Heterogeneous Attentions for Solving Pickup and Delivery Problem via Deep Reinforcement Learning

Jingwen Li[1], Liang Xin[2], Zhiguang Cao[1*], Andrew Lim[1], Wen Song[3], Jie Zhang[2]

***Abstract*—Recently, there is an emerging trend to apply deep reinforcement learning to solve the vehicle routing problem (VRP), where a learnt policy governs the selection of next node for visiting. However, existing methods could not handle well the pairing and precedence relationships in the pickup and delivery problem (PDP), which is a representative variant of VRP. To address this challenging issue, we leverage a novel neural network integrated with a heterogeneous attention mechanism to empower the policy in deep reinforcement learning to automatically select the nodes. In particular, the heterogeneous attention mechanism specifically prescribes attentions for each role of the nodes while taking into account the precedence constraint, i.e., the pickup node must precede the pairing delivery node. Further integrated with a masking scheme, the learnt policy is expected to find higher-quality solutions for solving PDP. Extensive experimental results show that our method outperforms the state-of-the-art heuristic and deep learning model, respectively, and generalizes well to different distributions and problem sizes.**

***Index Terms*—Heterogeneous attention, deep reinforcement learning, pickup and delivery problem**

I. Introduction

VEHICLE routing problem (VRP) is a fundamental topic in both communities of ITS (Intelligent Transportation Systems) and Operations Research, which has ubiquitous applications in industry, such as the logistics for harbor port [1], airport [2] and the general warehouse [3]. Instead of sharing a depot for all customers as VRP, in reality, a customer may always have his own delivery point, such as in intra-city express service [4] and ride-sharing [5]. The route planning for all applications of this type could be naturally described as a *pickup and delivery problem* (PDP), which is a representative variant of VRP. Generally, the PDP is characterized by pairing and precedence relationships, in which a pickup point must precede the pairing delivery point. Although widely studied, it is still challenging for conventional methods including exact and heuristic algorithms to optimally solve PDP with short computation time due to its NP-hard nature. Recently, there is an increasing attention on applying deep reinforcement learning (DRL) to automatically learn the rules in conventional heuristic methods for solving routing problems including Travelling Salesman Problem (TSP) and Capacitated VRP (CVRP), which delivers appealing results with much faster computation [6]–[9]. Inspired by them, in this paper, we aim to solve PDP by exploring deep reinforcement learning.

Although some success has been achieved, most of the DRL based solutions are only able to handle the typical VRP with a shared delivery point, which is less effective for coping with the pairing and precedence relationships in PDP. Intuitively, the masking scheme in existing DRL models could be adapted to represent those relationships in PDP, where the delivery points should be always masked until the pairing pickup points are visited. However, this adaption might be far from enough given the following two issues: 1) The masking scheme only takes effects in the output layer of the policy network, and a more desirable solution should enable the main body of the deep architecture intrinsically aware of the pairing and precedence relationships; 2) Different from the typical VRP, the nodes in PDP have more heterogeneous roles, such as ego pickup point, ego delivery point, peer pickup point, peer delivery point and depot. The complex relationship among them may render the decision making for selecting the next node more arduous.

To cope with those challenging issues, in this paper, we propose a deep reinforcement learning based method that integrates with a heterogeneous attention mechanism for solving PDP. In particular, the policy network in the DRL is characterized by an encoder-decoder structure, and it learns constructing a solution by iteratively selecting a pickup or delivery point at each step. Regarding the heterogeneous attention mechanism, six types of attentions are specifically designed in addition to the original one. Among them, three types of attentions are used to learn the relationship between each ego pickup point to the points of other roles, and the remaining three types are used for each ego delivery point. Together with a masking scheme that adaptively masks out the invalid points to guarantee feasibility, the heterogeneous attention mechanism is supposed to empower the policy network to more intrinsically perceive and learn the pairing and precedence relationships. Extensive experimental results show that the proposed method outperforms both the state-of-the-art deep reinforcement learning method and conventional heuristic methods with much higher solution quality and shorter computation time. More importantly, the learned policy generalizes well to problems

* Corresponding Author.

[1]Jingwen Li, Zhiguang Cao and Andrew Lim are with the Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore (emails: lijingwen@u.nus.edu, zhiguangcao@outlook.com, isealim@nus.edu.sg).

[2]Liang Xin and Jie Zhang are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore (email: xinl0003@e.ntu.edu.sg, zhangj@ntu.edu.sg).

[3]Wen Song is with the Institute of Marine Science and Technology, Shandong University, China (email: wensong@email.sdu.edu.cn).

This work is supported by the National Research Foundation of Singapore under grant NRF-RSS2016-004, the National Natural Science Foundation of China (Grant No. 61803104), the Young Scholar Future Plan of Shandong University (Grant No. 62420089964188), and the Singtel-NTU Cognitive & Artificial Intelligence Joint Lab through the NRF corporate lab@university scheme.

with different distributions and sizes.

The remainder of the paper is organized as follows. Section II briefly reviews conventional methods and deep models for routing problems. Section III introduces the PDP with mathematical formulation. Section IV reformulates the problem in RL manner and elaborates our DRL solution. Section V provides the computational experiments and analysis. Finally, Section VI concludes the paper and presents future works.

II. Related Work

In this section, we review the related works on conventional methods for PDP, and (deep) learning models for routing problems.

A. Exact and Heuristic Methods

Most of exact methods for solving PDP adopt branch-and-bound or its variants as framework. The precedence relations was first considered and added into classical TSP in [10], where an exact brand-and-bound method was adopted to tackle the problem. The general PDP was first surveyed in [11], summarizing different variants of PDP with formulations and several exact methods such as brand-and-bound with additive bounding procedure and column generation scheme to handle these variants. One of the conclusions is that dynamic programming performed well on Single-vehicle Pickup and Delivery Problem (PDP) in small scale instances. A branch-and-cut algorithm was presented in [12] to solve PDP based on three constraints: subtour elimination and precedence constraints, generalized order constraints and order matching constraints. A new branch-and-cut-and-price algorithm was proposed to solve PDP in [13] by considering two subproblems of pricing in column generation. However, despite delivering optimal solutions, the exact methods suffer from prohibitively heavy computation for large-scale ones given the exponential complexity.

In contrast, heuristic methods for PDP and VRP are able to tackle large-scale problems with relatively high computation efficiency [14]–[17]. Based on the shift, exchange and rearrange operators, a tabu-embedded simulated annealing algorithm was proposed to cope with large-scale PDP in [18]. An adaptive large neighborhood search heuristic method was presented in [19] to handle PDP by incorporating regret insertion method and six removal strategies such as shaw removal, cluster removal, etc. Another adaptive large neighborhood search method was proposed in [20] based on destroy and recreat strategies. Starting from an initial solution generated by a greedy insertion heuristic, various efficient removal operators and insert principles were applied to improve the solution iteratively. A hybrid three-stage heuristic approach was presented to solve PDTSP in [21]. With initial solutions generated using several local search operators, multi-start and variable neighborhood decent heuristic methods were integrated to improve the solution, where three shaking procedures were introduced to perturb solutions from local minimum. However, despite the desirable performance, the hand-engineered rules in the conventional heuristic methods heavily depend on human expertise and experience, which leave much room to improve the solution quality further.

