


ITC 2/49 Information Technology and Control Vol. 49 / No. 2 / 2020 pp. 237-248 DOI 10.5755/j01.itc.49.2.24613	A Pointer Neural Network for the Vehicle Routing Problem with Task Priority and Limited Resources	
	Received 2019/11/13	Accepted after revision 2020/03/22
	 <a href="http://dx.doi.org/10.5755/j01.itc.49.2.24613">http://dx.doi.org/10.5755/j01.itc.49.2.24613</a>	

**HOW TO CITE:** Ma, H., Sheng, Y., Xia, W. (2020). A Pointer Neural Network for the Vehicle Routing Problem with Task Priority and Limited Resources. *Information Technology and Control*, 49(1), 237-248. <https://doi.org/10.5755/j01.itc.49.2.24613>

# A Pointer Neural Network for the Vehicle Routing Problem with Task Priority and Limited Resources

**Huawei Ma, Yuxiang Sheng, Wei Xia**

School of Management, Hefei University of Technology, Hefei 230000, P.R. China,  
e-mails: colt\_mhw@126.com, shengyx777@163.com, xiawei@hfut.edu.cn

Corresponding author: colt\_mhw@126.com

The vehicle routing problem with task priority and limited resources (VRPTPLR) is a generalized version of the vehicle routing problem (VRP) with multiple task priorities and insufficient vehicle capacities. The objective of this problem is to maximize the total benefits. Compared to the traditional mathematical analysis methods, the pointer neural network proposed in this paper continuously learns the mapping relationship between input nodes and output decision schemes based on the actual distribution conditions. In addition, a global attention mechanism is adopted in the neural network to improve the convergence rate and results. To verify the effectiveness of the method, we model the VRPTPLR and compare the results with those of genetic algorithm and differential evolution algorithm. The parameter sensitivity of each algorithm is assessed using different datasets. Then, comparison experiments with the three algorithms employing optimal parameter configurations are performed for the validation sets, which are generated at different instance scales. It is found that the solution time of the pointer neural network is much shorter than that of the genetic algorithm and the proposed method provides better solutions for large-scale instances.

**KEYWORDS:** Task Priority, Limited Resources, Pointer Neural Network, Global Attention Mechanism.

---

## 1. Introduction

The vehicle routing problem with task priority and limited resources (VRPTPLR) is a type of combinatorial optimization problem that maximizes the total benefit by arranging tasks with different priorities and appropriately constructing heterogeneous vehicle routes. In this approach, the resources cannot meet all the demands, and the priorities are given in advance according to the importance degree and the urgency demands of the customer. This problem has been addressed in many fields, such as the emergency distribution of goods [28], and research to date has focused on the design and implementation of efficient algorithms.

A number of operations research methods have been proposed to solve this problem, which can be divided into exact algorithms [7] and heuristic algorithms [19]. The former always finds the optimal solution, but their efficiency in solving large-scale problems cannot satisfy the practical requirements. The latter can obtain feasible solutions quickly; however, the algorithm performance cannot be validated by mathematical theory. Therefore, based on mathematical abstracting and modeling, there is often a trade-off between solution performance and solution time.

In recent years, with the rapid development and successful application of machine learning technology in various fields, such as management operation research [10, 16], medicine [4, 22], computer science [11], etc., several technologies have been introduced to solve combinatorial optimization problems [26, 29]. Vinyals et al. [29] proposed a model consisting of two recurrent neural networks (RNNs) and an attention mechanism to solve combinatorial optimization problems. However, this method cannot effectively solve combinatorial optimization problems in the absence of label data. Subsequently, Bello et al. [3] improved the efficiency of the model and adjusted the parameters through reinforcement learning; experiments were then performed based on the traveling salesman problem (TSP) with 20, 50, and 100 node instances. The results showed that the performance of reinforcement learning methods was better than that of Christofide's heuristic algorithm [5]. The type of model presented by Bello does not require mathematical modeling and achieves a trade-off between solution time and performance, thereby providing a new

way of obtaining an adaptive solution to real-time scheduling problems, e.g., VRPTPLR.

Compared with the traditional VRP, the VRPTPLR studied in this paper can only access some customer nodes due to limited resources, and each vehicle should visit the customers following the order of priorities. Therefore, a pointer neural network model is proposed in this paper to select served customers first and then assign tasks to heterogeneous vehicles.

The primary contributions of the paper are as follows. 1) The latent rule that relates the vehicle routing sets and objective function values is learned by the pointer neural network by combining deep learning and reinforcement learning, and an excellent feasible solution can be obtained in a short time after training. 2) Because the attention mechanism can selectively learn input data and because the sequence of the model output is related to this mechanism, a global attention mechanism [14] is introduced in the proposed pointer neural network to calculate the weights of all nodes and exclude those nodes that violate certain constraints. Then, a polynomial sampling distribution is used to select the next node to be accessed by the vehicle.

This paper is structured as follows: Section 2 summarizes the findings for similar types of VRPs; Section 3 details the model of the pointer neural network; Section 4 describes the parameter experiments conducted with the three algorithms, presents the experimental results and compares the results achieved with different datasets; and Section 5 summarizes the work of this paper.

---

## 2. Related Work

The VRP and its generalized problems, including VRPTPLR, have been a focus of studies involving combinatorial optimization problems, and various algorithms with different mechanisms have been proposed. In addition to numerous operations research methods, reinforcement learning methods have recently been introduced to solve such problems [17].

As mentioned above, exact algorithms and heuristic algorithms are the primary classes of operation re-

search methods. In some instances, exact algorithms, such as dynamic programming [20], column generation [8] and the branch-and-price [2] method, have been used. Yu et al. [32] propose an improved branch-and-price (BAP) algorithm to precisely solve the heterogeneous fleet green vehicle routing problem with time windows, and a multi-vehicle approximate dynamic programming algorithm was designed to speed up the pricing problem in BAP. Then, the effectiveness of the BAP algorithm is verified by extensive computational experiments performed on the Solomon benchmark instances. In addition to a specific type of VRP, Pessoa et al. [21] introduce a Branch-Cut-and-Price algorithm for a family of VRP variants. The algorithm was extensively tested in instances of the literature and was shown to be significantly better than previous exact algorithms.

