Mathematical Biosciences and Engineering

MBE, 19(11): 10864–10881.
DOI: 10.3934/mbe.2022508
Received: 07 May 2022
Revised: 03 July 2022
Accepted: 11 July 2022
Published: 01 August 2022

http://www.aimspress.com/journal/MBE

Research article

Deep reinforcement learning algorithm for solving material emergency dispatching problem

Huawei Jiang*, Tao Guo, Zhen Yang and Like Zhao

College of Information Science and Engineering, Henan University of Technology, Zhengzhou 450001, China

* **Correspondence:** Email: lhwcad@126.com; Tel: +8613523061207.

Abstract: In order to solve the problem that the scheduling scheme cannot be updated in real time due to the dynamic change of node demand in material emergency dispatching, this article proposes a dynamic attention model based on improved gated recurrent unit. The dynamic codec framework is used to track the change of node demand to update the node information. The improved gated recurrent unit is embedded between codecs to improve the representation ability of the model. By weighted combination of the node information of the previous time, the current time and the initial time, a more representative node embedding is obtained. The results show that compared with the elitism-based immigrants ant colony optimization algorithm, the solution quality of the proposed model was improved by 27.89, 27.94, 28.09 and 28.12% when the problem scale is 10, 20, 50 and 100, respectively, which can effectively deal with the instability caused by the change of node demand, so as to minimize the cost of material distribution.

Keywords: dynamic vehicle routing problem; attention mechanism; deep reinforcement learning; gated recurrent unit; encoder-decoder

1. Introduction

Material emergency dispatching (MED) is the key link of emergency management in disaster areas. Due to the complex and changeable situation of disaster areas, the material demand of affected points will gradually change with the passage of time, which increases the complexity of solving MED problem. As a result, under the influence of dynamic factors, it is difficult for vehicles to complete the

material distribution task of all affected point in the shortest time. Therefore, in order to minimize the loss in the disaster area, it is very important to study and formulate a scientific and efficient dispatching scheme.

MED is the practical application of vehicle routing problem (VRP) [1]. At present, the algorithms used to formulate scheduling schemes mainly include accurate algorithm [2,3] and swarm intelligence algorithm [4,5]. Among them, the accurate algorithm mainly includes branch and bound method, column generation method and dynamic programming algorithm, which adopt mathematical methods to solve the problem. This kind of algorithm can obtain the global optimal solution of the problem, but it can only be used to solve small-scale problems [6]. When the scale of the problem becomes larger, the solution time will increase explosively. Compared with the accurate algorithm, swarm intelligence algorithm can obtain the approximate optimal solution of large-scale problems in an acceptable time [7], but the solution performance of the algorithm depends on various manually designed rules, and the performance of the algorithm is difficult to be verified by mathematical theory. Therefore, the relevant mathematical abstraction and modeling methods often have an imbalance between time and performance when solving the problem. In recent years, with the wide application of machine learning technology in computer science and other fields, it provides new ideas for solving VRP and other combinatorial optimization problems [8−11]. For example, deep reinforcement learning (DRL) algorithm is used to solve VRP [12]. Due to the DRL combines the perception ability of deep learning and the decision-making ability of reinforcement learning (RL), it can automatically learn the hidden relationship between data and avoid the limitations caused by manual design features. The DRL algorithm is based on the Encoder-Decoder framework to build the problem-solving model, although recurrent neural network (RNN), long and short term memory (LSTM) network and attention mechanism (AM) [13] are used in the existing framework to build relevant models to solve combinatorial optimization problems, there are still a disadvantage of low performance in solving VRP and other problems. Therefore, Kool et al. [14] used multi head attention mechanism in Encoder-Decoder framework and use REINFORCE algorithm with baseline to train the model, which improved the performance of solving combinatorial optimization problems such as VRP. Even so, when there are dynamic factors in VRP related problems, the fixed model framework will make the extracted node features deviate from the actual features, and the model is difficult to adapt to the changes of uncertain factors.

Due to the dynamic factors in the actual MED problem, the demand of the affected pointers will change dynamically with time under the influence of the disaster. If the above model is used to solve the problem, the node embedding is only calculated once and remains unchanged in the subsequent decoding steps, which making the fixed node embeddings unable to accurately express the current node characteristics when the demand of the effected points changes. Moreover, the existing model only uses the features of the current node to solve the problem, without considering the influence of the previous node information on the current moment. Therefore, this article proposes a dynamic attention mechanism based on improved gated recurrent unit (DAM-IGRU), which updates node embeddings in real time to adapt to the change of node demand, so that the currently known node information can be fully utilized when constructing solutions step by step. In addition, an improved gated recurrent unit (IGRU) is introduced between the encoder and the decoder to express the influence of known node information on the current moment. The reset gate (RG) and update gate (UG) of IGRU is used to weight the node information of the current moment, the previous moment and the initial moment, so as to obtain a higher representation of the node embedding at the current moment.

The rest of this paper is organized as follows. Section 2 describes the existing related work. Section 3 introduces the relevant background knowledge. Section 4 introduces the proposed model. Section 5 shows the experimental results. Section 6 is discussed and analyzed, and the conclusion follows in Section 7.

2. Related works

To solve VRP, researchers have proposed a variety of algorithms. According to the different solution methods, it can be divided into accurate algorithm, heuristic algorithm and machine learning algorithm.

The accurate algorithm is to establish the corresponding mathematical model for a specific problem and then use mathematical methods to solve it, which is usually used to solve the optimal solution of the problem. To improve the speed of the algorithm to solve the optimal solution of VRP, Fukasawa et al. [15] used the branch pricing algorithm to solve the root node when solving the VRP with limited capacity, and find the lower bound through the column generation method, so as to improve the overall efficiency of the algorithm. Pecin et al. [3] used the branch price and cut method to solve VRP with time window. It uses arc subset instead of node subset to reduce the computational complexity. For improving the quality of the solution, in solving VRP in waste management system, Ceselli et al. [16] proposed a specific accurate algorithm for pricing problem by dynamically generating rows and columns under the framework of branch and bound, so as to improve the quality of solution. Marinelli et al. [2] decomposed the main problem into multiple sub-problems by FCM clustering, and use an accurate algorithm to obtain the two-level solution of each sub-problem, so as to continuously optimize the feasible solution of the problem. Although the exact algorithm can get the optimal solution of the problem, it usually exchanges the quality of the solution at the cost of time and space complexity. Due to the introduction of strict mathematical methods in the solution, the calculation time will increase explosively with the increase of the scale of the problem, resulting in the performance of the exact algorithm is greatly reduced, even unable to find the optimal solution. Therefore, the exact algorithm can only be applied to the small-scale VRP solution.