B. Learning Based Methods

Recently, there is a growing trend to apply deep reinforcement learning to solve VRPs, and most of them are characterized by a policy network with an encoder-decoder structure. Generally, the encoder maps the 2-dimensional locations of customers (or nodes) into feature embedding to extract useful information from data, and the decoder addresses the problem in two different fashions, i.e., *construction* or *improvement*, respectively. Regarding the former, the decoder starts with an empty sequence, and iteratively selects a node at each step to construct a complete solution. Regarding the latter, the decoder starts with a complete initial solution, and constantly selects either candidate nodes or heuristic operators for certain operation at each step, so that the new solution could be improved over previous ones until the termination criterion is met. To force that each customer is only visited once, a masking scheme is always applied to mask the visited or invalid nodes. Further integrated with attention mechanisms or graph neural networks, those DRL models are potentially empowered to produce higher-quality solutions.

In specific, the first seminal deep architecture proposed to cope with routing problems was Pointer Network (PtrNet), which solved TSP in a supervised manner [22] and was later extended to the framework of reinforcement learning [6]. Furthermore, PtrNet was adopted to solve CVRP based on the reinforcement learning, where the customer information (i.e., location and demands) was dynamic and the route length was uncertain [7]. To speed up training process, the Recurrence Neural Network (RNN) structure in the encoder was removed since sequential and positional information was not meaningful for CVRP. A Transformer-based architecture was proposed in [8], [23] by adopting self-attention instead of Seq2Seq structure in both encoder and decoder to engender higher-quality solutions. A hybrid of local search and deep reinforcement learning was introduced in [24] to solve both VRP and VRPTW (time window). The reinforcement learning was adopted to solve the electric vehicle fleet dispatch problem in [25], in which a novel framework was proposed using the decentralized learning and centralized decision making. With the online routing problem transformed into a vehicle tour generation problem, a deep reinforcement learning based method was proposed using a structural graph embedded pointer network to construct those tours [26]. Rather than learning construction heuristics as the methods mentioned above, the NeuRewriter in [27] and the improvement heuristic method in [28] were proposed to learn refining an initial yet complete solution iteratively with local search operators.

III. Preliminary

In this section, we introduce the preliminary for PDP and describe its mathematical formulation.

With n customer requests represented as pickup node (point) set $P = \{x_i\}_{i=1}^{n}$ and delivery node (point) set $D = \{x_i\}_{i=n+1}^{2n}$, the pickup node x_i and the delivery node x_{i+n} are bound as a pair with precedence relationship. With node 0 corresponding to depot, the complete node set is defined as $X = \{x_0\} \cup P \cup D$. Let $X' = X \cup \{x_{2n+1}\} = \{x_i\}_{i=0}^{2n+1}$,

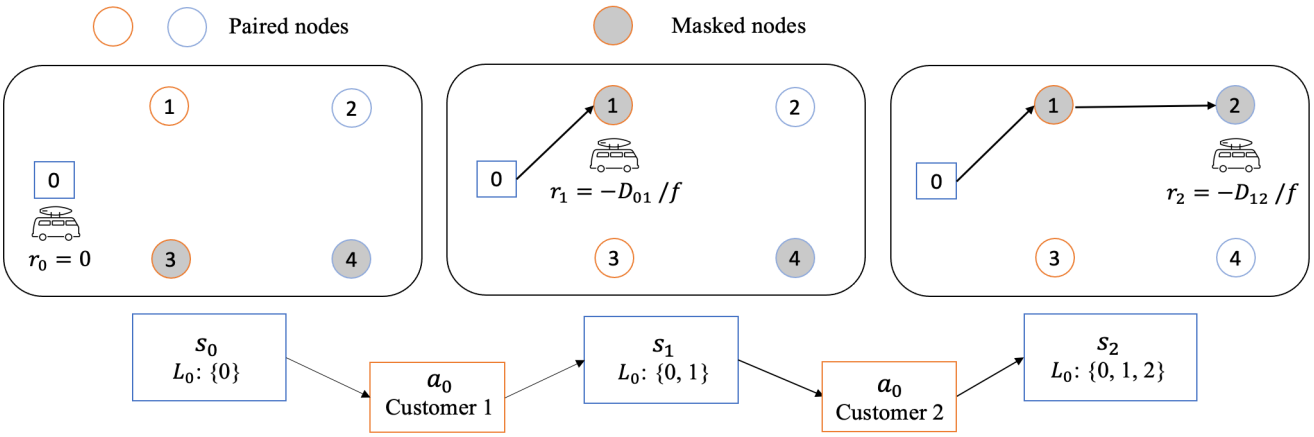


Fig. 1: An illustrative example of MDP with 1 vehicle and 4 nodes. Node x_1 and x_2 are pickup nodes, and node x_3 and x_4 are their delivery nodes, where node x_1 and x_3, node x_2 and x_4 are paired. At the beginning, node x_3 and node x_4 are masked due to the precedence constraint. At the first step, node x_1 is selected, i.e., $a_1 = x_1$. In response to this action, a) node x_1 is masked and node x_3 becomes unmasked; b) the partial route is updated, i.e., $L_1 = \{0, 1\}$; c) the accumulative reward for this step is the negative value of the travel time of the vehicle travelling from the depot to node x_1, i.e., $r_1 = -D_{01}/f$. At the second step, node x_2 is selected, and the state, action, reward are updated similarly.

where node x_{2n+1} is the copy of depot. Each node $x_i \in R^2$ is defined as $\{c_i\}$, and c_i contains 2-dimensional coordinates of location of x_i. We assume that all items picked from x_i will be totally delivered to x_{i+n} by a vehicle v with infinity capacity (i.e., for the sake simplification). With the above settings, the PDP describes a process that starting from the depot, a vehicle sequentially visits all pickup nodes and delivery nodes exactly once to perform the service, and finally returns to the depot, with the objective of minimizing the total travel time. Note that here PDP allows consecutive pickups, or deliveries, or mix of them as long as it satisfies the precedence constraint.

Let D_{ij} be the Euclidean distance between node x_i and node x_j, f be the speed of vehicle v, $y_{ij} \in \{0, 1\}$ be a binary variable to indicate whether the vehicle v travels directly from node x_i to node x_j, B_i be the arrival time of node x_i, and M be a large-enough number. Accordingly, the objective function of PDP is formulated as follows,

$$min \sum_{i \in X} \sum_{j \in X} \frac{D_{ij}}{f} y_{ij}. \tag{1}$$

Meanwhile, the PDP satisfies the following constraints,

$$\sum_{j \in X} y_{ij} = 1, \qquad i \in X', \tag{2}$$

$$\sum_{i \in X} y_{ij} = 1, \qquad j \in X', \tag{3}$$

$$B_j \geq B_i + \frac{D_{ij}}{f} - M(1 - y_{ij}), \qquad i \in X', j \in X', \tag{4}$$

$$B_{i+n} \geq B_i + \frac{D_{i,i+n}}{f}, \qquad i \in P, i + n \in D, \tag{5}$$

$$y_{ij} = \{0, 1\}, \qquad i \in X', j \in X', \tag{6}$$

$$B_i \geq 0, \qquad i \in X'. \tag{7}$$

The objective is to minimize the total travel time for the vehicle v. Constraint (2) and (3) ensure that each node including the depot (x_0) and its copy (x_{2n+1}) is visited exactly once. Constraint (4) indicates the arrival time calculation for two adjacent nodes in the route. Constraint (5) refers to the precedence constraint, and guarantees that the arrival time of a pickup node is earlier than that of its delivery node. Constraint (6) defines the binary decision variable and constraint (7) imposes the non-negativity of the arrival time. Note that we focus on a single vehicle in this paper, based on which it could be easily extended to multiple vehicles.

