Deep Dynamic Attention Model with Gate Mechanism for Solving Time-dependent Vehicle Routing Problems

Anonymous authors
Paper under double-blind review

Abstract

Vehicle routing problems (VRPs) are a type of classical combinatorial optimization problems widely existing in logistics and transportation operations. There has been an increasing interest to use deep reinforcement learning (DRL) techniques to tackle VRPs, and previous DRL-based studies assumed time-independent travel times between customers. However, travel times in real-world road networks are time-varying, which need to be considered in practical VRPs. We thus propose a Deep Dynamic Attention Models with Gate Mechanisms (DDAM-GM) to learn heuristics for time-dependent VRPs (TDVRPs) in real-world road networks. It extracts the information of node location, node demand, and time-varying travel times between nodes to obtain enhanced node embeddings through a dimension-reducing MHA layer and a synchronous encoder. In addition, we use a gate mechanism to obtain better context embedding. On the basis of a 110-day travel time dataset with 240 time periods per day from an urban road network with 408 nodes and 1250 directed links, we conduct a series of experiments to validate the effectiveness of the proposed model on TDVRPs without and with consideration of time windows, respectively. Experimental results show that our model outperforms significantly two state-of-the-art DRL-based models.

1 Introduction

The vehicle routing problem (VRP) is a classical combinatorial optimization problem and one of the most widely investigated problems in transportation science and logistics. VRPs aim to determine the set of routes for a fleet of vehicles to serve a given set of customers in a road network so that one or more objectives can be optimized without violating constraints imposed. Travel speeds in real-world road network are time-dependent. That is, the travel speeds (time) on a road link are different in different time periods. VRPs with time-dependent travel speeds (time) are called as time-dependent VRPs (TDVRPs). A practical TDVRP can be defined on a directed graph $\mathcal{G} = (\mathbb{V}, \mathbb{L})$, where $\mathbb{V} = \{0, \dots, V\}$ is a set of vertices (nodes), and $\mathbb{L} = \{1, \dots, L\}$ is the set of directed links connecting the nodes in $\mathbb{V}$. Node 0 is the depot at which K vehicles with capacity Q are based. Let $\mathbb{N} = \{1, \dots, N\} (\mathbb{N} \subset \mathbb{V})$ denote the set of customers nodes to be visited (served). Each node $i \in \mathbb{V}$ is associated with a feature vector consisting of the horizontal coordinate x_i^h, the vertical coordinate x_i^v and the demand x_i^d of the node, i.e., $\boldsymbol{x}_i = (x_i^h, x_i^v, x_i^d)$. Each customer node has a certain customer demand and the demands of other nodes are 0. That is, $x_i^d = 0$ for $i \in \mathbb{V} \backslash \mathbb{N}$. Let $T_{i,j,p}$ denote the travel time between nodes i and j $(i, j \in \mathbb{V})$ at time period p, and $\boldsymbol{T}_p$ denote the travel time matrix composed of all $T_{i,j,p}$. That is, $\boldsymbol{T}_p = \{T_{i,j,p}\}_{(V+1)\times(V+1)}$.

Techniques for solving VRPs can be classified roughly into traditional techniques and deep learning-based techniques. Traditional techniques include exact techniques (Dabia et al., 2013; Spliet et al., 2018; Lera-Romero et al., 2020), heuristics (Malandraki & Daskin, 1992; Kok et al., 2010; Huart et al., 2016), and metaheuristics (Donati et al., 2008; Rincon-Garcia et al., 2020; Gmira et al., 2021). Exact techniques can obtain the optimal solution in theory, but the (worst-case) computational complexity is exponential. It is very difficult (if not impossible) for these techniques to generate quickly effective solutions to practical VRPs. Some heuristic techniques can obtain feasible solutions quickly to VRPs. However, these techniques are usually problem-dependent and generate poor-quality solutions. Various metaheuristics have been widely used to handle VRPs in recent years because they are problem-independent and have the potentials of providing better solutions than heuristics. However, they are prone to getting stuck in local optima and cannot provide effective solutions to practical VRPs within a reasonable computation time.

In recent years, deep learning (DL) and deep reinforcement learning (DRL) techniques have attracted more and more optimization researchers' attention because it is promising to use neural networks to directly and quickly learn heuristics from data without any hand-engineered reasoning (Bengio et al., 2021). Some researchers have developed some DRL-based methods to solve effectively several travelling salesmen problems (TSPs) (Bello et al., 2016; Ma et al., 2019) and VRPs (Kool et al., 2018; Nazari et al., 2018), which used much less computation time to find near-optimal solutions than benchmarking techniques did. Although the great potential of using DRL for such combinatorial optimization problems as VRPs, related studies are still in their early stage. In the DRL field, TDVRPs in real road networks have not been investigated so far, which is the focus of this paper.

This paper investigate TDVRPs, without and with consideration of time windows respectively, based on a road network with 408 network nodes and 1250 road links of Chengdu City, China. We used a 110-day link travel speed dataset with 240 2-minute time periods per day to represent the time-dependency of travel speeds of the Chengdu road network. We develop a deep dynamic attention models with gate mechanisms (DDAM-GM) based on the MARDAM (Bono et al., 2020). The novelty of this model consists of three improvements. First, instead of using 3-dimensional model inputs in previous studies (Nazari et al., 2018; Kool et al., 2018; Bono et al., 2020), we propose 4-dimensional model inputs with travel time information to handle TDVRPs effectively, and develop a dimension-reducing MHA to convert reduce 4-dimensional model inputs to 3-dimensional node embeddings and extracts the static travel time information. Second, we propose a synchronous encoder for synchronous coding, so that the model can extract time-varying traveling time information. Third, a gate mechanism used by Parisotto et al. (2020) is introduced to our model to obtain the better context embedding.

