A Hybrid of Deep Reinforcement Learning and Local Search for the Vehicle Routing Problems

Jiuxia Zhao, Minjia Mao, Xi Zhao, *Senior Member, IEEE*, and Jianhua Zou, *Member, IEEE*

***Abstract*—Different variants of the Vehicle Routing Problem (VRP) have been studied for decades. State-of-the-art methods based on local search have been developed for VRPs, while still facing problems of slow running time and poor solution quality in the case of large problem size. To overcome these problems, we first propose a novel deep reinforcement learning (DRL) model, which is composed of an actor, an adaptive critic and a routing simulator. The actor, based on the attention mechanism, is designed to generate routing strategies. The adaptive critic is devised to change the network structure adaptively, in order to accelerate the convergence rate and improve the solution quality during training. The routing simulator is developed to provide graph information and reward with the actor and adaptive cirtic. Then, we combine this DRL model with a local search method to further improve the solution quality. The output of the DRL model can serve as the initial solution for the following local search method, from where the final solution of the VRP is obtained. Tested on three datasets with customer points of 20, 50 and 100 respectively, experimental results demonstrate that the DRL model alone finds better solutions compared to construction algorithms and previous DRL approaches, while enabling a 5- to 40-fold speedup. We also observe that combining the DRL model with various local search methods yields excellent solutions at a superior generation speed, comparing to that of other initial solutions.**

***Index Terms*—VRP, VRPTW, routing simulator, deep reinforcement learning, adaptive critic, local search.**

I. Introduction

INTELLIGENT transportation system (ITS) is an essential issue in smart cities [1]. Nowadays, under the development of e-commerce, the delivery cost has become a top burden [2]. Moreover, Bräysy found out that distribution costs represent almost half of total logistic costs [3]. In food and beverage industry, distribution costs account for 70 percent of the additional cost of goods [4]. This indicates that ITS should pay attention to effective route planning. The vehicle routing problem (VRP) is a typical NP-hard problem in combinatorial optimization and has been studied for decades [5]. In ITS, applications of VRPs include express delivering [6], production planning [7] and airline scheduling [8], indicating wide practical significance.

Manuscript received November 22, 2019; revised March 24, 2020; accepted June 3, 2020. Date of publication July 15, 2020; date of current version November 1, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 91746111 and Grant 71702143 and in part by the Key Research and Development Program of Shaanxi Province under Grant 2020 ZDLGY 09-08. The Associate Editor for this article was H. A. Rakha. *(Jiuxia Zhao and Minjia Mao contributed equally to this work.) (Corresponding author: Xi Zhao.)*

Jiuxia Zhao and Jianhua Zou are with the School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: jiuxia@stu.xjtu.edu.cn).

Minjia Mao is with the School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an 710049, China.

Xi Zhao is with the School of Management, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: zhaoxi1@mail.xjtu.edu.cn).

Digital Object Identifier 10.1109/TITS.2020.3003163

VRPs are generally given an approximate solution by heuristic methods due to the complexity of the problem. Two main categories of heuristic algorithms are construction algorithms and local search algorithms [9]. Construction algorithms give solutions of VRPs through hand-crafted rules, sacrificing solution quality for high efficiency. For the second category, existing methods using local search include simulated annealing [10], tabu search [11] and large neighborhood search (LNS) [12]. Starting from an initial solution, local search uses different search operators to search for better solutions. In modern operation research, construction algorithms are usually used to generate initial solutions [13], [14]. However, the final iterative results of local search algorithms largely depend on the initial solutions especially for problems with large sizes [15], because the search space of local search algorithms is near the initial solution. Therefore, unsuitable initial solutions may lead to plenty of calculation time and local optimum.

Recently, deep reinforcement learning (DRL) has achieved appealing results in several challenging problems including Go [16], app recommendation [17] and combinatorial optimization [18]–[20]. More specifically, DRL-based models have been proposed to solve several NP-hard optimization problems, consisting of Maximum Cut (MC) [21], Travelling Salesman Problem (TSP) [22] and VRP [20], [23]. Two characteristics of the DRL model make it promising account for VRPs. First, DRL can estimate useful patterns which are difficult to find by manual heuristics especially in large scale problems [24]. Second, DRL has great potential for solving time-sensitive VRPs, due to the fast route generation (inference) process [23].

Nonetheless, several challenges arise when solving routing problems using DRL: (1) Although the inference speed is fast, the training process of the DRL model is time consuming [23]. Hence, it is a challenge to accelerate the convergence rate of DRL models. (2) An interactable environment and a great number of instances are required to train the DRL model. A simulator is needed for generating data and interacting with the model. (3) The solution quality of existing DRL based models is not comparable to state-of-the-art heuristic algorithms like LNS, representing the necessity of solution quality improvement.

To resolve the aforementioned challenges, we introduce a novel DRL model for generating routing strategies. The DRL model consists of an actor, an adaptive critic and a routing simulator. The actor is designed to produce routing policies, the adaptive critic aims to optimize the parameters of the actor, and the routing simulator serves as the environment to assist the training and validation of the actor and critic. Specifically, attention mechanism [25] is used in the actor network, which is trained with the well known policy gradient method [26]. The adaptive critic is proposed to compute the baseline of the policy gradient method, in order to reduce the variance of training. The network structure of the adaptive critic changes adaptively as the training process moves on, in order to improve convergent performance and achieve better results. The simulator is devised to generate numerical instances for different variants of VRPs, update graph information after each node selection and calculate the reward of the given routing solution.

Moreover, to further obtain satisfactory solutions for large scale problems with a reasonable running time, we propose a two-stage framework for solving VRPs. The first stage is implemented by the aforementioned DRL model, while the second stage considers local search heuristic algorithms. As an end-to-end model, the DRL model can efficiently generate a solution when given the information of a new VRP instance, but the results are generally effective in quality. Using this solution as the initial solution in the second stage, the solution quality and calculation time can be enhanced by combining the local search method.

Our major contributions are summarized as follows:

- We propose a DRL model to generate routing solutions for VRPs. In this model, an adaptive critic is devised to train the parameters of the neural network, which can accelerate the convergence rate and improve the performance of the DRL model simultaneously.
- We develop a routing simulator served as the environment of the DRL model. It simulates the real world scenario, generating massive instances and interacting with the DRL model.
- We propose a two-stage framework to obtain effective routing solutions. By combining the DRL model and the local search method, the framework is capable of producing satisfactory routing solutions for VRPs with large problem size at a reasonable calculation speed.
- We validate the performance of the proposed DRL model and the two-stage framework. We also investigate the affect of the adaptive critic on the solution quality and convergence rate.

The rest of the paper is organized as follows. We first analyze the related work in Section 2. Section 3 elaborates the problem statement. We describe the detailed DRL model in Section 4. And the experimental results are presented in Section 5. Finally, we conclude this paper in Section 6.

II. Related Work

A. Exact and Heuristic Algorithms

VRPs have been studied for decades and researchers have developed extensive solution approaches, consisting of exact algorithms and heuristic algorithms. Exact algorithms, including branch and bound method and column generation, can get the optimal solutions [27]. However, with the scale of the problem grows, the computational complexity increases exponentially, causing the problems unsolvable in a reasonable time [28].

Heuristic algorithms are widely employed in practice. VRPs can be solved by heuristic algorithms via designing hand-crafted features with expert knowledge. The local search method, often referred as neighborhood search, is an important category of heuristic algorithms. Starting from an initial solution, local search methods improve the solution quality by searching in space near the initial solution. Based on the original design, a variety of heuristic algorithms are derived including simulated annealing [10], tabu search [11] and LNS [12]. Local search algorithms have been adopted to different variants of the VRP, such as the Vehicle Routing Problem with Time Window (VRPTW) [29], split delivery VRP [30] and electric VRP [31]. However, the search procedure stops when there is no improvement of solutions in the neighborhood space, causing the algorithms falling into local optima [32].