IV. Methodology

In this section, we first reformulate the PDP as the form of reinforcement learning (RL), then a policy network based on the encoder-decoder structure is designed to learn node selection for solution construction, which is empowered by a heterogeneous attention mechanism.

A. Reformulation as RL Form

For solving PDP, the route construction process could be essentially deemed as a sequence of decision making, which can be naturally formulated as the form of RL and solved accordingly. We model such route construction process as a Markov Decision Process (MDP), where an example of the MDP is presented in Fig. 1. In specific, the elements of the MDP, i.e., the state space, the action space, the transition rule and the reward function are defined as follows.

State. The state $s_t = (L_t)$ represents the partial solution constructed at time step t, where L_t contains all visited nodes until step t, and L_0 refers to the depot.

Action. During the route construction process, the action a_t is represented as (x_j), i.e., selecting node x_j at step t.

Transition. The next state $s_{t+1} = (L_{t+1}) = (L_t; \{x_j\})$ is originated from s_t by selecting a node at step t, where ";" means concatenating the partial solution at last step with the newly selected node.

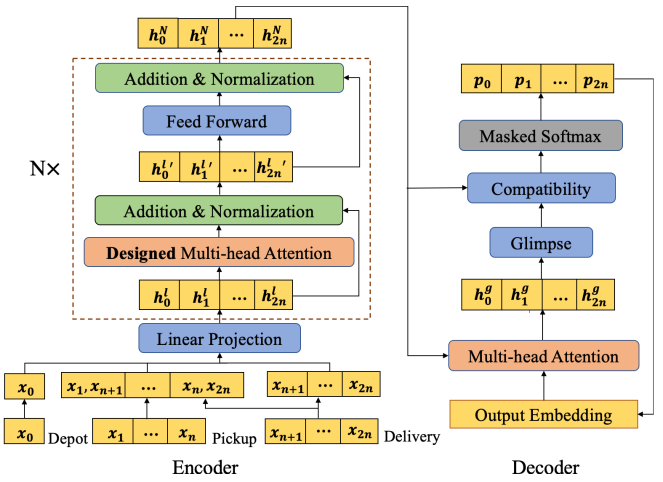


Fig. 2: Policy Network (left: encoder; right: decoder).

Reward. To minimize the total travel time of routes, we define the reward as the negative of the objective value, which is calculated by accumulating the negative value of travel time of all steps. Consequently, the reward is represented as $R = \sum_{t=1}^{T} \boldsymbol{r}_t$, where $\boldsymbol{r}_t$ is the negative value of the incremental travel time at step t. Suppose that node x_i and x_j are selected at step t and $t+1$, respectively, then $\boldsymbol{r}_{t+1}$ is expressed as a m-vector as follows,

$$\boldsymbol{r}_{t+1} = r(s_{t+1}, a_{t+1}) = r((L_{t+1}), (x_j)) = -\frac{D_{ij}}{f}, \quad (8)$$

where D_{ij}/f is the time for traveling from node x_i to x_j.

Policy. The stochastic policy p_θ automatically selects a node at each time step under the precedence constraint. This process is repeated iteratively until all pickup-delivery services are completed. The final outcome engendered by performing the policy is a permutation of all nodes, which prescribes the order of each node for the vehicle to visit, i.e., $\pi = \{\pi_0, \pi_1, ..., \pi_T\}$. Based on the chain rule, the probability of an output solution is factorized as follows,

$$P(\pi|X) = \prod_{t=0}^{T-1} p_\theta(\pi_t|X, \pi_{1:t-1}), \quad (9)$$

where X is the input of an problem instance. And the decision making about the node selection will be performed based on the learnt p_θ.

B. Policy Network based on Heterogeneous Attentions

The action space of a routing problem is discrete and grows exponentially as problem scales up, where the policy-based reinforcement method consisting an actor network and a critic network is often adopted to solve such problem. At each time step, the actor network generates a probability vector over all actions given the current state and then selects an action accordingly, which is iteratively repeated until the terminal condition. The reward of the actor network is calculated by summing up the accumulative reward at each step for the whole process. The critic network, as a baseline of the actor network, calculates the baseline reward only depending on the initial state to reduce variance. After receiving the reward of the actor network and the baseline reward of the critic network, the policy gradient method is adopted to update the parameters of two networks accordingly, where the actor network is trained towards finding solutions with higher quality.

To learn the policy p_θ, we design a policy network based on an encoder-decoder structure as depicted in Fig. 2, which is integrated with a heterogeneous attention scheme. Given the properties of PDP, the heterogeneous attentions are supposed to learn the relationship among the nodes of different roles, so that the precedence constraint could be intrinsically captured. Consequently, the parameters of this deep architecture also refer to the ones in p_θ.

1) Encoder: Regarding the encoder, we adopt a structure similar to the one in [8], with which we integrate a designed heterogeneous attention mechanism. Before elaborating the attentions, we first enhance the representations of inputs to the embedding by taking into account the pairing relation between the pickup node and delivery node. To this end, we concatenate the pickup nodes with the corresponding delivery nodes to strengthen the pairing relation, i.e., $x_i = (x_i; x_{i+n}), \forall\ i \in P$, where ";" refers to concatenation of two vectors. The enhanced inputs are then linearly projected to initial node embeddings h^0 with dimension $d_h = 128$. Afterwards the node embeddings go through N attention layers, each of which encompasses a multi-head attention sublayer and a feed-forward (FF) sublayer.

Regarding the multi-head attention sublayer, we design heterogeneous attentions to better capture the precedence relation, and the details of the architecture are presented in Fig. 3, where we leave out the depot for simplification. Let $h_i^{l-1}, i \in X$ denote the node embedding of attention layer $l-1$ ($l \in \{1, ..., N\}$), where i is the node index. Let d_k, d_v be the *query/key* dimension and d_v be the *value* dimension, where $d_k = d_v = \frac{d_h}{M}$ and $M = 8$ is the number of heads.

Given the input sequence X, the self-attention mechanism learns the relations between arbitrary two elements of the sequence to compute a representation of this sequence for better feature extraction [29]. To capture the relations, three vectors, i.e., *query, key* and *value* are created based on the input X, which are expressed as follows,

$$Q_i = W^Q h_i^{l-1},\ K_i = W^K h_i^{l-1},\ V_i = W^V h_i^{l-1},\ i \in X, \quad (10)$$

where $W^Q, W^K \in R^{d_h \times d_k}$ and $W^V \in R^{d_h \times d_v}$ are trainable parameter matrices, and Q, K and V refer to *query*, *key* and *value*, respectively. The scaled dot product is selected as a compatibility function of the *query* and the *key* to measure the importance between arbitrary two nodes, then the compatibility is further processed by a softmax function to compute the weights of all nodes. Following this, the original self-attention for solving the typical VRP in [8] is calculated as follows,

$$a_{ij} = softmax(\frac{Q_i^T K_j}{\sqrt{d_k}}),\ i, j \in X, \quad (11)$$

where higher attention value a_{ij} means that the node x_i depends more on the node x_j.