We test our DDAM-GM on problem instance sets with different number of customers and the results show that the proposed model achieves the better performance than two learning-based baselines. The main contributions of this paper are as follows.
1) This paper is the first to address practical TDVRPs with time-varying travel time in real road network.
2) We propose a novel DRL model, DDAM-GM, to learn problem-solving heuristics, which can provide superior solutions to the investigate TDVRPs over two learning-based models.
3) We propose three improvements for learning-based models, which can improve effectively the performance of DRL models for TDVRPs.

2 RELATED WORK

A pioneering work by Vinyals et al. (2015) developed the Pointer Networks (PtrNets) to solve multiple combinatorial optimization problems (e.g., TSP) based on an encoder-decoder framework (Sutskever et al., 2014). The model was trained off-line and supervised by example solutions. Since then, DL applications in routing problems have attracted increasing attention (Bello et al., 2016; Kool et al., 2018; Bono et al., 2020).

Bello et al. (2016) extended PtrNets by using REINFORCE algorithm (Williams, 1992) to train PtrNets without supervised solutions, which is the first to introduce DRL to handling combinatorial optimization problems. The graph PtrNets developed by Ma et al. (2019) integrated a graph embedding layer and hierarchical RL (Haarnoja et al., 2018) into PtrNets, which extended the applications of PtrNets to large-scaled TSPs with time windows but increased the computation time heavily.

There has been an increasing interest to using DRL in tackling VRPs since they are extensions of TSPs. Some DRL-based models work like constructive heuristics, which construct each vehicle's route by starting with an emtpy route and adding new customer nodes to visit in turn until a complete route is formed. In these models, the new node is chosen out based on the attention values (i.e., selection probability) of available nodes in node decoding. Nazari et al. (2018) used element-wise projections to encode nodes instead of using LSTM (Hochreiter & Schmidhuber, 1997) in PtrNets to solve a capacitated vehicle routing problem (CVRP). Kool et al. (2018) proposed the Attetion Model (AM) by using a transformer model (Vaswani et al., 2017) for node encoding and a self-attention mechanism for node decoding, which exhibted superior performances over several benchmarking models (e.g., OR-Tools, PtrNets) on several routing problems. Duan et al. (2020) proposed a graph convolution network-based DRL model to effectively solve a practial CVRP in a real road network. These previous studies construct each vehile route in turn in optimization process. Following AM (Kool et al., 2018), Bono et al. (2020) developed a MARDAM model to solve a dynamic CVRP with stochastic customers based on manually generated travel speeds and problem sets, which integrate fleet state and fleet state representation modules into the AM model and construct multiple vehicle routes simultaneously .

Similar to improvement heuristics, some DRL-based models, integrating DRL with heurisitcs, iteratively immprove the solutions based on a given initial solution (Chen & Tian, 2019; Lu et al., 2019; Gao et al., 2020). Chen & Tian (2019) proposed a neural-based DRL model, NeuRewriter, by learning a policy to pick heuristics and rewrite the local components of the current solution to solve a CVRP. Following this study, Lu et al. (2019) proposed a transformer-based model by employing a rich set of improvement and perturbation operators to solve a CVRP . Similarly, Gao et al. (2020) designed a graph attention network-based DRL model to learn the local-search heuristics to handle CVRPs with and without time windows. However, these improvement-type DRL models were much more time-consuming in finding optimal solutions than constructive-type DRL models did.

In summary, previous studies were conducted usually based on manually generated problem sets and have not considered time-varying travel speeds in real road networks. It is thus worthy to explore novel and effective DRL models for TDVRPs in real road networks.

3 Deep Dynamic Attention Models with Gate Mechanisms

3.1 Model Overview

Our DDAM-GM is developed based on the MARDAM in Bono et al. (2020) by integrating three novel improvements with the MARDAM to handle TDVRPs effectively.

Solving a VRP can be modelled as a sequential Multi-agent Markov Decision Process (sMMDP) with T time steps (Bono et al., 2020). In each time step t, the current vehicle k selects an unserved customer π_t^k to visit. This step is repeated T times until a complete solution $\boldsymbol{\pi} = \left\{\boldsymbol{\pi}^1, \cdots, \boldsymbol{\pi}^k, \cdots, \boldsymbol{\pi}^K\right\} (1 \leq k \leq K)$ is formed, where $\boldsymbol{\pi}^k$ denotes the k^{th} vehicle's route. This procedure of constructing the solution $\boldsymbol{\pi}$ is called as an episode. The objective of learning in DDAM-GM is to obtain a policy p^k for each vehicle so that the sum of all routes' objective values (rewards) are optimized. We consider homogeneous vehicles only. We can thus simplify the learning objective by making all vehicles' policies share a same group of parameters θ. That is, we only need to learn a policy parameterized by θ and formulated in equation 1. The sMMDP can be solved based on the REINFORCE algorithm (Williams, 1992)

$$p_\theta(\boldsymbol{\pi} \mid s) = \prod_{t=1}^{T} p_\theta\left(\pi_t \mid s, \boldsymbol{\pi}_{1:t-1}\right) \tag{1}$$

Like the MARDAM, the DDAM-GM follows a similar encoder-decoder framework with the Multi-Head Attention (MHA) layer introduced by (Vaswani et al., 2017), which is designed for fully connected road networks in which any node pairs are connected. However, any road nodes in a real road network are only connected with several neighboring nodes. We thus convert the real road network to a fully connected directed road network consisting of only customer nodes by using the method in Huang et al. (2017) to find the travel paths with the shortest travel time between any two customer nodes. Moreover, the encoder in MARDAM only extracts the location and demand information of each road network, and cannot extract time-varying travel time information between road nodes. Intuitively, travel time information between the current node and other nodes are critical to choose next node to visit. Furthermore, the MARDAM cannot represent important relevant information (e.g., the local and global traffic information in the road network) that are critical to select next nodes to visit in TDVRPs.