The heuristic algorithm is proposed relative to the exact algorithm and gives a feasible solution of the combinatorial optimization problem within an acceptable range, but the deviation degree between the feasible solution and the optimal solution can not be estimated. Compared with the traditional exact algorithm, the heuristic algorithm has higher robustness and feasibility in dealing with large-scale VRP. However, this algorithm has some disadvantages such as falling into local extremum and slow convergence speed. To prevent the algorithm from falling into local optimum, Zhang et al. [17] introduced a mutation operator to search the neighborhood of randomly selected solution in ant colony algorithm and used exchange operator, shift operator, and inverse operator to maintain the diversity of population and prevent it from falling into local optimum too early. In addition, when solving the multi-site vehicle routing problem, Yao et al. [18] adopted a scanning strategy to help the ant colony algorithm jump out of the local optimal solution, and introduced crossover operation to expand the search space of the algorithm. To accelerate the convergence speed of the algorithm, Cattaruzza et al. [19] introduced local search algorithm and segmentation algorithm into the genetic algorithm to solve the VRP with time window, and selectively carried out local search on a certain individual to accelerate the convergence speed and thus improve the performance of the genetic algorithm. Wang et al. [20] proposed an adaptive genetic algorithm with local search for solving multi-site VRP. According to the

fitness value, the adaptive rate strategy was adopted to update the crossover and mutation probabilities, and combined with local search to accelerate the convergence speed of the algorithm. Although a lot of work has been done to solve the two shortcomings of the algorithm, its solving performance mainly depends on various rules designed by humans, which makes it difficult to achieve the optimal quality of the solution.

With the rapid development and successful application of machine learning technology in the fields of management operations research and computer science, more and more researchers apply it to solve combinatorial optimization problems (such as VRP), and have obtained certain achievement. For example, Hopfield et al. [21] used Hopfield neural network to quickly solve the small-scale traveling salesman problem (TSP), so that the application of neural network in related problems has been widely concerned and discussed by researchers [22]. Researchers have done a lot of work around TSP, VRP and other issues. For example, Vinyas et al. [23] proposed pointer network (PN) and regarded the combinatorial optimization problem as a sequence problem, in which the input is the sequence of nodes and the output is the arrangement of input nodes. In addition, the AM [13] was used to create pointers to input elements, and the softmax probability distribution is used as a "pointer" to represent the variable length sequence, so as to eliminate the limitation that the output length is dependent on the input. In the above work, the recurrence network with non-parametric softmax is trained in a supervised way, and the supervised signals given by the approximate solver are used to train the model, so as to predict the sequence of the visited nodes, but it is difficult to obtain the optimal label of TSP. Therefore, Bello et al. [24] proposed an Encoder-Decoder framework that uses neural network and RL to solve combinational optimization problems, and used the RNN in encoder and decoder respectively. When training the model, the negative path length is used as the reward value, and REINFORCE algorithm is adopted to optimize the parameters of the neural network, which has achieved certain results in solving TSP and other problems. As an extension of TSP, VRP is more difficult to calculate because of its more complex form. In order to solve this problem, Nazari et al. [25] proposed an end-to-end framework using RL algorithm, which is composed of an RNN decoder and an AM. Due to paying attention to the change of node demand during solution, it takes the embedding of static elements as the input of RNN decoder, and inputs the output of RNN and the embedding of dynamic elements into the AM to select the next node. At the same time, Yu et al. [26] constructed a deep neural network model to solve the online VRP. It uses an unsupervised auxiliary network to train the model parameters and form a new neural combination optimization strategy based on RL, so that the trained model can effectively solve the dynamic system. In addition, in order to obtain better models and training methods, Kool et al. [14] proposed an attention-based model that is more effective than PN. The model used the multi head attention mechanism in the Encoder-Decoder framework, and used the REINFORCE algorithm with baseline during model training, which improved the performance of the model to solve combinational optimization problems such as TSP and VRP. In addition, Ma et al. [27] adopted pointer neural network combined with deep learning and RL to solve VRP with task priority and limited resources. According to the actual distribution of nodes, the model continuously learns the mapping relationship between input nodes and output decision-making scheme. Moreover, a global attention mechanism was used in the model to improve the convergence speed and solution performance of the model, which provided a better solution for VRP.

When solving TSP and VRP combinatorial optimization problems, the performance of the above work has better advantages than the traditional heuristic algorithm, which lays a foundation for solving related problems in the future. However, when solving the VRP Problem with dynamic factors, its

performance still has some limitations. Therefore, this article will improve the DRL algorithm to adapt to the MED problem with the change of material demand at the actual affected point. Therefore, the following research work will be carried out to improve the effectiveness of the algorithm.

3. Background

3.1. Encoder-decoder model based on multi-head attention mechanism

The Encoder-Decoder model based on multi head attention mechanism (MHA) [14] uses MHA in encoder and decoder respectively, and obtains node embedding with stronger representation ability after fusion improvement by integrating different results of multiple identical attention. When solving VRP, its encoder is used to generate intermediate node embedding and output access nodes in turn through the decoder. The input feature dimension of VRP problem is $d_x = 3$ (including node coordinates and demand). The initial node embedding (dimension $d_h = 128$) is obtained through linear transformation, and the node embedding is updated through the n-layer attention layer in encoder, and the final embedding h_i^N and graph embedding $\bar{h}^N$ of each node are obtained by Eqs(3.1)–(3.5).

$$Q_i^m = W_Q^m h_i, K_i^m = W_K^m h_i, V_i^m = W_V^m h_i \tag{3.1}$$

$$h_i^m = softmax\left(\frac{Q^{m^T} K^m}{\sqrt{d_k}}\right) V^m \tag{3.2}$$

$$h_i' = \sum_{m=1}^{M} W_O h_i^m \tag{3.3}$$

$$h_i^N = \tanh\big(\tanh(h_i + h_i') + FF(\tanh(h_i + h_i'))\big) \tag{3.4}$$

$$\bar{h}^N = \frac{1}{n}\sum_{i=1}^{n} h_i^N \tag{3.5}$$

Where $m \in \{1, \ldots, M\}$, M is the number of heads, h_i represents the embedding of node i, $W_Q^m \in R^{d_k \times d_h}$, $W_K^m \in R^{d_k \times d_h}$, $W_V^m \in R^{d_v \times d_h}$, $W_O \in R^{d_h \times d_v} (d_k = d_v = d_h/M = 16)$ is a learnable parameter, Q_i^m, K_i^m and V_i^m are the $query$ vector, key vector and $value$ vector of node the i-th in the m-th head, h_i' is the embedding of node i after one attention mechanism, FF is the feedforward sublayer.