However, different from the typical VRP, the nodes in PDP may have heterogeneous roles, and thus impose more complex

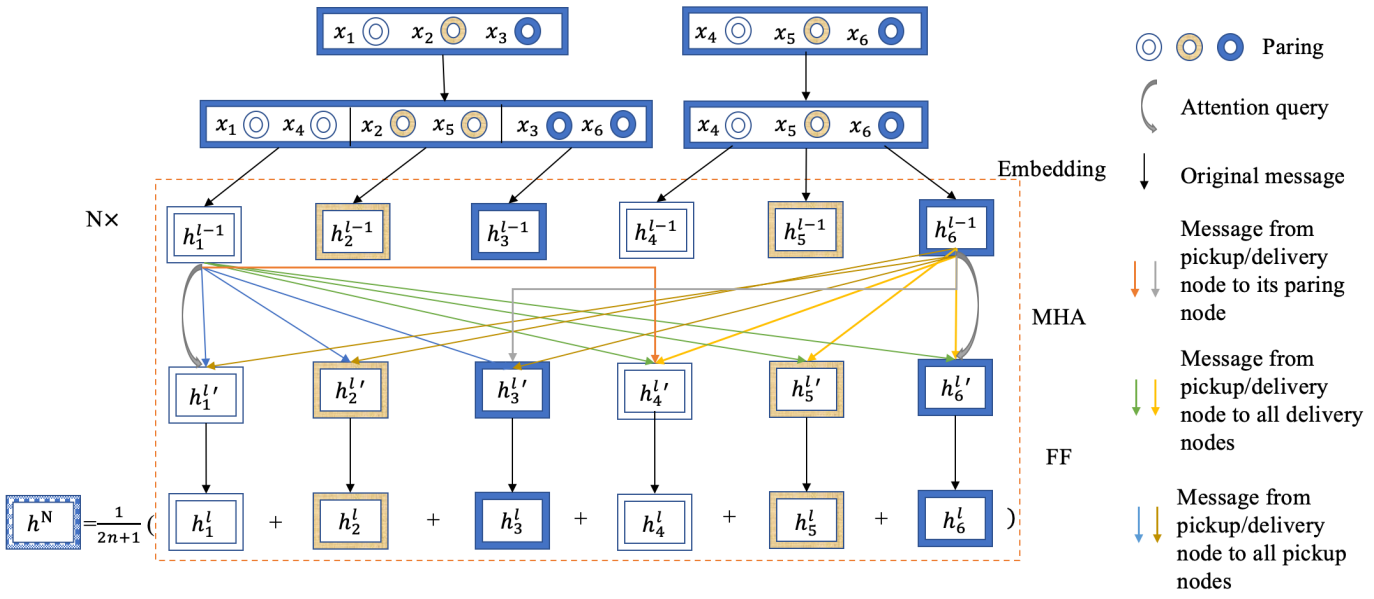


Fig. 3: Illustration of the encoder. Taking pickup node x_1 as an example, it is concatenated with its delivery node x_4 to strengthen the pairing relation. After embedded into 128 dimensions, three types of attentions are calculated between node x_1 and other nodes of heterogeneous roles, i.e., the paired delivery node x_4, all pickup nodes (x_1, x_2, x_3) and all delivery nodes (x_4, x_5, x_6), respectively. The new attentions are supposed to capture the precedence relation and heterogeneous roles of nodes.

relations. For instance, the roles of pickup node and delivery node are different; the pickup node must precede the pairing delivery node; multiple pickup (or delivery) nodes are allowed to be visited consecutively, etc. Therefore, homogeneously treating those nodes with an attention may limit the quality of the solutions. To alleviate this issue, we design six types of attentions in addition to the original one. Among them, three types of attentions are constructed for each pickup node to learn the relations to its pairing delivery node, all pickup nodes and all delivery nodes, respectively. And another three types of attentions are constructed for each delivery node in a similar way.

The details of the heterogeneous attention scheme are as follows. Suppose that h_i^P is the embedding of the ith pickup node, and h_j^D is the embedding of the jth delivery node, then the node embedding of layer $l-1$ can be expressed as $h^{l-1} = Concat(h_0, h_i^P, h_j^D), \forall\ i \in P, \forall\ j \in D$, where we omit $l-1$ for readability. Accordingly, the core of attentions between each pairing nodes is calculated as follows,

$$\begin{aligned} Q_i^{pd} &= W^{Q_{pd}} h_i^P,\ K_i^{pd} = W^K h_{i+n}^D,\ V_i^{pd} = W^V h_{i+n}^D, \\ Q_i^{dp} &= W^{Q_{dp}} h_i^D,\ K_i^{dp} = W^K h_{i-n}^P,\ V_i^{dp} = W^V h_{i-n}^P. \end{aligned} \tag{12}$$

Similarly, the core of attentions from each pickup node to arbitrary pickup nodes and arbitrary delivery nodes is calculated as follows,

$$\begin{aligned} Q_i^{pP} &= W^{Q_{pP}} h_i^P,\ K_i^{pP} = W^K h_i^P,\ V_i^{pP} = W^V h_i^P, \\ Q_i^{pD} &= W^{Q_{pD}} h_i^P,\ K_i^{pD} = W^K h_i^D,\ V_i^{pD} = W^V h_i^D, \end{aligned} \tag{13}$$

and the core of attentions from each delivery node to arbitrary pickup nodes and arbitrary delivery nodes is calculated as follows,

$$\begin{aligned} Q_i^{dP} &= W^{Q_{dP}} h_i^D,\ K_i^{dP} = W^K h_i^P,\ V_i^{dP} = W^V h_i^P, \\ Q_i^{dD} &= W^{Q_{dD}} h_i^D,\ K_i^{dD} = W^K h_i^D,\ V_i^{dD} = W^V h_i^D, \end{aligned} \tag{14}$$

where all parameter matrices are trainable and have same sizes with the original one. Note that we share parameter matrices of all *keys* and *values* for the seven types of attentions to speed up the training, while keep all parameter matrices of *queries* independently reserved to learn problem properties from different perspectives.

Consequently, the immediate attentions of different types could be achieved based on above cores. Regarding the two types of paired nodes, the attentions are calculated as follows,

$$\begin{aligned} a_{i,i+n}^{pd} &= softmax(\frac{Q_i^{pd} * K_{i+n}^{pd}}{\sqrt{d_k}}),\ i \in P, \\ a_{i,i-n}^{dp} &= softmax(\frac{Q_i^{dp} * K_{i-n}^{dp}}{\sqrt{d_k}}),\ i \in D, \end{aligned} \tag{15}$$

where "*" means the element-wise product. Regarding the remaining four types, the immediate attentions are calculated as follows,

$$a_{ij}^y = softmax(\frac{Q_i^{yT} K_j^y}{\sqrt{d_k}}),\ y \in \{pP, pD, dP, dD\}. \tag{16}$$

Then, the multi-head vector concatenates different information from M heads as follows,

$$\text{MultiHead}(Q_i^y, K_j^y, V_j^y) = Concat(h_i^1, ..., h_i^M) W^O, \tag{17}$$

where $W^O \in R^{d_h \times d_h}$ is the trainable parameter matrices, and the single head vector h_i is calculated by adding up the heads

from all types of attentions as follows,

$$h_i^m = a_{ij}V_j + a_{ij}^{pd} * V_j^{pd} + a_{ij}^{dp} * V_j^{dp} + \sum_y \sum_j a_{ij}^y V_j^y, \quad y \in \{pP, pD, dP, dD\},\ m \in \{1, ..., M\}. \tag{18}$$

Note that to enable the policy network to perceive and learn the pairing relationships and the precedence constraint, the heads are added to different parts of node embeddings according to the types of attentions. E.g., the heads with attentions from pickup nodes to other roles of nodes only contribute to the pickup node embeddings, where the heads added to the delivery node embeddings are all zero.