We thus introduce three novel improvements to the MARDAM to overcome the drawbacks of the MARDAM for TDVRPs. First, we propose a dimension-reducing MHA layer, detailed in section 3.2, to extract travel time information in the encoder. Second, instead of using a one-time encoding process to obtain node embedding only once during encoding in MARDAM, we propose a synchronous encoder to perform an encoding process in each time step t and obtain time-varying node embedding in each encoding so that the time-varying travel time information in real road network can be extracted. Meanwhile, we mask the customer nodes visited in each encoding to improve the optimization performance. Third, we use the travel times from a vehicle's location to other nodes to represent the local traffic information of this vehicle, and use the travel times travelling between nodes of all customer node pairs in the road network to represent the global traffic information at a certain time. These traffic information and other relevant information are aggregated by the gate mechanism in Parisotto et al. (2020) to represent the fleet state in our DDAM-GM.

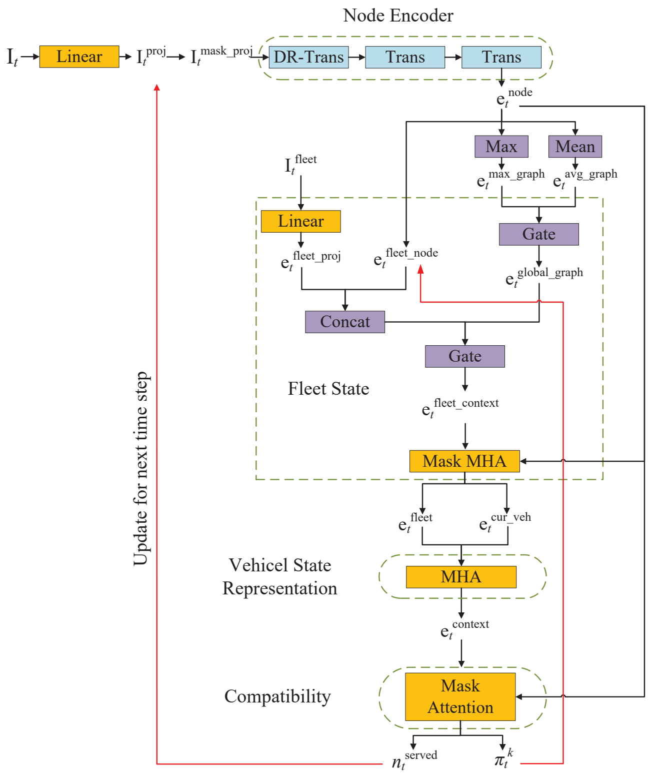

Figure 1: Architecture of one encoding-decoding procedure in the DDAM-GM

Figure 1 shows the architecture of one encoding-decoding procedure in the proposed DDAM-GM. The model input $\mathbf{I}_t$ represents TDVRPs-related feature values of the depot node and N customer nodes in the road network at the t^{th} encoding-decoding procedure. Let $T_{i,j,t}$ denote the travel time from node i to node j at time step t. We set $\boldsymbol{f}_{i,j,t} = (x_i^h, x_i^v, x_i^d, T_{i,j,t})$ and have $\mathbf{I}_t = \{\boldsymbol{f}_{i,j,t}\}_{B\times(N+1)\times(N+1)}$, where B represents the number of problem instances (i.e. batch size). Based on input $\mathbf{I}_t$, the initial node embeddings $\mathbf{I}_t^{\text{proj}}$ is obtained through an linear projection, which is then converted to $\mathbf{I}_t^{\text{mask\_proj}}$ by using a mask operation to mask all served nodes. The $\mathbf{I}_t^{\text{mask\_porj}}$ is then inputted into a "Node Encoder" consisting of a "DR-Trans" block and two same "Trans" block proposed in Vaswani et al. (2017), by which it is converted into node embeddings $\mathbf{e}_t^{\text{node}}$. The "DR-Trans" block is same to the "Trans" block except for using a DR-MHA layer, described in section 3.2, to replace the MHA layer in "Trans" block. We extract the node embedding of each vehicle's current location from $\mathbf{e}_t^{\text{node}}$, based on which the node embeddings $\mathbf{e}_t^{\text{fleet\_node}}$ of the fleet can be obtained as the local traffic information of all vehicle locations in the road network. Instead of averaging $\mathbf{e}_t^{\text{node}}$ as the graph embedding at the time step t in the AM (Kool et al., 2018), we take the average and the maximum of $\mathbf{e}_t^{\text{node}}$ to obtain the average graph embedding $\mathbf{e}_t^{\text{avg\_graph}}$ and the maximum graph embedding $\mathbf{e}_t^{\text{max\_graph}}$, and then obtain the global graph embedding $\mathbf{e}_t^{\text{global\_graph}}$ of the whole road network at the time step t by using a gate mechanism proposed in Parisotto et al.

