Reinforcement Learning for Practical Express Systems with Mixed Deliveries and Pickups

JINWEI CHEN and ZEFANG ZONG, Tsinghua University
YUNLIN ZHUANG, Hitachi (China) Research Development Corporation
HUAN YAN, DEPENG JIN, and YONG LI, Tsinghua University

In real-world express systems, couriers need to satisfy not only the delivery demands but also the pick-up demands of customers. Delivery and pickup tasks are usually mixed together within integrated routing plans. Such a mixed routing problem can be abstracted and formulated as Vehicle Routing Problem with Mixed Delivery and Pickup (VRPMDP), which is an NP-hard combinatorial optimization problem. To solve VRPMDP, there are three major challenges as below. (a) Even though successive pickup and delivery tasks are independent to accomplish, the inter-influence between choosing pickup task or delivery task to deal with still exists. (b) Due to the two-way flow of goods between the depot and customers, the loading rate of vehicles leaving the depot affects routing decisions. (c) The proportion of deliveries and pickups will change due to the complex demand situation in real-world scenarios, which requires robustness of the algorithm. To solve the challenges above, we design an encoder-decoder based framework to generate high-quality and robust VRPMDP solutions. First, we consider a VRPMDP instance as a graph and utilize a GNN encoder to extract the feature of the instance effectively. The detailed routing solutions are further decoded as a sequence by the decoder with attention mechanism. Second, we propose a Coordinated Decision of Loading and Routing (CDLR) mechanism to determine the loading rate dynamically after the vehicle returns to the depot, thus avoiding the influence of improper loading rate settings. Finally, the model equipped with a GNN encoder and CDLR simultaneously can adapt to the changes in the proportion of deliveries and pickups. We conduct the experiments to demonstrate the effectiveness of our model. The experiments show that our method achieves desirable results and generalization ability.

CCS Concepts: • **Computing methodologies → Planning for deterministic actions;**

Additional Key Words and Phrases: Practical express systems; vehicle routing problem with mixed deliveries and pickup; reinforcement learning

Jinwei Chen and Zefang Zong contributed equally to this research.

This work was supported in part by The National Key Research and Development Program of China under grant 2020AAA0106000, the National Natural Science Foundation of China under U20B2060, 61971267, 61972223.

Authors' addresses: J. Chen, Z. Zong, H. Yan (corresponding author), D. Jin, and Y. Li, Tsinghua University, 30 Shuangqing Rd, Haidian Qu, Beijing Shi 100190, China; emails: {cjw20, zongzf19}@mails.tsinghua.edu.cn, {yanhuan, jindp, liyong07}@tsinghua.edu.cn; Y. Zhuang, Hitachi (China) Research Development Corporation, Beijing Shi 100190, China; email: ylzhuang@hitachi.cn.

1556-4681/2023/02-ART33 $15.00
https://doi.org/10.1145/3546952

ACM Reference format:
Jinwei Chen, Zefang Zong, Yunlin Zhuang, Huan Yan, Depeng Jin, and Yong Li. 2023. Reinforcement Learning for Practical Express Systems with Mixed Deliveries and Pickups. *ACM Trans. Knowl. Discov. Data.* 17, 3, Article 33 (February 2023), 19 pages.
https://doi.org/10.1145/3546952

1 INTRODUCTION

The rapid development of online shopping has led to a surge in logistics demand, which brings significant challenges to the efficiency of the existing express systems. Intelligent express systems are developed to find better and faster logistics plans to satisfy the needs of customers. In real-world scenarios, customers have demands to either deliver or pick up packages. An express system needs to distinguish diverse demands and dispatch vehicles to efficiently accomplish the tasks. In addition, the tasks with both delivery demands and pickup demands are usually mixed to improve efficiency in complex environments.

Research on optimizing the routing strategy of express systems in both academia and industry has drawn great attention in recent years [1, 27, 42, 48]. The inner routing process of express systems as described can be formulated as an extension of the famous **Vehicle Routing Problem** (**VRP**) [18]. In 1988, Casco, Golden, and Wasil et al. proposed a problem to extend the standard VRP, named the **Vehicle Routing Problem with Mixed Delivery and Pickup** (**VRPMDP**) [4]. As shown in Figure 1, routing tasks include both deliveries from the depot to customers and pickups from customers back to the depot. Many heuristics and meta-heuristics methods have been proposed to solve the VRPMDP, such as ant colony algorithm [40], tabu search [10] and so on. Although these methods can produce relatively good solutions, they need to spend massive time dealing with large-scale problems.

Recently, researchers begin to apply **deep reinforcement learning** (**DRL**) to solve VRP, and more general combinatorial optimization problems [9, 17, 33]. DRL can extract features and explore the optimization space of combinatorial optimization problems effectively, and learn the policy through interaction between the agent and the outer environment [26]. In addition, the model trained via offline data can compute the solutions for online instances quickly, which is very effective in practical applications. However, current methods cannot be directly used for VRPMDP due to three challenges. First, Even though successive pickup and delivery tasks are independent to accomplish, the inter-influence between choosing pickup task or delivery task to deal with still exists. Second, due to the two-way flow of parcels between the depot and customers, the loading rate, i.e., how many parcels a vehicle shall take when leaving the depot, needs to be carefully considered. Different loading rate initialization affects the entire routing quality. Finally, the proportion of deliveries and pickups changes in different express scenarios, while the algorithm applied to VRPMDP cannot adapt to the environment dynamics.

To solve the challenges above, we propose a DRL-based method to solve VRPMDP. The detailed neural network structure follows an encoder-decoder structure. First, we consider a VRPMDP instance as a graph and utilize a GNN encoder to extract the feature of instance effectively. The detailed routing solutions are decoded as a sequence by the decoder with attention mechanism. Second, we propose a **Coordinated Decision of Loading and Routing** (**CDLR**) mechanism. CDLR mechanism determines the loading rate after the vehicle return to the depot to avoid the influence of improper setting of loading rate on route decision. Finally, the model equipped with the GNN encoder and CDLR simultaneously can adapt to the changes in the proportion of deliveries and pickups.

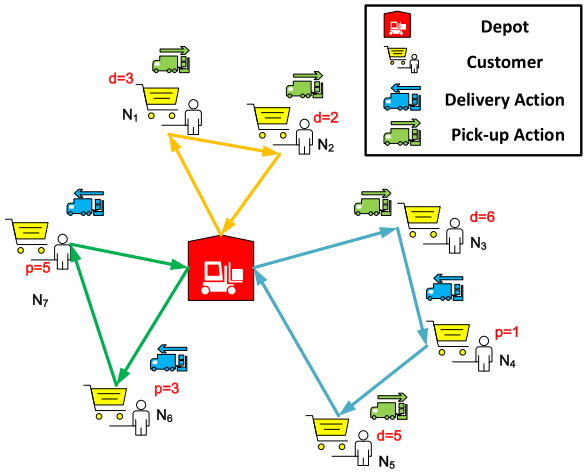


Fig. 1. A visualized instance of vehicle routing problem with mixed delivery and pickup. In the figure, d and p represent the customer's delivery demand and pick-up demand, respectively. A vehicle needs to transport parcels from the depot to customers who have delivery demand. Meanwhile, it also needs to take parcels back from customers who have pick-up demand.

To summarize, the main contributions of our work are as follows:

— To the best of our knowledge, we are the first one to solve Express Systems with Mixed Deliveries and Pickups by reinforcement learning.
— We propose an encoder-decoder based framework to solve VRPMDP. We utilize a GNN encoder to extract the features of instance effectively, and the detailed routing solutions are further decoded as a sequence by the decoder with attention mechanism. In addition, we propose the CDLR mechanism to determine the loading rate for solution improvement.
— Extensive experiments are conducted to demonstrate the effectiveness of our framework. The experiments show that our method outperforms other baselines. We also show the desirable generalization ability of our method.

The remainder of this article is as follows. We first introduce the related works of VRPMDP and deep learning for combinatorial optimization in Section 2. We then formulate the VRPMDP and present our approach in Sections 3 and 4, respectively. We show the experiment results in Section 5. Lastly, we conclude our work in Section 5.