Because the efficiency of exact algorithms declines greatly in large-scale instances, heuristic algorithms have attracted considerable attention for large-scale instance problems; such algorithms include the ant colony algorithm [13, 33], tabu search algorithm [15], genetic algorithm [23] and others. Arnold et al. [1] and Triki et al. [27] designed heuristic algorithms for their specific large-scale mathematical problems to obtain improved feasible solutions. Zhang et al. [33] proposed a multi-objective solution strategy based on the ant colony algorithm and three mutation operators to solve the multi-objective vehicle routing problem with flexible time windows. The performance of the proposed approach was evaluated on Solomon benchmark instances, and experimental results show that the suggested approach is comparative to the best known results in the literature. Moreover, in recent years, hybrid heuristic algorithms have gained increasing interest for solving combinatorial optimization problems [12, 24]. Kucukoglu et al. [9] consider efficient search procedures based on the constraints of the problem, a modified solution acceptance criterion and an advanced tabu list structure, and finally proposed the hybrid simulated annealing/tabu search algorithm. The experimental results based on benchmark problems indicated that the efficiency and solution quality were both better than those of well-known solution approaches. Sedighizadeh et al. [25] proposed a hybrid heuristic algorithm based on particle swarm optimization and an artificial bee colony algorithm,

and comparison experiments between non-hybrid algorithms and hybrid algorithms were performed for multiple instances.

In addition to mathematical programming methods, machine learning methods have been increasingly used to solve VRPs; for example, Cooray et al. [6] enhanced the genetic algorithm with machine learning technology and decreased the calculation time to below that of the general genetic algorithm. Moreover, machine learning can not only be combined with heuristic algorithms but can also be used to construct neural network models to solve VRPs. Wang et al. [30] built neural network models for TSP and then trained neural network models and adjusted the parameters iteratively. Experiments verified that the solution quality of the well-trained neural network was better than that of state-of-the-art results of learning algorithms. Yu et al. [31] propose a novel deep reinforcement learning-based neural combinatorial optimization strategy, which used an unsupervised auxiliary network to train the model parameters. The simulation results show that the proposed strategy can significantly outperform conventional strategies with limited computation time in both static and dynamic logistic systems. Nazari et al. [18] present an end-to-end framework for solving VRP using deep reinforcement learning. After problem instances are sampled from a given distribution, the model is trained by observing the reward signals and following feasibility rules. The experimental results show that the tour length of the model after training is not longer than a recent method for solving TSP, and at the same time, the framework can be applied in variants of VRP.

At present, the most popular methods solving combinatorial optimization problems are still to obtain optimal or near-optimal solutions through mathematical modeling and algorithm designing, and for large-scale problems the heuristic algorithms are always the focus due to the high efficiency requirements. However, the booming deep learning and reinforcement learning technologies provide us with a new idea for solving the problems. We can build neural networks and output excellent feasible solutions in very short time after training the networks with a large number of samples, while complex modeling and validation of mathematical models is not required.

### 3. Establishment of the Pointer Neural Network Model

For the VRPTPLR, the pointer neural network model is established in this paper. This model simultaneously takes into account the allowance of the current vehicle and the priority of accessing the customer during the processing step. Then, vehicle routing plans are established based on the total benefits. To improve the convergence effect of the model, the strategy gradient method is used to make the neural network converge by adjusting its parameters in accordance with the objective function values of two consecutive batches.

#### 3.1. Description of the Problem

The VRPTPLR is common in real environments, especially in peak periods. This problem can be described as follows: each customer is associated with a priority in view of the urgency of demands, and a fleet of heterogeneous vehicles is used to serve the corresponding customers. Because the number of vehicles is limited, only a portion of the customers can be assigned to vehicles considering the priorities and side constraints. The objective of the problem is to maximize the total benefits by choosing the appropriate customers and constructing the optimal vehicle routes. Here, the total benefit can be calculated by the total rewards of serving customers minus the total travel costs of vehicles.

In this paper, to solve the above problem, a pointer neural network is proposed. The network inputs include the indexed vehicle and node data, and the outputs are the resulting orders for each vehicle. Because the inputs and outputs are both described as sequences, a type of encoder-decoder framework based on deep learning is introduced. In this framework, the encoder transforms the input sequences into a fixed-length vector, and the decoder outputs the visiting orders of appointed vehicles considering the capacity of each vehicle and the task priorities. Moreover, to adequately extract the network features, embedding is performed ahead of the encoder-decoder process to improve the learning effects of the neural network.

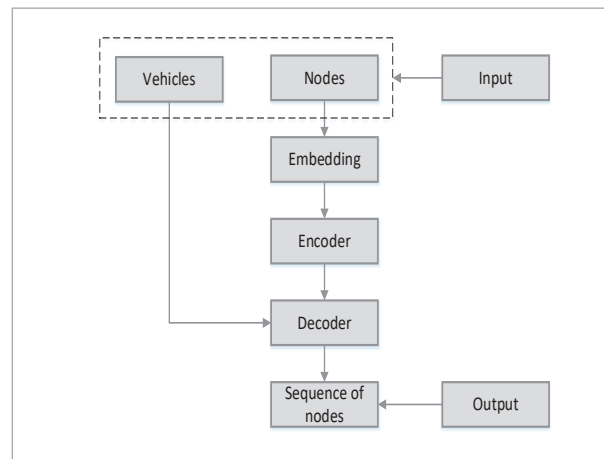
#### 3.2. Components of the Model

The pointer neural network model in this paper consists of an embedding layer, an encoder layer and a

decoder layer. Initially, the input data are raised dimensions in the embedding layer. Then, the embedded information is passed to the encoder layer to obtain the fixed-length vector of the input data features. Finally, the vector is transformed into the visiting orders of specified vehicles by the decoder layer, and the relative total benefits are calculated. The model is shown in Figure 1, in which the encoder layer and decoder layer are both long short-term memory (LSTM) networks.

Figure 1

The pointer neural network model



**1 Vehicle and node information:** Vehicle information refers to the vehicle properties, including the vehicle capacity and travel costs per kilometer. Node information refers to the depot and customers properties, which include coordinates  $(x, y)$ , customer demands, task priorities and the visiting rewards. The corresponding symbol definitions are as follows.

It is assumed that  $V = \{y^j, j = 1, \dots, m\}$ , where  $m$  is the total number of vehicles,  $j$  is the set of vehicle properties,  $y^j = (c^j, f^j)$   $j = 1, \dots, m$  is associated with the  $j$ th vehicle,  $c^j$  is the capacity of the  $j$ th vehicle, and  $f^j$  is the travel cost of the  $j$ th vehicle per unit kilometer.