(2020) to aggregate $\mathbf{e}_t^{\text{avg\_graph}}$ and $\mathbf{e}_t^{\text{max\_graph}}$. We then obtain the context embedding $\mathbf{e}_t^{\text{fleet\_context}}$ of a fleet by combining $\mathbf{e}_t^{\text{global\_graph}}$, $\mathbf{e}_t^{\text{fleet\_node}}$ and fleet state projection $\mathbf{e}_t^{\text{fleet\_proj}}$ together based on the same gate mechanism. $\mathbf{e}_t^{\text{fleet\_proj}}$ is determined by a linear projection of $\mathbf{I}_t^{\text{fleet}}$. $\mathbf{I}_t^{\text{fleet}}$ represented by the location, remaining capacity, and completion time of serving next customer of each vehicle in the fleet. Then, we use $\mathbf{e}_t^{\text{fleet\_context}}$ and $\mathbf{e}_t^{\text{node}}$ as the input of an Mask MHA layer to obtain the fleet state representation $\mathbf{e}_t^{\text{fleet}}$ and the current vehicle embedding $\mathbf{e}_t^{\text{cur\_veh}}$. During each encoding-decoding procedure, the vehicle that completes its current customer service first is chosen as the current vehicle for which its next customer to visit is determined in this decoding. “Mask” in ”Mask MHA” refers to forcing the attention value of customers who cannot be served to 0 (i.e. the customers that have been served and the customers whose demand exceeds the vehicle capacity). Next through an MHA layer, the “Vehicle State Representation” block gathers all vehicles’ state representations in $\mathbf{e}_t^{\text{fleet}}$ on the current vehicle, so as to get a context embedding $\mathbf{e}_t^{\text{context}}$. Given $\mathbf{e}_t^{\text{node}}$ and $\mathbf{e}_t^{\text{context}}$, the attention value for each node is then computed by a Mask Attention layer in the “Compatibility” block, and the next node π_t^k to visit for vehicle k and a binary vector n_t^{served} are determined finally.

3.2 Dimension-reducing MHA layer

In previous studies (Kool et al., 2018; Nazari et al., 2018; Bono et al., 2020), DRL-based model inputs have three dimensions, including problem instance, node, and feature. Let d_1, d_2, and d_3 denote their dimension sizes, respectively. d_1 is the number of problem instances (i.e., batch size) inputted in each iteration in model inference. d_2 is equal to $N+1$ since we consider the depot and N customer nodes. d_3 is equal to 3 since each node is characterized by 3 feature values consisting of two coordinates and one demand value (i.e, a 1×3 vector). To handle TDVRPs in real road networks, one of our contribution in methodology is to add time-varying travel times as new features in the model inputs, which extends the feature dimension to a $(N+1)\times 4$ vector, and thus leads to a 4-dimensional network input and much larger memory use and computational complexity.

To reduce memory use and improve computational efficiency, we propose a dimension-reducing MHA (DR-MHA) layer to extract the information of node location, node demand, and travel time between nodes simultaneously, and convert the 4-dimensional inputs into 3-dimensional embeddings. Specifically, using $\mathbf{I}_t^{\text{mask\_proj}}$ as inputs, the DR-MHA generates three $d_1 \times (N+1) \times (N+1) \times d_k$ tensors, including the query $\mathbf{Q}$, the key $\mathbf{K}$, and the value $\mathbf{V}$. Let $\mathbf{A}^{\mathrm{T}(x,y)}$ denote the transpose of the x^{th} dimension and y^{th} dimension of multidimensional array $\mathbf{A}$, and $\mathbf{A}^{\mathrm{T}(x,y)\&\mathrm{T}(x',y')}$ denotes that the transpose $A^{\mathrm{T}(x,y)}$ and $A^{\mathrm{T}(x',y')}$ are performed on tensor $\mathbf{A}$ in turn. We then compute the single-head function by the equation below,

$$\mathrm{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{softmax}\left(\frac{\mathrm{diagnoal}\left(\mathbf{QK}^{\mathrm{T}(2,3)\&\mathrm{T}(3,4)}\right)}{\sqrt{d_k}}\right)\mathbf{V}^{\mathrm{T}(2,3)} \tag{2}$$

where $\mathrm{diagnoal}(\cdot)$ is a function of getting the value of the diagonal of the matrix. If the dimension of $\mathbf{QK}^{\mathrm{T}(2,3)\&\mathrm{T}(3,4)}$ is $d_1 \times (N+1) \times d_k \times d_k$, and $\mathrm{diagnoal}(\mathbf{QK}^{\mathrm{T}(2,3)\&\mathrm{T}(3,4)})$ means to get an tensor with dimension $d_1 \times (N+1) \times d_k$.

Next, to attend jointly information from different sub-networks at different nodes, we compute the multi-head function by the equation below according to the method in Vaswani et al. (2017),

$$\begin{aligned} \text{MultiHead}(\mathbf{Q},\mathbf{K},\mathbf{V}) &= \text{Contact}(head_1,\ldots,head_h)W^O \\ \text{where} \quad head_i &= \text{Attention}(\mathbf{Q}W_i^Q, \mathbf{K}W_i^K, \mathbf{V}W_i^V) \end{aligned} \tag{3}$$

where $W_i^Q \in \mathbb{R}^{d_h\times d_k}$, $W_i^K \in \mathbb{R}^{d_h\times d_k}$, $W_i^V \in \mathbb{R}^{d_h\times d_v}$ and $W^O \in \mathbb{R}^{hd_h\times d_v}$ the are parameter matrices of linear projections. This paper sets $h=8$, $d_h=128$, and $d_k=d_v=d_h/h=16$.

3.3 Synchronous encoder

The node encoder with DR-MHA layer can only obtain the node embeddings with the information of node location, node demand and travel time between nodes in a given time step t, and cannot obtain time-varying travel time information. We develop a dynamic encoding method, called as synchronous encoder, to represent time-varying travel times in real road network, which is synchronized with its corresponding decoding process (i.e., encoding once, decoding once).