2 RELATED WORK

2.1 Traditional Solutions for VRP with Mixed Delivery and Pickup

Casco et al. first formulated the VRP with Delivery and Pickup in 1988 [4]. The VRPDP can be further divided into more detailed problems according to different cases [43]. The case where a customer wishes to both deliver and pick up parcels is called *combined demands* and can be modeled as ***VRP with Simultaneous Deliveries and Pickups (VRPSDP)*** [31]. Another case where a customer only delivers or picks up parcels is called *single demands* and can be modeled as *VRPMDP*. *VRPMDP* can be considered as a special case of *VRPSPD*.

The main non-learning based methods currently used for VRPMDP include exact algorithms, heuristic algorithms and meta-heuristic algorithms. Hernández-Pérez et al. proposed a branch-and-cut method to obtain the exact solution of VRPMDP [13]. Due to its NP-hard nature, the exact

algorithm cannot get a solution in polynomial time. As an alternative, there are many heuristic methods proposed to solve VRPMDP. Psaraftis et al. firstly proposed the local search to solve VRPMDP [34]. On the basis, Lenstra and Schuur et al. proposed a two-phase method, which constructs the initial solution at the first phase and executes the local search at the second phase [22]. Tabu search [10] is one of the most popular meta-heuristic methods to guide a local heuristic search process, which explores the entire solution space and tries to overcome local optimality. Crispim J proposed a method, which integrates the advantages of tabu search and variable neighborhood descent, and applied this method on VRPMDP [7]. In addition, the ant colony optimization method and adaptive large neighborhood search are also popular VRPMDP methods. Wade and Salhi et al. developed a further improvement on the standard ant colony approach for VRPMDP [40]. Majidi et al. proposed an adaptive large neighborhood search heuristic including new removal and insertion operators to solve VRPMDP [25]. In VRPMDP, the heuristic methods are indeed better than the exact method in solution speed. However, shortcomings still exist on the unsatisfactory computation cost on growing problem scales. These methods are difficult to apply to decision-making in a timely manner.

2.2 Deep Reinforcement Learning

Deep learning methods can effectively extract high-dimensional features of the environment [19]. Reinforcement learning allows the agent to continuously interact with the environment so that the agent learns to conduct the optimal decision. To combine the feature extraction advantage of deep learning with the decision-making advantage of reinforcement learning, DRL methods are commonly used as an end-to-end decision-making learning method in recent years [32]. Mnih et al. proposed DQN [29], which utilizes a network (Q network) to calculate the Q-value, and they apply DQN to Atari for evaluating its effectiveness. For improving the convergence performance, they proposed an improved DQN [30]. The improved DQN used two Q networks with the same structure to calculate the target Q value and update the Q network parameters, respectively. Actor-Critic algorithm combines the policy gradient algorithm with DQN by introducing actor network and critic network [16]. Lillicrap et al. proposed DDPG [21] to make decisions for the continuous action space by combing Actor-Critic and DQN. DRL has numerous applications in robot control, recommendation system, routing decision, and so on.

2.3 DRL for Combinatorial Optimization

In the early stage, reinforcement learning has been applied for combinatorial optimization by Zhang et al. in 1995 [44]. In recent years, DRL has been applied to combinatorial optimization problem [36]. Vinyals et al. [39] proposed the Pointer Network that integrated the attention mechanism into the sequence-to-sequence model and trained the model to solve TSP. This encoder-decoder architecture is used by many researchers. Bello et al. [2] trained the PN model by reinforcement learning instead of supervised learning aiming at TSP. Nazari et al. [33] replaced the RNN encoder of Pointer network by a linear embedding layer with shared parameters, and this model is applied to the VRP. Kool et al. [17] used the adapted Transformer [38] for embedding, and optimized the policy by policy-gradient reinforcement learning. Based on the model proposed by Kool et al., Xin et al. [46] improved the diversity of policy by utilizing multi-decoder to train multiple construction policies. In addition, they proposed novel beam search scheme and Embedding Glimpse layer to improve the quality of the solution.

Some researchers have found that instances in combinatorial optimization problems can be modeled as graphs, and use graph neural networks as encoder to extract instance features. Dai et al. [15] used the graph embedding method and deep Q-learning algorithm [28] to solve

combinatorial optimization problems. Joshi et al. [14] proposed a model based on the **graph convolution network** (**GCN**) and supervised learning to solve TSP. This work considered a TSP instance as a graph and used the residual gated graph ConvNets [3] to embed the nodes and edges in the graph for supervised learning. Duan et al. [9] designed a joint training method for VRP. They used the adapted GCN to embed the nodes and edges in the graph and designed one decoder for sequence prediction and one decoder for edge classification. Joining the results of two decoders makes the training process converge quickly and improve the solution quality. Ma et al. [24] proposed **Graph Pointer Network** (**GPN**) to solve large-scale TSP. In addition, a hierarchical RL framework is utilized with GPN to solve TSP with a time window constraint. Lei et al. [20] proposed a E-GAT Network to capture edge information in the graph structure and consider the residual connections for effective embedding. They utilized PPO and REINFORCE algorithm to optimize the network, respectively.

Besides the construction based methods generating partial solutions step by step as above, others also focus on improving current feasible solutions continuously. Deudou et al. [8] proposed a method to learn the heuristics for TSP using reinforcement learning and the performance is as good as the well-known heuristics solver OR-Tools [11]. Chen and Tian [5] considered the optimization problem as a rewriting problem. They propose Neuwriter to refine the initial solution towards the optimum. While Lu et al. [23] proposed a "Learn to Improve" (L2I) model to start with an initial solution and iteratively refine the solution by an improvement operator, which was applied on VRP and obtained nice results which outperform a state-of-the-art heuristic solver LKH-3 [12].

Although the research of DRL for combinatorial optimization has developed for several years, there is no research on utilizing DRL for VRPMDP as far as we know.

3 PROBLEM FORMULATION

In this section, we formulate the problem of VRPMDP. VRPMDP needs to consider not only parcels transported from the depot to customers for delivery, but also pickups from customers to the depot, which is an extension of the VRP.

In the VRPMDP, a depot, a vehicle and a set of customers with either delivery or pick-up demand (one customer only have one type of demand in a fundamental setting) are given. It can be defined on a graph $G = (N, A)$, where $N = C \cap \{0\} = \{0, 1, \ldots, n\}$ is the node set and $A = \{(i, j)|i, j \in V\}$ is the edge set. And C is the customer set which can be divided into two categories. In order to distinguish two different customers, we set two different attributes (delivery demand d_i and pick-up demand p_i) for each customer. For the delivery customer, its pickup demands will be set to 0, vice versa. The reason for this setting is to avoid confusion between different parcels in the process of parcel assembling. Simply using positive or negative values for indication may lead to confusion on individual pickup and delivery demands. V is the capacity of the vehicle.

The delivery and pickup orders of VRPMDP are mixed, which means a vehicle shall decide how many parcels to take when it departs from the depot. Therefore, the concept of loading rate η is introduced to characterize the ratio of parcels carried according to the vehicle capacity (the remaining space allows performing the pick-up operations). The loading rate η is calculated as $\eta = \frac{V_0}{V}$ (where V_0 means the amount of parcels carried when the vehicle leaves the depot). The major parameters of VRPMDP are organized in Table 1.