B. Neural Combinatorial Optimization

The first attempt to use neural network (NN) for combinatorial optimization problems can trace back to Hopfield, who solved the TSP of small instances with NN [33]. Recently, based on the sequence-to-sequence model [34], Vinyals *et al.* proposed the pointer network trained with supervised learning for TSP [35]. For combinatorial optimization, supervised learning based methods generally requires optimal solutions as labels. However, they are hard to obtain in practice. On the contrary, DRL learns from the reward signals, without requiring the label in advance. Bello *et al.* made use of the pointer network to solve the TSP and trained the networks with reinforcement learning techniques [18]. Experiments demonstrated its scalability in the KnapSack problem and TSP, compared with traditional algorithms. Based on the pointer network, Nazari *et al.* proposed an attention-based network to solve the VRP and split delivery VRP, leading outperformed results on medium-size problems compared with construction algorithms and Google OR-Tools [20].

On the other hand, Khalil *et al.* reformulated the nodes of graph structure by a struct2vec network and applied a deep Q-network to train the network [21]. Yu *et al.* combined the pointer network with the struct2vec network to generate routing strategies for online VRP [23], regarded as a guideline for future DRL-based routing problems. Moreover, Li employed guided tree search with the graph convolutional network for maximal independent set (MIS) problem, achieving satisfactory results [24].

III. Problem Statement

In a typical VRP instance, a set of n customer points and one depot are denoted as a graph $G = (V, E)$. For each point v_i, it has several characteristics including coordinate ($c_i = (x_i, y_i)$), and demand (q_i). Service time ($t_{s,i}$) and time window ($[t_{e,i}, t_{l,i}]$) are added in a VRPTW instance. The detailed information of variables is shown in Table I. As depicted in Figure 1, several vehicles are needed to meet the customers according to the following requirements including capacity and time constraints:

- The starting points and ending points of all vehicles are the depot v_0.

TABLE I
VARIABLES

Variable	Definition
$V = \{v_0, v_1, ..., v_n\}$	vertex set
v_0	depot
$v_i, i \neq 0$	customer point i
n	number of customer points
$E = \{(v_i, v_j)\|i \neq j\}$	edge set
x_i	coordinate of point i in X-axis
y_i	coordinate of point i in Y-axis
$c_i = (x_i, y_i)$	coordinate of point i
Q	max load of vehicle
Q'	current load of vehicle
q_i	demand of point i
T	the whole time interval
t	the current time
$t_{e,i}$	early time window of point i
$t_{l,i}$	late time window of point i
$t_{s,i}$	service time of point i
$t_{a,i}$	arrival time of point i
$d_{i,j}$	distance between i and j
$t_{i,j}$	time cost between i and j
π	solution sequence
L	total routing length

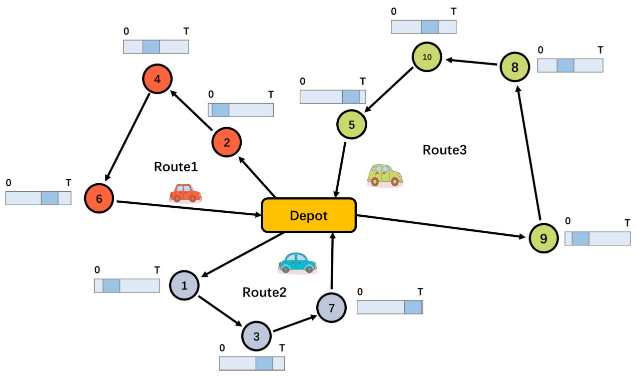


Fig. 1. An illustration of VRPTW. Three vehicles serve 10 customers in given time windows.

- All vehicles have a fixed maximum load Q, and the planning route cannot exceed Q.
- The time windows are set hard, i.e., vehicles are not allowed to serve customers outside of the time windows. When arriving early $t_{a,i} < t_{e,i}$, the vehicle should wait until the early time. Meanwhile, it is not allowed to arrive later than the late time $t_{l,i}$.

Given π as the solution sequence, our goal is to minimize the total routing length L:

$$L(\pi|G) = \sum_{i=1}^{|\pi|-1} \left\| c_{\pi(i)} - c_{\pi(i+1)} \right\|_2, \quad (1)$$

where $\|\cdot\|_2$ is the ℓ_2 norm and $|\pi|$ is the length of the sequence π.

To tackle this problem, we propose a two-stage framework, including the DRL stage and local search stage. As depicted in Figure 2, initial solutions obtained by DRL pass through the local search stage to get the final solutions.

Given a graph G, the proposed DRL model, parameterized by θ, needs to define a strategy $p(\pi|G)$ and select a sequence π. It is a stochastic policy and can be factorized as

$$p_\theta(\pi|G) = \prod_{t=1}^{|\pi|} p_\theta(\pi(t)|\pi(<t), G). \quad (2)$$

In order to minimize the objective in Eq(1), we use the DRL model to compute every component in the right-hand side of Eq(2) and the basic elements of the DRL model are defined as follows.

A. Agent

We consider an operating vehicle as an agent. At every time step t, the agent is in a given state of the environment and chooses an action $\pi(t)$ according to its policy. Receiving reward $-L$ from the environment, the agent learns to generate satisfactory results.

B. State

The state is divided into static state s and dynamic state d_t. Static state, which stays unchangeable all the time, includes coordinates for VRP, considering service time $(t_{s,i})$ and time windows $[t_{e,i}, t_{l,i}]$ of customers for VRPTW. Dynamic state consists of demands of customers q_i, load Q' and position of the operating vehicle at current time t.

C. Action

The action $\pi(t)$ specifys the next destination at current time t and a joint action π instructs the routing strategy of operating vehicles for considered instance. $\pi(t)$ is sampled from the right-hand side of Eq(2) during training time and obtained by greedy or beam search strategies in the test process.

D. Reward

The negative of the objective function Eq(1) is calculated as reward when all the demands are satisfied, since DRL is set to maximize the reward (minimize the tour length).

IV. REINFORCEMENT LEARNING MODEL

The proposed DRL model can be depicted as Figure 2, including an actor, an adaptive critic and a routing simulator. The routing simulator (described in Section A) is used to be the environment of VRPs, returning new states and reward to the actor and adaptive critic. The actor (described in Section B) is designed to generate the routing policy of the given graph. The adaptive critic (described in Section C) aims to optimize the parameters of the actor via estimating the reward adaptively.

A. Routing Simulator

A reliable environment is necessary for DRL. Due to the high complexity of real world problems, devising the simulator is a challenge. In this section, the design of the routing simulator is introduced. This simulator first generate massive VRP instances for training and validation. In addition to data generation, the simulator needs to interact with the vehicle as it moves. Once the vehicle chooses the next customer point, the simulator has two main aspects to update. One is to update the dynamic state, and the other is to mask the unavailable points at next time [18].