Previous DRL-based VRP studies masked served customers only in decoding by setting the probability of selecting served customers to 0. Intuitively, served customers should be irrelevant to further node-selection decisions, and the embedding information of served customer nodes contained in $\mathbf{e}_t^{\text{node}}$ could have negative effects on $\mathbf{e}_t^{\text{cur\_veh}}$ since $\mathbf{e}_t^{\text{cur\_veh}}$ is highly related to $\mathbf{e}_t^{\text{node}}$. The model performance could be improved if the served customer are not considered in further encoding and decoding. Thus, we mask the served customers further before decoding by the following equation.

$$\mathbf{I}_t^{\text{mask\_proj}} = \mathbf{I}_t^{\text{proj}} \odot \boldsymbol{n}_t^{\text{served}} \tag{4}$$

where $\boldsymbol{n}_t^{\text{served}}$ is a binary vector in which each element represents a customer node. The element value is set to 0 if the corresponding node has been served, otherwise it is set to 1. $\odot$ means element by element multiplication.

3.4 Gate Mechanisms

We use a gate mechanism proposed in Parisotto et al. (2020) to obtain effectively the global graph embedding $\mathbf{e}_t^{\text{global\_graph}}$ of the whole road network and the context embedding $\mathbf{e}_t^{\text{fleet\_context}}$ of a fleet at time step t. The gate mechanism is implemented by the formula below.

$$\mathbf{r}_t = \sigma(W_r \mathbf{e}_t^2 + U_r \mathbf{e}_t^1) \tag{5}$$

$$\mathbf{z}_t = \sigma(W_z \mathbf{e}_t^2 + U_z \mathbf{e}_t^1 - b_g) \tag{6}$$

$$\mathbf{h}_t = \tanh(W_g \mathbf{e}_t^2 + U_g(\mathbf{r}_t \odot \mathbf{e}_t^1)) \tag{7}$$

$$g(\mathbf{e}_t^1, \mathbf{e}_t^2) = (1 - \mathbf{z}_t) \odot \mathbf{e}_t^1 + \mathbf{z}_t \odot \mathbf{h}_t \tag{8}$$

where W_r, W_z, W_g, U_r, U_z and U_g re parameter matrices of linear projections, $b_g = 0.1$, $\sigma(\cdot)$ is the Sigmoid function. We calculate $\mathbf{e}_t^{\text{global\_graph}}$ by setting $\mathbf{e}_t^1$ to $\mathbf{e}_t^{\text{avg\_graph}}$ and $\mathbf{e}_t^2$ to $\mathbf{e}_t^{\text{max\_graph}}$. To calculate $\mathbf{e}_t^{\text{fleet\_context}}$, we set $\mathbf{e}_t^1$ to $\mathbf{e}_t^{\text{global\_graph}}$ and $\mathbf{e}_t^2$ to $\text{Concat}(\mathbf{e}_t^{\text{fleet\_node}}, \mathbf{e}_t^{\text{fleet\_proj}})$.

3.5 Model extension for TDVRP with time windows

The model described above is designed for the TDVRP without customer time windows. It can be easily extended to handle the TDVRP with time windows (TDVRP-TW). Let tw_i denote the desired time window of serving customer node i. We have $\text{tw}_i = [a_i, b_i]$, where a_i and b_i represent the lower bound and the upper bound of the expected arrival time at node i. Our TDVRP-TW considers soft time windows. That is, if the actual arrival time of a vehicle arriving node i is less than a_i or greater than b_i, an earliness or tardiness penalty occurs. Setting sufficiently large earliness and tardiness penalty rates is equivalent to considering hard time windows.

Compared with the model for TDVRP without time windows, the model made two changes to adapt the TDVRP-TW. First, the input $\mathbf{I}_t$ of DDAM-GM consists of node coordinates x_i^h, x_i^v, demand x_i^d, travel time $T_{i,j,t}$ from node i to node j at time step t, and time window tw_i of each node i ($i \in \{\mathbb{V}, 0\}$). That is, we have $\boldsymbol{f}_{i,j,t} = (x_i^h, x_i^v, x_i^d, T_{i,j,t}, a_i, b_i)$ and have $\mathbf{I}_t = \{\boldsymbol{f}_{i,j,t}\}_{B\times(N+1)\times(N+1)}$. Second, the model does not mask the served nodes in encoding because our experiments show that masking served nodes in encoding will reduce the optimum-seeking performance for TDVRP-TW.

3.6 Model Training

Our DDAM-GM is trained by policy gradient using REINFORCE algorithm (Williams, 1992). The objective $\mathcal{L}(\theta\,|s)$ is the expected loss, which can be estimated with respect to the parameters θ

$$\nabla\mathcal{L}(\theta\,|s) = \mathbb{E}_{p_\theta(\boldsymbol{\pi}|s)}\left[(L(\boldsymbol{\pi}\,|s) - b(s))\nabla \log p_\theta(\boldsymbol{\pi}\,|s)\right] \tag{9}$$

where $L(\boldsymbol{\pi}\,|s)$ is the objective value to be minimized of solution $\boldsymbol{\pi}$, $b(s)$ is a baseline to reduce variance. We adopt either critic network or rollout randomly as baseline $b(s)$. The critic network ϕ, shares the parameters of DDAM-GM and connects two fully connected layers behind the decoder of DDAM-GM to output the estimated expected objective value. Rollout is similar to the baseline with the best performance in Kool et al. (2018).