In the decoder, the next node to be accessed is selected based on the currently generated partial solution $\pi_{1:t-1}$ and each node embedding. The context embedding is calculated according to Eq (3.6), and then the importance of each node for context embedding is calculated respectively, and the node embedding after fusion promotion is obtained through the MHA mechanism. The output probability of the node is obtained by a single head attention mechanism and calculated with Eqs (3.7) and (3.8).

$$h_c = concate\big(\tilde{h}^N, h_{\pi_{t-1}}^N, h_{\pi_1}^N\big) \tag{3.6}$$

$$q_c = W^Q h_c, K_i = W^K h_i, v_i = W^V h_i \tag{3.7}$$

$$p_j = softmax\left(C \cdot \tanh\left(\frac{q_c^T k_j}{\sqrt{d_k}}\right)\right) \tag{3.8}$$

Where $W^Q \in R^{d_k \times d_h}$, $W^K \in R^{d_k \times d_h}$, $W^V \in R^{d_v \times d_h}$, C is used to map the result to $[-C, C]$ ($C = 10$).

3.2. Gated recurrent unit

The gated recurrent unit (GRU) [28] aims to solve the gradient disappearance problem in the standard RNN, which can be regarded as the optimization of the structural complexity of the LSTM. The principle of GRU is similar to that of LSTM. Through the gating mechanism, GRU can not only remember the past information, but also selectively forget some unimportant information. In terms of structure, GRU only includes two gating structures: reset gate and update gate. Its structure is shown in Figure 1.

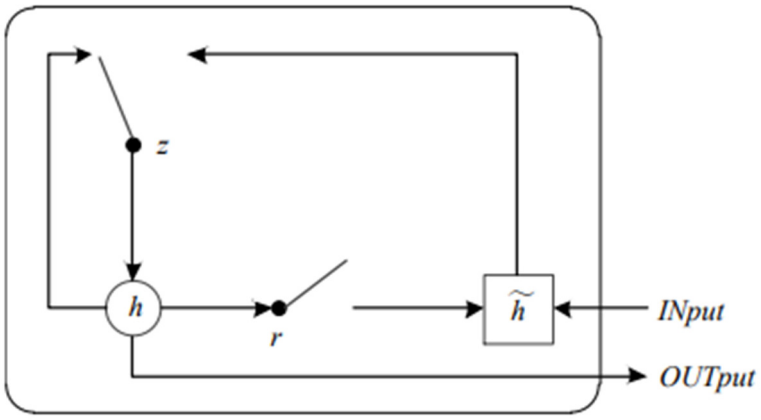

Figure 1. The structure diagram of GRU.

The RG is used to determine how to combine the new input and the previous memory information, and the UG determines how much previous memory information to retain. The RG r_t and UG z_t can be calculated by Eqs (3.9) and (3.10) respectively:

$$r_t = \sigma(W^r x_t + U^r h_{t-1} + b^r) \tag{3.9}$$

$$z_t = \sigma(w^z x_t + U^z h_{t-1} + b^z) \tag{3.10}$$

Where σ is the sigmoid function, x_t is the input at time t, h_{t-1} is the previous hidden state, and W^r, U^r, W^z and U^z are the learned weight matrix respectively.

The candidate hidden status and hidden status can be updated by Eqs (3.11) and (3.12) respectively.

$$\tilde{h}_t = \phi(W x_t + r_t \odot U h_{t-1}) \tag{3.11}$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t \tag{3.12}$$

Where ϕ is the tanh function, W and U are the learnable weight matrix respectively, $\odot$ is the Hadamard product, that is, the product of the corresponding elements in the two matrices.

4. Proposed models

4.1. The structure of proposed model

In the MED problem with dynamic demand, each affected point contains two static features (coordinates) and one dynamic feature (demand). The existing model is mainly used to solve the static problem. The node embedding is only calculated once in the solution, that is, the node features remain unchanged in the subsequent decoding process. Therefore, when the node demand changes, the fixed node embedding will not be able to accurately express the current node characteristics, thus losing some node information and difficult to fit the dynamic characteristics in the problem. In addition, when constructing the solution step by step, the existing model only uses the current node embedding, the previous output node and context embedding. The context embedding is mainly used to reflect the load information of the current vehicle, and does not make full use of the information of the intermediate output node, so that when solving the next output node, the importance between the location of this node and the relative location of other nodes accessed by the current vehicle is ignored.

Therefore, this article proposes the DAM-IGRU. Aiming at the problem of the dynamic change of the demand of the affected points, the model adopts the dynamic Encoder-Decoder framework to recalculate the node embedding in each decoding step, so as to obtain the current accurate node characteristics. At the same time, IGRU is introduced between encoder and decoder, and its RG and UG are used to weighted combine the node information of the previous time, the current time and the initial time to obtain a more representative node embedding, so as to effectively mine the hidden structure information between nodes. The structure of DAM-IGRU model in this article is shown in Figure 2.

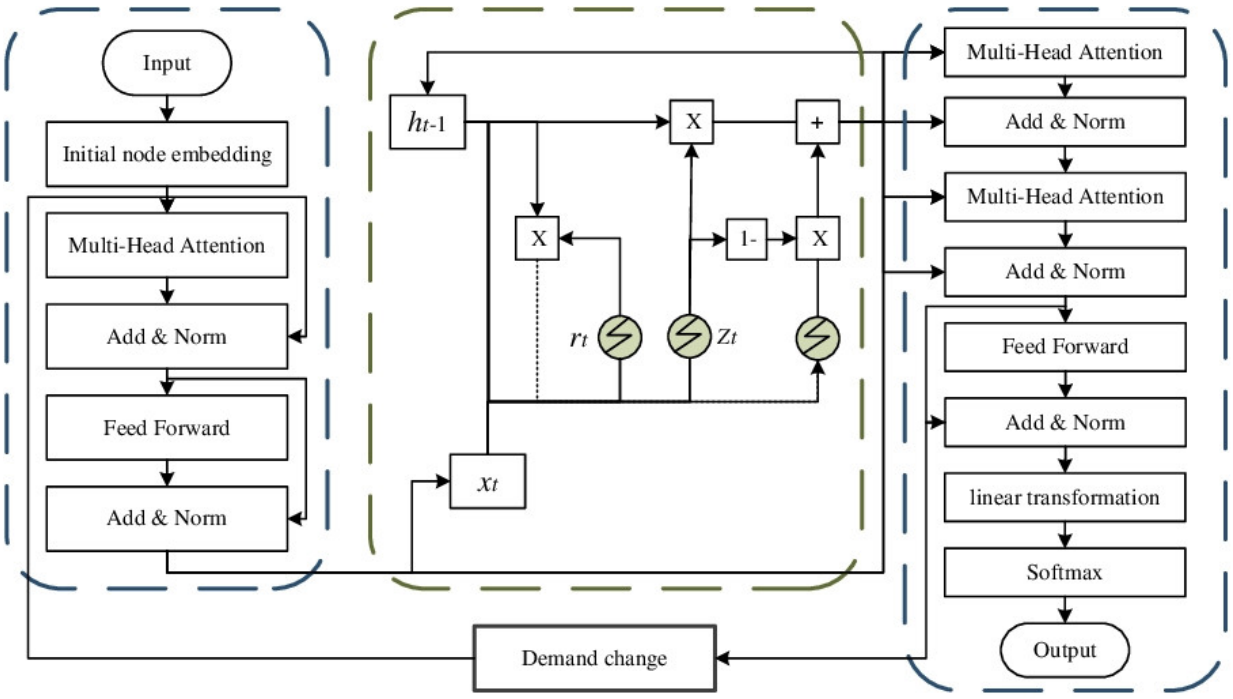

Figure 2. The structure of DAM-IGRU model.

