257.96%

5 Conclusion

A novel deep reinforcement learning method is proposed in this paper for solving the EVRP-BSS using an encoder–decoder framework. The encoder integrates GCN and self-attention to extract edge features and embed richer information, while the decoder employs a masking scheme to ensure solution feasibility. Experiments on both random and benchmark instances show that the method consistently outperforms traditional heuristics and neural baselines. Future research will explore alternative optimization techniques for addressing EVRP and its variants [19–21].

References

1. Asghari, M., Al-e-hashem, S.: Green vehicle routing problem: A state-of-the-art review. International Journal of Production Economics 231, (2021)
2. Conrad, R.G., Figliozzi, M.A.: The Recharging Vehicle Routing Problem. IISE Annual Conference. Proceedings 1-8 (2011)
3. Zhong, X.Q., Zhong, W.F., Liu, Y., Yang, C., Xie, S.L.: Cooperative operation of battery swapping stations and charging stations with electricity and carbon trading. Energy 254, (2022)
4. Festa, P.: A brief introduction to exact, approximation, and heuristic algorithms for solving hard combinatorial optimization problems. 2014 16th International Conference on Transparent Optical Networks (ICTON) 1-20 (2014)
5. Vinyals, O., Fortunato, M., Jaitly, N.: Pointer networks. Advances in neural information processing systems 28, (2015)
6. Kool, W., Van Hoof, H., Welling, M.: Attention, learn to solve routing problems! arXiv preprint arXiv:1803.08475 (2018)
7. Kwon, Y.-D., Choo, J., Kim, B., Yoon, I., Gwon, Y., Min, S.: Pomo: Policy optimization with multiple optima for reinforcement learning. Advances in Neural Information Processing Systems 33, 21188-21198 (2020)
8. Wu, Y., Song, W., Cao, Z., Zhang, J., Lim, A.: Learning improvement heuristics for solving routing problems. IEEE transactions on neural networks and learning systems 33, 5057-5069 (2021)
9. Ma, Y., Li, J., Cao, Z., Song, W., Zhang, L., Chen, Z., Tang, J.: Learning to iteratively solve routing problems with dual-aspect collaborative transformer. Advances in Neural Information Processing Systems 34, 11096-11107 (2021)
10. Tang, M., Zhuang, W., Li, B., Liu, H., Song, Z., Yin, G.: Energy-optimal routing for electric vehicles using deep reinforcement learning with transformer. Applied Energy 350, (2023)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778. (2016)
12. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022 (2016)
13. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning 8, 229-256 (1992)
14. Goeke, D., Schneider, M.: Routing a mixed fleet of electric and conventional vehicles. European Journal of Operational Research 245, 81-99 (2015)
15. Gurobi Optimization, L.: Gurobi Optimizer Reference Manual. (2024)
16. Zhang, S., Gajpal, Y., Appadoo, S.S., Abdulkader, M.M.S.: Electric vehicle routing problem with recharging stations for minimizing energy consumption. International Journal of Production Economics 203, 404-413 (2018)
17. Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T., Subramanian, A.: New benchmark instances for the Capacitated Vehicle Routing Problem. European Journal of Operational Research 257, 845-858 (2017)
18. Gehring, H., Homberger, J.: A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In: Proceedings of EUROGEN99, pp. 57-64. Citeseer, (1999)
19. Chen, X., Li, J., Wang, Z., Chen, Q., Gao, K., Pan, Q.: Optimizing dynamic flexible job shop scheduling using an evolutionary multi-task optimization framework and genetic programming. IEEE Transactions on Evolutionary Computation (2025)

20. Du, Z.-s., Li, J.-q., Song, H.-n., Gao, K.-z., Xu, Y., Li, J.-k., Zheng, Z.-x.: Solving the permutation flow shop scheduling problem with sequence-dependent setup time via iterative greedy algorithm and imitation learning. Mathematics and Computers in Simulation 234, 169-193 (2025)
21. Zhu, M., Li, J., Li, J., Gao, K., Xu, Y., Yu, X., Li, W.: CNV_IWOABP: Collaboration of improved whale optimization algorithm and BP neural networks for copy number variations. Complex System Modeling and Simulation (2025)