In training, we use the Monte Carlo sampling to approximate the gradients of parameters θ as:

$$\nabla \mathcal{L}(\theta) \approx \frac{1}{B} \sum_{i=1}^{B} \left[\left(L\left(\boldsymbol{\pi}_i^s \left| s_i \right. \right) - b\left(s_i \right) \right) \nabla \log p_\theta \left(\boldsymbol{\pi}_i^s \left| s_i \right. \right) \right] \tag{10}$$

where B is the batch size, $\boldsymbol{\pi}_i^s$ is the solution constructed by sample rollout to instance s_i. For rollout baseline, $b\left(s_i\right) = L\left(\boldsymbol{\pi}_i^g \left| s_i \right.\right)$, where $\boldsymbol{\pi}_i^g$ is the solution constructed by greedy rollout to instance s_i. For critic network ϕ, $b(s_i)$ is the output value obtained by taking instance s_i as the input of ϕ. We use the Adam optimizer (Kingma & Ba, 2014) to update the model's parameters.

4 EXPERIMENTS

Our models are programed with Pytorch, and executed on a server with Intel Xeon Platinum 8260 CPU and NVIDIA RTX3090 GPU. Our code of the DDAM-GM will be made available on github [1].

4.1 EXPERIMENTAL SETTING

4.1.1 GENERATION OF INSTANCE SETS

The experiments are conducted on the basis of a real urban road network and a time-varying travel speed dataset from a megacity in China, Chengdu. For TDVRPs either with or without time windows, we train and test our DDAM-GM model based on 3 different problem instance sets with 10, 20, and 50 customers respectively. The objective of our TDVRP without time windows is to minimize the total travel time (minutes) while the objective of our TDVRP with time windows is to minimize the sum of the total travel time, the total earliness and tardiness penalty, and the penalty of unserved customers. The unit earliness and tardiness penalties are set to 1 and 30 per minute respectively, and the unit penalty of unserved customers is set to 30.

The road network contains 408 nodes and 1250 directed edges within the first ring road in the network presented in Zhang et al. (2021). Using the method in Guo et al. (2019), we obtain a 110-day travel speed dataset based on the raw GPS trajectory data of floating taxis within the first-ring road from June 1 to September 17, 2017. In each day, we consider 240 consecutive 2-minute time periods from 8am to 16pm. For an instance set with a certain N, we select randomly a depot and N customer nodes from the 408 road nodes, the coordinates of which are represented in the Universal Transverse Mercator Grid System. Customer demands are sampled randomly between 1 and 9. In all TDVRP instances, we calculate the objective function values by using the travel speed dataset of the 110th day. For a road link directly connected in the road network in a time period, we use the median of its all historical travel speeds during the first 100 days to represent its historical travel speed. Then we obtain the shortest travel time $\boldsymbol{T}_{i,j,p}$ between any two nodes from the depot and the N customer nodes in each time period p based on the method in Huang et al. (2017). Hence, we have a total of 240 $(N+1) \times (N+1)$ matrices consisting of shortest travel times. Together with node coordinates and demands, these matrices are contained in the input $\mathbf{I}_t$ of the DDAM-GM in time step t. The current time period p is determined by the time of the current vehicle completing its current customer service at time step t.

Based on the Chengdu road network, we generate the time windows of customers in each instance according to the rules in Bono et al. (2020). The full time horizon is $[0, 480]$ since we consider 8 hours from 8am to 16pm. The lower and upper bounds of time windows are sampled uniformly from $[10, 30]$ and $[60, 90]$, respectively. We set the vehicle capacity to 30 in all problem instances.

4.1.2 HYPERPARAMETERS

Our DDAM-GM has the same hyperparameter settings as MARDAM. The node embedding dimension is 128. The encoder consists of a "DR-Trans" block and two "Trans" block. The decoder consists of two MHAs with eight attention heads followed by a single-head attention layer. The tanh clip is applied with $C = 10$.

We train the models for 100 epochs. For each epoch, we generate 1,280,000 instances on the fly and train with batch size of 512 (except for TDVRP and TDVRP-TW with 50 customers, where we

[1]https://github.com/WM19998

generate 160,000 instances and train with batch size of 64). We use the Adam optimizer and set the learning rate as 10^{-4} without decay. We generated 10,240 validation instances to judge whether the current model has improved compared with the previous model after training 100 batches. If there is any improvement, save the model. We sample 10,000 instances from the same distributions used for training and validation to evaluate model.

4.1.3 Algorithms and models for comparison

The investigated TDVRPs consider time-varying travel time in a real urban road network, which leads to 240 time periods. Existing methods in the vehicle routing area have not considered so many time periods and cannot tackle our TDVRPs well. We compare the performances of the following models, in which the last two are our DDAM-GM with different decoding strategies.
1) Greedy algorithm (GRA): The nearest and unvisited node is selected as the next node to visit.
2) AM (g): AM (Kool et al., 2018) with greedy decoding strategy.
3) AM-I (g): A DRL model integrating AM (g) with the three improvements described in sections 3.2-3.4.
4) MARDAM (g): MARDAM (Bono et al., 2020) with greedy decoding strategy.
5) MARDAM (s): MARDAM (Bono et al., 2020) with sampling decoding strategy.
6) DDAM-GM (g): Our DDAM-GM with greedy decoding strategy.
7) DDAM-GM (s): Our DDAM-GM with sampling decoding strategy.

The greedy decoding strategy indicates that the model selects the node with the largest probability calculated in "Compatibility" block each time step in decoding. The sampling decoding strategy indicates that the model uses a largest probability to select the node with the largest probability calculated in "Compatibility" block each time step in decoding. Moreover, in our sampling decoding, we randomly generate 1280 solutions for each instance, and select the solution with the smallest objective value as the final solution to this instance.