In DAM-IGRU model, when the demand of nodes changes, the encoder is used to re-extract the characteristics of nodes to generate new node embedding and graph embedding. In order to reduce the calculation time and improve the efficiency of model solving, only the unselected nodes and distribution centers are calculated when updating the node embedding each time.

When solving the problem, obtain the node embedding h_i^N and graph embedding $\bar{h}^N$ of all currently unreachable nodes and distribution centers through encoder, and input the node embedding into IGRU. In IGRU, in order to make full use of the known node information and the hidden relationship between nodes, it not only retains the node information at the previous time, but also retains the node information at the initial time, and uses the RG and UG to weight the node embedding at different times, so as to obtain the high-level node embedding with stronger expression ability. The RG is used to determine how to combine the node information of the current time, the memory information of the previous time and the initial time, and the UG determines how much memory information of the previous time and the initial time should be retained. In this article, the improved method shown in Eqs (4.1)–(4.3) was adopted to calculate RG r_t and UG z_t in IGRU.

$$R = \begin{cases} W^r[0{:}\,x_t]_{n-t-1} + U^r[0{:}\,h_{t-1}]_{n-t} + V^r h_0 + b^r, & t \geq 1 \\ W^r[0{:}\,x_t]_{n-t-1} + V^r h_0 + b^r, & t = 0 \end{cases} \tag{4.1}$$

$$Z = \begin{cases} W^z[0{:}\,x_t]_{n-t-1} + U^z[0{:}\,h_{t-1}]_{n-t} + V^z h_0 + b^z, & \text{t >= 1} \\ W^z[0{:}\,x_t]_{n-t-1} + V^z h_0 + b^z, & \text{t} = 0 \end{cases} \tag{4.2}$$

$$r_t = sigmoid(R_{[n-t+1:]}), z_t = sigmoid(Z_{[n-t+1:)} \tag{4.3}$$

Where $[0{:}\,x_t]_{n-t-1}$ indicates that the matrix x_t is supplemented with zero elements in the first n-t-1 column, $[0{:}\,h_{t-1}]_{n-t}$ indicates that the matrix h_{t-1} is supplemented with zero elements in the first n-t column, t indicates the current decoding step, $R_{[n-t+1:]}$ indicates that the $n-t+1$ column of matrix R is taken to the last column, and $Z_{[n-t+1:]}$ indicates that the n-t+1 column of matrix Z is taken to the last column. When the Encoder-IGRU-Decoder process is completed once, the access node at the current time will be output. In the next Encoder-IGRU-Decoder process, the node information output at the previous t-2 time will be deleted from the information of the previous time. Therefore, when the node embedding is promoted and fused in IGRU, the number of embedded nodes input each time will differ by 1 from the previous time.

After the corresponding operations of RG and UG, in this article, the current candidate hidden state $\tilde{h}_t$ and hidden state h_t is calculated by the improved Eqs (4.4) and (4.5). The hidden state h_t has two functions: one is to feed back the hidden information to IGRU at the next time, and the other is to embed it into the decoder as the current node. In the decoder, in addition to h_t, its input also includes the graph embedding $\bar{h}^N$ obtained by the encoder, which is decoded according to the two kinds of information to obtain the output of the current decoding step.

$$\tilde{h}_t = tanh(W[0{:}\,x_t]_{n-t-1} + r_t \odot U h_{t-1}) \tag{4.4}$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t \tag{4.5}$$

The basic steps of the DAM-IGRU model are as follows:

Step 1: For the original input node sequence, the initial node embedding of the node at $t' = 0$ is

obtained by linear transformation.

Step 2: The node embedding is input into encoder for coding, and the node embedding and graph embedding after improved fusion are obtained through MHA mechanism.

Step 3: Input the node embedding into IGRU, determine whether to delete or retain the information of t-1 time and initial time by UG, and use the RG to combine the information of t time, t-1 time and initial time, so as to obtain the node embedding with stronger representation ability after improved fusion.

Step 4: Input the node embedding and graph embedding into the decoder to obtain the output probability of all unvisited nodes, and select the node with the highest probability to output.

Step 5: Judge whether the access of all nodes is completed, if the access is completed, it ends, and the complete access sequence is obtained. Otherwise, the demand of unvisited nodes changes, and return to Step 2.

4.2. Model training algorithm

The performance of the DAM-IGRU model is not only related to the model structure, but also related to the model parameters. Therefore, to obtain a high-performance DAM-IGRU model, it is necessary to find the optimal model parameters to maximize the superiority of the model. RL is an important branch of machine learning technology. Through the interaction between the agent and the environment, features can be automatically learned from the data to maximize benefits. REINFORCE is one of the most basic strategy gradient algorithms in RL, which uses stochastic gradient descent to update strategy parameters, that is, given an instance X, the gradient of parameter θ is:

$$\nabla_\theta R(\theta) = E_{\pi \sim p_\theta(\cdot|X)}[(L(\pi \mid X) - b(X))\nabla_\theta log p_\theta(\pi \mid X)] \tag{4.6}$$

Where $L(\pi \mid X)$ is the path length of the solution to π, and $b(X)$ is the baseline function used to estimate the expected path length of the instance X, which can reduce the gradient variance. During training, the instance X comes from the same distribution, then the gradient of the parameter θ in Eq (4.6) can be approximated by Monte Carlo sampling:

$$\nabla_\theta R(\theta) \approx \frac{1}{B}\sum_{i=1}^{B}(L(\pi_i^s \mid X_i) - L(\pi_i^g \mid X_i))\nabla_\theta log_\theta(\pi_i^s \mid X_i) \tag{4.7}$$

Where B is the batch size, π_i^s and π_i^g are the solutions of the instance X_i constructed by sample rollout and greedy rollout, respectively. The specific training algorithm is shown in Algorithm 1.