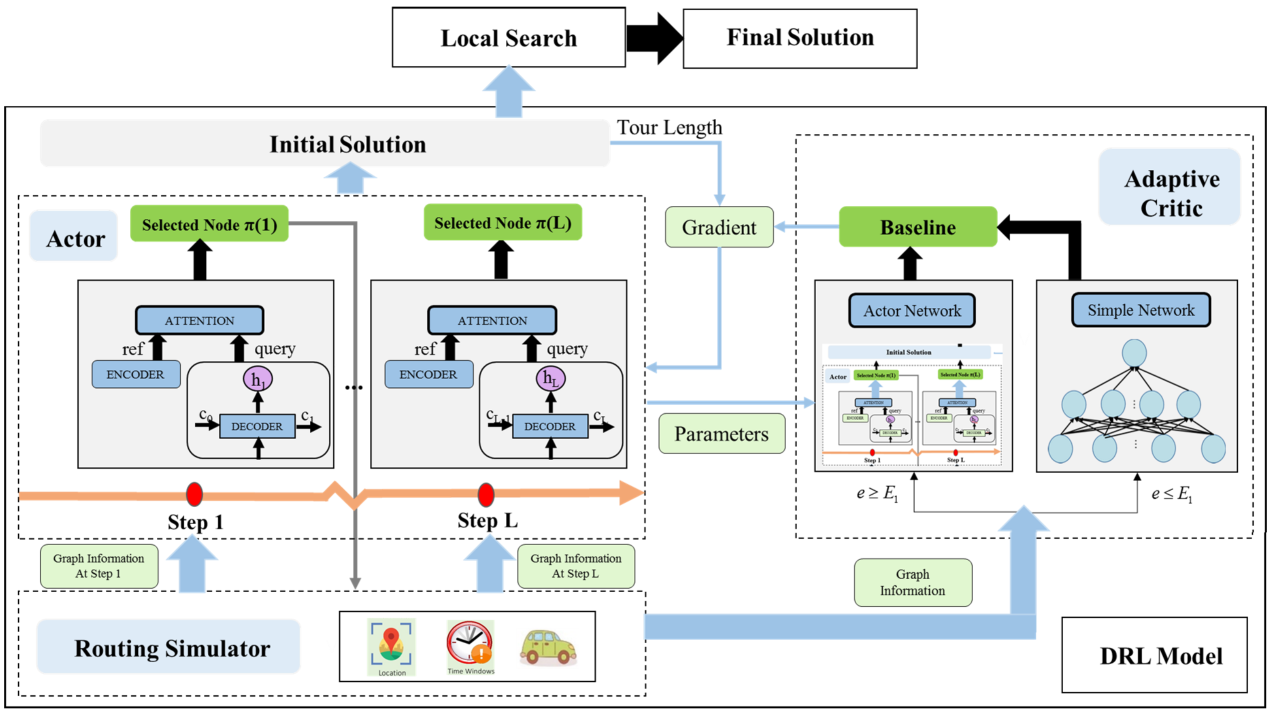


Fig. 2. The architecture of the model.

1) Data Generation: Well-known datasets of VRPs include Augerat dataset [36] for VRP and Solomon benchmark [37] for VRPTW. However, the number of instances in existing datasets cannot meet the need of DRL. Hence the routing simulator firstly tackles the data generation problem. For the sake of brevity, we only discuss the data generation of the VRPTW, because VRP can be transferred from VRPTW by removing time-related elements.

According to the data distribution of Solomon benchmark, the routing simulator generates instances with feasible solutions. In a VRPTW instance, the simulator first creates n customers and one depot. Each customer has nine features which are divided mainly into two parts, static and dynamic elements. The static part consists of the constant variables over time including coordinates (x_i, y_i), time windows $[t_{e,i}, t_{l,i}]$ and service time $t_{s,i}$. Both the service time $t_{s,i}$ and the length of time window Δt are randomly generated from a closed interval. Then the early time $t_{e,i}$ and the due time $t_{l,i}$ are calculated as follows:

$$t_{e,i} = \frac{e_i \times d_{0i}}{v}, e_i \in [1, \frac{T - t_{s,i} - \Delta t_i}{d_{0i}} \times v - 1], \quad (3)$$

$$t_{l,i} = t_{e,i} + \Delta t_i, \quad (4)$$

where v is the speed of the vehicle. d_{0i} denotes the distance between depot v_0 and customer point v_i. T is the whole time interval.

For dynamic elements, the variables change with time. The dynamic state includes the demand of each customer, current load, time and position of the vehicle. The demand q_i of a customer point is an integer and the demand of depot is 0. The maximize load Q varys with the problem sizes n, reffering to Nazari [20].

2) Updating Dynamic: After choosing the next customer to serve, the state changes according to the movement of the operating vehicle. The static state remains unchanged, while the dynamic state updates as the vehicle moves forward. Once the vehicle decides to serve customer j, the position variable equals the index of the chosen customer. Meanwhile, the demand of the chosen customer becomes zero and the load of the vehicle updates by subtracting the demand value $Q' = Q' - q_j$. In addition, the time is set to be the end of this service $t' = \max\{t_{a,j},\ t_{e,j}\} + t_{s,j}$ where $t_{a,j}$ is the time of arrival.

3) Updating Mask: Owing to the capacity and time constraints, there exist points that the vehicle can not reach at current time t. And we utilize the mask to present the available customer points of the operating vehicle. The customer points that violate the capacity and time constraints are masked and the already served points are also masked.

The different mask conditions are shown in Eq(5), Eq(6) and Eq(7) seperately. And the ultimate mask is a conbination of the above situations, shown in Eq(8):

$$C^q_{t,j} = \begin{cases} 1 & q_j \geq 1 \\ 0 & \text{otherwise} \end{cases}, \quad (5)$$

$$C^Q_{t,j} = \begin{cases} 1 & q_j \leq Q' \\ 0 & \text{otherwise} \end{cases}, \quad (6)$$

$$C^t_{t,j} = \begin{cases} 1 & t_{a,j} \leq t_{l,j} \\ 0 & \text{otherwise} \end{cases}, \quad (7)$$

$$\text{Mask}^t_j = C^q_{t,j} \odot C^Q_{t,j} \odot C^t_{t,j}, \quad (8)$$

where $\odot$ means the element-wise product.

B. Actor Network Architecture

1) Encoder: The encoder is designed to map the state of the current graph, making the agent understanding the representation of each vertex. Like previous network architecture [20], the static state and the dynamic state are embedded respectively. In basic attention mechanism [25], the model is sensitive to the order of the input sequence. A graph embedding is used in this paper to overcome this problem. As shown in Eq(9) and Eq(10), the static and dynamic state are embedded into a D-dimensional space. The embeddings are sent as the reference information of the attention layer.

$$s_i^r = W^s s_i + b^s, \tag{9}$$

$$d_{i,t}^r = W^d d_{i,t} + b^d, \tag{10}$$

where s_i means the static state and $d_{i,t}$ means the dynamic state of point i at step t.

2) Decoder: Taking the embedding of current vehicle information as input, the decoder decodes the present state into high dimensional hidden state sequentially. A Long Short-Term Memory (LSTM) layer [38] is used to generate hiddden state h_t, which is the query in the attention layer.

3) Attention Layer: The attention layer is used to describe the similarity between the current state and each customer point. At step t, the hidden state h_t of decoder is used as a query to calculate the relevance with the graph information (s^r, d_t^r), where (s^r, d_t^r) is the output of the encoder. The relevant parameter a_t is set as

$$u_i^t = v_a^T \tanh\left(W_a\left[s_i^r; d_{i,t}^r; h^t\right]\right), \tag{11}$$

$$a_t = \text{softmax}\left(u^t\right), \tag{12}$$

where $[\cdot\ ;\ \cdot]$ is the concatenation operator between vectors.

In VRPs, the network will take a mask to determine whether the next customer point meets the demand and time constraints. The mask is given by the routing simulator and shown as

$$\text{Mask}_i^t = \begin{cases} 1, & \text{if node } i \text{ is valid at time } t \\ 0, & \text{otherwise} \end{cases}. \tag{13}$$

At last, consider M to be a large constant, the probability of each next optional customer point can be represented as follows:

$$p(\pi(t)|(\pi(<t), G) = \text{softmax}\left(a_t + M \cdot \text{Mask}^t\right). \tag{14}$$

The detailed processing procedure of the proposed model is depicted in Algorithm 1.