Let  $S = \{s_i, i = 0, 1, \dots, n\}$  be the node set, in which node 0 is a depot and the other nodes are customers, where  $n$  is the total number of nodes. The five-tuple  $s_i = (x_i, y_i, d_i, p_i, b_i)$ ,  $i = 0, 1, \dots, n$ ,  $s_i$  represents the properties of the  $i$ th node, where the first two variables are X-Y coordinates, the third variable is the demand, the

fourth variable is the task priority level, and the final variable is the reward value of visiting the node.

**2 Encoder-Decoder:** The origin node properties are represented by the abovementioned five-dimensional vector, and it is very difficult to estimate the benefit of visiting each node based on low-dimension data, especially when the dimensional features are similar. Hence, an encoder-decoder model integrated with embedding operations is proposed to learn the decision schemes, as shown in Figure 2.

In the encoder scheme, the embedded vectors are input to an LSTM network, and a reflect vector  $ref = Encoder\_outputs$  is output, where  $ref$  is an  $N \times M$  matrix in which  $N$  represents the quantity of nodes and  $M$  represents the increased dimension after embedding. Here,  $E_i$  is the embedded vector associated with the  $i$  th node. Let  $c$  and  $h$  be the long-term unit state and the hidden state, respectively, of the LSTM network, where  $c_i$  and  $h_i$  are the unit state and hidden state, respectively, when encoding the  $i$  th node. The encoder network is shown in the upper right portion of Figure 2.

As shown in Figure 2, the decoder consists of four components: an LSTM network, an attention mechanism, constraint verification and a node selection operation. During each iteration, the component of the reflect vector associated with the current node  $idx_i$

is input into the LSTM network to obtain the hidden state  $h'_{i+1}$  at step  $i$ . Then,  $ref$ ,  $h'_{i+1}$  and the vehicle information are input into the attention mechanism to calculate the visiting probability of each node in the next iteration. To ensure the feasibility of the solution, constraint verification is performed before selecting the visiting node in the next iteration.

The details of the attention mechanism are shown in Figure 3. The reflect vector from the encoder, the hidden state  $h'$  from the LSTM network of the decoder and the vehicle properties from the input dataset are used for the attention mechanism. First, the weight of the  $i$  th node of the  $j$  th vehicle is calculated. Then, the visiting probability of each node is obtained by the softmax function. Finally, a polynomial sampling function is used to choose the node most likely to be visited.

$$u_i^j = \begin{cases} v^T \cdot \tanh(W_a [ref : q]) & \text{if } \sum_{k=1}^l d_{\pi^i(k)} < c^j \\ & \text{and } p_i > p_{\pi^i(l)} \\ & \text{and } i \neq \pi^j(k) \\ & \text{for } k = 1, \dots, l, j = 1, \dots, m \\ -\infty & \text{otherwise} \end{cases} \quad (1)$$

$$p(C_i^j) = \text{soft max}(u_i^j) \quad (2)$$

**Figure 2**  
Encoder-decoder model

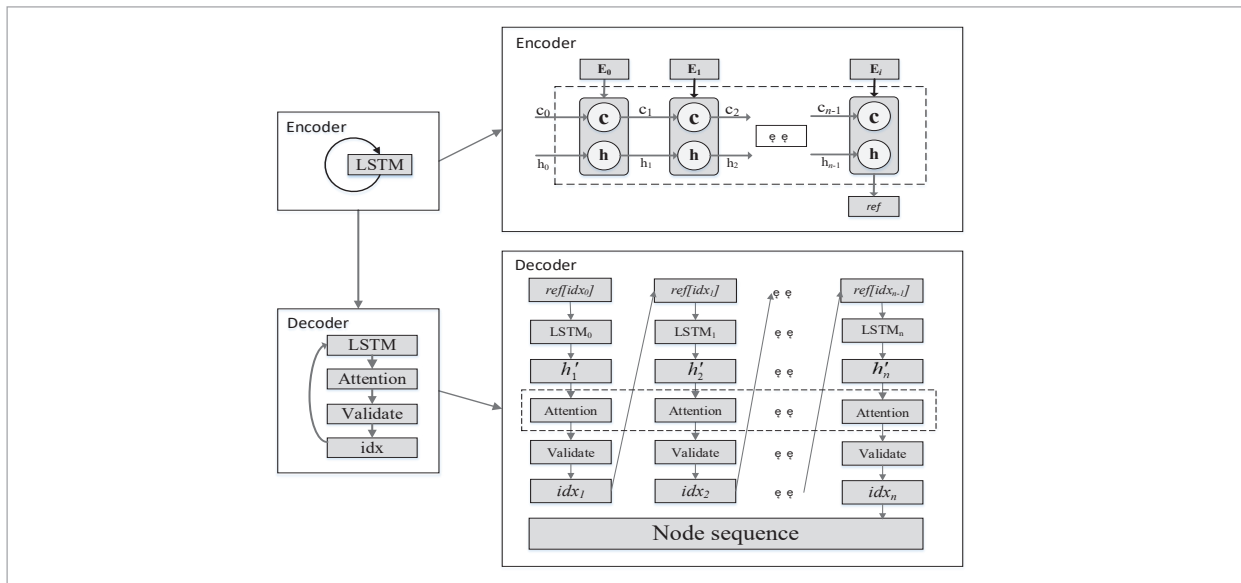
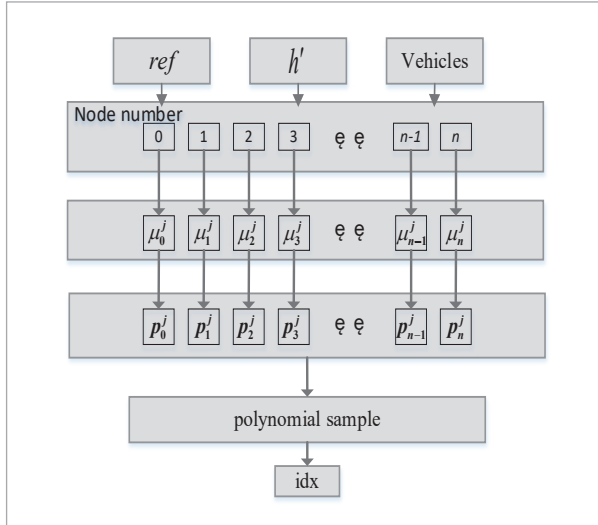