Algorithm 1: REINFORCE algorithm	
1	Input: number of epochs E, steps per epoch F, batch size B
2	Initialize parameters θ
3	for epoch = 1.., E do
4	for step = 1.., F do
5	$X_i \leftarrow RandomInstance() for i \in \{1,\cdots,B\}$
6	$\pi_i^s \leftarrow SampleRollout(P_\theta(\cdot \mid X_i)) for i \in \{1,\cdots,B\}$

Continued on next page

7	$\pi_i^g \leftarrow GreedyRollout(P_\theta(\cdot \mid X_i)) for i \in \{1,\cdots,B\}$
8	$g_\theta \leftarrow \frac{1}{B}\sum_{i=1}^{B}(L(\pi_i^s \mid X_i) - L(\pi_i^g \mid X_i))\nabla_\theta log_\theta(\pi_i^s \mid X_i)$
9	$\theta \leftarrow Adam(\theta, g_\theta)$
10	end for
11	end for

5. Experiment results

Our experiments were written in Python 3.8, compiled on PyCharm, and run on an Intel(R) Core(TM) i5-10400f CPU @ 2.10GHz, Windows 10 operating system.

Since the node demand in the MED problem in this paper change dynamically, and the currently public data sets (such as Solomon [29]) lack data to simulate the demand change, the randomly generated data can be used to approximate the demand change of each affected point in the disaster area environment. In addition, in order to comprehensively evaluate the solution performance of the algorithm under different problem sizes, four problem sizes of 10, 20, 50 and 100 are selected [30]. The node positions are evenly and randomly selected in the unit square $[0, 1]^2$, and the node demand is randomly generated between [0, 9] (the demand of the distribution center is 0), and the vehicle loads corresponding to different node numbers are 20, 30, 40 and 50 respectively. According to the solution goal of MED problem, that is, to complete the material distribution task of all disaster affected points in the disaster area in the shortest time, the solution of the problem is measured by the total driving distance of all vehicles when completing this task. The larger the total driving distance is, the higher the cost will be, and for the converse situation, the lower the cost will be.

5.1. Model training

The solution performance of the model is not only related to the model structure, but also depends on the model parameters. Appropriate model parameters can maximize the superiority of the model, so as to obtain better problem solutions when solving problems. Therefore, for improving the solution performance of the model, it is necessary to optimize the model parameters according to the number of nodes. Since REINFORCE is an unsupervised training, this article will perform iterative training on the model under the condition that the baseline is greedy rollout. The hyper-parameters used in the experiment are shown in Table 1, and Figures 3–6 shows the average cost change of the validation set under the baseline greedy rollout when the number of nodes is 10.

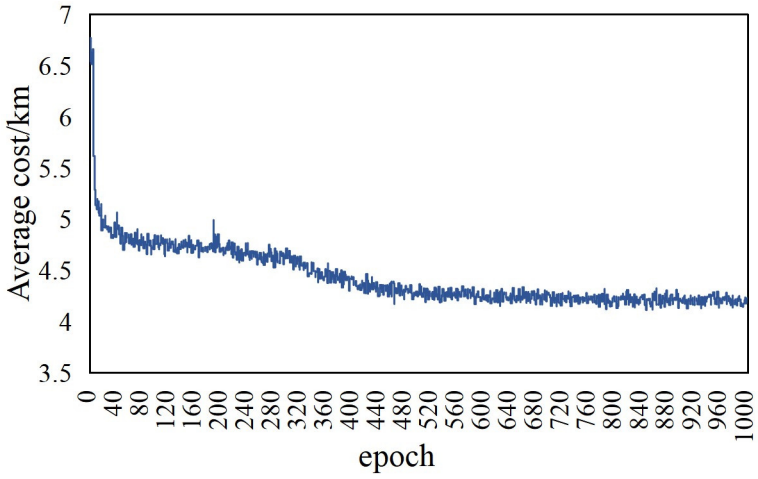

Figure 3. Average cost change of validation set in 1000 training.

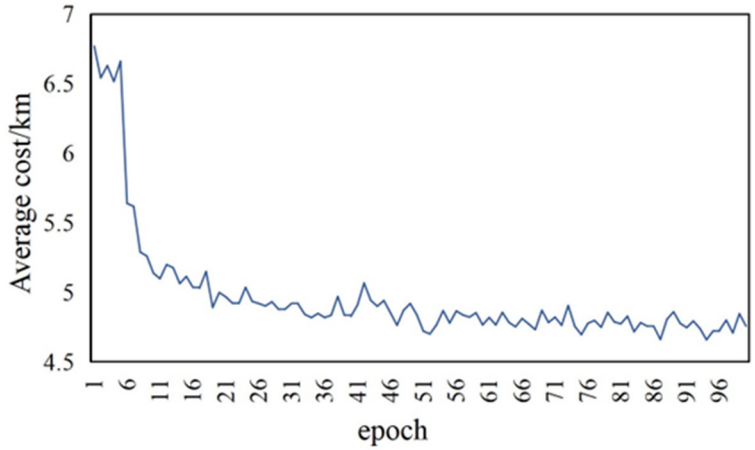

Figure 4. Average cost change of validation sets in the first 100 training.

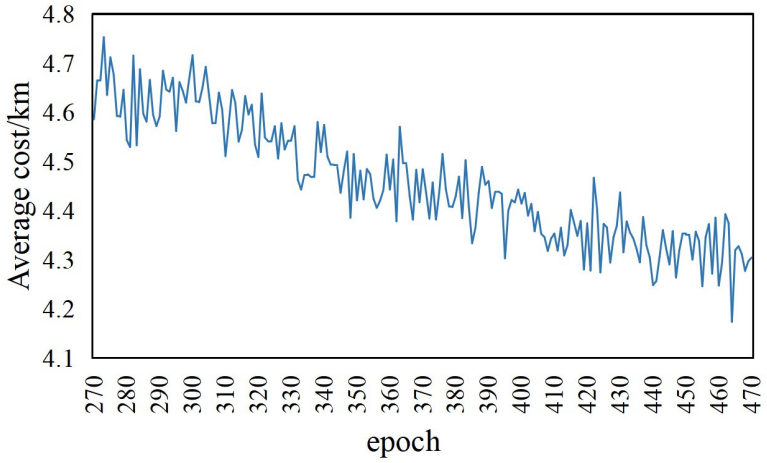

Figure 5. Average cost change of validation set in 270–470 training.

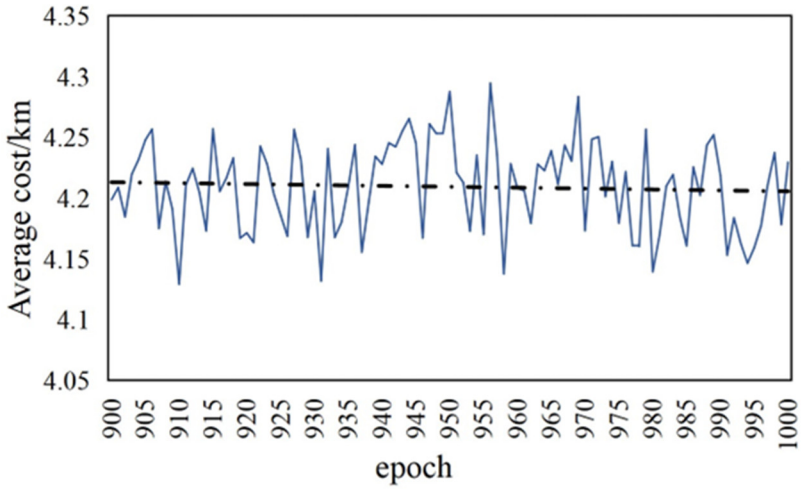

Figure 6. Average cost change of validation set in the last 100 training.