$\pi = \{\pi_1, \ldots, \pi_k, \ldots, \pi_K\}$ denotes the solution with π_k representing the sub-route vehicle departing from the depot for the kth time. Specifically, π_k records the order of visited nodes, i.e., $\pi_k = (v_{k,1}, \ldots, v_{k,t}, \ldots,)$ with $v_{k,t}$ denoting the node that vehicle visits in the tth step in π_k. On the premise of visiting all customers, our goal is to minimize the total distance traveled by the vehicle. We give the formulation of VRPMDP as below:

Table 1. Major Parameter Definitions of VRPMDP

Parameter	Definition
C	The customers need to be visited
N	The nodes(customers and depot)
A	The edges connecting each two nodes
V	The capacity of vehicles
T	The distance cost matrix
d_i	The delivery demands of the customer i
p_i	The pick-up demands of the customer i
η	The loading rate of vehicle leaving the depot

$$\min \sum_{k=1}^{K} \sum_{i=1}^{|\pi_k|-1} T(v_{k,i}, v_{k,i+1}), \tag{1}$$

$$s.t. \quad \pi_1 \cup \pi_2 \cup \cdots \cup \pi_k \cdots \cup \pi_K = N, \tag{2}$$

$$\pi_m \cap \pi_n = 0, \forall m, n \leq K, m \neq n, \tag{3}$$

$$0 \leq \eta_k - \sum_{i=r}^{s} d_{k,i} + \sum_{i=r}^{s} p_{k,i} \leq V, 0 \notin \pi_k^{r:s}, \forall k \leq K. \tag{4}$$

where constraints (2) and (3) ensure all customers visited and only visited once, constraint (4) ensures the total amount of parcels on the vehicle during the whole process does not exceed the capacity of vehicle.

4 METHODOLOGY

Given a VRPMDP instance as input, we train an encoder-decoder based model to output the routes and corresponding loading rate of vehicles. The encoder computes the h-dimensional embeddings for every node and the decoder takes the embeddings as input to produce the solution of VRPMDP. When the decoder parses the solution step by step, the CDLR mechanism constantly refines the range of loading rate η and finally determines it when returning to the depot. In addition, we use REINFORCE algorithm to train the entire network structure. The framework for VRPMDP we propose is shown in Figure 2. The framework consists of two parts: (1) Training Part: The training instances are fed into our encoder-decoder model to train the model parameter. (2) Inferring Part: The model with trained parameters takes testing instances as input and outputs the vehicle routing and corresponding loading rate.

4.1 RL Setting for VRPMDP

In reinforcement learning, the agent gets feedback from interactions with the environment and makes decisions based on the feedback. To design the RL strategy for VRPMDP, we formulate the VRPMDP solution as a **Markov decision process** (**MDP**) and the key components of our RL settings such as state, action, reward, and policy are explained as follows:

— **State** represents the information used by the agent to make decisions. The state includes the current partial vehicle routing solution, the unvisited customers, the remaining capacity for delivery and pickup parcels. When the agent takes an action, the state updates based on the changes in environmental information immediately.
— **Action** represents the decision made by the agent based on the current state. In the routing decision, the action is to select the next customer to visit.

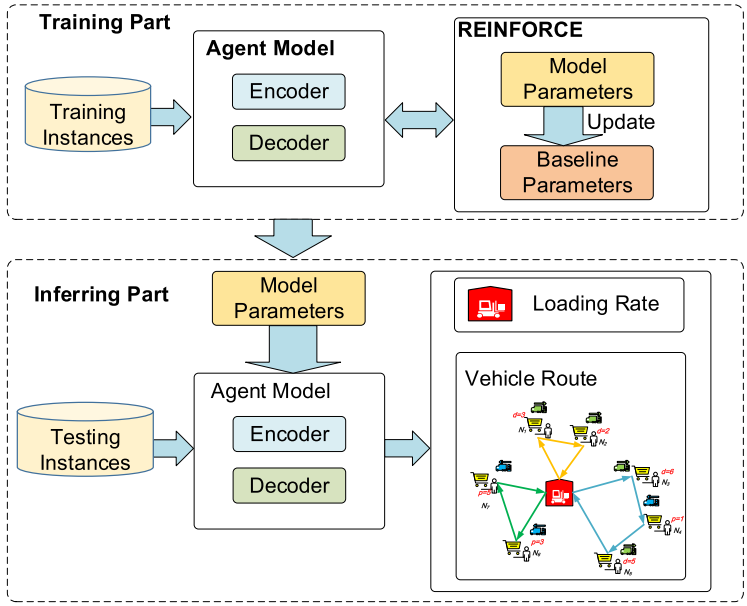


Fig. 2. The framework includes two parts: (1) Training Part: The training instances are fed into our encoder-decoder model. And the parameter of model is optimized by REINFORCE algorithm with rollout baseline, which belongs to Policy Gradient methods. (2) Inferring Part: The model with trained parameters takes testing instances as input and outputs the vehicle routing and corresponding loading rate.

— **Reward** is the feedback of the environment to the agent's actions. Given a VRPMDP instance s, the solution is represented as $\hat{\pi}|s = \{\hat{\pi}_1, \hat{\pi}_2, \ldots, \hat{\pi}_n\}$, where $\hat{\pi}_i \in \{0, 1, \ldots, |C|\}$. The actions of the agent should minimize the total distance of the vehicle routes, thus we define the reward as follows:

$$r(\hat{\pi}|s) = -\sum_{i=1}^{|\hat{\pi}|-1} T(\hat{\pi}_{i+1}, \hat{\pi}_i), \tag{5}$$

— **Policy** determines what action to choose in different states. With $\hat{\pi} = \{\hat{\pi}_1, \hat{\pi}_2, \ldots, \hat{\pi}_n\}$ as a solution of a VRPMDP instance, the policy $p(\hat{\pi}|s)$ can be characterized by the network we utilized. With the parameter of network model θ, the definition of policy can be given as

$$p(\hat{\pi}|s;\theta) = \prod_{i=1}^{n} p(\hat{\pi}_i|s, \hat{\pi}_{1:i-1};\theta). \tag{6}$$

4.2 GNN-Based Encoder

With the coodinates of nodes $x_i (i = 0, \ldots, |C|)$ and delivery demands $d_i (i = 1, \ldots, |C|)$ and pick-up demands $p_i (i = 1, \ldots, |C|)$ of customers, we use a linear projection to get the initial embedding of all nodes:

$$h_i^{(0)} = \begin{cases} W_0^x x_i + b_0^x, & i = 0, \\ W^x [x_i, d_i, p_i] + b^x, & i = 1, \ldots, |C|, \end{cases} \tag{7}$$

where W_0, b_0^x, W^x, b^x are trainable parameters. The initial embedding does not express the relationship between different nodes. Considering that the graph neural network can effectively capture and represent the relationship among nodes, we utilize a GNN-based method to embed the instance as illustrated in Figure 3.

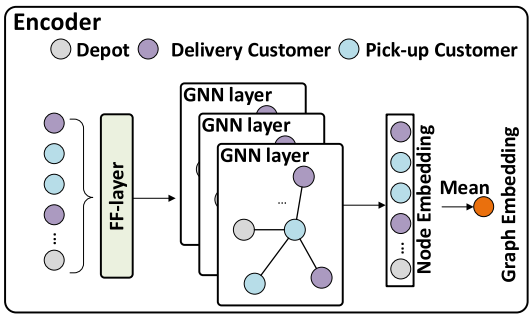


Fig. 3. The Framework of our GNN-based Encoder. The FF-layer projects the instance to the initial embedding, and L-stacked GNN layers capture the relationship between each node in the instance and produce the embedding of all nodes. Finally, the graph embedding is calculated by averaging the embedding of all nodes.

GNNs are effective methods to represent the graphs [35]. Neighborhood aggregation strategy is utilized to represent the nodes by recursively aggregating and transforming representation vectors of the neighboring nodes. In an express system, all customers along with the depot and their potential connection relationships compose a graph structure. We expect to construct the VRPMDP instance as the fully connected graph and get the representation vector of nodes through the GNN. The Encoder consists of L consecutive GNN layers [24, 47], each layer of the GNN is represented as

$$h_i^l = \epsilon h_i^{l-1} \Psi + (1-\epsilon) AGG\left(\left\{h_j^{l-1}\right\}_{j \in adj(i) \cup i}\right), \tag{8}$$

where $h_i^l \in \mathbb{R}^{d_l}$ is the output of the lth layer ($l \in 1, \ldots, L$), $\Psi \in \mathbb{R}^{d_{l-1} \times d_l}$ is a linear projection to capture information about node i itself, $adj(i)$ is the set of nodes which are adjacent to node i, $AGG : \mathbb{R}^{d_{l-1}} \rightarrow \mathbb{R}^{d_l}$ is a linear projection to aggregate the information of $adj(i)$, and ϵ is a trainable parameter to adjust the weight of sum.