C. Adaptive Critic Training

Vinyals *et al.* used supervised learning to train the neural network for the TSP [35]. However, high quality labels are difficult to obtain but are the key to supervised learning. On the contrary, DRL only needs the reward signal which is the negative of the total tour length to train the network. By setting the DRL parameters as θ and providing the graph G, we can get the training objective of the network:

$$J(\theta|G) = \mathbb{E}_{\pi\sim p_\theta(\cdot|G)} L(\pi|G). \tag{15}$$

The policy gradient is used to train the parameters [26]:

$$\nabla_\theta J(\theta|G) = \mathbb{E}_{\pi\sim\theta}\left[(L(\pi|G) - b(G))\nabla_\theta \log p_\theta(\pi|G)\right], \tag{16}$$

where $b(G)$ means a baseline which demonstrates the routing length of different instances and is used to reduce the training variance.

Algorithm 1 Processing Procedure of the DRL Model

```
Input Static state $s_i$ for $i \in \{1,\ldots,n\}$, Dynamic state $d_{i,0}$ for $i \in \{1,\ldots,n\}$
1:  Compute embeddings of static state $s_i^r$ for $i \in \{1,\ldots,n\}$ by Encoder.
2:  Initialize current position $\pi(0) \leftarrow 0$.
3:  for t = 1, ..., N do
4:      Compute embeddings of dynamic state $d_{i,t}^r$ for $i \in \{1,\ldots,n\}$ by Encoder.
5:      Compute the query $h_t$ according to $s_{\pi(t-1)}^r$ by Decoder.
6:      Compute the attention value $a_t$ by Eq(11) and Eq(12).
7:      Compute the Mask $\mathrm{Mask}_i^t = C_{t,i}^q * C_{t,i}^Q * C_{t,i}^t$ for $i \in \{1,\ldots,n\}$ by Routing Simulator.
8:      Compute the probability of each point $p(\pi(t)|(\pi(<t), G) =$
        $\text{softmax}\left(a_t + M \cdot \text{Mask}^t\right)$.
9:      $\pi(t) = \text{Greedy}(p(\pi(t)|(\pi(<t), G))$ if choose greedy mode.
10: end for
11: return Joint action $\pi = \{\pi(t)\}_1^N$ to get the solution.
```

In practice, the expected value is replaced with the average of Monte Carlo sampling in a batch size B:

$$\nabla_\theta J(\theta) \approx \frac{1}{B}\sum_{i=1}^{B}\left(L(\pi_i|G_i) - b(G_i)\right)\nabla_\theta \log p_\theta(\pi_i|G_i). \tag{17}$$

In the actor-critic framework, the critic is set to predict the advantage function, which is exactly the $b(G)$. As depicted in Fig 2, the adaptive critic consists of a simple network and an actor network. During the early stage of training, a multi-layer simple network is designed to formulate a normal critic. The loss of the normal critic is calculated as

$$\mathcal{L}(\theta^C) = \frac{1}{B}\sum_{i=1}^{B}\left\|L(\pi_i|G_i) - V_{\theta^C}(G_i)\right\|_2^2, \tag{18}$$

where $V_{\theta^C}(G_i)$ is the output baseline of the normal critic.

However, the simple network structure is hard to describe the difference between instances effectively. To improve the performance of critic, we borrow the idea of self critic employed in image captioning [39]. One problem of self critic is that in the early stage of training, the critic, as a copy of actor, cannot calculate the baseline accurately, since the actor itself performs bad at the early stage.

To overcome this problem, the adaptive critic copies the network structure and parameters of the actor when the actor is trained for several epochs. By denoting the parameters of the adaptive critic as θ^c, the baseline is the route length of a greedy solution:

$$b(G) = L(\pi|G)|_{\pi=\text{Greedy}(\pi\sim p_{\theta^C}(\cdot|G))}. \tag{19}$$

The adaptive critic will reacquire the parameters of the actor θ only when the t-test of the two recent testing results has a confidence of 95%, ensuring that the parameters of the actor are updated in a positive direction. Detailed algorithm is depicted in Algorithm 2.

Algorithm 2 Adaptive Critic Training

Input Batch size B, Training epoch E_1, E_2, Number of sampling N

```
1: Initialize actor params $\theta$
2: Initialize adaptive critic params $\theta^C \leftarrow \theta$
3: for epoch $= 1, \ldots, \mathrm{E}_1 + \mathrm{E}_2$ do
4:   Generate training set $M_e$
5:   Generate validation set $M_t$
6:   for $n = 1, \ldots, N$ do
7:     $G_i \sim \mathrm{SampleInput}(M_e)$ for $i \in \{1, \ldots, B\}$
8:     $\pi_i \sim \mathrm{SampleSolution}\left(p_\theta(\cdot|G_i)\right)$ for $i \in \{1, \ldots, B\}$
9:     if $e \le E_1$ then
10:      $b_i \leftarrow V_{\theta^{\mathrm{C}}}(G_i)$ for $i \in \{1, \ldots, B\}$
11:      $g_\theta \leftarrow \frac{1}{B}\sum_{i=1}^{B}\left(L\left(\pi_i|G_i\right) - b_i\right)\nabla_\theta \log p_\theta\left(\pi_i|G_i\right)$
12:      $g_{\theta^{\mathrm{C}}} \leftarrow \frac{1}{B}\sum_{i=1}^{B}\|L(\pi_i|G_i) - b_i\|_2^2$
13:      $\theta \leftarrow \mathrm{ADAM}(\theta, \mathrm{g}_\theta)$
14:      $\theta^{\mathrm{C}} \leftarrow \mathrm{ADAM}(\theta^{\mathrm{C}}, \mathrm{g}_{\theta^{\mathrm{C}}})$
15:    else
16:      $\pi_i^C \sim \mathrm{Greedy}\left(p_{\theta^{\mathrm{C}}}(\cdot|G_i)\right)$ for $i \in \{1, \ldots, B\}$
17:      $b_i \leftarrow L(\pi_i^{\mathrm{C}}|G_i)$ for $i \in \{1, \ldots, B\}$
18:      $g_\theta \leftarrow \frac{1}{B}\sum_{i=1}^{B}\left(L\left(\pi_i|G_i\right) - b_i\right)\nabla_\theta \log p_\theta\left(\pi_i|G_i\right)$
19:      $\theta \leftarrow \mathrm{ADAM}(\theta, \mathrm{g}_\theta)$
20:      if T-TEST$(L\left(\pi|G_{test}\right), b_i) < 5\%$ then
21:        $\theta^C \leftarrow \theta$
22:      end if
23:    end if
24:  end for
25: end for
```

1) Simple Network Architecture: We explain the architecture of the simple network now, which aims to mapping the input graph into a baseline $b(G)$. The network includes three modules: 1) an encoder which has the same structure as the actor network, 2) a Recurrent Neural Network (RNN) block and 3) a two-layer linear network. The encoder is designed to embed the information into static state, dynamic state and hidden state h. Similar to Bello [18], the RNN block repeats to update the hidden state h by glimpsing at the static and dynamic state. After the input is embedded by the encoder and sent to a block, a two-layer linear network is employed to calculate the baseline.

V. Experiments

We conduct extensive computational experiments to investigate the performance of the proposed DRL model and two-stage framework, in terms of solution quality and computational time. We tested on two kinds of routing problems — VRP and VRPTW, with problem sizes of 20, 50 and 100, respectively. The test sets are generated based on the routing simulator, named cohort-20, cohort-50, cohort-100 for VRP and cohort-20TW, cohort-50TW and cohort-100TW for VRPTW. These cohorts are used only in the test procedure to ensure reliable comparison. Computational experiments are conducted on a server with a configuration of single GeForce GTX 1080 Ti and 4 Intel(R) Xeon(R) CPUs at 2.40GHz with 12 cores. We first present the experimental details in Section A. In Section B and C, we analyze the performance of the DRL model alone and two-stage framework, compared with other baselines. Finally we investigate the convergence rate and solution quality influenced by the proposed adaptive critic in Section D.

A. Experimental Details

1) Testing Strategies: To further improve the results, we use two approaches during the testing process, greedy and beam search [34]. For each step, the greedy strategy selects the highest probability customer point as the action. However, given beam width k, beam search stores top-k choices at a time and outputs the final results according to the maximum probability of the sequence.