4.1.4 Evaluation Metric

We measure the performance ζ of each model by the percentage change of its objective value relative to the objective value generated by the AM(g)), which is formulated as follows,

$$\zeta = \frac{V - V_{AM(g)}}{V_{AM(g)}} \times 100\% \tag{11}$$

4.2 Results and discussion

Table 1 shows the performance comparison of our DDAM-GM model and 5 baselines. We report the mean of objective function values of all test instances. Each model is trained and tested on problem instances with the same number of customer nodes. We have not presented the results of AM-I for the TDVRP-TW instance with N=50 because (1) the performances of AM-I are worse than the DDAM-GM's at $N = 10$ and N=20, and (2) the training of AM-I is very time-consuming (approximately more than 10 days in the server we used) at $N = 50$.

It can be found from Table 1 that,
1) For both TDVRPs with and without time windows, two models (i.e., AM-I and DDAM-GM) with our three improvements are clearly superior over the corresponding original models (i.e., AM and MARDAM) by reducing the objective value by 0.41% to 6.28%. That is, our three improvements on DRL models are helpful to improve DRL models' performances on TDVRPs.
2) For TDVRPs without time windows, both DDAM-GM and the MARDAM are inferior to the AM and AM-I, which is similar to the results in Bono et al. (2020) where the AM outperformed the MARDAM in a CVRP. It indicates that it is not important for TDVRPs without time windows to construct multiple vehicle routes simultaneously.
3) For TDVRP-TWs, our DDAM-GM outperforms AM, AM-I and MARDAM largely. Its performance superiority increases with the number of customer nodes, which ranges from 0.41% to 20.57%. It indicates that constructing multiple vehicle routes simultaneously is helpful for TDVRP-TWs to improve the solution performance.
4) The greedy algorithm performs the worst, for the 6 test instance sets, the performances of the GRA are 10.96%-23.17% worse than those of the AM (g).

Table 1: DDAM-GM vs baselines

		Instances with N=10			Instances with N=20			Instances with N=50		
	Method	Obj.	Percent change	Time	Obj.	Percent change	Time	Obj.	Percent change	Time
TDVRP	AM (g)	62.98	-	0.04s	109.05	-	0.08s	241.85	-	0.20s
	AM-I (g)	62.41	-0.91%	0.15s	106.88	-1.99%	0.68s	234.07	-3.22%	5.28s
	MARDAM (g)	68.62	8.96%	0.04s	119.57	9.65%	0.07s	277.47	14.73%	0.19s
	DDAM-GM (g)	64.31	2.11%	0.14s	113.06	3.68%	0.66s	260.04	7.52%	5.31s
	MARDAM (s)	64.00	1.62%	32.00s	109.23	-0.17%	71.96s	256.79	6.18%	226.12s
	DDAM-GM (s)	62.53	-0.71%	156.94s	107.77	-1.17%	814.46s	244.05	0.91%	114m
	GRA	71.42	13.40%	29.49s	128.41	17.75%	61.98s	274.50	13.50%	263.09S
TDVRP-TW	AM (g)	86.79	-	0.05s	152.53	-	0.09s	356.86	-	0.20s
	AM-I (g)	81.76	-5.80%	0.13s	146.29	-4.09%	0.64s	***	***	***
	MARDAM (g)	82.76	-4.64%	0.04s	142.96	-6.27%	0.07s	330.01	-7.52%	0.19s
	DDAM-GM (g)	80.89	-6.80%	0.12s	138.15	-9.43%	0.61s	316.22	-11.39%	4.98s
	MARDAM (s)	75.70	-12.78%	35.86s	127.29	-16.55%	74.08s	293.68	-17.70%	221.60s
	DDAM-GM (s)	75.39	-13.14%	136.55s	125.89	-17.47%	762.70s	283.44	-20.57%	106m
	GRA	107.90	24.32%	29.68s	187.18	22.72%	74.77s	395.97	10.96%	314.52s

Table 2: Results of Ablation study for DDAM-GM Structure

	Obj.	Percent change.	Time
DDAM-GM (g)	113.06	-	0.66s
DDAM-GM (no DR-MHA) (g)	115.05	1.76%	0.29s
DDAM-GM (SE change 1) (g)	113.66	0.53%	0.38s
DDAM-GM (SE change 2) (g)	114.05	0.88%	0.60s
DDAM-GM (no GM) (g)	114.78	1.52%	0.64s

4.3 Ablation Study

To evaluate the effectiveness of three improvement, we perform an ablation study for the DDAM-GM structure based on a TDVRP with 20 customers. The corresponding experimental results are presented in Table 2. In this table, "DDAM-GM (g) w/o DR-MHA" and "DDAM-GM (g) w/o GM" represent the DDAM-GM without improvement 1 and improvement 3, respectively. "DDAM-GM (g) with 1 encoding 2 decoding" and "DDAM-GM (g) w/o customer nodes mask" are used to investigate the effectiveness of improvement 2. The former is to encode nodes every two time steps, but decode nodes every time step. The latter is to cancel out the node mask operation before the node-encoder. Table 2 shows the objective values of the 5 models, percent changes relative the DDAM-GM (g), and computation time used.

It can be found from Table 2 that three improvements are all helpful to improve the performance of MARDAM. The performance of our DDAM-GM decreases the most (1.76%) without improvement 1 although improvement 1 increases largely the computation time. Without the gate mechanism, the computation time is almost the same, but the model performance will decrease by 1.52%.

5 Conclusion

This paper proposes a novel DDAM-GM to tackle practical TDVRPs with time-varying travel time in real road networks. In this model, three improvements are integrated into the MARDAM in Bono et al. (2020) to adapt the practical TDVRPs, which include a dimension-reducing MHA layer, a synchronous encoder, and a gate mechanism. We contain time-varying travel time in the road network as model input. Our experimental results show that our model outperforms significantly two state-of-the-art DRL-based models and a greedy algorithm. The future work can extend the proposed model to solve other real-world combinatorial optimization problems and compare the performance of the proposed model with commercial software and other traditional methods.