To extract the environment information more comprehensively, we also need to obtain the embedding of the entire instance. We consider averaging the embedding of each node to calculate instance graph embedding $\overline{h}$:

$$\overline{h} = \frac{1}{N} \sum_{i=0}^{n} h_i. \tag{9}$$

4.3 Sequence Decoder

Taking the encoded information as input, the decoder outputs the solution of a VRPMDP instance, which consists of a sequence of nodes and the corresponding vehicle loading rate. Our decoder contains two attention layers following Kool et al. [17] and the decoding process is shown in Figure 4. The first attention layer calculates the context embedding by the multi-head mechanism which can empower the model to integrate information from different sub-spaces [38]. And the second layer utilizes the single-head mechanism to compute the probability of selecting the nodes.

At the timestep $t (t \geq 1)$, the decoder produces the next node to visit based on the context embedding. Considering that the vehicles carry both delivery and pickup parcels, the context $c^{(t)}$ is consist of the graph embedding $\overline{h}$, the embedding of node visited at last step $h_{\pi_{t-1}}$, the amount of delivered goods on board $D^{(t)}$ and the amount of picked up goods on board $P^{(t)}$:

$$c^{(t)} = \begin{cases} [\overline{h}; h_{\pi_{t-1}}; D^{(t)}; P^{(t)}], & t \geq 2, \\ [\overline{h}; h_0; D^{(t)}; P^{(t)}], & t = 1, \end{cases} \tag{10}$$

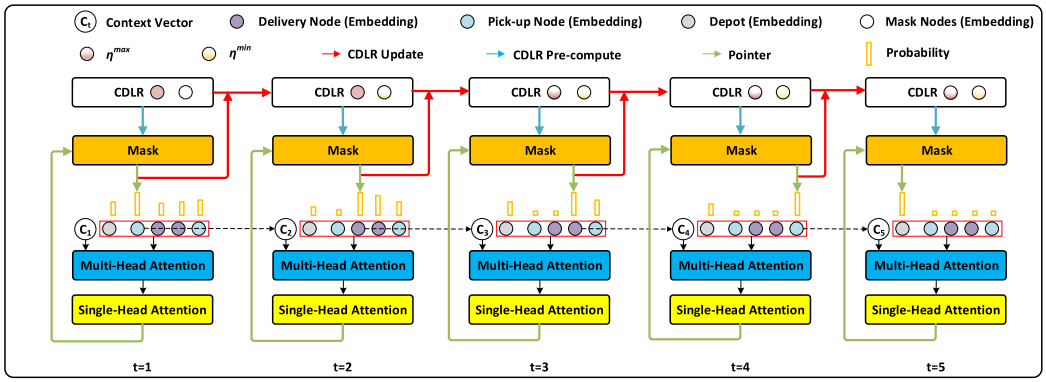


Fig. 4. The architecture of our decoder, which parses the routes step by step. We use multi-head attention mechanism to further process node embeddings based on current states, and generate unfiltered selection weights of each node via a single-head attention structure. A mask mechanism filters out unqualified nodes, either already visited or violating capacity constraints. The identification of capacity feasibility relies on the CDLR mechanism, which updates η^{min} and η^{max} step by step. Finally, we sample the node to visit based on masked probability distributions, and update the range of loading rate.

where [;] represents the concatenation operator. And the context vector is fed into the multi-head attention layer for update. First, we calculate the query vector $q_{(c)} \in \mathbb{R}^{d_k}$, key vector $k_i \in \mathbb{R}^{d_k}$ and value vector $v_i \in \mathbb{R}^{d_v}$ for each nodes based on the context embedding and node embedding:

$$q_{(c)} = W_Q c^{(t)}, \tag{11}$$

$$k_i = W_K h_i, \tag{12}$$

$$v_i = W_V h_i, \tag{13}$$

where $W_Q \in \mathbb{R}^{d_k \times d_h}$, $W_K \in \mathbb{R}^{d_k \times d_h}$ and $W_V \in \mathbb{R}^{d_v \times d_h}$ are projection matrix, i represents index of the node. In the decoder, we design a mask to filter out inaccessible nodes, which may be nodes have been visited or the demand of customer cannot be satisfied. And the mask is also used to determine the range of loading rate which will be described in the next sub-section. With the query vector and value vector, we can compute the compatibilities of all nodes as

$$u_{(c)j} = \begin{cases} \frac{q_{(c)}^T k_j}{\sqrt{d_k}} & if \quad j \notin \pi_{t'}, \forall t' < t \\ -\infty & otherwise. \end{cases} \tag{14}$$

Then we calculate the attention weight for all node embedding h_i (i = 0, 1, . . . , n) and update the context vector by H-head (H=8) attention mechanism:

$$a_{(c)j} = \frac{e^{u_{(c)j}}}{\sum_i e^{u_{(c)i}}}, \tag{15}$$

$$c^{(t)} = \sum_{h=1}^{H} W_h \sum_{i=1}^{n} a_{(c)t}^h v_i^h, \tag{16}$$

where W_h is the parameter matrix for every head. Then the updated context embedding is taken as input of the single-head attention layer. The compatibilities of the query $u_{(c)j}$ with all nodes are

calculated, and the results are clipped in [-10,10] [2]:

$$u_{(c)j} = \begin{cases} 10\frac{q_{(c)}^T k_j}{\sqrt{d_k}} & if \quad j \neq \pi_{t'}, \forall t' < t, \\ -\infty & otherwise. \end{cases} \tag{17}$$

By normalizing the compatibility using softmax, we can obtain p_i^t, which is the probability of choosing customer i at timestep t. With the probability P_i^t, we can generate the probability of a partial VRPMDP tour P':

$$P_i^t = P(\hat{\pi}_{t=i} | \hat{\pi}_{1:t-1}; \theta) = \frac{e^{u_{(c)i}}}{\sum_j e^{u_{(c)j}}}, \tag{18}$$

$$P^{'} = \prod_{i \in \hat{\pi}} P_i^t. \tag{19}$$

Based on the calculated probability distribution, the sampling search strategy is utilized to select the next nodes and constructs the solution step by step. The sampling search strategy can sample several solutions and determine the best one as the result [6].

4.4 Coordinated Decision of Loading and Routing (CDLR)

In previous VRPMDP methods [41], the *loading rate* is considered as a fixed value when vehicle leaving the depot. Such a fixed setting is not flexible due to: (1) Inappropriate loading rate will result in limited decision space, (2) The visiting state of each customer changes with the operation of the vehicle, and a fixed loading rate is difficult to cope with the continuous changes of visiting state. Figure 5 shows four possible situations if the loading rate is fixed as an inappropriate value. Figure 5(a) shows that when vehicle capacity $V = 20$ and $\eta = 0.3$, the delivery demand of N_3 cannot be satisfied in the entire process. Similarly, N_7 in Figure 5(b) cannot be visited at the first step otherwise violating the constraints. In Figure 5(c), the remaining two nodes N_1 and N_2 cannot be visited in one tour due to the delivery goods in one vehicle cannot satisfy the demands of two nodes in one time. This means the vehicle needs to visit two nodes in two tours which increases the distance inevitably. Figure 5(d) shows a similar situation that the fixed inappropriate loading rate leads to an increased route distance.

In order to avoid the inflexibility and performance degradation caused by fixed loading rate, we propose the CDLR mechanism to determine the *loading rate* with every tour.