Finally, the attention mechanism works as follows,

$$h_i^{l'} = BN^l(h_i^{l-1} + \text{MultiHead}_i^l(Q_i^y, K_j^y, V_j^y)), \tag{19}$$

$$h_i^l = BN^l(h_i^{l'} + FF^l(h_i^{l'}), \tag{20}$$

where h_i^l is the node embeddings at the lth layer, and parameters are independently reserved for different layers. Each multi-head attention layer and feed-forward layer consists of a skip-connection [30] and a batch normalization (BN) layer [31].

Subsequently, the graph embedding of inputs h^N is computed as the mean of node embeddings at final layer, i.e., $\bar{h}^N = \frac{1}{2n+1}\sum_{i=0}^{2n} h_i^N$. Both the node embeddings h_i^N and graph embedding $\bar{h}^N$ are taken as the input of the decoder.

2) Decoder: Given the graph embedding and node embeddings from the encoder, the decoder will generate a probability vector for selecting a node at each decoding step.

To achieve this, a *context* h^c (output embedding) is always needed at the beginning, which only consists of graph embedding and last node embedding at time step t as follows,

$$h^c = Concat(\bar{h}^N, h_{\pi_{t-1}}^N). \tag{21}$$

At the first time step, the node embedding is usually replaced with trainable parameters. Similar to [32], the *glimpse* h^g used to aggregate the contributions from different parts of node information is expressed as follows,

$$h^g = \text{MultiHead}(W_g^Q h^c, W_g^K h^N, W_g^V h^N), \tag{22}$$

where $W_g^Q, W_g^K \in R^{d_h \times d_k}, W_g^V \in R^{d_h \times d_v}$ are trainable parameter matrices. Given that $q = W^Q h^g$ and $k_i = W^K h_i^N$, the *compatibility* with all nodes at step t is calculated as follows,

$$\hat{h^t} = C \cdot tanh(h^t), \tag{23}$$

where

$$h_i^t = \begin{cases} \frac{q^T k_i}{\sqrt{d_k}}, & if\ i \notin \pi_{t'},\ \forall\ t' < t, \\ -\infty, & \text{otherwise}, \end{cases} \tag{24}$$

and C is set to 10 to clip the result for better exploration. Meanwhile, all invalid nodes are dynamically masked at each step to guarantee feasibility. In particular, upon departure from the depot, all delivery nodes would be masked. After visiting the first pickup node, the pairing delivery node would be unmasked, and the visited nodes, including the depot, would be masked. The masked nodes are allowed to be unmasked only when feasibility is satisfied. Finally, we compute the probability vector using the softmax function as follows,

$$p(\pi_t | X, L_{t-1}) = softmax(\hat{h^t}), \tag{25}$$

where element p_i^t represents the probability of selecting node x_i at step t. This process iteratively continues until all nodes are visited and the vehicle returns to the depot. Regarding the decoding strategy, we could choose the node with the maximum probability at each step in a greedy manner, and we could also sample multiple solutions and retrieve the best one, which will be investigated in the experiments.

ALGORITHM 1: Reinforcement Learning Algorithm

input : number of episodes I; batched size B; step limits for route constructions T;
actor network p_θ with parameters θ;
critic network v_ϕ with parameters ϕ;

1 **foreach** $epoch = 1, 2, ..., I$ **do**
2 Generate B problem instances randomly;
3 Reset gradients $d_\theta \leftarrow 0$;
4 **foreach** $b = 1, 2, ..., B$ **do**
5 Initiate state $s_t^b = (L_0^b)$;
6 **while** $t < T$ **do**
7 $a_t^b \sim p_\theta(\cdot|s_t^b)$;
8 Pick an action a_t^b in sampling strategy using the policy network p_θ;
9 Receive reward r_t^b and transit to next state;
10 $t = t + 1$;
11 **end**
12 $R^b = \sum_{t=1}^{T} r_t^b$;
13 Receive baseline reward $v_\phi(X^b)$ using GreedyRollout policy v_ϕ;
14 **end**
15 $d_\theta \leftarrow \frac{1}{B}\sum_{b=1}^{B}(R^b - v_\phi(X^b))\nabla_\theta log\ p_\theta(\pi^b|X^b)$;
16 Update θ using d_θ;
17 **if** *OneSidedPairedTTest*(p_θ, v_ϕ) $<\alpha$ **then**
18 Replace v_ϕ with p_θ;
19 **end**
20 **end**

C. Training Algorithm

The training of the proposed policy for solving PDP is summarized in Algorithm 1, where we adopt the reinforcement learning method with roll-out baseline in [8]. The policy gradient method is characterized by two networks: 1) actor network, i.e., the p_θ mentioned above, governs the actions of node selection by generating a vector of probabilities over those actions, and sampling according to the probabilities to better explore the action space; 2) self-critic network v_ϕ, i.e., a roll-out baseline with similar structure as the actor network, calculates the reward given the initial state by selecting the node with maximum probability to eliminate variance. After receiving the reward of the actor network R and the baseline reward of the critic network $v_\phi(X)$, the reinforcement learning algorithm updates the parameters of two networks accordingly using the policy gradient method. In specific, at each episode, we construct a route for each instance and calculate the reward with respect to this solution in line 12, and the parameters of

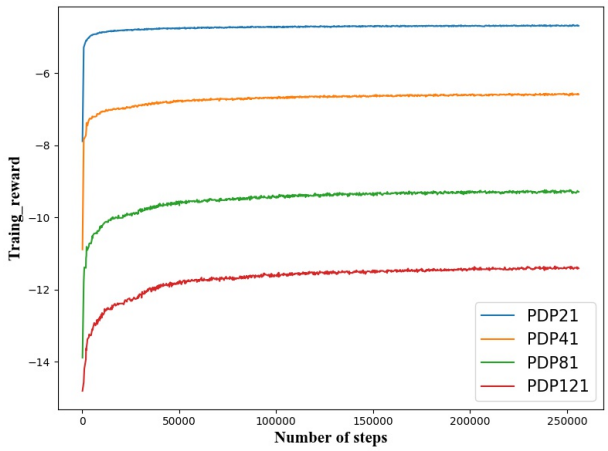


Fig. 4: The Reward Curve.

actor network are updated in line 16. Moreover, the expected reward of critic network $v_\phi(X^b)$ for instance b is obtained from a greedy roll-out of the policy in line 13. Furthermore, the parameters of critic network are replaced with that of actor network when the performance of the latter is significantly superior according to a paired t-test on several fixed number of instances in line 18 [8]. By updating the two networks, the policy p_θ is trained towards finding solutions of higher quality.

V. Computational Experimentation and Analysis

In this section, we conduct experiments to verify the performance of the proposed DRL model for solving PDP. Specifically, with the customers divided into paired pickup and delivery nodes, a vehicle starts at depot, and visits all customers exactly once with pairing relations and precedence constraint, i.e., the pickup node must be visited before its delivery node. The objective is to minimize the total travel time. Note that, PDP is an NP-hard problem, and the computation complexity grows exponentially as problem scales up.