Acknowledgments

This work was supported by The General Program of National Natural Science Foundation of China (Grant No.s, 72171159 and 71872118), Sichuan Provincial Science and Technology Planning Project (Grant No., 2020YJ0043), and Sichuan University (Grant No., SKSYL201819).

REFERENCES

Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2): 405–421, apr 2021. doi: 10.1016/j.ejor.2020.07.063.

Guillaume Bono, Jilles S. Dibangoye, Olivier Simonin, Laetitia Matignon, and Florian Pereyron. Solving multi-agent routing problems using deep attention mechanisms. *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2020. doi: 10.1109/tits.2020.3009289.

Xinyun Chen and Yuandong Tian. Learning to perform local rewriting for combinatorial optimization. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 6281–6292, 2019.

Said Dabia, Stefan Ropke, Tom van Woensel, and Ton De Kok. Branch and price for the time-dependent vehicle routing problem with time windows. *Transportation Science*, 47(3):380–396, aug 2013. doi: 10.1287/trsc.1120.0445.

Alberto V. Donati, Roberto Montemanni, Norman Casagrande, Andrea E. Rizzoli, and Luca M. Gambardella. Time dependent vehicle routing problem with a multi ant colony system. *European Journal of Operational Research*, 185(3):1174–1191, mar 2008. doi: 10.1016/j.ejor.2006.06.047.

Lu Duan, Yang Zhan, Haoyuan Hu, Yu Gong, Jiangwen Wei, Xiaodong Zhang, and Yinghui Xu. Efficiently solving the practical vehicle routing problem. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, jul 2020. doi: 10.1145/3394486.3403356.

Lei Gao, Mingxiang Chen, Qichang Chen, Ganzhong Luo, Nuoyi Zhu, and Zhixin Liu. Learn to design the heuristics for vehicle routing problem. *arXiv preprint arXiv:2002.08539*, 2020.

Maha Gmira, Michel Gendreau, Andrea Lodi, and Jean-Yves Potvin. Tabu search for the time-dependent vehicle routing problem with time windows on a road network. *European Journal of Operational Research*, 288(1):129–140, jan 2021. doi: 10.1016/j.ejor.2020.05.041.

Feng Guo, Dongqing Zhang, Yucheng Dong, and Zhaoxia Guo. Urban link travel speed dataset from a megacity road network. *Scientific Data*, 6(1), may 2019. doi: 10.1038/s41597-019-0060-3.

Tuomas Haarnoja, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine. Latent space policies for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pp. 1851–1860. PMLR, 2018.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, nov 1997. doi: 10.1162/neco.1997.9.8.1735.

Yixiao Huang, Lei Zhao, Tom Van Woensel, and Jean-Philippe Gross. Time-dependent vehicle routing problem with path flexibility. *Transportation Research Part B: Methodological*, 95:169–195, jan 2017. doi: 10.1016/j.trb.2016.10.013.

Vincent Huart, Sylvain Perron, Gilles Caporossi, and Christophe Duhamel. A heuristic for the time-dependent vehicle routing problem with time windows. In *Lecture Notes in Economics and Mathematical Systems*, pp. 73–78. Springer International Publishing, 2016. doi: 10.1007/978-3-319-20430-7_10.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

A. L. Kok, E. W. Hans, J. M. J. Schutten, and W. H. M. Zijm. A dynamic programming heuristic for vehicle routing with time-dependent travel times and required breaks. *Flexible Services and Manufacturing Journal*, 22(1-2):83–108, jun 2010. doi: 10.1007/s10696-011-9077-4.

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

Gonzalo Lera-Romero, Juan J. Miranda Bront, and Francisco J. Soulignac. Linear edge costs and labeling algorithms: The case of the time-dependent vehicle routing problem with time windows. *Networks*, 76(1):24–53, mar 2020. doi: 10.1002/net.21937.

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

Qiang Ma, Suwen Ge, Danyang He, Darshan Thaker, and Iddo Drori. Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning. November 2019.

Chryssi Malandraki and Mark S. Daskin. Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Science*, 26(3):185–200, aug 1992. doi: 10.1287/trsc.26.3.185.

Mohammadreza Nazari, Afshin Oroojlooy, Martin Takáč, and Lawrence V Snyder. Reinforcement learning for solving the vehicle routing problem. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 9861–9871, 2018.

Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *International Conference on Machine Learning*, pp. 7487–7498. PMLR, 2020.

Nicolas Rincon-Garcia, Ben Waterson, Tom J. Cherrett, and Fernando Salazar-Arrieta. A metaheuristic for the time-dependent vehicle routing problem considering driving hours regulations – an application in city logistics. *Transportation Research Part A: Policy and Practice*, 137: 429–446, jul 2020. doi: 10.1016/j.tra.2018.10.033.

Remy Spliet, Said Dabia, and Tom Van Woensel. The time window assignment vehicle routing problem with time-dependent travel times. *Transportation Science*, 52(2):261–276, mar 2018. doi: 10.1287/trsc.2016.0705.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pp. 2692–2700, 2015.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Dongqing Zhang, Stein W. Wallace, Zhaoxia Guo, Yucheng Dong, and Michal Kaut. On scenario construction for stochastic shortest path problems in real road networks. *Transportation Research Part E: Logistics and Transportation Review*, 152:102410, aug 2021. doi: 10.1016/j.tre.2021.102410.