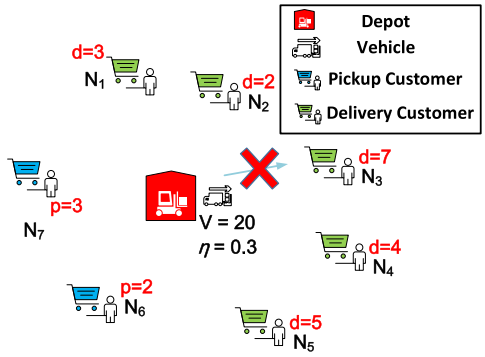
As we know, in the whole process of the vehicle visiting each customer, the following capacity constraints need to be met:

$$\begin{cases} \eta V + P_k^{sum} - D_k^{sum} \leq V, & k \geq 0, \\ D_k^{sum} \leq \eta V, & k \geq 0, \end{cases} \tag{20}$$

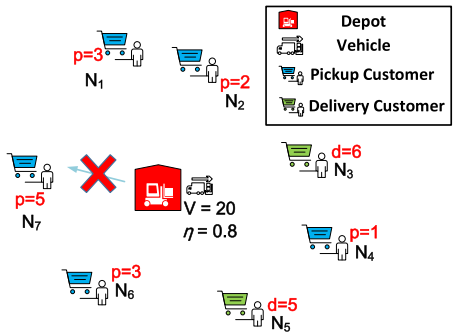
where V is the capacity of the vehicle, k is the timestamp, P_k^{sum} and D_k^{sum} represent currently picked parcels and delivered parcels respectively (be set to zero when the vehicle is at the depot). Based on the Constraint above, We set two factors as the maximum loading rate η^{max} and the minimum loading rate η^{min}, which represent the upper and lower bounds of the loading rate. We update η^{max} and η^{min} while determining the next customer to visit, so as to narrow the value interval for selection. The update process is as below:

$$\eta_k^{max} = \begin{cases} 1, & k = 0, \\ min\left(\eta_{k-1}^{max}, 1 - \frac{P_k^{sum} - D_k^{sum}}{V}\right), & k > 0, \end{cases} \tag{21}$$

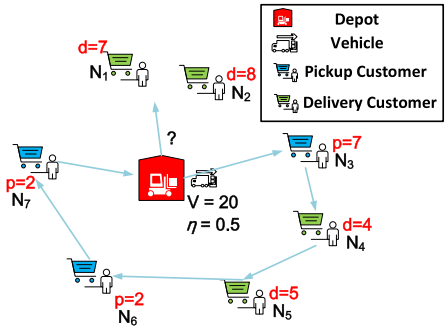
$$\eta_k^{min} = \begin{cases} 0, & k = 0, \\ \frac{D_k^{sum}}{V}, & k > 0, \end{cases} \tag{22}$$



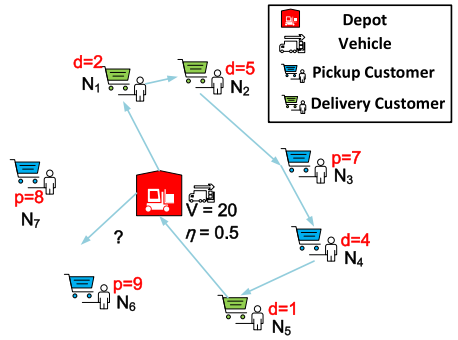
(a) situation 1:delivery customer N_3 can not be satisfied



(b) situation 2:pickup customer N_3 can not be satisfied



(c) situation 3:delivery customers N_1 and N_2 can not be satisfied in one tour



(d) situation 4:pickup customers N_6 and N_7 can not be satisfied in one tour

Fig. 5. Four situations may occur when the loading rate is fixed. (a) and (b) show the situation that some nodes cannot be satisfied when the inappropriate loading rate is set. (c) and (d) show that the distance will be increased due to unnecessary distance when the loading rate is not set properly.

And they must meet the constraints as below:

$$\eta_k^{max} \geq \eta_k^{min}, \tag{23}$$

When a vehicle needs to choose the next nodes to visit, we will pre-calculate the η_{max} and η_{min} to choose the remaining customers as below:

$$\eta_k^{max(i)} = min\left(\eta_k^{max}, 1 - \frac{P_k^{sum} + P^{(i)} - D_k^{sum} - D^{(i)}}{V}\right), \tag{24}$$

$$\eta_k^{min(i)} = \eta_k^{min} + \frac{D^{(i)}}{V}, \tag{25}$$

where i means the remaining ith node, $D^{(i)}$ and $P^{(i)}$ are corresponding delivery demand and pickup demand of the remaining ith node. When pre-calculated $\eta_k^{max(i)}$ and $\eta_k^{min(i)}$ cannot satisfy the constraint (23) with any remaining customer node, the vehicle returns to the depot and we set η_k^{min} as the loading rate of the current tour. The pre-calculate process is incorporated into the mask.

Compared to the previous methods, we determine the loading rate after vehicles returning to the depot as a coordinated decision-making process, instead of fixing the loading rate directly before

the vehicle leaves the depot. Our mechanism can effectively determine the loading rate according to the constructed routes and reduce the bad influence of loading rate on route strategy, which will be further proved in the experiment.

4.5 Training Method

Using the network we demonstrate, we can obtain a probability distribution $p_\theta(\hat{\pi}|\theta_s)$ with a given instance s. We define the loss as

$$L(\theta|s) = \mathbb{E}_{p_\theta(\hat{\pi}|\theta s)}[Dis(\hat{\pi})], \tag{26}$$

where $\hat{\pi}|s$ is the solution of instance s, $Dis(\hat{\pi})$ is the total length of solution. The training method we use is the REINFORCE algorithm [45], which belongs to the category of policy-gradient methods [37]. REINFORCE algorithm uses the probability distribution defined by the policy model to select actions during the interaction between the agent and the environment. Once the agent finds an action with a high reward value, it updates the parameter of the model to increase the probability of selecting this action in the subsequent interaction process. We use REINFORCE algorithm with rollout baseline, which was first proposed by Kool et al. [17]:

$$\nabla L(\theta|s) = \mathbb{E}_{p_\theta(\hat{\pi}|s)}[(Dis(\hat{\pi}) - b(s))\nabla log p_\theta(\hat{\pi}|s)]. \tag{27}$$

The baseline $b(s)$ is a deterministic greedy rollout policy that is defined by the best model so far. $b(s)$ that is not related to action a given by the trajectory policy can effectively improve the convergence speed. The baseline is compared with the training policy on the evaluation dataset using greedy search strategy at the end of every epoch. The greedy search strategy drives the agent to select the node with high probability and provide a locally optimal solution quickly. If the current model has significant improvement according to paired t-test (significance level $\alpha = 0.05$) on the validation set, we update the baseline parameter with the parameter of the current model. The whole training process of our model is shown in Algorithm 1.

5 EVALUATE

In this section, we conduct experiments on two datasets to answer the following research questions:

- **RQ1:** How does our proposed method perform on VRPMDP compared to the existing heuristic and reinforcement learning methods?
- **RQ2:** How is the generalization ability of our method on extended testing datasets with different sizes and different delivery-pickup ratios?
- **RQ3:** How does the CDLR mechanism of contribute to the overall result?
- **RQ4:** How do the results differ when the hyper-parameter of GNN and the instance graph constructing method changes?

5.1 Datasets

We evaluate the method on two datasets as follows:

- **Random Generated Dataset.** We generate the random dataset following the settings as below:
  Locations: The locations of customer i, (x_i,y_i) is sampled from a 1×1 square uniformly. And each x_i and y_i is distributed in (0,1) uniformly. The depot is also randomly sampled in the square as described.
  Demands: For each delivery customer, the demand for delivery is distributed in [1,9] and is set as 0 for pick-up. For each pick-up customer, the demand for pick-up is distributed in [1,9]