Figure 3

Attention mechanism



In formula (1),  $\pi^j$  indicates the sequence of the current travel set of the  $j$ th vehicle as  $[s_1, s_2, s_3, s_4]$ ,  $k$  is the index of  $s$ , and  $l$  indicates the length of the sequence set of the  $j$ th vehicle. Thus,  $\pi^j(k)$  represents the specific node that was accessed at step  $k$  by the  $j$ th vehicle, and  $\pi^j(l)$  represents the last node in the travel set of the  $j$ th vehicle.  $q$  is an  $N \times M$  matrix based on a hidden state  $h^i$  transform of the dimension.  $[ref : q]$  is the transpose matrix after  $q$  contacts with  $ref$ , and the shape is  $[2M \times N]$ .  $p(C_i^j)$  is the probability that the  $i$ th node of the  $j$ th vehicle is selected.  $v^T$  and  $w_a$  are learnable parameters of the model, where  $v^T$  is a  $1 \times M$  vector and  $w_a$  is an  $M \times 2M$  matrix.

**3 Optimization algorithm:** At the end of the decoding process, the node sequence of each vehicle to be accessed is output, and the vehicles distribution distance is calculated as follows:

$$L(\pi^j | S) = \sum_{k=1}^{h_j} \|x_{\pi^j(k)} - x_{\pi^j(k+1)}\|_2, \quad (3)$$

where  $h_j$  indicates the length of the sequence set of the  $j$ th vehicle when all vehicles are considered.  $L(\pi^j | S)$  denotes the distance traveled by the  $j$ th vehicle.

The objective function maximizes the benefit of all vehicles after completion of their access tasks and is calculated as follows:

$$z = \max \sum_{j=1}^m \left( \sum_{k=1}^{h_j} b_{\pi^j(k)} - f^j L(\pi^j | S) \right). \quad (4)$$

In the above formula,  $\sum_{k=1}^{h_j} b_{\pi^j(k)}$  represents the customer rewards of the  $j$ th vehicle.  $f^j L(\pi^j | S)$  represents the transportation cost of the vehicle.

To optimize the parameters of the neural network, the strategy gradient method of the Adam (adaptive moment estimation) optimization algorithm is adopted in the pointer neural network. The gradient is calculated according to the objective function difference of two consecutive batch decision schemes to adjust the parameters in the neural network. The optimization algorithm can dynamically adjust the learning rate of each parameter, and the learning rate has a certain range in each iteration. This approach has good adaptability for the problem explored in this paper.

## 4. Experiments and Results

To test the performance of the pointer neural network algorithm, the training and test datasets are generated, as described in Section 4.1, and a comparison with genetic algorithm (GA) and differential evolution algorithm (DE) is performed. All the experiments are run on a computer with an AMD Ryzen 7 1700X eight-core processor, 8 GB of memory, and Windows 10 (64 bits). The algorithms were coded in the Python language to implement the deep learning methods.

### 4.1. Experimental Datasets

For the pointer neural network algorithm, we define the generation rules of the input data. The node coordinates are chosen randomly in the range of  $[0,1] \times [0,1]$ . The quantity demand and priority are drawn uniformly at random in  $[10,30]$  and in  $[0,10]$  uniform distribution, respectively, and the reward value of accessing the node is equal to  $dp^2$ , indicating that a high priority number corresponds to a high return value. Three types of vehicles were defined. The capacity  $c$  was set to 50, 70, and 100, and the corresponding transportation cost per unit kilometer  $f$  was set to 100, 150, and 200, respectively. The three types of vehicles were randomly selected when generating the vehicle sets. In the parameter sensitivity test, 1.28 million groups of nodes

and vehicle datasets were generated according to the distribution rule for 50 nodes with 10 vehicles, 75 nodes with 15 vehicles, 100 nodes with 20 vehicles and 150 nodes with 30 vehicles. After parameter sensitivity tests were performed, the pointer neural network was trained with the optimal parameter combination based on 8 test problem sets, and verification sets (C1, C2, C3, C4, C5, C6, C7, and C8) were generated at the same scales, where C1 was 50 nodes with 8 vehicles, C2 was 50 nodes with 10 vehicles, C3 was 75 nodes with 10 vehicles, C4 was 75 nodes with 15 vehicles, C5 was 100 nodes with 15 vehicles, C6 was 100 nodes with 20 vehicles, C7 was 150 nodes with 25 vehicles, and C8 was 150 nodes with 30 vehicles.

## 4.2. Algorithm Experiments

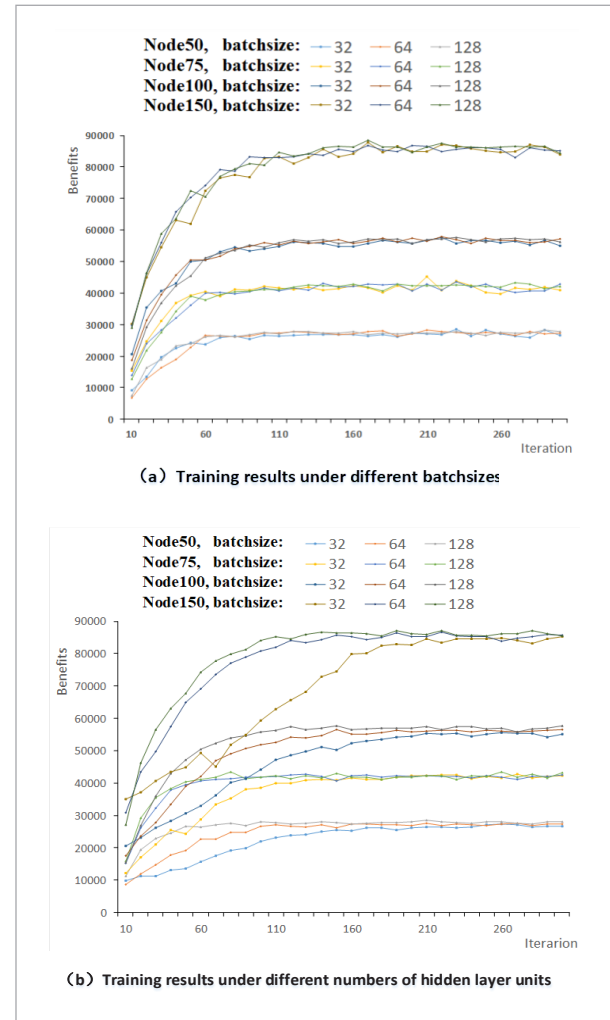
In this section, two types of experiments are described. The first experiment is a parameter sensitivity test for the three algorithms, and the optimal parameter configurations are found for different datasets. The second experiment involves three types of performance indicators, including the solution time, solution result and corresponding stability. For these three indicators, the three algorithms were compared considering their optimal parameter configurations based on verification sets to test the performance of the pointer neural network.