A. Experimentation Settings

Following the settings in existing works [6]–[8], [27], the locations of the depot and customer (pickup and delivery pair) nodes are randomly and independently generated using a 2-dimensional uniform distribution in the range of 0 and 1, where the distance between two nodes are calculated based on Euclidean space. For expression simplification, the vehicle speed f in Eq. (1) and Eq. (8) is fixed to 1, however, our method is capable of handling varying speeds. Regarding the scale, the problem sizes are set to 21, 41, 81, and 121 (including one depot and all nodes of pickup and delivery), and termed as PDP21, PDP41, PDP81, PDP121, respectively.

Regarding training, all instances are generated on the fly. We run 800 epochs, and in each epoch, 2500 batches with 512 instances are processed. For each instance, the 2-dimension locations of nodes are first linearly projected to a 128-dimension vector, then processed by a 3-layers heterogeneous attention mechanism before fed into the decoder that shares the same hidden layer dimension with the encoder. Moreover, we adopt the Adam Optimizer to train the policy network with constant learning rate 10^{-4}. Each epoch takes 4.37 mins for PDP21, 27.48 mins for PDP41 with single GPU (2080Ti) 35.47 mins for PDP81 with two GPUs and 44.11 mins for PDP121 with three GPUs, respectively. Regarding testing, instances are fixed for our method and all baselines, where 10000 instances are generated for each problem size, using the same distribution with the training one.

TABLE I: Diverse DRL models for PDP.

Method	PDP81 Obj.	PDP81 Gap	PDP81 Time	PDP121 Obj.	PDP121 Gap	PDP121 Time
4 attns-no-sharing	9.009	0.54%	1.72s	11.024	0.57%	3.64s
4 attns-sharing	8.987	0.29%	1.72s	11.371	3.73%	3.69s
7 attns-no-sharing	9.001	0.45%	1.78s	10.965	0.03%	3.78s
7 attns-sharing	8.961	0%	1.94s	10.962	0%	3.83s

B. Diversity of the Heterogeneous Attentions

With respect to the proposed DRL model, we apply two versions of decoding strategy during testing: 1) *Greedy*, always selects the node with maximum probability at each decoding step under the precedence constraint; 2) *Sampling*, engenders $\mathcal{N}$ solutions for each instance by sampling the probability distribution in Eq. (25), and retrieves the best one, where $\mathcal{N}$ is set to 1280 and 12800, and termed as DRL(*Sample1280*) and DRL(*Sample12800*), respectively. Before comparing with others, we first depict the training curves of our DRL method for all problem sizes in Fig. 4. From the reward curves we observe that the rewards increase very fast for all cases until converged. As the number of customers increases, the rewards become smaller and converge relatively slower, which is reasonable since reward is inversely proportional to the route length and the computation complexity grows when the problem scales up.

To verify the impacts of different attention mechanisms on the performance, we evaluate different combinations of attentions. Apart from the seven types of attentions in our method, we also consider a mechanism with four types of attentions. Regarding the latter, besides the original attention, it only integrates the attentions from each ego pickup node to nodes of other three roles, i.e., the ego delivery node, all pickup nodes and delivery nodes. We term them as *7 attns-sharing* (ours) and *4 attns-sharing*, respectively. For the two mechanisms, we also consider their variants by disabling the parameter sharing for *keys* and *values*, which are termed as *7 attns-no-sharing* and *4 attns-no-sharing*, respectively. We concentrate on Sample12800 for PDP81 and PDP121, and the results are recorded in Table I. We observe that the policy with 7 attns-sharing (ours) achieves the smallest objective value with only slightly longer computation time, which well justified the rationale for the design of our heterogeneous attentions. Therefore, we adopt *7 attns-sharing* in our method.

C. Comparison Analysis

The proposed DRL model is compared with five baselines: 1) CPLEX [33], a state-of-the-art exact solver for combinatorial optimization problems; 2) OR-Tools [34], a constraint-solver based method that widely used for combinatorial optimization problems; 3) Simulated Annealing (SA) [35], a metaheuristic method with guided neighborhood search for

TABLE II: DRL Method VS Baselines for PDP.

Method	PDP21			PDP41			PDP81			PDP121		
	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
CPLEX	**4.560**	**0%**	433.72s	-	-	-	-	-	-	-	-	-
SA	4.602	0.92%	48.14s	6.409	1.50%	155.44s	9.098	1.53%	305.58s	11.298	3.07%	497.70s
VNS	4.608	1.05%	90.78s	6.592	4.40%	139.25s	9.732	8.60%	583.90s	12.307	12.27%	1035.83s
OR-Tools	4.704	3.16%	0.29s	6.586	4.31%	7.45s	9.188	2.53%	233.72s	11.173	1.92%	1774.58s
AM(Greedy)	5.021	10.11%	**0.09s**	7.180	13.72%	**0.15s**	10.042	12.06%	**0.37s**	12.334	12.52%	**0.52s**
AM(Sample1280)	4.731	3.75%	0.26s	6.739	6.73%	0.43s	9.469	5.67%	0.91s	11.674	6.50%	1.78s
AM(Sample12800)	4.691	2.87%	0.64s	6.673	5.69%	1.00s	9.382	4.70%	1.83s	11.565	5.50%	3.51s
DRL(Greedy)	4.668	2.37%	0.11s	6.536	3.52%	0.19s	9.229	2.99%	0.41s	11.322	3.50%	0.70s
DRL(Sample1280)	4.593	0.72%	0.29s	6.338	0.38%	0.49s	8.996	0.39%	0.98s	11.025	0.84%	1.90s
DRL(Sample12800)	4.585	0.55%	0.84s	**6.314**	**0%**	1.12s	**8.961**	**0%**	1.94s	**10.962**	**0%**	3.83s

solving location routing problem with simultaneous pickup and delivery; 4) Variable Neighborhood Search (VNS) [36], a heuristic method based on different neighborhood structures for solving PDP with loading constraints; 5) the DRL based attention model (AM) [8], learning a policy of node selection in a constructive manner for TSP and CVRP. The time limit of CPLEX for solving PDP20 is set to 500 seconds for reasonable computation time. We adapt the pairing and precedence constraints of PDP to all baselines so that they share the same problem characteristics with ours.

We record the performance of our DRL method and baselines for all problem sizes in Table II, where the objective value, optimality gap, and computation time of an instance are adopted for evaluation. Note that, according to our experiments, the exact solver CPLEX is only available for PDP21, and prohibitively time-consuming for larger sizes. Therefore, the gap is defined by comparing with CPLEX for PDP21, and with the best one among all methods for other sizes. From Table II, we observe that our DRL method with Sample12800 outperforms all conventional heuristic methods and AM model for all cases. Although the computation time of our method is slightly longer than that of AM, it is significantly shorter than that of all conventional baselines. Regarding three variants of our DRL method and AM, both Sample1280 and Sample12800 produce smaller objective values and gaps than Greedy, which demonstrates the effectiveness of sampling strategy in improving the solution quality, despite of slightly longer computation time. Among all baselines, CPLEX can engender optimal solutions for PDP21, but suffers from the longest computation time for solving a single instance, i.e., 433.72 seconds. SA achieves better performance in terms of objective values and gaps than other heuristic methods for most cases except for PDP121, where OR-Tools achieves smaller objective value but longer computation time. Furthermore, AM(Greedy) achieves shortest computation time and highest objective values. Moreover, given that Sample12800 is superior to Sample1280, and Sample1280 superior to Greedy, our Greedy alone outperforms all variants of AM regarding objective values and gaps, which verifies that the proposed heterogeneous attention mechanism is capable of empowering the policy network to intrinsically capture and learn the precedence relations. Another phenomenon is that, even our sole Sample1280 outperforms all conventional heuristic methods and AM in all aspects. Besides, as the problem scales up, the objective value becomes larger and computation time becomes longer for all methods, however, the computation time of DRL based methods almost increases linearly, with that of conventional methods growing exponentially. From Table II, our DRL method achieves the best overall performance in comparison with AM and all conventional methods.