ALGORITHM 1: Model Training via REINFORCE with Rollout Baseline

Input: Number of epochs E, batch size $\mathcal{B}$, significance α, VRPMDP instance graph $G(V, E)$, validation set Set^V.
Output: Parameter of model θ.
Initialize model parameter θ and baseline parameter θ_B: $\theta, \theta_B \leftarrow \theta$;
for $epoch = 1 \rightarrow E$ **do**
  Generating batches $\{Batch_1, Batch_2, \ldots, Batch_T\}$;
  for $j = 1 \rightarrow T$ **do**
    Get random instance s_i from $Batch_j$;
    Initialize $\hat{\pi} = \emptyset$, $\eta_0^{max} = 1$, $\eta_0^{min} = 0$;
    for *step* $t = 1$ *until* $\bigcup_t v_t = V$ **do**
      Pre-calcute $\eta_t^{max(k)}$, $\eta_t^{min(k)}$ in CDLR for remaining nodes;
      Update mask m_t;
      Generate v^t via the policy model with θ;
      Update η_t^{max}, η_t^{min} in CDLR;
      if v^t *is the depot* **then**
        Set η_t^{min} as loading rate of this tour;
        Reset $\eta_t^{max} = 1$, $\eta_t^{min} = 0$;
    Calculate $\hat{\pi}|s_i$ with $p_\theta(\hat{\pi}|s_i)$via the policy model with θ;
    Calculate $\hat{\pi}_B|s_i$ and $p_{\theta_B}(\hat{\pi}_B|s_i)$ via the policy model with θ_B;
    $L(\hat{\pi}|s_i) = \mathbb{E}_{p_\theta(\hat{\pi}|\theta)}[Dis(\hat{\pi}|s_i)]$;
    $L(\hat{\pi}_B|s_i) = \mathbb{E}_{p_{\theta_B}(\hat{\pi}|\theta_B)}[Dis(\hat{\pi}_B|s_i)]$;
    $\nabla\mathcal{L} = \sum_{i=1}^{\mathcal{B}}(L(\hat{\pi}|s_i) - L(\hat{\pi}_B|s_i))\nabla_\theta \log p_\theta(\hat{\pi}|s_i)$;
    $\theta \leftarrow Adam(\theta, \nabla\mathcal{L})$;
  if $OneSidedPariedTTest(p_\theta, p_{\theta_B}) < \alpha$ *on* Set^V **then**
    $\theta_B \leftarrow \theta$

and is set as 0 for delivery. The ratio of the number of delivery customers to the number of pick-up customers is approximately 1:1.

— **Real World Dataset.** We use the pickup-delivery requests, which consist of 226 delivery services and 226 pick-up services located in Guangdong Province, China as a real-world dataset. We construct problems with specific scales by randomly selecting subsets from all services. We also control the ratio of the number of delivery customers to the number of pick-up customers at approximately 1:1.

For both datasets, we sample N customers and a depot as an instance randomly and conduct three experiments with $N = 20, 50, 100$. We embed each node's feature into 128-dimensional input. For the GNN-based encoder, we use 3-layer GNN and the dimension of hidden layers in the decoder is set as 128. We train our model with the Adam optimizer with the learning rate of 10^{-3}. All the parameters are initialized by Uniform($1/\sqrt{d}$, $1/\sqrt{d}$)(d is the input dimension) and $L2$ norm of gradients is clipped to 1.0. We process 2500 batches of 512 batch size in one single epoch for training on the randomly-generated dataset, and 800 batches of 64 batch size in one epoch on the real-world dataset. The experiments are conducted using Pytorch 1.7 on 2080Ti GPUs. We trained 100 epochs on both random datasets and real-world datasets.

5.2 Performance Comparison(RQ1)

We first compare our model with the existing two widely recognized heuristics methods:

Table 2. Overall Performance Comparison

Model	Random Dataset						Real World Dataset					
	N = 20		N = 50		N = 100		N = 20		N = 50		N = 100	
	Cost	Time	Cost	Time	Cost	Time	Cost	Time	Cost	Time	Cost	Time
ALNS	6.0350	0.4008s	11.0839	0.5729s	17.2316	0.8201s	240.3	0.3911s	430.9	0.7092s	680.2	1.2039s
ACO	6.5741	0.3227s	11.5936	1.2680s	16.9393	1.6352s	254.6	0.5742s	439.7	0.7752s	666.5	2.4456s
improvement	+14.3%	4.7x	+23.9%	3.2x	+26.1%	1.9x	+6.2%	5.7x	+11.9%	3.1x	+11.0%	2.3x
AM(greedy)	6.6382	0.0001s	10.4170	0.0002s	14.9888	0.0003s	268.3	0.0001s	470.0	0.0002s	731.0	0.0004s
AM(sample 1280)	6.0937	0.0564s	9.6170	0.1531s	13.9791	0.3846s	246.0	0.1551s	413.3	0.2671s	683.3	0.4109s
MDAM(greedy)	6.4183	0.0099s	10.5874	0.0152s	15.0868	0.0508s	251.9	0.0114s	439.2	0.0130s	637.3	0.0202s
MDAM(beam 50)	5.8868	0.1347s	9.5132	0.2586s	13.8979	0.5015s	234.1	0.1082s	399.3	0.2713s	594.7	0.4889s
E-GAT(sample 1280)	6.2710	0.1636s	9.9992	0.7874s	15.9925	1.4357s	256.2	0.1534s	399.6	0.7820s	615.0	1.7249s
improvement	+12.2%	–	+11.4%	–	+10.0%	–	+3.7%	–	+4.9%	–	+0.3%	–
Ours	**5.1666**	0.0678s	**8.4265**	0.1766s	**12.5076**	0.4271s	**225.5**	0.0690s	**379.8**	0.2256s	**593.1**	0.5234s

The best result in each column is bold and the improvement row shows the performance gain of our solution compared to the best heuristics/RL solution.

— **Ant Colony Optimization** [40] is one of the famous heuristics to solve the problem. The ant colonies are constructed to optimize the objective functions.
— **Adaptive Large Neighborhood Search** [25] is commonly applied to routing and scheduling problems. The algorithm expands the search space of the solution by designing multiple sets of destruction operators and repair operators. And in each iteration, each destruction and repair operator is selected and weighted according to the past performance, so as to find the optimal solution.

We also compare our method with the existing two RL-based methods for solving combinatorial optimization:

— **AM-VRP** [14] proposes an encoder-decoder framework based on the attention mechanism. The model is optimized by REINFORCE algorithm, which utilizes the rollout as its baseline. We migrate AM-VRP to the VRPMDP, and set the vehicle loading rate as 0.7.
— **MDAM-VRP** [46] further improves the diversity of policy based on the AM model by utilizing multi-decoder to train multiple construction policies. A novel beam search scheme and Embedding Glimpse layer are also applied to the model to improve the quality of the solution. We migrate MDAM-VRP to the VRPMDP, and set the vehicle loading rate as 0.7.
— **E-GAT-VRP** [20] proposes a E-GAT Network to capture edge information in the graph structure and consider the residual connections for effective embedding, and utilized PPO and REINFORCE algorithm to optimize the network, respectively. We migrate E-GAT-VRP with REINFORCE algorithm to the VRPMDP, and set the vehicle loading rate as 0.7.

We calculate the average solution distance and average time cost for all testing instances in two datasets. And the performance of all methods is shown in Table 2. The table is divided into three sections: the heuristic methods, the RL-based models and ours. And the best results in each experiment are in bold. We also calculate the improvement of our method compared with the best heuristics and RL methods.

In terms of average solution distance, our proposed approach outperforms all baselines in all experiments. Compared to other RL baselines, our method reduces the average distance by 12.2% and 3.7%, 11.4% and 4.9%, 10.0% and 0.3% for 20, 50, 100 customers on the random datasets and Real World datasets, respectively. Compared to heuristics baselines, Our method reduces the average distance by 14.3% and 6.2%, 23.9% and 11.9%, 26.1% and 11.0% for 20, 50, 100 customers on the random and Real World datasets, respectively.

In terms of time cost, our method also has desirable results. Compared to heuristics baselines, the inference time improvement is remarkable. The solution can be calculated in one second by

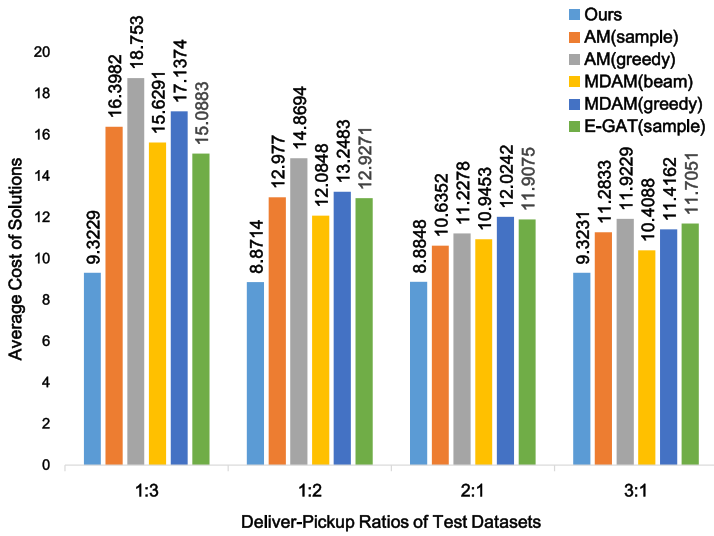


Fig. 6. Comparison of RL methods on test instances with different ratio of the amount of delivery customers and pickup customers α, $\alpha = 1:2, 1:3, 2:1, 3:1$.

our method while costing much more time by the heuristics, and the inference speed of ours can be up to 5.7 times better than that of heuristics. Compared to RL baselines, there is still a gap in time consumption between our method and AM. But compared to other RL baselines, our method is faster.