2) Baselines: To evaluate the proposed DRL model, we compare the DRL model against different baselines: construction algorithms, ant colony optimization (ACO) [40], and the DRL method presented by Nazari *et al.* [20]. Construction algorithms include Clarke-Wreight Savings [41], Random Insertion, Nearest Insertion [42] and Nearest Neighbor [43].

To investigate the effectiveness of the proposed two-stage framework, we compute the results of the above baselines as initial solutions for local search. Google OR-Tools and LNS [12] are considered as the representatives of local search algorithms. The results with different initial solutions of each local search algorithm are presented in the second stage of comparison.

3) Data Generation Parameters: In this section, we describe the parameters of data generation in the routing simulator. For the VRPTW, the coordinate of the depot and customer points (x_i, y_i) are generated from within a range of $[0, 1] \times [0, 1]$. Both the service time $t_{s,i}$ and the length of time window Δt are randomly generated from $[0.15, 0.2]$. The whole time interval is set as $T = 4.6$ and the vehicle speed is set as $v = 1$. The integer demand q_i of a customer point is generated from 1 to 9. We set the maximize load $Q = 30$ when $n = 20$, $Q = 40$ when $n = 50$, $Q = 50$ when $n = 100$, refering to Nazari [20].

4) Hyperparameters: As for the DRL model, the encoder embeds the graph information into a 128-dimension vector, while the decoder is a single LSTM layer with 128 hidden units, of which the dropout is 0.1. We take batch size as 256 and 200 epochs for training, randomly sampling 512000 instances in each epoch. We clip the L2 norm of gradient to 2.0. After fine tuning, we set E_1 as 20, which will be described in Section D. The model is trained with Adam optimizer [44] with a learning rate of 10^{-4}. When testing, we use beam search (BS) of size 10.

Since Google OR-Tools is functioned as one of the local search algorithms, the *local_search_metaheuristic* parameter is set to *GUIDED_LOCAL_SEARCH*, and the time limit for each instance is $30 * n\ ms$, where n is the problem size of this instance. For LNS algorithm, the number of customers to remove ($toRemove$) and the maximum number of iterations ($Iter$) vary with the problem sizes. The parameters are set as ($toRemove = 3, Iter = 150$) at problem size 20, ($toRemove = 6, Iter = 200$) at size 50, and ($toRemove = 10, Iter = 250$) at size 100. Except that LNS is implemented in Matlab, other code is conducted in Python.

TABLE II
RESULTS WITHOUT LOCAL SEARCH

Method	$n = 20$			$n = 50$			$n = 100$		
	Aver.	Gap	Time[1]	Aver.	Gap	Time	Aver.	Gap	Time
VRP									
Clarke-Wreight Saving (CW)	7.09	15.28%	7.56	12.40	18.21%	49.92	18.73	9.47%	278.71
Nearest Insertion (NI)	8.59	39.67%	74.50	17.78	69.49%	201.39	30.95	80.89%	1379.89
Random Insertion (RI)	12.24	99.02%	60.05	29.24	178.74%	104.81	57.08	233.61%	678.69
Nearest Neighbor (NN)	7.51	22.11%	12.90	13.01	24.02%	18.19	19.45	13.68%	59.99
Ant Colony Optimization (ACO)	7.18	16.75%	8820.47	12.62	20.31%	44327.63	19.22	12.33%	153305.55
Nazari *et al.* (greedy)	6.34	3.09%	10.12	10.79	2.86%	11.01	17.39	1.64%	14.27
Nazari *et al.* (10)	6.16	0.16%	10.44	10.57	0.76%	17.69	17.13	0.00%	37.39
DRL (greedy)	6.30	2.44%	9.36	10.68	1.81%	11.27	17.44	1.93%	14.67
DRL (10)	**6.15**	0.00%	10.35	**10.49**	0.00%	17.63	**17.11**	0.00%	35.74
VRPTW									
Gurobi	**8.80**	0.00%	113.77	**16.53**	0.00%	667.88	**26.29**	0.00%	3224.22
Clarke-Wreight Saving (CW)	10.77	22.39%	9.09	22.54	36.36%	69.08	38.54	46.60%	420.71
Nearest Insertion (NI)	10.99	24.89%	21.41	22.93	38.72%	206.77	40.12	52.61%	1460.77
Random Insertion (RI)	12.18	38.41%	19.66	27.10	63.94%	218.44	49.65	88.86%	1522.30
Nearest Neighbor (NN)	13.16	49.55%	5.62	27.31	65.21%	18.57	45.31	76.15%	60.32
Ant Colony Optimization (ACO)	10.68	21.36%	2825.00	22.75	37.63%	14435.00	38.25	45.49%	47734.00
Nazari *et al.* (greedy)	9.58	8.86%	9.76	18.58	12.40%	12.18	30.37	15.52%	17.15
Nazari *et al.* (10)	9.21	4.66%	9.08	18.05	9.20%	20.15	29.87	13.62%	41.71
DRL (greedy)	9.51	8.07%	11.16	18.40	11.31%	12.05	30.03	14.23%	16.60
DRL (10)	**9.16**	4.09%	9.12	**17.90**	8.29%	16.08	**29.50**	12.21%	32.55

[1] The running time is calculated by running 1280 instances of a cohort.

5) Running Time: The running time is hard to compare but important. Since the model is end-to-end, the training time is not considered when analyzing the overall performance of the proposed methods. The running time of each method is calculated by running 1,280 instances in total. For each two-stage algorithm, the running time of the overall process is listed in the table. All the reported calculation time is in seconds.

B. Results Without Local Search

The proposed DRL model is compared with the baselines mentioned in Section A for VRP and VRPTW respectively in Table II. The first column reports the baseline adopted to solve the corresponding cohorts, in which "DRL" is the DRL model proposed in Section 4. Columns with headers "Aver", "Gap" and "Time" report the average tour lengths, gap and total calculation time on different cohorts. For the VRP, the best result is set as gap (=0%) since the optimal results is hard to obtain even for cohort-20, while the optimal solutions calculated by Gurobi is set as gap (=0%) for the VRPTW.

For both VRPs, the proposed DRL model can achieve better results than construction algorithms with significantly shorter computation time even in large problem size. Most importantly, our results outperform work of Nazari [20] both in greedy and beam search (beam size = 10) mode for all cohorts. The reason is that the highly effective adaptive critic is capable of giving better assessment of instances when training the model, compared with the normal critic.

C. Results With Local Search

We compare the performance on enhancing the Google OR-Tools and LNS by our proposed DRL model with that by other baselines mentioned above. Similar to the results without local search, Table III and Table IV report the average tour lengths, gap and total calculation time on each cohort of Google OR-Tools and LNS with different initial solutions, respectivley.

Using Google OR-Tools in the local search stage, the results are presented in Table III. We can observe that results enhanced by DRL methods, including our and Nazari's model, significantly outperform that enhanced by construction algorithms both in solution quality and calculation time. Taking cohort-100TW as an example, the results enhanced by DRL models achieve average solution quality at 27.37 with an average solution time of 486.40s, while the results enhanced by construction algorithms get the best solution quality at 27.67, with a solution time of 2337.23s. We focus on the comparison between OR-Tools with Nazari's DRL model and with our DRL model, in terms of solution quality and calculation time. Taking solution quality into consideration, OR-Tools with our DRL model becomes more competitive for the VRP at all problem size. While for the VRPTW, the proposed method only achieves better results for cohort-100TW and gets the same solution quality for cohort-20TW and cohort-50TW. For the calculation time, note that these two DRL models have the same characteristics of high inference speed, their computational time is similar.