Table 1. Experimental hyper-parameters.

Hyper-parameters	Value
batch size	128
epoch size	1280
val size	512
learning rate	0.0001

As can be seen from Figure 3, during the entire training process, the average cost of the validation set showed a fluctuating downward trend and finally stabilizes. As shown in Figure 4, from the beginning of training to about the 10th time, the average cost of the validation set decreased significantly, and then showed a slow downward trend. It can be seen from Figure 5 that when the number of training times is from 270 to 470, the average cost of the validation set showed a significant downward trend. From Figure 6, it can be seen that the average cost of the validation set still fluctuates at the end of training, but its overall trend has gradually stabilized. This kind of turbulence in training processes is because REINFORCE belongs to unsupervised training, and there is no expected actual data label as a reference. The baseline greedy rollout is added in training to provide a definite benchmark for model training, that is, when the result obtained by using a certain strategy is better than the current benchmark, the probability of the strategy is increased, and vice versa, the probability of the strategy is reduced. Therefore, the average cost of the validation set during the training process of the model shows an oscillating downward trend. And because the algorithm has the ability to learn automatically, it will gradually obtain better model parameters through a lot of training, so that the fluctuation range of the average cost of the verification set will gradually stabilize. That is, it can be considered that better model parameters are obtained through a lot of training on the model.

When using DAM-IGRU model to solve the problem, in order to obtain better model parameters and improve the solution performance of the model, the model is trained under different problem scales, and the corresponding training time and calculation time are shown in Table 2. It can be seen from Table 2 that with the increase of the problem scale, the time required for model training increases significantly, and the corresponding calculation time also increases gradually, but its calculation time

is far less than the training time of the model, and the model only needs to be trained once to solve the problem. Therefore, DAM-IGRU model has high solution efficiency.

Table 2. Model training and calculation time under different number of nodes.

Scales	DAM-IGRU Training time/h	Computing time/ms
10	2.8	1.749
20	10.3	7.266
50	56	50.694
100	330	261.931

5.2. IGRU validity verification

In order to verify the effectiveness of IGRU in the model, DAM model without IGRU and DAMIGRU model are used to solve the MED problem respectively. Firstly, when the number of nodes is 10, 20, 50 and 100 respectively, the same super parameters are used to train the DAM model. Then 10 test sets are generated under different node numbers. Each test set contains 2000 samples. Two models are used to calculate the test set respectively. The calculation results are shown in Table 3 and Figure 7.

Table 3. Experimental results under the two models.

Scales	DAM Average cost/km	Average time/ms	DAM-IGRU Average cost/km	Average time/ms	Average cost deviation	Average time difference/ms
10	4.632	1.539	4.179	1.749	9.78%	0.210
20	6.426	7.063	5.965	7.366	7.17%	0.303
50	11.506	49.482	10.802	50.694	6.11%	1.212
100	18.777	260.311	17.232	261.931	8.23%	1.620

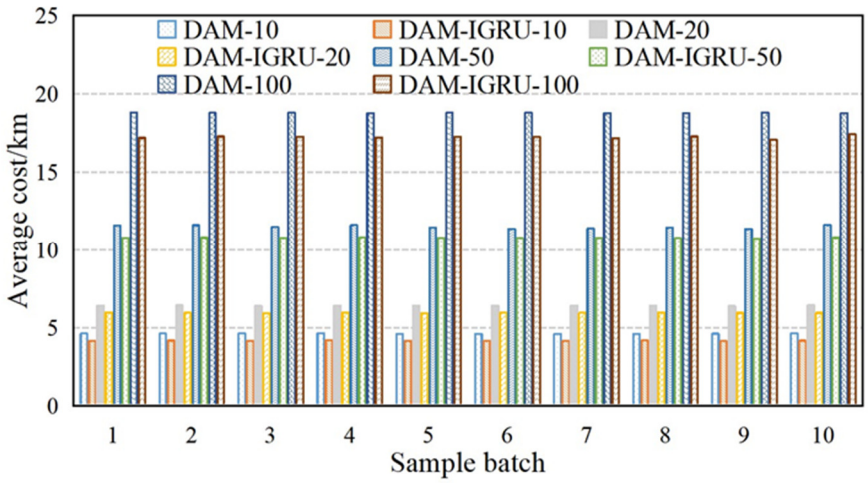

Figure 7. Calculation results of different node numbers under the two models.

As can be seen from Figure 7, when DAM and DAM-IGRU models are used to solve problems of different scales, the average cost of problem solving obtained by DAM-IGRU model is significantly better than that of DAM model. According to Table 3, when the number of nodes is 10, 20, 50 and 100, the average cost of the 10 test sets calculated by DAM-IGRU model decreases by 9.78, 7.17, 6.11 and 8.23%, respectively, compared with the DAM model obtained by training under the same hyper-parameter. The difference is whether IGRU is used between the encoder and the decoder. The difference in solution performance between the two may be because IGRU can further improve the fusion of node features, and the obtained node embedding has more representation ability, so as to improve the solution performance of the model, which can show the effectiveness of IGRU in the model to a certain extent. In addition, in terms of calculation time cost, the average time required by DAM-IGRU model is longer than that of DAM model, and the average time difference between the two models increases with the increase of node number. This is because in DAM-IGRU model, the node embedding obtained by encoder needs to be upgraded and integrated through IGRU again, which takes a certain amount of time.

5.3. Model validity verification

In order to verify the effectiveness of the DAM-IGRU model proposed in this article, the elitismbased immigrants ant colony optimization (EIACO) algorithm proposed by Mavrovouniotis et al. [31] is used to solve this problem. The calculation results are shown in Table 4.

Table 4. Comparison of calculation results of EIACO and DAM-IGRU.