Overall, our method achieves desirable results in terms of both average solution distance and inference time.

5.3 Generalization Ability(RQ2)

The main comparison results shown above are conducted with pickup and delivery customer ratio as about 1:1. However, in more applicable real-world scenarios, the ratio of delivery customers and pick-up customers varies. For general usage, a robust model should be adaptive to different data distributions with different ratios. Thus we test the above models on instances of the same scale ($N = 50$) but with different customer ratios $\alpha = 1:2, 1:3, 2:1, 3:1$, respectively.

The results are shown in Figure 6, which compares the average routing length of different solutions. When applying the AM model ($\alpha = 1:1$) and MDAM model ($\alpha = 1:1$) to test instances with $\alpha \neq 1:1$, the performances of models deteriorate in different degrees. And the performances of the baseline models on instances with $\alpha = 2:1, 3:1$ are better than on instances with $\alpha = 1:2, 1:3$. This is because setting the loading rate to 0.7 is appropriate for fewer pickup nodes. This also proves that the fixed loading rate cannot deal with the environment dynamics. We find that our model has stable generalization ability and the best performance on different α. We attribute the superior generalization ability to CDLR instead of fixing the loading rate.

We also conduct experiments on testing datasets with different scales with the training datasets and the results are shown in Figure 7. We find that the more difference between the scale of the training instance and the scale of the test instance, the worse the result. For example, each model trained by VRPMDP-20 instances achieves worse result on VRPMDP-100 testing instances than one trained by VRPMDP-50 or VRPMDP-100 instances. Then we find our method has a desirable generalization ability compared to other baselines. The performance on VRPMDP-100 by our

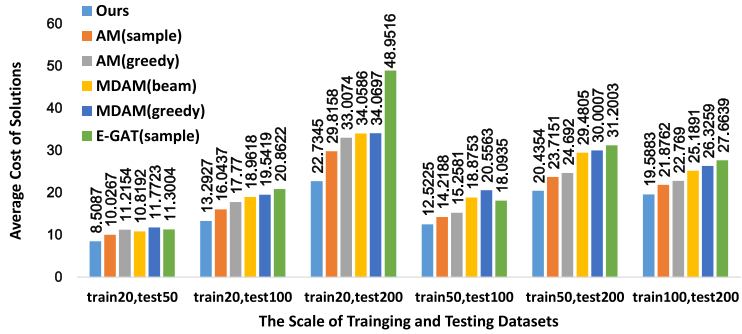
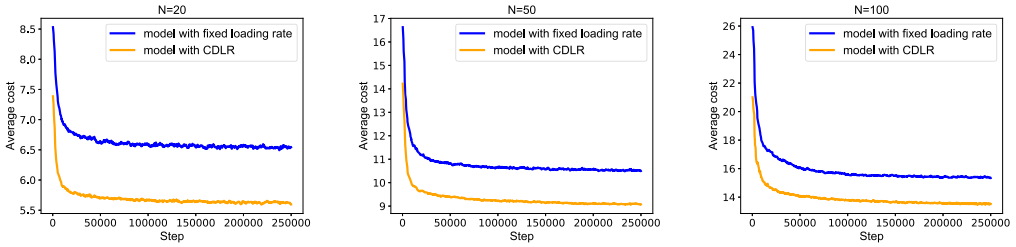


Fig. 7. Comparison of RL methods on test instances with different scales from the training instances. For instance, train 20, test 50 means the model is trained on VRPMDP20 instances and applied on VRPMDP50 instances.



(a) Learning curve on VRPMDP-20 with two different loading rate settings

(b) Learning curve on VRPMDP-50 with two different loading rate settings

(c) Learning curve on VRPMDP-100 with two different loading rate settings

Fig. 8. Learning curve, respectively, on VRPMDP-20, VRPMDP-50 and VRPMDP-100 with different loading rate setting, which the blue lines represent the model with fixing the loading rate and the orange lines represent the model with CDLR.

models trained on VRPMDP-20 and VRPMDP-50 instances are 6.2% and 0.1% worse than one trained on VRPMDP-100, while the loss is acceptable for the generalization task. We attribute this to the information aggregation performance of GNN and the superior generalization ability to CDLR instead of fixing the loading rate and then making routing decisions.

5.4 Ablation Study (RQ3, RQ4)

To evaluate the effectiveness of the CDLR, we conduct the corresponding ablation study in this section. We assess the contribution of CDLR by omitting it and fixing the loading rate as 0.7 in the entire routing decision process. The results on the random datasets are shown in Table 3. The performance of the model with CDLR is all better than that with fixed loading rate with up to 13.22%. While improving the quality of solutions, the model with CDLR consumes only a little more time than the model with fixed loading rate. We also draw the learning curve of the above two methods in Figure 8. The overall training and final convergence quality of the model with CDLR are better than the model with fixed loading rate. The results all prove the effectiveness of the CDLR mechanism.

We also conduct experiments to study the effectiveness of GNN hyperparameter. The number of GNN layers in our encoder is changed from 2 to 4 and the results of different settings are shown in Table 4. The results show that our setting outperforms other settings slightly in VRPMDP-50 and VRPMDP-100 instances while is worse than other settings on VRPMDP-20 instances. We can

Table 3. Comparison on Model Fixed Loading Rate and Coordinated Decision-making

	N = 20		N = 50		N = 100	
	Cost	Time	Cost	Time	Cost	Time
fixed loading rate	5.9542	0.0587s	9.5688	0.1555s	14.1425	0.3856s
coordinated decision-making	5.1666	0.0678s	8.4265	0.1766s	12.5076	0.4271s

Table 4. Comparison on GNN with Different Hyper-parameters

	N = 20		N = 50		N = 100	
	Cost	Time	Cost	Time	Cost	Time
ours	5.1666	0.0678s	8.4265	0.1766s	12.5076	0.4271s
GNN(2-layer)	5.1515	0.0657s	8.4723	0.1785s	12.5373	0.4300s
GNN(4-layer)	5.1895	0.0664s	8.4478	0.1802s	12.5265	0.4307s

Table 5. Comparison on Different Method of Instance Graph Construction

	N = 20		N = 50		N = 100	
	Cost	Time	Cost	Time	Cost	Time
ours	5.1666	0.0678s	8.4265	0.1766s	12.5076	0.4271s
k = 5 nearest	5.1724	0.0678s	8.6062	0.1808s	12.8657	0.4341s
k = 10 nearest	5.2014	0.0695s	8.5977	0.1775s	12.8334	0.4292s
k = 15 nearest	5.2625	0.0686s	8.6401	0.1793s	12.7933	0.4295s

find that the results are desirable even though the number of GNN layers is only 2. The results show that our model with 3-layer GNN encoder can tradeoff performance and computation time.

At last, we consider the effect of different graph constructing methods on performance. There are many methods to convert a VRPMDP instance to a graph. Consider that all customers are connected, we construct the instance graph as a fully-connected graph [15] which can fully capture the connectivity of the customers. While in other literature [9], the constructing method is different. In the process of constructing the graph, they only consider the edges that connect each node to its k nearest nodes, and the whole instance graph is constructed iteratively in this way. This method may reduce the training time due to discarding some information. Thus we conduct experiments to compare our method to these methods. The results are shown in Table 5. We find that the performance of k-nearest generation methods are all worse than ours while the inference time is indeed less than ours. We suppose that the selective determination of the connectivity between customers discards certain information, resulting in slightly poorer performance.