We also test our framework by using LNS in the local search stage. The results are summarized in Table IV. We can see that the proposed framework (DRL+LNS) significatly surpasses others for all cohorts and gets much closer to optimal results (from around 7% to 4% for cohort-100TW and about 3% improvement of cohort-100). The results imply that the DRL model can enhance the local search method

TABLE III
RESULTS WITH GOOGLE OR-TOOLS

Method	$n = 20$			$n = 50$			$n = 100$		
	Aver.	Gap	Time	Aver.	Gap	Time	Aver.	Gap	Time
VRP									
OR-Tools	6.00	1.01%	31.70	10.55	4.87%	83.64	16.50	5.97%	193.81
CW+OR-Tools	6.57	10.61%	24.64	11.47	14.02%	137.36	17.46	12.14%	612.09
NI+OR-Tools	7.40	24.58%	95.87	14.33	42.45%	336.70	23.84	53.11%	2100.34
RI+OR-Tools	9.22	55.22%	83.53	20.04	99.20%	264.30	36.84	136.61%	1441.75
NN+OR-Tools	6.81	14.65%	30.65	11.81	17.40%	108.56	17.83	14.52%	459.67
Nazari *et al.* (greedy)+OR-Tools	6.00	1.01%	23.48	10.20	1.39%	77.78	15.71	0.90%	333.16
Nazari *et al.* (10)+OR-Tools	5.96	0.34%	22.99	10.13	0.70%	79.68	15.68	0.71%	342.86
DRL (greedy)+OR-Tools	5.99	0.84%	22.42	10.17	1.09%	76.38	15.59	0.13%	338.98
DRL (10)+OR-Tools	**5.95**	0.17%	23.52	**10.11**	0.50%	80.42	**15.58**	0.06%	340.74
VRPTW									
Gurobi	**8.80**	-	113.77	**16.53**	-	667.88	**26.29**	-	3224.22
OR-Tools	9.06	2.95%	7692.30	18.31	10.77%	19245.65	32.42	23.32%	38556.57
CW+OR-Tools	9.03	2.61%	30.44	17.35	4.96%	199.56	28.23	7.38%	995.87
NI+OR-Tools	8.95	1.70%	45.88	17.08	3.33%	353.50	27.68	5.25%	2151.88
RI+OR-Tools	8.96	1.82%	45.63	17.09	3.39%	381.31	27.67	5.25%	2337.23
NN+OR-Tools	8.98	2.05%	32.82	17.23	4.23%	179.72	27.80	5.74%	789.09
Nazari *et al.* (greedy)+OR-Tools	8.94	1.59%	30.40	17.02	2.96%	119.34	27.47	4.49%	488.85
Nazari *et al.* (10)+OR-Tools	**8.89**	1.02%	27.52	**16.94**	2.48%	122.45	27.37	4.11%	512.98
DRL (greedy)+OR-Tools	8.94	1.59%	31.47	17.02	2.96%	123.01	27.38	4.15%	487.81
DRL (10)+OR-Tools	**8.89**	1.02%	34.53	**16.94**	2.48%	130.76	**27.25**	3.65%	**455.98**

TABLE IV
RESULTS WITH LARGE NEIGHBORHOOD SEARCH (LNS)

Method	$n = 20$			$n = 50$			$n = 100$		
	Aver.	Gap	Time	Aver.	Gap	Time	Aver.	Gap	Time
VRP									
CW+LNS	6.03	1.51%	20.41	10.40	3.38%	180.39	16.05	3.08%	1106.69
NI+LNS	6.26	5.39%	87.32	11.01	9.44%	332.922	17.38	11.62%	2205.87
RI+LNS	6.24	5.05%	72.87	10.91	8.45%	235.67	17.25	10.79%	1512.24
NN+LNS	6.10	2.69%	25.86	10.53	4.67%	149.23	16.18	3.91%	882.48
Nazari *et al.* (greedy)+LNS	5.96	0.34%	22.93	10.14	0.80%	144.91	15.69	0.77%	844.04
Nazari *et al.* (10)+LNS	**5.94**	0.00%	23.36	10.09	0.30%	150.71	15.63	0.39%	870.02
DRL (greedy)+LNS	5.96	0.34%	22.23	10.11	0.50%	145.283	15.60	0.19%	843.03
DRL (10)+LNS	**5.94**	0.00%	23.27	1**0.06**	0.00%	151.75	**15.57**	0.00%	857.40
VRPTW									
Gurobi	**8.80**	-	113.77	**16.53**	-	667.88	**26.29**	-	3224.22
CW+LNS	8.99	2.16%	23.31	17.20	4.07%	227.59	27.98	6.43%	1396.04
NI+LNS	9.13	3.75%	35.54	17.54	6.11%	362.72	28.46	8.26%	2446.80
RI+LNS	9.18	4.32%	33.73	17.57	6.29%	375.94	28.57	8.67%	2496.17
NN+LNS	9.05	2.84%	19.87	17.34	4.90%	177.64	28.22	7.34%	1042.94
Nazari *et al.* (greedy)+LNS	8.93	1.48%	23.60	17.03	3.01%	168.432	27.52	4.68%	986.72
Nazari *et al.* (10)+LNS	8.90	1.14%	23.15	16.97	2.67%	175.74	27.49	4.56%	1005.84
DRL (greedy)+LNS	8.93	1.48%	25.32	17.00	2.84%	182.17	27.52	4.68%	978.42
DRL (10)+LNS	**8.89**	1.02%	23.16	**16.94**	2.48%	175.20	**27.44**	4.37%	1003.36

like Google OR-Tools and LNS both in solution quality and calculation time. Moreover, the competitive calculation speed makes the proposed method having the potential to be enlarged to accommodate the growth of problem size.

We present a more detailed comparison about solution quality of VRPTW utilizing Google OR-Tools as local search algorithm. Figure 3 shows the result comparison for VRPTW cohorts in terms of their winning rate on instance level. Each table represents the comparison result of the problem with a specific size. The value in the table is the proportion of total 1,280 instances in which the method on the row outperforms those on the column. For example, the value at the coordinate (1,2) in table (a), means 43.2% of total instances that OR-Tools outperforms CW+OR-Tools on cohort-20TW. We can observe that DRL(10)+OR-Tools outperforms OR-Tools with other initial solutions, e.g., at a

	OR-Tools	CW+OR-Tools	NI+OR-Tools	RI+OR-Tools	NN+OR-Tools	Na+OR-Tools	Na10+OR-Tools	RL+OR-Tools	RL10+OR-Tools
OR-Tools	0	43.2	37.4	36.3	40.7	26.3	29.1	29.1	28.2
CW+OR-Tools	56.9	0	27.3	28.5	28.9	22.7	19.1	20.2	16
NI+OR-Tools	62.7	37.8	0	25.9	34.6	30.4	23.8	29.6	24.3
RI+OR-Tools	63.8	39	23.7	0	35.7	29.8	23.9	27.3	23.1
NN+OR-Tools	59.4	34.1	28.9	29.5	0	24.8	19.8	23	18.7
Na+OR-Tools	65.5	37.3	32.3	33.2	33.8	0	9.77	16.8	13.9
Na10+OR-Tools	71	42.1	36.9	37.7	38.8	20.9	0	23	15.9
RL+OR-Tools	66.9	37.8	32.7	34.5	35.3	19.5	15.5	0	9.38
RL10+OR-Tools	71.9	43	37.4	38.7	40.2	26.2	18.3	20	0