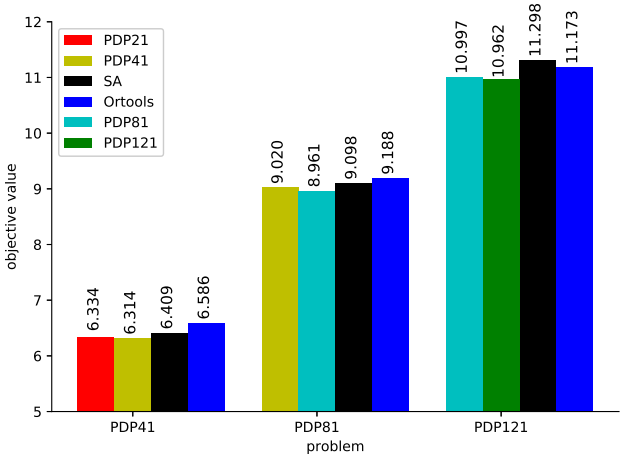


Fig. 5: Generalization on larger problem size.

D. Generalization Analysis

To verify the generalization of our method, we continue to conduct two types of experiments: 1) apply the policy learnt for a problem size to a larger one, 2) apply the policy learnt for uniform distribution to others, i.e., Gaussian distribution.

Regarding the former, we focus on sample12800, and apply the policy learnt for a problem size to its larger neighbour, e.g., apply sample12800 for PDP21 to PDP41. In Fig. 5, the horizontal coordinate refers to the problems to be solved, and the vertical one refers to the objective values of different policies or methods, where we consider SA and OR-Tools given their good performance in Table II. We can observe that for each problem size, the corresponding policy performs best, which is slightly superior to the one learnt for a different size. However, both of them outstrip the best conventional methods, from which we conclude that our DRL method has desirable capability of generalization on larger problem sizes.

Regarding the latter, we focus on PDP81 and PDP121 since they are the largest two, where we apply the policies learnt with uniform distribution to Gaussian distribution with different standard deviations. In Table III, three standard deviations

TABLE III: Generalization on a Different Distribution (i.e.,Gaussian Distribution).

Method	PDP81(sdv=0.6)		PDP121(sdv=0.6)		PDP81(sdv=0.8)		PDP121(sdv=0.8)		PDP81(sdv=1.0)		PDP121(sdv=1.0)	
	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time
SA	8.916	342.05s	11.092	505.84s	8.984	344.79s	11.205	507.03s	9.013	364.59s	11.228	515.68s
OR-Tools	9.027	233.83s	11.048	1740.23s	9.103	242.38s	11.119	1747.34s	9.129	244.07	11.139	1753.76s
AM(Greedy)	9.882	0.31s	12.188	0.55s	9.954	0.31s	12.257	0.57s	9.987	0.32s	12.288	0.56s
AM(sample1280)	9.291	0.83s	11.518	1.93s	9.372	0.83s	11.592	1.93s	9.406	0.84s	11.622	1.94s
AM(sample12800)	9.204	1.76s	11.405	3.83s	9.283	1.77s	11.481	3.85s	9.318	1.78s	11.512	3.88s
DRL(Greedy)	9.082	0.42s	11.141	0.70s	9.145	0.42s	11.219	0.69s	9.173	0.41s	11.253	0.68s
DRL(sample1280)	8.844	0.89s	10.840	1.97s	8.911	0.89s	10.919	1.98s	8.940	0.88s	10.953	1.99s
DRL(sample12800)	8.807	1.94s	10.787	3.95s	8.876	1.93s	10.866	3.98s	8.904	1.94s	10.900	3.97s

are adopted, and the one with 1.0 is closer to the original uniform distribution. We observe that our DRL method with Sample12800 always outperforms all other methods in all cases in terms of objective value, and AM(Greedy) consumes shortest computation time. Specifically, both Sample1280 and Sample12800 achieves smaller objective value than conventional heuristics with significantly shorter computation time, and AM with comparable computation time. SA performs better in terms of objective value than OR-Tools for PDP81 but consumes longer computation time, which is the other way round for PDP121. Given the strong superiority to the conventional heuristics, we conclude that our method has satisfactory generalization on a new distribution.

VI. Conclusion and Future Work

In this paper, we cope with a challenging variant of VRP, i.e., the pickup and delivery problem (PDP), which is characterized by a precedence constraint that a pickup node must precede the pairing delivery node. To solve this problem, we propose a deep reinforcement learning method integrating with a heterogeneous attention mechanism, which empowers the policy network to capture and learn the precedence relation and heterogeneous roles of different nodes. Experimental results show that our DRL method achieves the best overall performance compared with existing modern (i.e., AM) and conventional heuristic methods, respectively. Moreover, our method generalizes well to larger problem size and new distribution. Additionally, our method with diverse combinations on heterogeneous attention mechanisms outperform AM and all conventional heuristic methods.

Exploring deep reinforcement learning to solve vehicle routing problem is a newly emerged direction, which is still at a preliminary stage despite of the good potential. The work in this paper is an early attempt on the learning based method for solving PDP, which delivers superior results but may perform inferior for other types of VRPs, such as VRP with time windows constraint and dynamic customer requests. In the future, we plan to investigate the following aspects, 1) multiple vehicles with capacity constraint; 2) time window constraint and dynamic customer requests (or traffic conditions); and 3) real data or other classical benchmark datasets for testing.

References

[1] A. Agra, M. Christiansen, A. Delgado, and L. M. Hvattum, "A maritime inventory routing problem with stochastic sailing and port times," *Computers & Operations Research*, vol. 61, pp. 18–30, 2015.

[2] J. Tang, Y. Yu, and J. Li, "An exact algorithm for the multi-trip vehicle routing and scheduling problem of pickup and delivery of customers to the airport," *Transportation Research Part E: Logistics and Transportation Review*, vol. 73, pp. 114–132, 2015.

[3] S. Anily and A. Federgruen, "One warehouse multiple retailer systems with vehicle routing costs," *Management science*, vol. 36, no. 1, pp. 92–114, 1990.

[4] J. Renaud, F. F. Boctor, and J. Ouenniche, "A heuristic for the pickup and delivery traveling salesman problem," *Computers & Operations Research*, vol. 27, no. 9, pp. 905–916, 2000.

[5] N. Agatz, A. Erera, M. Savelsbergh, and X. Wang, "Optimization for dynamic ride-sharing: A review," *European Journal of Operational Research*, vol. 223, no. 2, pp. 295–303, 2012.

[6] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," in *International Conference on Learning Representations*, 2017.

[7] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác, "Reinforcement learning for solving the vehicle routing problem," in *Advances in Neural Information Processing Systems*, pp. 9839–9849, 2018.

[8] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!," in *International Conference on Learning Representations*, 2018.

[9] L. Xin, W. Song, Z. Cao, and J. Zhang, "Multi-decoder attention model with embedding glimpse for solving vehicle routing problems," in *35th AAAI Conference on Artificial Intelligence*, 2021.