**1 Parameter sensitivity test:** During the training process of the pointer neural network for four types of training datasets, the influence of the input data batch size and number of units in the LSTM hidden layer were tested. The batch size was 32, 64, and 128, and the number of hidden layer units was 32, 64, and 128. The sensitivity test results for these parameters of the pointer neural network are shown in Figure 4.

Then, sensitivity tests considering the population size, crossover probability, mutation probability and update ratio of the GA were conducted. The range of the population size was set to  $[100, 2000]$ , and the step size was 100; the range of the crossover probability was set to  $[0.1, 0.95]$ , and the step size was 0.05; the range of the mutation probability was set to  $[0.025, 0.5]$ , and the step size was 0.025; and the range of the update ratio was set to  $[0.05, 1]$ , and the step size was 0.05. The test results for the various parameter settings are shown in Figure 5.

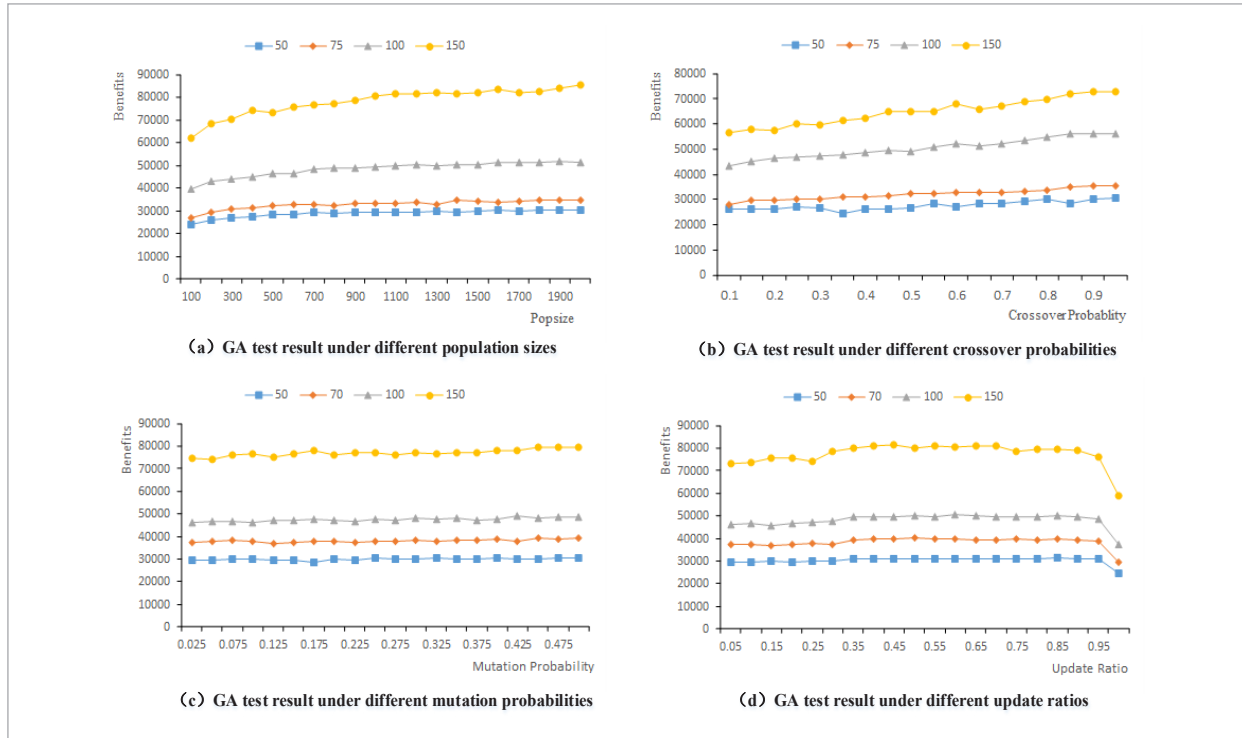
**Figure 4**

The parameter sensitivity test results for the pointer neural network



Last, sensitivity tests considering the population size, crossover probability and impact factor of the DE were conducted. The range of the population size was set to  $[100, 2000]$ , and the step size was 100; the range of the crossover probability was set to  $[0.5, 1]$ , and the step size was 0.025; the range of the impact factor was set to  $[0.1, 2]$ , and the step size was 0.025; and the range of the update ratio was set to  $[0.1, 2]$ , and the step size was 0.1. The test results for the various parameter settings are shown in Figure 6.