(a) Comparison for Size 20

	OR-Tools	CW+OR-Tools	NI+OR-Tools	RI+OR-Tools	NN+OR-Tools	Na+OR-Tools	Na10+OR-Tools	RL+OR-Tools	RL10+OR-Tools
OR-Tools	0	13.4	5.78	5.16	8.05	3.67	3.2	4.92	3.36
CW+OR-Tools	86.6	0	32.8	33.9	41.7	27.8	21.9	27.5	22.3
NI+OR-Tools	94.2	65.9	0	50.5	59.7	43.4	38.7	45.3	38.2
RI+OR-Tools	94.8	65.6	47.2	0	58.2	44	37	43.9	36.9
NN+OR-Tools	92	56.8	38.4	40.1	0	34.1	29.2	35.9	27.9
Na+OR-Tools	96.4	69.9	53.8	53.9	62.6	0	27.3	44.9	35.6
Na10+OR-Tools	96.9	76.3	58.8	61.1	68.4	45	0	51.6	43
RL+OR-Tools	95.2	70.8	52.6	54.1	62.3	44.4	37.7	0	24.7
RL10+OR-Tools	96.7	75.7	58.6	60.8	69.9	54.3	45.9	42.9	0

(b) Comparison for Size 50

	OR-Tools	CW+OR-Tools	NI+OR-Tools	RI+OR-Tools	NN+OR-Tools	Na+OR-Tools	Na+OR-Tools	RL+OR-Tools	RL10+OR-Tools
OR-Tools		1.02	0.16	0.08	0.39	0.08	0.08	0	0
CW+OR-Tools	98.9	0	24.5	24.5	39.4	17.4	12.5	13.4	9.38
NI+OR-Tools	99.8	75.4	0	49.5	65.9	37.2	31.7	31.9	26.1
RI+OR-Tools	99.9	75.5	50.5	0	65	36.9	32.3	30.9	25.1
NN+OR-Tools	99.6	60.6	34.1	34.9	0	22.7	18.4	18.8	13.7
Na+OR-Tools	99.9	82.4	62.8	63.1	77.3	0	41.6	44.1	35.9
Na10+OR-Tools	99.9	87.5	68.3	67.7	81.6	55.5	0	50.9	43.4
RL+OR-Tools	99.9	86.6	68	69.1	81.3	55.2	49	0	37.3
RL10+OR-Tools	100	90.7	76.1	75	86.4	64.1	56.6	56.9	0

(c) Comparison for Size 100

Fig. 3. The proportion of all instances which the result of the method on the row outperforms the method on the column.

TABLE V

RESULTS OF DIFFERENT E_t

E_t%	0	5	10	20	30
Aver.	9.429	9.418	9.412	9.415	9.424
E_t%	40	60	80	100	
Aver.	9.422	9.425	9.433	9.429	

winning rate above 56% in cohort-100TW. From this perspective, the proposed method achieves consistent performance across different instance to exceed other approaches, rather than a stochastic event.

D. Adaptive Critic

As mentioned above, a suitable baseline helps to reduce the training variance. In this paper, we introduce the adaptive critic which is an combination of self critic [22] and normal critic, conducted by simple network structure. One hyperparameter of the adaptive critic is the transition point, determining when to change the network structure from normal critic to self cirtic. It is defined as the percentage of epochs at the time of the transition, and symbolized as E_t. Note that when E_t equals 100%, the adaptive critic degenerates into a simple network; when E_t equals 0%, it is exactly the same as the self critic [22]. Table V reports a comparison of the average tour length and training time per epoch on different set of E_t, when testing on cohort-20. We can observe that the best solution is achieved when E_t equals 10%, and the self critic and normal critic have a consistent result under this situation.

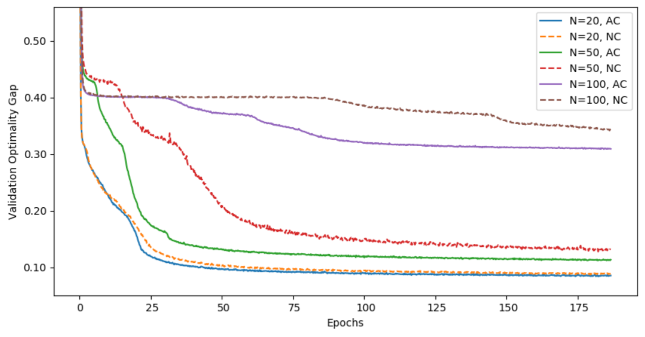


Fig. 4. Different convergence rates using different critics.

The convergence rates of the DRL model with adaptive critic and normal critic are compared in Figure 4. The Y-axis is the validation optimality gap and the X-axis is the training epochs of corresponding methods. It can be observed that the convergence rates and the solution quality of the DRL model are largely improved with adaptive critic, since this design makes use of the actor with inference mode to estimate the baseline after training for several epochs. On the contrary, normal critic is likely to be confused with different instances, and the self critic does not perform well at the beginning since the actor is poorly trained at start.

VI. CONCLUSION

In this paper, we first propose a novel DRL model for solving VRPs, consisting of an actor to generate routing solutions and an adaptive critic to train the parameters of the actor. Like previous work [20], the actor is implemented by the attention mechanism together with graph embbeding to construct vehicle routing tours iteratively. To reduce the training variance, an adaptive critic is devised to optimize the parameters of the actor. In addition, a routing simulator is designed to help the training and evaluation of the DRL model. Moreover, a two-stage framework consisting of DRL and local search is proposed for solving VRPs. The DRL method is capable of finding out strong initial solutions for local search to enhance the solution quality, achieving near optimal solution.

The simulation results show that the proposed DRL model outperforms the construction algorithms and other DRL model, and the adaptive critic can effectively accelerate the convergence rate and improve the solution quality. We conduct comprehensive experiments to evaluate the performance of the framework, using the Google OR-Tools and LNS as classical local search algorithms. Experimental results indicate that the proposed framework can generate outstanding routing strategies, comparing to local search with initial solutions produced by other construction algorithms. Besides the improvement of solution quality, the route generation of the proposed framework costs short computing time, making the framework promising for VRPs with large sizes. Furthermore, using DRL results as an initial solution can be a general approach to combining DRL with heuristic methods to solve combinatorial optimization problems.

The future research can be highlighted as follows. On one hand, adavanced techniques of reinforcement learning, e.g., multi-header attention, can be employed to improve the performance of the proposed DRL model. On the other, the proposed two-stage framework can be applied to more complicated routing problems and other combinatorial optimization problems.

REFERENCES

[1] L. Zhu, F. R. Yu, Y. Wang, B. Ning, and T. Tang, “Big data analytics in intelligent transportation systems: A survey,” *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 1, pp. 383–398, Jan. 2019.

[2] M. Joerss, J. Schröder, F. Neuhaus, C. Klink, and F. Mann, *Parcel Delivery: The Future of Last Mile*. New York, NY, USA: McKinsey & Company, 2016.

[3] O. Bräysy and M. Gendreau, “Vehicle routing problem with time windows, part I: Route construction and local search algorithms,” *Transp. Sci.*, vol. 39, no. 1, pp. 104–118, Feb. 2005.

[4] L. R. Dopson and D. K. Hayes, *Food and Beverage Cost Control Study Guide*. Hoboken, NJ, USA: Wiley, 2015.

[5] G. Kim, Y.-S. Ong, C. K. Heng, P. S. Tan, and N. A. Zhang, “City vehicle routing problem (city VRP): A review,” *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 4, pp. 1654–1666, Aug. 2015.

[6] G. Perboli and M. Rosano, “Parcel delivery in urban areas: Opportunities and threats for the mix of traditional and green business models,” *Transp. Res. Part C: Emerg. Technol.*, vol. 99, pp. 19–36, Feb. 2019.

[7] Y. Li, F. Chu, C. Feng, C. Chu, and M. Zhou, “Integrated production inventory routing planning for intelligent food logistics systems,” *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 3, pp. 867–878, Mar. 2019.

[8] B. D. Brouer, J. F. Alvarez, C. E. M. Plum, D. Pisinger, and M. M. Sigurd, “A base integer programming model and benchmark suite for liner-shipping network design,” *Transp. Sci.*, vol. 48, no. 2, pp. 281–312, May 2014.