Scales	EIACO Average cost/km	Average time/ms	DAM-IGRU Average cost/km	Average time/ms	Average cost deviation	Average time difference/ms
10	5.795	5.8	4.179	1.749	27.89%	4.05
20	8.278	20.5	5.965	7.366	27.94%	13.13
50	15.021	129.7	10.802	50.694	28.09%	79.01
100	23.973	496.6	17.232	261.931	28.12%	234.67

As can be seen from Table 4, compared with EIACO algorithm, when calculating MED problems with different nodes, the average cost of the problem solution calculated by DAM-IGRU model is lower. When the problem scale is 10, 20, 50 and 100 respectively, the quality of the problem solution calculated by the DAM-IGRU model is improved by 27.89, 27.94, 28.09 and 28.12% respectively. It can be seen that the DAM-IGRU model has better performance in solving, and the average deviation between the two algorithms increases gradually with the increase of the problem scale. This is because when the problem scale is small, that is, the number of nodes is small, the solution space of the problem is relatively small, and the probability of obtaining a better solution by using EIACO algorithm is relatively large. When the problem scale is large, the solution space of the problem increases explosively, and the trained DAM-IGRU model can effectively deal with the huge solution space, and eliminate the influence of the dynamic change of node demand, so as to obtain a better quality solution.

In addition, it can be seen from Table 4 that the calculation time of DAM-IGRU model and EIACO algorithm increases significantly with the increase of problem scale. However, compared with EIACO algorithm, the average time required for solving DAM-IGRU model is less. Under different

node numbers, the solving time is reduced by 4.05, 13.13, 79.01 and 234.67 respectively. It also shows that when dealing with large-scale problems, the well-trained DAM-IGRU model can quickly obtain the corresponding solution by automatically learning the characteristics of input nodes.

6. Discussion

1) The reason for the difference between the heuristic algorithm and the DAM-IGRU model.

It can be seen from Table 4 that the heuristic algorithm and the DAM-IGRU model have certain differences in solving the same problem, and the reason for this difference may be the different principles on which they solve. The heuristic algorithm iteratively optimizes the solution of the problem according to the rules designed in advance, which makes its solution performance effected by various manually designed rules, and the iterative rules of the algorithm make it easy to fall into local extremum too early. At the same time, with the increase of the scale of the problem, the solution space of the problem increases greatly, which reduces the solution performance of the algorithm under the same parameters (such as the number of iterations of the algorithm), making it difficult to fully explore the solution space of the problem. Compared with the heuristic algorithm, the DAM-IGRU model will automatically learn features from the data, and can mine the deep-level information hidden between the data without manual participation. This method can avoid the limitations of manual design features to a certain extent and improve the solution efficiency of problem. In addition, the experimental analysis also shows that the heuristic algorithms are greatly effected by the parameter setting, and its randomness in the solution spaces search makes the solution different. Therefore, if the optimal parameters of the heuristic algorithm are found for experimental calculation, the quality of the solution may be close to the DAM-IGRU model. In the follow-up work, this problem needs to be deeply studied and analyzed.

2) Time complexity of DAM-IGRU model.

Although the DAM-IGRU model has achieved good results, the encoder is repeatedly used to update the node embedding in the solution process, which increases the time complexity of the algorithm. It can also be seen from Table 3 that the solution time of DAM-IGRU model is increased compared with DAM model. In the experimental calculation, when the decoder outputs an access node, the demand of the remaining unreached nodes will change. In order to obtain the current accurate node embedding, it is necessary to re-encode the node information. Therefore, in the whole solution process, the update times of node embedding are related to the length of the final output sequence. The longer the output sequence is, the more time it takes to solve the problem. In order to reduce the complexity of the problem, this article only considers the increase or decrease of the demand of the existing affected points. But in the actual problem, there may be new affected points. If this situation is taken into account, the complexity of the problem will increase and the time spent in solving may also increase significantly. How to reduce the solving time while improving the solving performance of the model is a problem worthy of our in-depth thinking. In the following work, in view of the high time complexity, we will carry out the optimal solution research on the MED problem with changing demand.

3) Inconsistent data sets caused by changes in node access order and demand

In addition, in the MED problem solved in this article, the demand attribute of the affected point changes dynamically over time, that is, in each decoding step, when one access node is output, the demand of the remaining unreached nodes will change. Therefore, for the same set of initial data (the same coordinates and initial demand), the final demand of each node may not be consistent in the

subsequent decoding process due to different output sequences. For example, for a group of nodes (0, 1, 2, 3, 4, 5), assuming that their initial demands are (0, 0.2, 0.1, 0.3, 0.4, 0.3), the model is used to solve the sample. When the first output node is 2, the generated access order is 0–2, and then the demands of the unvisited nodes (1, 3, 4, 5) will change. If the first output node is 3 and the generated access order is 0-3, the demands of the unvisited nodes (1, 2, 4, 5) will change. Therefore, in the comparative experiment, even if the initial data set is the same, with the output of the access sequence, the demand of the remaining unreachable nodes will change continuously, it will lead to the inconsistency of the sample data. In this case, its impact on the experimental results needs to be considered. For example, in Table 3, for the same batch of samples, the calculation result of DAM-IGRU model is better than that of DAM model. In order to minimize the impact of this problem on the results, although we used a large number of samples to train, verify and test the model, it is not known whether this result is still effected by the inconsistency of node demand. And if so, we want to know how much impact it will have. So, in the next step, the demand inconsistent of the unvisited nodes will be further studied based on the change of the output sequence.

7. Conclusions

In this article, a DAM-IGRU model is proposed to solve the MED problem of changing demand. The model adopts a dynamic Encoder-Decoder structure to adapt to the characteristics of dynamic changes in node demand, and updates the current node characteristics in real time in different decoding steps, so that get accurate node information. In addition, through the introduction of IGRU, the node information of the previous moment, the current moment and the initial moment can be effectively combined, and the hidden structural information between the nodes can be deeply excavated, so as to obtain the node features with more representational ability and improve the solving ability of the model.

Meanwhile, the DAM-IGRU model is applied to the MED problem and compared with the improved heuristic algorithm EIACO. The experimental results show that when solving MED problems of different scales, the DAM-IGRU model can obtain higher quality solutions than EIACO in a shorter time. In addition, through further analysis and discussion of the experimental results, the following conclusions are drawn: 1) DAM-IGRU model and heuristic algorithm have different solving performance due to different basic principles in solving. 2) Compared with the DAM model, since the DAM-IGRU model needs to repeatedly calculate the node embedding, the required solution time will increase to varying degrees. 3) When solving the MED problem, in order to simulate the dynamic change of the demand, it may lead to inconsistency in the data set. For 2) and 3), we will find relevant solutions for these two problems in the future, and further optimize the model to improve its performance.

Acknowledgments

This research was supported by Key Scientific Research Projects of Colleges and Universities in Henan Province (Grant No.22A520003), Innovative Funds Plan of Henan University of Technology (Grant No.2021ZKCJ18).

Conflict of interest

The authors declare there is no conflict of interest.