**Figure 5**  
The parameter sensitivity test results for the GA



**Figure 6**  
The parameter sensitivity test results for the DE

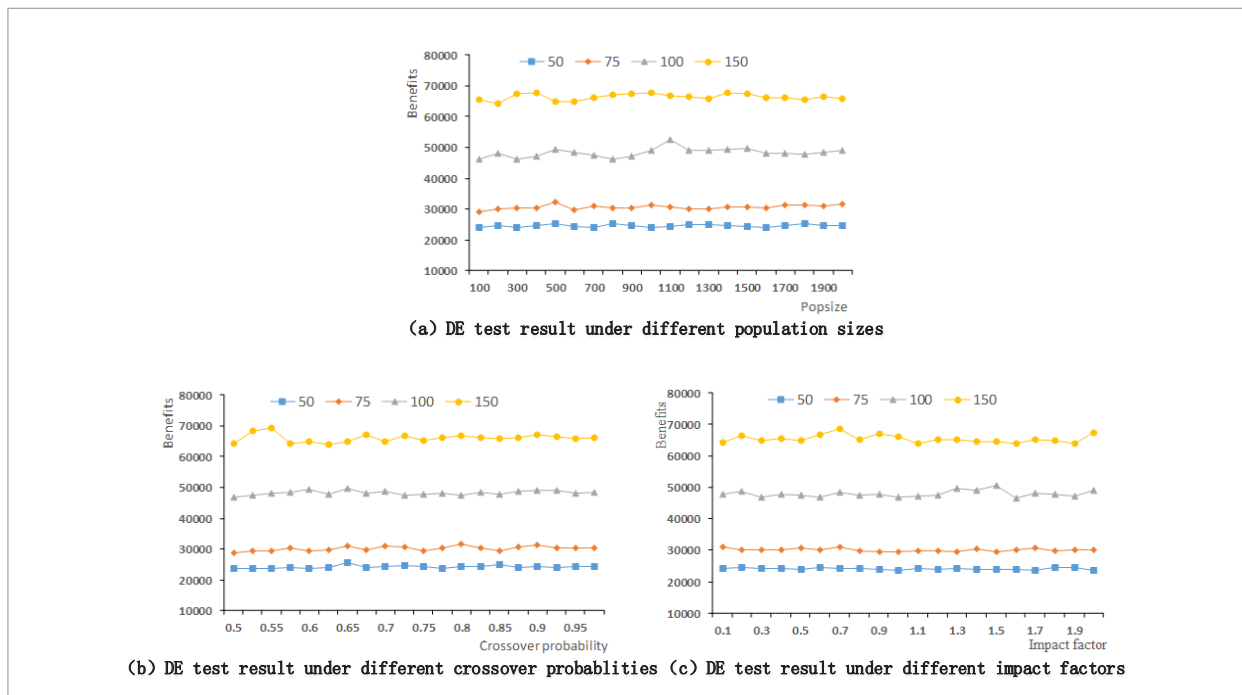
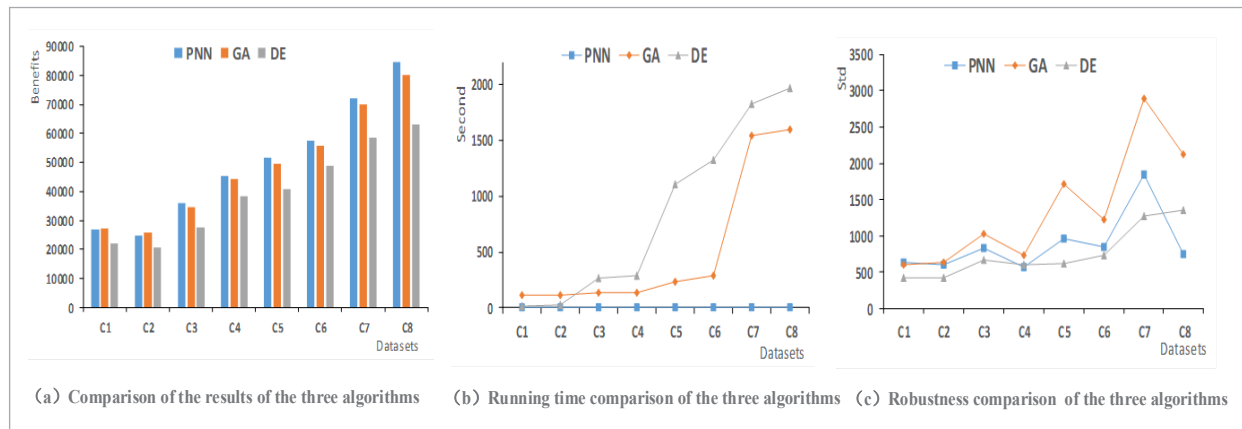




Figure 7

Performance of the three algorithms



**2 Algorithm comparison:** For each optimal parameter set, the pointer neural network, GA and DE were implemented based on 8 verification datasets (C1, C2, C3, C4, C5, C6, C7, and C8); then, the quality of the solutions, running times and robustness of the algorithms were compared. A comparison of the performance of the three algorithms is shown in Figure 7.

### 4.3. Results

Here, we describe the performance of the three algorithms. In the parameter sensitivity test, the parameters of the three algorithms have different effects on the convergence of the results, and the four parameters in the GA have stronger influences on the experimental results than do the pointer neural network parameters. In the algorithm comparison experiment, the results of the three algorithms differ for the different performance indicators, and the pointer neural network displays better performance for the medium- and large-scale datasets.

In the parameter sensitivity test of the pointer neural network, the convergence rate is higher when more data are used, and the fluctuation is large, as shown in Figure 4(a). However, the number of groups in each batch does not affect the final convergence results. As shown in Figure 4(b), upon increasing the number of hidden units in the LSTM network, the convergence speed of the neural network increases significantly, but the final convergence results remain the same. We nonetheless test the number of hidden layers in the LSTM network and the parameters in the Adam optimization algorithm, including the learning rate and

the attenuation rate. These parameters only affect the convergence speed of the pointer neural network training and do not affect the final convergence results. Therefore, to ensure a reasonable convergence speed and the stability of the pointer neural network, the number of groups and the number of hidden units in each batch are set to 128.

In the parameter sensitivity test of the GA, the growth of the population size from 50 nodes to 75 nodes has no marked influence on the vehicle benefits, as shown in Figure 5(a), and when the node size exceeds 75, revenue growth gradually increases as the population scale increases. As shown in Figure 5(b), for different numbers of nodes, the yield value increases gradually with increasing crossover probability, and when the crossover probability exceeds 0.9, the benefits decrease. As shown in Figure 5(c), the probability of variation has no significant influence on the benefits of the four instance sizes. As shown in Figure 5(d), when the update ratio is between 0.05 and 0.3 or between 0.45 and 0.9, the benefits are largely stable. When the update ratio is approximately 0.4, the benefits at the four instance scales peak, and when the update ratio exceeds 0.9, the benefits decrease significantly. Therefore, for the four instance sizes, the population size of the GA is set to 1500, 1500, 1900, and 2000; the crossover probability of the GA is set to 0.95, 0.9, 0.9, and 0.9; the mutation probability of the GA is set to 0.25, 0.45, 0.475, and 0.45; and the update ratio of the GA is set to 0.4, 0.5, 0.4, and 0.45.