[10] F. Lokin, "Procedures for travelling salesman problems with additional constraints," *European Journal of Operational Research*, vol. 3, no. 2, pp. 135–141, 1979.

[11] M. W. Savelsbergh and M. Sol, "The general pickup and delivery problem," *Transportation science*, vol. 29, no. 1, pp. 17–29, 1995.

[12] K. Ruland and E. Rodin, "The pickup and delivery problem: Faces and branch-and-cut algorithm," *Computers & mathematics with applications*, vol. 33, no. 12, pp. 1–13, 1997.

[13] S. Ropke and J.-F. Cordeau, "Branch and cut and price for the pickup and delivery problem with time windows," *Transportation Science*, vol. 43, no. 3, pp. 267–286, 2009.

[14] G. Kim, Y. S. Ong, T. Cheong, and P. S. Tan, "Solving the dynamic vehicle routing problem under traffic congestion," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 8, pp. 2367–2380, 2016.

[15] Z. Cao, H. Guo, J. Zhang, D. Niyato, and U. Fastenrath, "Improving the efficiency of stochastic vehicle routing: A partial lagrange multiplier method," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3993–4005, 2016.

[16] C. Ghorai, S. Shakhari, and I. Banerjee, "A spea-based multimetric routing protocol for intelligent transportation systems," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[17] Z. Cao, H. Guo, W. Song, K. Gao, Z. Chen, L. Zhang, and X. Zhang, "Using reinforcement learning to minimize the probability of delay occurrence in transportation," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 3, pp. 2424–2436, 2020.

[18] H. Li and A. Lim, "A metaheuristic for the pickup and delivery problem with time windows," *International Journal on Artificial Intelligence Tools*, vol. 12, no. 02, pp. 173–186, 2003.

[19] S. Ropke and D. Pisinger, "A unified heuristic for a large class of vehicle routing problems with backhauls," *European Journal of Operational Research*, vol. 171, no. 3, pp. 750–775, 2006.

[20] V. Ghilas, E. Demir, and T. Van Woensel, "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines," *Computers & Operations Research*, vol. 72, pp. 12–30, 2016.

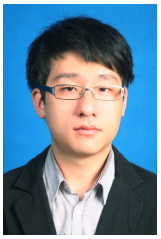
[21] H. Hernández-Pérez, I. Rodríguez-Martín, and J.-J. Salazar-González, "A hybrid heuristic approach for the multi-commodity pickup-and-delivery traveling salesman problem," *European Journal of Operational Research*, vol. 251, no. 1, pp. 44–52, 2016.
[22] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Advances in Neural Information Processing Systems*, pp. 2692–2700, 2015.
[23] L. Xin, W. Song, Z. Cao, and J. Zhang, "Step-wise deep learning models for solving routing problems," *IEEE Transactions on Industrial Informatics*, 2020.
[24] J. Zhao, M. Mao, X. Zhao, and J. Zou, "A hybrid of deep reinforcement learning and local search for the vehicle routing problems," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1 – 11, 2020.
[25] J. Shi, Y. Gao, W. Wang, N. Yu, and P. A. Ioannou, "Operating electric vehicle fleet for ride-hailing services with reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 11, pp. 4822–4834, 2019.
[26] J. James, W. Yu, and J. Gu, "Online vehicle routing with neural combinatorial optimization and deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 10, pp. 3806–3817, 2019.
[27] X. Chen and Y. Tian, "Learning to perform local rewriting for combinatorial optimization," in *Advances in Neural Information Processing Systems*, pp. 6278–6289, 2019.
[28] Y. Wu, W. Song, Z. Cao, J. Zhang, and A. Lim, "Learning improvement heuristics for solving routing problems," *https://arxiv.org/abs/1912.05784v2*, 2020.
[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, pp. 5998–6008, 2017.
[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
[31] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, pp. 448–456, 2015.
[32] O. Vinyals, S. Bengio, and M. Kudlur, "Order matters: Sequence to sequence for sets," in *International Conference on Learning Representations*, 2015.
[33] IBM, "Cplex, ibm ilog cplex optimization tools," *URL https://www.ibm.com/analytics/cplex-optimizer*, 2016.
[34] Google, "Or-tools, google optimization tools," *URL https://developers.google.com/optimization/routing*, 2016.
[35] V. F. Yu and S.-Y. Lin, "Solving the location-routing problem with simultaneous pickup and delivery by simulated annealing," *International Journal of Production Research*, vol. 54, no. 2, pp. 526–549, 2016.
[36] T. Pinto, C. Alves, and J. V. de Carvalho, "Variable neighborhood search algorithms for pickup and delivery problems with loading constraints," *Electronic Notes in Discrete Mathematics*, vol. 58, pp. 111–118, 2017.

Jingwen Li received the bachelor's degree in the field of computer science from University of Electronic Science and Technology of China, China, in 2018. She is currently pursuing the phd's degree with the department of Industrial Systems Engineering and Management, National University of Singapore (NUS). Her research interests include deep reinforcement learning for combinatorial optimization problems, especially for vehicle routing problems.

Liang Xin received the bachelor's degree from Tongji University and the master's degree from Carnegie Mellon University. He is currently pursuing the phd's degree with the School of Computer Science and Engineering, Nanyang Technological University, Singapore (NTU). His research interests include deep learning for combinatorial optimization problems.

Zhiguang Cao received the Ph.D degree from Interdisciplinary Graduate School, Nanyang Technological University, Singapore, 2017. He received the B.Eng. degree in Automation from Guangdong University of Technology, Guangzhou, China, in 2009 and the M.Sc. degree in Signal Processing from Nanyang Technological University, Singapore, in 2012, respectively. He worked as a Research Fellow with Future Mobility Research Lab, and Energy Research Institute @ NTU (ERI@N), Singapore. He is currently a Research Assistant Professor with the Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore. His research interests focus on applying deep (reinforcement) learning to solve combinatorial optimization problems.

Andrew Lim received the Ph.D. degree in computer science in 1992 from the University of Minnesota, Minneapolis. He is currently a Professor with the department of Industrial Systems Engineering and Management, National University of Singapore (NUS). His works have been published in key journals such as Operations Research and Management Science, and disseminated via international conferences and professional seminars. Before Andrew was recruited by NUS under The National Research Foundation's Returning Singaporean Scientists Scheme in 2016, he spent more than a decade in Hong Kong where he held professorships in The Hong Kong University of Science and Technology and City University of Hong Kong.



Wen Song en Songen SongW received the B.S. degree in automation and the M.S. degree in control science and engineering from Shandong University, China, in 2011 and 2014, respectively, and the Ph.D. degree in computer science from the Nanyang Technological University, Singapore, in 2018. He was a Research Fellow in the Singtel Cognitive and Artificial Intelligence Lab for Enterprises (SCALE@NTU). He is currently an Associate Professor with the Institute of Marine Science and Technology, Shandong University, China.

His current research interests include artificial intelligence, planning and scheduling, multi-agent systems, and operations research.

Jie Zhang received the Ph.D. degree from Cheriton School of Computer Science, University of Waterloo, Canada, in 2009. He is an Associate Professor with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. During his Ph.D. study, he received the prestigious NSERC Alexander Graham Bell Canada Graduate Scholarship for top Ph.D. students across Canada. His research has been focused on the design of effective and robust intelligent software agents, through the modeling (trustworthiness, preferences) and simulation of different agents in a wide range of environments, using AI techniques (data mining, machine learning, and probabilistic reasoning) and multi-agent technologies.