References

1. K. Dorling, J. Heinrichs, G. G. Messier, S. Magierowski, Vehicle routing problems for drone delivery, *IEEE Trans. Syst. Man Cybern.: Syst.*, **47** (2017), 70–85. https://doi.org/10.1109/TSMC.2016.2582745
2. M. Marinelli, A. Colovic, M. Dell'Orco, A novel dynamic programming approach for two-echelon capacitated vehicle routing problem in city logistics with environmental considerations, *Transp. Res. Procedia*, **30** (2018), 147–156. https://doi.org/10.1016/j.trpro.2018.09.017
3. D. Pecin, C. Contardo, G. Desaulniers, E. Uchoa, New enhancements for the exact solution of the vehicle routing problem with time windows, *Informs J. Comput.*, **29** (2017), 377–580. https://doi.org/10.1287/ijoc.2016.0744
4. D. Zhang, S. Cai, F. Ye, Y. W. Si, T. T. Nguyen, A hybrid algorithm for a vehicle routing problem with realistic constraints, *Inf. Sci.*, **394–395** (2017), 167–182. https://doi.org/10.1016/j.ins.2017.02.028
5. D. M. Pierre, N. Zakaria, Stochastic partially optimized cyclic shift crossover for multiobjective genetic algorithms for the vehicle routing problem with time-windows, *Appl. Soft Comput.*, **52** (2017), 863–876. https://doi.org/10.1016/j.asoc.2016.09.039
6. M. Desrochers, J. Desrosiers, M. Solomon, A new optimization algorithm for the vehicle routing problem with time windows, *Oper. Res.*, **40** (1992), 199–415. https://doi.org/10.1287/opre.40.2.342
7. H. F. Wu, X. Q. Chen, Q. H. Mao, Q. N. Zhang, S. C. Zhang, Improved ant colony algorithm based on natural selection strategy for solving TSP problem, *J. Commun.*, **34** (2013), 165–170.
8. I. Sutskever, O. Vinyals, Q. Le, Sequence to sequence learning with neural networks, preprint, arXiv:1409.3215.
9. H. J. Dai, E. B. Khalil, Y. Y. Zhang, B. Dilkina, L. Song, Learning combinatorial optimization algorithms over graphs, preprint, arXiv:1704.01665.
10. Y. Bengio, A. Lodi, A. Prouvost, Machine learning for combinatorial optimization: a methodological tour d'horizon, preprint, arXiv:1811.06128.
11. M. Deudon, P. Cournut, A. Lacoste, Y. Adulyasak, L. M. Rousseau, Learning heuristics for the TSP by policy gradient, in *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, (2018), 170–181. https://doi.org/10.1007/978-3-319-93031-2_12
12. J. Zhao, M. Mao, X. Zhao, J. Zou, A hybrid of deep reinforcement learning and local search for the vehicle routing problems, *IEEE Trans. Intell. Transp. Syst.*, **22** (2021), 7208–7218. https://doi.org/10.1109/TITS.2020.3003163
13. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, et al., Attention is all you need, preprint, arXiv:1706.03762.
14. W. Kool, H. van Hoof, M. Welling, Attention, learn to solve routing problems, preprint, arXiv:1803.08475.
15. R. Fukasawa, H. Longo, J. Lysgaard, M. P. de Aragão, M. Reis, E. Uchoa, et al., Robust branch-and-but-and-price for the capacitated vehicle routing problem, *Math. Program.*, **106** (2006), 491–511. https://doi.org/10.1007/s10107-005-0644-x
16. A. Ceselli, G. Righini, E. Tresoldi, Vehicle routing problems with different service constraints: a branch-and-cut-and-price algorithm, *Networks*, **64** (2014), 282–291. https://doi.org/10.1002/net.21584

17. H. Z. Zhang, Q. W. Zhang, L. Ma, Z. Y. Zhang, Y. Liu, A hybrid ant colony optimization algorithm for a multi-objective vehicle routing problem with flexible time windows, *Inf. Sci.*, **490** (2019), 166–190. https://doi.org/10.1016/j.ins.2019.03.070
18. B. Yao, C. Chen, X. Song, X. Yang, Fresh seafood delivery routing problem using an improved ant colony optimization, *Ann. Oper. Res.*, **273** (2019), 163–186. https://doi.org/10.1007/s10479-017-2531-2
19. D. Cattaruzza, N. Absi, D. Feillet, The multi-trip vehicle routing problem with time windows and release dates, *Trans. Sci.*, **50** (2016), 363–761. https://doi.org/10.1287/trsc.2015.0608
20. S. H. Wang, Z. Y. Lu, L. Wei, G. L. Ji, J. Q. Yang, Fitness-scaling adaptive genetic algorithm with local search for solving the multiple depot vehicle routing problem, *Inf. Technol. Control*, **92** (2016), 601–616. https://doi.org/10.1177/0037549715603481
21. J. Hopfield, D. W. Tank, "Neural" computation of decisions in optimization problems, *Biol. Cybern.*, **52** (1985), 141–152. https://doi.org/10.1007/BF00339943
22. K. A. Smith, Neural networks for combinatorial optimization: a review of more than a decade of research, *Oper. Res.*, **11** (1999), 1–123. https://doi.org/10.1287/ijoc.11.1.15
23. O. Vinyals, M. Fortunato, N. Jaitly, Pointer networks, preprint, arXiv:1506.03134.
24. I. Bello, H. Pham, Q. V. Le, M. Norouzi, S. Bengio, Neural combinatorial optimization with reinforcement learning, preprint, arXiv:1611.09940.
25. M. Nazari, A. Oroojlooy, L. V. Snyder, M. Takáč, Reinforcement learning for solving the vehicle routing problem, preprint, arXiv:1802.04240.
26. J. J. Q. Yu, W. Yu, J. T. Gu, Online vehicle routing with neural combinatorial optimization and deep reinforcement learning, *IEEE Trans. Intell. Transp. Syst.*, **20** (2019), 3806–3817. https://doi.org/10.1109/TITS.2019.2909109
27. H. W. Ma, Y. X. Sheng, W. Xia, A pointer neural network for the vehicle routing problem with task priority and limited resources, *Inf. Technol. Control*, **49** (2020), 237–248. https://doi.org/10.5755/j01.itc.49.2.24613
28. K. Cho, B. V. Merrienboer, C. Gulcehre, D. Bahdanau, F, Bougares, H. Schwenk, et al., Learning phrase representations using RNN Encoder-Decoder for statistical machine translation, in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (2014), 1724–1734. https://doi.org/10.3115/v1/D14-1179
29. M. M. Solomon, VRPTW benchmark problems, 2003.
30. G. A. Bula, C. Prodhon, F. A. Gonzalez, H. M. Afsar, N. Velasco, Variable neighborhood search to solve the vehicle routing problem for hazardous materials transportation, *J. Hazard. Mater.*, **324** (2017), 472–480. https://doi.org/10.1016/j.jhazmat.2016.11.015
31. M. Mavrovouniotis, S. Yang, Ant algorithms with immigrants schemes for the dynamic vehicle routing problem, *Inf. Sci.*, **294** (2015), 456–477. https://doi.org/10.1016/j.ins.2014.10.002

AIMS Press