In the parameter sensitivity test of the DE, the sensitivity test results of the three parameters from 50 nodes to 75 nodes showed no significant influence, as

shown in Figure 6. However, the peak of the population size of 100 nodes and 150 nodes occurred at 1100 and 1400, respectively, as shown in Figure 6(a). As shown in Figure 6(b), when the crossover probability of 150 nodes increases to 0.55, the revenue grows rapidly to the peak. As shown in Figure 6(c), the peak of the impact factor of 100 nodes and 150 nodes has a distinct difference; the peak of 100 nodes is at 1.5, and that of 150 nodes is at 0.7. Therefore, for the four instance sizes, the population size of the DE is set to 800, 500, 1100, and 1400; the crossover probability of the DE is set to 0.65, 0.8, 0.65, and 0.55; and the impact factor of the DE is set to 0.6, 0.7, 1.5, and 0.7.

As shown in Figure 7(a), we can find that the results of DE are consistently the worst of the three algorithms, while the results of the pointer neural network are close to those of the GA when the node number is 50. As the node number is increased to 75 and 100, the results of the pointer neural network gradually exceed those of the GA; the performance of the network for C3 to C6 is superior to that of the GA and DE. Furthermore, when the node number is increased to 150, the results of the DE are differ significantly from the other two algorithms. Except for the comparison of results, the time required by the pointer neural network is much lower than that required by the GA and DE. The solution time of the eight test datasets is consistently within 2 seconds, and the stability is high, as shown in Figure 7(b). However, the solution time of the two heuristic algorithms increases significantly as the problem scale increases, and the running time of the DE is longer than that of the GA. When the problem scale increases to 150, more than 25 minutes are required to solve the problem. As shown in Figure 7(c), the volatility of the GA maintains a largely stable level in the 50-node case, and the volatility increases significantly as the node number increases. Compared with the GA, the pointer neural network and DE are more stable at different scales, and the standard deviation of the DE results is the minimum of the three algorithms.

In light of different test dataset scales, the parameters in the neural network are largely stable after the pointer neural network is adequately trained, so the functional relationship between the input and output can be effectively learned. The pointer neural network can be used to obtain a feasible solution in a

very short time, and the output is consistently stable. Compared with the pointer neural network, the GA or DE can stably output a feasible solution based on small-scale examples. In large-scale instances, the end of the evolution can easily arrive at different local optimal values due to the large scale of the problem. The results obtained in each experiment converge to local extreme points, and the difference in the results is large, which leads to the high volatility of the solution results. Therefore, the pointer neural network method proposed in this paper is most suitable for medium- to large-scale instances. After providing sufficient data training, the proposed method can output multiple feasible solutions in a very short time, and the decision makers can choose the best feasible solution.

---

## 5. Conclusion

In the VRPTPLR, the available resources and urgency of customer tasks are considered common constraints, and the supplier filters the visited customers with the objective of maximizing profits and arranging the vehicles to meet customer demands. We applied a novel machine learning method to improve the current vehicle routing scheme. First, a pointer neural network model was proposed to ensure the feasibility of solution. Then, training datasets were used to find the optimal parameter configuration and continuously optimize the parameters of the neural network until the trained neural network yielded excellent feasible solutions for the verification sets. According to the experimental results, the performance of the pointer neural network in medium- and large-scale instances is similar to that of the GA, and the performance of the proposed method for large-scale instances is superior to that of the GA and DE based on three indicators, including the running time, solution quality and robustness. The machine learning method not only solves the problem studied in this paper but also provides a new solution to variants of the VRP.

## Acknowledgments

This research was supported by the National Natural Science Foundation of China (71472058, 71871079, 71671059). The authors thank American Journal Experts for English language editing during the preparation of this manuscript.

## References

1. Arnold, F., Gendreau, M., Sorensen, K. Efficiently Solving Very Large-scale Routing Problems. *Computers & operations Research*, 2019, 107, 32-42. <https://doi.org/10.1016/j.cor.2019.03.006>
2. Battarra, M., Erdogan, G., Vigo, D. Exact Algorithms for the Clustered Vehicle Routing Problem. *Operations Research*, 2014, 62(1), 58-71. <https://doi.org/10.1287/opre.2013.1227>
3. Bello, I., Pham, H., Le, Q., Norouzi, M., Bengio, S. *Neural Combinatorial Optimization with Reinforcement Learning*. Arxiv, 2016.
4. Bradley, R., Tagkopoulos, I., Kim, M., Kokkinos, Y., Pagniotakos, T., Kennedy, J., De Meyer, G., Watson, P., Elliott, J. Predicting Early Risk of Chronic Kidney Disease in Cats Using Routine Clinical Laboratory Tests and Machine Learning. *Journal of Veterinary Internal Medicine*, 2019, 33(6), 2644-2656. <https://doi.org/10.1111/jvim.15623>
5. Christofides, N. Worst-Case Analysis of a New Heuristic for the Traveling Salesman Problem. *Technical Report*, 1976.
6. Cooray P. L. N. U., Rupasinghe T. D. Machine Learning-Based Parameter Tuned Genetic Algorithm for Energy Minimizing Vehicle Routing Problem. *Journal of Industrial Engineering*, 2017, 2017, 3019523. <https://doi.org/10.1155/2017/3019523>
7. Hintsch, T., Irnich, S. Exact Solution of the Soft-Clustered Vehicle-Routing Problem. *European Journal of Operational Research*, 2020, 280(1), 164-178. <https://doi.org/10.1016/j.ejor.2019.07.019>
8. Ibn Faiz, T., Vogiatzis, C., Noor-E-Alam, M. A Column Generation Algorithm for Vehicle Scheduling and Routing Problems. *Computers & Industrial Engineering*, 2019, 130, 222-236. <https://doi.org/10.1016/j.cie.2019.02.032>
9. Kucukoglu, I., Dewil, R., Cattrysse, D. Hybrid Simulated Annealing and Tabu Search Method for the Electric Travelling Salesman Problem with Time Windows and Mixed Charging Rates. *Expert Systems with Applications*, 2019, 134, 279-303. <https://doi.org/10.1016/j.eswa.2019.05.037>
10. Li, J. L., Fu, D. W., Yuan, Q., Zhang, H. H., Chen, K. H., Yang, S., Yang, F. C. A Traffic Prediction Enabled Double Rewarded Value Iteration Network for Route Planning. *IEEE Transactions on Vehicular Technology*, 2019, 68(5), 4170-4181. <https://doi.org/10.1109/TVT.2019.2893173>
11. Li, R., Zhang, B. P., Kang, D. J., Teng, Z. Deep Attention Network for Person Re-Identification with Multi-loss. *Computers & Electrical Engineering*, 2019, 79. <https://doi.org/10.1016/j.compeleceng.2019.106455>
12. Liu, R., Jiang, Z. B. A Hybrid Large-neighborhood Search Algorithm for the Cumulative Capacitated Vehicle Routing Problem with Time-Window Constraints. *Applied Soft Computing*, 2019, 80, 18-30. <https://doi.org/10.1016/j.asoc.2019.03.008>
13. Lu, L. C., Ue, T. W. Mission-oriented Ant-Team ACO for Min-Max MTSP. *Applied Soft Computing*, 2019, 76, 436-444. <https://doi.org/10.1016/j.asoc.2018.11.048>
14. Luong, M. T., Pham, H., Manning, C. Effective Approaches to Attention-based Neural Machine Translation. *Association for Computational Linguistics*, 2015. <https://doi.org/10.18653/v1/D15-1166>
15. Molina, J. C., Eguia, I., Racero, J. An Optimization Approach for Designing Routes in Metrological Control Services: a case study. *Flexible Services and Manufacturing Journal*, 2018, 30(4), 924-952. <https://doi.org/10.1007/s10696-016-9265-3>
16. Mukhutdinov, D., Filchenkov, A., Shalyto, A., Vyatkin, V. Multi-agent Deep Learning for Simultaneous Optimization for Time and Energy in Distributed Routing System. *Future Generation Computer Systems-The International Journal of Escience*, 2019, 94, 587-600. <https://doi.org/10.1016/j.future.2018.12.037>
17. Nakajima, H., Iima, H. The Solution of Combinatorial Optimization Problems Based on Reinforcement Learning. *Proceedings of the 2017 International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence*, 2017, 78-82. <https://doi.org/10.1145/3059336.3059342>
18. Nazari, M., Oroojlooy j. A., Snyder, L., Takáč, M. Deep Reinforcement Learning for Solving the Vehicle Routing Problem. *Advances in Neural Information Processing Systems*, 2018.
19. Pei, J., Mladenovic, N., Urosevic, D., Brimberg, J., Liu, X. B. Solving the Traveling Repairman Problem with Profits: A Novel Variable Neighborhood Search Approach. *Information Sciences*, 2020, 507, 108-123. <https://doi.org/10.1016/j.ins.2019.08.017>
20. Pengle, Z., Yajie, D. A fast Dynamic Programming Algorithm to a Varied Capacity Problem in Vehicle Routing. *International Journal of Applied Decision Sciences*, 2018, 11(2), 146-167. <https://doi.org/10.1504/IJADS.2018.090924>