[9] D. C. Paraskevopoulos, G. Laporte, P. P. Repoussis, and C. D. Tarantilis, “Resource constrained routing and scheduling: Review and research prospects,” *Eur. J. Oper. Res.*, vol. 263, no. 3, pp. 737–754, 2017.

[10] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by simulated annealing,” *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[11] F. Glover, “Tabu search—Part I,” *ORSA J. Comput.*, vol. 1, no. 3, pp. 190–206, 1989.

[12] P. Shaw, “Using constraint programming and local search methods to solve vehicle routing problems,” in *Proc. Int. Conf. Princ. Pract. Constraint Program.*, 1998, pp. 417–431.

[13] P. Grangier, M. Gendreau, F. Lehuédé, and L.-M. Rousseau, “An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization,” *Eur. J. Oper. Res.*, vol. 254, no. 1, pp. 80–91, Oct. 2016.

[14] I. Žulj, S. Kramer, and M. Schneider, “A hybrid of adaptive large neighborhood search and tabu search for the order-batching problem,” *Eur. J. Oper. Res.*, vol. 264, no. 2, pp. 653–664, Jan. 2018.

[15] M. Zirour, “Vehicle routing problem: Models and solutions,” *J. Qual. Meas. Anal.*, vol. 4, no. 1, pp. 205–218, 2008.

[16] D. Silver *et al.*, “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, p. 484, 2016.

[17] Z. Shen, K. Yang, W. Du, X. Zhao, and J. Zou, “DeepAPP: A deep reinforcement learning framework for mobile application usage prediction,” in *Proc. 17th Conf. Embedded Netw. Sensor Syst.*, Nov. 2019, pp. 153–165.

[18] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, “Neural combinatorial optimization with reinforcement learning,” 2016, *arXiv:1611.09940*. [Online]. Available: http://arxiv.org/abs/1611.09940

[19] H. Dai, B. Dai, and L. Song, “Discriminative embeddings of latent variable models for structured data,” in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2702–2711.

[20] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takac, “Reinforcement learning for solving the vehicle routing problem,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 9839–9849.

[21] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, “Learning combinatorial optimization algorithms over graphs,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6348–6358.

[22] W. Kool and M. Welling, “Attention, learn to solve routing problems!” in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–25.

[23] J. J. Q. Yu, W. Yu, and J. Gu, “Online vehicle routing with neural combinatorial optimization and deep reinforcement learning,” *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 10, pp. 3806–3817, Oct. 2019.

[24] Z. Li, Q. Chen, and V. Koltun, “Combinatorial optimization with graph convolutional networks and guided tree search,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 539–548.

[25] A. Vaswani *et al.*, “Attention is all you need,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[26] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Mach. Learn.*, vol. 8, nos. 3–4, pp. 229–256, May 1992.

[27] G. Laporte, “The vehicle routing problem: An overview of exact and approximate algorithms,” *Eur. J. Oper. Res.*, vol. 59, no. 3, pp. 345–358, Jun. 1992.

[28] G. Laporte, M. Gendreau, J.-Y. Potvin, and F. Semet, “Classical and modern heuristics for the vehicle routing problem,” *Int. Trans. Oper. Res.*, vol. 7, nos. 4–5, pp. 285–300, Sep. 2000.

[29] D. Pisinger and S. Ropke, “Large neighborhood search,” in *Handbook of Metaheuristics*. Boston, MA, USA: Springer, 2010, pp. 399–419.

[30] W. Gu, D. Cattaruzza, M. Ogier, and F. Semet, “Adaptive large neighborhood search for the commodity constrained split delivery VRP,” *Comput. Oper. Res.*, vol. 112, 2019, Art. no. 104761.

[31] J. Hof, M. Schneider, and D. Goeke, “Solving the battery swap station location-routing problem with capacitated electric vehicles using an AVNS algorithm for vehicle-routing problems with intermediate stops,” *Transp. Res. B, Methodol.*, vol. 97, pp. 102–112, Mar. 2017.

[32] T. Erdelić and T. Carić, “A survey on the electric vehicle routing problem: Variants and solution approaches,” *J. Adv. Transp.*, vol. 2019, pp. 1–48, May 2019.

[33] J. J. Hopfield and D. W. Tank, “Neural computation of decisions in optimization problems,” *Biological*, vol. 52, no. 3, pp. 141–152, 1985.

[34] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.

[35] O. Vinyals, M. Fortunato, and N. Jaitly, “Pointer networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2692–2700.

[36] P. Augerat, J. M. Belenguer, E. Benavent, A. Corberán, and D. Naddef, “Separating capacity constraints in the CVRP using tabu search,” *Eur. J. Oper. Res.*, vol. 106, nos. 2–3, pp. 546–557, Apr. 1998.

[37] M. M. Solomon, “Algorithms for the vehicle routing and scheduling problems with time window constraints,” *Oper. Res.*, vol. 35, no. 2, pp. 254–265, Apr. 1987.

[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[39] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, "Self-critical sequence training for image captioning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 7008–7024.

[40] M. Dorigo and M. Birattari, "Ant colony optimization," in *Encyclopedia of Machine Learning*. Boston, MA, USA: Springer, 2011, pp. 36–39.

[41] G. Clarke and J. W. Wright, "Scheduling of vehicles from a central depot to a number of delivery points," *Oper. Res.*, vol. 12, no. 4, pp. 568–581, Aug. 1964.

[42] S. Joshi and S. Kaur, "Nearest neighbor insertion algorithm for solving capacitated vehicle routing problem," in *Proc. Int. Conf. Comput. Sustain. Global Develop.*, 2015, pp. 86–88.

[43] R. He, W. Xu, Y. Wang, and W. Zhan, "A route-nearest neighbor algorithm for large-scale vehicle routing problem," in *Proc. 3rd Int. Symp. Intell. Inf. Technol. Secur. Inform.*, Apr. 2010, pp. 390–393.

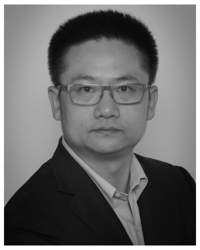
[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

Jiuxia Zhao received the bachelor's degree in the field of management from Xi'an Jiaotong University, Xi'an, China, in 2018, where she is currently pursuing the master's degree with the School of Automation Science and Technology.

Her research interests include the data mining and deep reinforcement learning for combinatorial optimization problems, especially for vehicle routing problems.

Minjia Mao received the B.Eng. degree from Tsinghua University in 2018. He is currently pursuing the master's degree with the School of Mathematics and Statistics, Xi'an Jiaotong University.

His research interests include deep learning and reinforcement learning, especially on applying learning methods for optimization problems.



Xi Zhao (Senior Member, IEEE) received the Ph.D. degree (Hons.) in computer science from the École Centrale de Lyon, Lyon, France, in 2010.

He is currently a Professor with the School of Management, Xi'an Jiaotong University. His research interests include deep reinforcement learning, behavior analysis, and big data. He has published over 70 articles indexed by SCI/ EI, and has been PI for more than 15 grants in these fields.

Jianhua Zou (Member, IEEE) received the Ph.D. degree with the Institute of Plasma Physics, Chinese Academy of Sciences, Beijing, China, in 1991.

He is the Vice Dean of the School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an, China. He conducted the post-doctoral research with the Huazhong University of Science and Technology, Wuhan, China, and Xi'an Jiaotong University from 1991 to 1993 and from 1993 to 1995, respectively. He became a Professor with the System Engineering Institute, Xi'an Jiaotong University. He was the Principle Investigator of two projects funded by the National Science Foundation of China, about ten key projects supported by industry and military. As a Senior Visiting Scholar, he has established partnerships with Eindhoven University, Eindhoven, The Netherlands, and Philips Company, Amsterdam, The Netherlands. He has published over 60 academic articles. His current research interests include intelligent control, computer vision and pattern recognition, data mining, and knowledge discovery.