6 CONCLUSIONS

In this article, we consider the practical express system with deliveries and pickups as VRPMDP, and utilize DRL to solve the problem. We adopt GNN as the encoder structure to extract the instance features, considering both independence and inter-influence between pickup and delivery. The detailed routing solutions are further decoded as a sequence by the decoder with attention mechanism. We develop a CDLR mechanism to determine the loading rate with minimal impact on routing decision, which guarantees robustness to the changing proportion of deliveries to pickups. Extensive experiments are conducted to demonstrate the effectiveness of our model. The experiments show that our method achieves desirable results and has desirable generalization ability.

REFERENCES

[1] Andrzej and Adamski. 2011. Hierarchical integrated intelligent logistics system platform. *Procedia Social & Behavioral Sciences* 20 (2011), 1004–1016.

[2] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio. 2016. Neural combinatorial optimization with reinforcement learning. arXiv preprint arXiv:1611.09940.

[3] X. Bresson and T. Laurent. 2017. Residual gated graph ConvNets. arXiv preprint arXiv:1711.07553.

[4] D. O. Casco, B. L. Golden, and E. A. Wasil. 2018. Vehicle routing with backhauls: Models, algorithms and case studies. *Computers & Operations Research* 91 (2018), 79–81.

[5] Xinyun Chen and Yuandong Tian. 2019. Learning to perform local rewriting for combinatorial optimization. *Advances in Neural Information Processing Systems* 32 (2019).

[6] Kyunghyun Cho. 2016. Noisy parallel approximate decoding for conditional recurrent language model. arXiv:1605.03835.

[7] José Crispim and José Brandão. 2005. Metaheuristics applied to mixed and simultaneous extensions of vehicle routing problems with backhauls. *Journal of the Operational Research Society* 56, 11 (2005), 1296–1302.

[8] Michel Deudon, Pierre Cournut, Alexandre Lacoste, Yossiri Adulyasak, and Louis-Martin Rousseau. 2018. Learning heuristics for the TSP by policy gradient. In *Proceedings of the International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer, 170–181.

[9] L. Duan, Y. Zhan, H. Hu, Y. Gong, and Y. Xu. 2020. Efficiently solving the practical vehicle routing problem: A novel joint learning approach. In *Proceedings of the KDD'20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

[10] F. Glover, M. Laguna, and R. Martí. 1997. Tabu search. *General Information* 106, 2 (1997), 221–225.

[11] Google. 2019. OR-Tools. Retrieved from https://developers.google.com/optimization/.

[12] Keld Helsgaun. 2017. An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. (2017).

[13] H. Hernández-Pérez and J. J. Salazar-González. 2004. A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. *Discrete Applied Mathematics* 145, 1 (2004), 126–139.

[14] C. K. Joshi, T. Laurent, and X. Bresson. 2019. An efficient graph convolutional network technique for the travelling salesman problem. arXiv preprint arXiv:1906.01227.

[15] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. 2017. Learning combinatorial optimization algorithms over graphs. *Advances in Neural Information Processing Systems* 30 (2017), 6348–6358.

[16] Vijay R. Konda and John N. Tsitsiklis. 2000. Actor-critic algorithms. In *Proceedings of the Advances in Neural Information Processing Systems*. 1008–1014.

[17] Wouter Kool, Herke van Hoof, and Max Welling. 2018. Attention, learn to solve routing problems!. In *Proceedings of the International Conference on Learning Representations*.

[18] Gilbert Laporte and Ibrahim H Osman. 1995. Routing problems: A bibliography. *Annals of Operations Research* 61, 1 (1995), 227–262.

[19] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[20] Kun Lei, Peng Guo, Yi Wang, Xiao Wu, and Wenchao Zhao. 2021. Solve routing problems with a residual edge-graph attention neural network. arXiv:2105.02730.

[21] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015).

[22] Vdb Ljj, Jjk Lenstra, and P. P. Schuur. 1990. A variable depth approach for the single-vehicle pickup and delivery problem with time windows. *American Journal of Clinical Nutrition* 86, 1 (1990), 64–73.

[23] Hao Lu, Xingwen Zhang, and Shuang Yang. 2019. A learning-based iterative method for solving vehicle routing problems. In *Proceedings of the International Conference on Learning Representations*.

[24] Qiang Ma, Suwen Ge, Danyang He, Darshan Thaker, and Iddo Drori. 2019. Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning. arXiv:1911.04936.

[25] S. Majidi, S. M. Hosseini-Motlagh, and J. Ignatius. 2017. Adaptive large neighborhood search heuristic for pollution-routing problem with simultaneous pickup and delivery. *Soft Computing* 22, 9 (2017), 2851–2865.

[26] Nina Mazyavkina, Sergey Sviridov, Sergei Ivanov, and Evgeny Burnaev. 2021. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research* 134 (2021), 105400.

[27] Duncan McFarlane, Vaggelis Giannikas, and Wenrong Lu. 2016. Intelligent logistics: Involving the customer. *Computers in Industry* 81 (2016), 105–115.

[28] V. Mnih. 2015. Artificial intelligence human-level control through deep reinforcement learning. *NATURE -LONDON-* 518, 7540 (2015), 529–533.

[29] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013).
[30] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.
[31] Fermín Alfredo Tang Montané and Roberto Diéguez Galvao. 2006. A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Computers & Operations Research* 33, 3 (2006), 595–619.
[32] Seyed Sajad Mousavi, Michael Schukat, and Enda Howley. 2016. Deep reinforcement learning: An overview. In *Proceedings of the SAI Intelligent Systems Conference*. Springer, 426–440.
[33] MohammadReza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takac. 2018. Reinforcement learning for solving the vehicle routing problem. *Advances in Neural Information Processing Systems* 31 (2018), 9839–9849.
[34] H. N. Psaraftis. 1983. k-interchange procedures for local search in a precedence-constrained routing problem. *European Journal of Operational Research* 13, 4 (1983), 391–402.
[35] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE Transactions on Neural Networks* 20, 1 (2008), 61–80.
[36] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*.
[37] Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the Advances in Neural Information Processing Systems*. 1057–1063.
[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Advances in Neural Information Processing Systems*. 5998–6008.
[39] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. *Advances in Neural Information Processing Systems* 28 (2015), 2692–2700.
[40] A. Wade and S. Salhi. 2003. An ant system algorithm for the mixed vehicle routing problem with backhauls. *Metaheuristics: Computer Decision-Making* 86 (2003), 699–719.
[41] Hsiao-Fan Wang and Ying-Yen Chen. 2012. A genetic algorithm for the simultaneous delivery and pickup problems with time window. *Computers & Industrial Engineering* 62, 1 (2012), 84–95.
[42] Jianxin Wang, Ming K. Lim, Yuanzhu Zhan, and XiaoFeng Wang. 2020. An intelligent logistics service system for enhancing dispatching operations in an IoT environment. *Transportation Research Part E: Logistics and Transportation Review* 135 (2020), 101886.
[43] N. A. Wassan and G. Nagy. 2013. The vehicle routing problem with deliveries and pickups: Modelling issues and solution approaches. (2013).
[44] Z. Wei and T. G. Dietterich. 1995. A reinforcement learning approach to job-shop scheduling. *Morgan Kaufmann Publishers Inc.* 95 (1995), 1114–1120.
[45] R. J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8, 3–4 (1992), 229–256.
[46] L. Xin, W. Song, Z. Cao, and J. Zhang. 2021. Multi-decoder attention model with embedding glimpse for solving vehicle routing problems. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*.
[47] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. 2018. How powerful are graph neural networks? In *International Conference on Learning Representations*.
[48] Mengke Yang, Movahedipour Mahmood, Xiaoguang Zhou, Salam Shafaq, and Latif Zahid. 2017. Design and implementation of cloud platform for intelligent logistics in the trend of intellectualization. *China Communications* 14, 10 (2017), 180–191. DOI: https://doi.org/10.1109/CC.2017.8107642

Received 2 January 2022; revised 3 May 2022; accepted 20 June 2022