21. Pessoa, A., Sadykov, R., Uchoa, E. Enhanced Branch-Cut-and-Price Algorithm for Heterogeneous Fleet Vehicle Routing Problems. *European Journal of Operational Research*, 2018, 270(2), 530-543. <https://doi.org/10.1016/j.ejor.2018.04.009>
22. Polap, D. Analysis of Skin Marks Through the Use of Intelligent Things. *IEEE Access*, 2019, 7, 149355-149363. <https://doi.org/10.1109/ACCESS.2019.2947354>
23. Ruiz, E., Soto-Mendoza, V., Barbosa, A. E. R., Reyes, R. Solving the Open Vehicle Routing Problem with Capacity and Distance Constraints with a Biased Random Key Genetic Algorithm. *Computers & Industrial Engineering*, 2019, 133, 207-219. <https://doi.org/10.1016/j.cie.2019.05.002>
24. Schermer, D., Moeini, M., Wendt, O. A Hybrid VNS/Tabu Search Algorithm for Solving the Vehicle Routing Problem with Drones and En Route Operations. *Computers & Operations Research*, 2019, 109, 134-158. <https://doi.org/10.1016/j.cor.2019.04.021>
25. Sedighzadeh D, Mazaheripour H. Optimization of Multi Objective Vehicle Routing Problem Using a New Hybrid Algorithm Based on Particle Swarm Optimization and Artificial Bee Colony Algorithm Considering Precedence Constraints. *Alexandria Engineering Journal*, 2018, 57(4):2225-2239. <https://doi.org/10.1016/j.aej.2017.09.006>
26. Sutskever, I., Vinyals, O., Le, Q. Sequence to Sequence Learning with Neural Networks. *Advances in Neural Information Processing Systems*, 2014, 4, 3104-3112.
27. Triki C, Akil J, Al-Azri N. Optimising the Periodic Distribution of Gas Cylinders with Customers Priority. *International Journal of Operational Research*, 2017, 28(2), 279-289. <https://doi.org/10.1504/IJOR.2017.081478>
28. Vahdani, B., Veysmoradi, D., Shekari, N., Mousavi, S.M. Multi-objective, Multi-period Location-routing Model to Distribute Relief After Earthquake by Considering Emergency Roadway Repair. *Neural Computing & Applications*, 2018, 30(3), 835-854. <https://doi.org/10.1007/s00521-016-2696-7>
29. Vinyals, O., Fortunato, M., Jaitly, N. Pointer Networks. *Advances in Neural Information Processing Systems*, 2015, 28, 2692-2700.
30. Wang, H. B., Gu, M.Z., Yu, Q., Tao, Y., Li, J. J., Fei, H. H., Yan, J., Zhao, W., Hong, T. J. Adaptive And Large-Scale Service Composition Based on Deep Reinforcement Learning. *Knowledge-Based Systems*, 2019, 180, 75-90. <https://doi.org/10.1016/j.knosys.2019.05.020>
31. Yu, J. J. Q., Yu, W., Gu, J. T. Online Vehicle Routing with Neural Combinatorial Optimization and Deep Reinforcement Learning. *IEEE Transactions on Intelligent Transportation Systems*, 2019, 20(10), 3806-3817. <https://doi.org/10.1109/TITS.2019.2909109>
32. Yu, Y., Wang, S. H., Wang, J. W., Huang, M. A Branch-and-Price Algorithm for the Heterogeneous Fleet Green Vehicle Routing Problem with Time Windows. *Transportation Research Part B-methodological*, 2019, 122, 511-527. <https://doi.org/10.1016/j.trb.2019.03.009>
33. Zhang, H. Z., Zhang, Q. W., Ma, L., Zhang, Z. Y., Liu, Y. A Hybrid Ant Colony Optimization Algorithm for a Multi-objective Vehicle Routing Problem with Flexible Time Windows. *Information Sciences*, 2019, 490, 166-190. <https://doi.org/10.1016/j.ins.2019.03.070>